Middle East Technical University
Institute of Applied Mathematics

Post-Quantum Cryptography and NTT as a Polynomial Multiplication Method

Aslı Ebru KAYA
(Cryptography)

Advisor: Assoc. Prof. Dr. Oğuz YAYLA

Term Project Report
January 2023

Abstract

Cryptology has been a crucial element in the rapidly evolving technology. Without a successful encryption, no one and no system can be secure. The advancements in another aspect of technology, namely the quantum computers, have turned out to be the destructive game changer for the schemes used in crypto-systems, which have been deemed to be safe until now in the era of classical computers. The widely used state of the art public-key crypto-schemes are not resistant to predicted quantum attacks. The threat posed by quantum computers gave a way for new research area, which is called post-quantum cryptology. Lattice based schemes seem to be the most promising crypto-systems among possible post-quantum cryptographic schemes. Polynomial multiplications are the bottleneck of the lattice based schemes since they are fundamentally used in these schemes and take considerable amount of time and power. There are different methods for polynomial multiplications with various levels of complexity. Simple and straightforward schoolbook method has a quadratic complexity which makes it unfeasible to be utilized in the crypto-schemes. Number Theoretic Transform, which is a specific case of Fast Fourier Transform, seems to be the most favorable polynomial multiplication method over finite fields with its almost linear complexity of $O(nlogn)$. Since post-quantum cryptographic lattice based schemes deal with polynomials that have coefficients of integers, Number Theoretic Transform, in which all procedure is applied in the ring of integers, suits well to avoid any round-off error and accelerate the computations while reducing computational complexity considerably.

Öz

Kriptoloji, hızla gelişen teknolojide çok önemli bir unsur olmuştur. Başarılı bir şifreleme olmadan hiç kimse ve hiçbir sistem güvende olamaz. Teknolojinin bir başka alanındaki gelişmeler, yani kuantum bilgisayarlar, klasik bilgisayarlar çağında bugüne kadar güvenli kabul edilen kripto-sistemlerde kullanılan şemalar için yıkıcı bir oyun değiştirici haline geldi. Yaygın olarak kullanılan son teknoloji açık anahtarlı kripto şemaları, gelecekteki kuantum saldırılarına karşı dirençli değildir. Kuantum bilgisayarların oluşturduğu tehdit, kuantum sonrası kriptoloji adı verilen yeni bir araştırma alanının doğmasına yol açtı. Kafes tabanlı şemalar, olası kuantum sonrası kriptografik şemalar arasında en umut verici şifreleme sistemleri olarak görünmektedir. Polinom çarpımları, bu şemalarda temelde kullanıldıkları ve önemli miktarda zaman ve güç harcadıkları için kafes tabanlı şemaların darboğazıdır. Çeşitli karmaşıklık seviyelerine sahip polinom çarpımları için farklı yöntemler bulunmaktadır. Basit ve anlaşılır olan standart yöntem, kripto şemalarında kullanılmasını olanaksız kılan ikinci dereceden bir karmaşıklığa sahiptir. Hızlı Fourier Dönüşümünün özel bir durumu olan Sayı Teoretik Dönüşüm, neredeyse doğrusal $O(nlogn)$ karmaşıklığı ile sonlu cisim üzerinde en uygun polinom çarpım yöntemi olarak değerlendirilmektedir. Kuantum sonrası kriptografik kafes tabanlı şemalar, tamsayı katsayılarına sahip polinomlarla ilgilendiğinden, tüm prosedürü tamsayılar alanında uygulanan Sayı Teoretik Dönüşüm metodu, hesaplama karmaşıklığını önemli ölçüde azaltırken, herhangi bir yuvarlama hatasından kaçınmak ve hesaplamaları hızlandırmak için de uygundur.

Contents

Chapter 1

Introduction

Advancement in technology is pleasant most of the time, but sometimes technology itself can be the enemy of the advancement. Interconnectedness and data are the very crucial characteristics of the technology as we know it for the last decade or two, and it seems that future technology will gradually be more dependent on information exchange. Devices are in a fast trend to become smart. Any frequently used technological device conducts a sort of communication with any other distant party on peer to peer level or via a cloud. Home appliances, cellular phones, watches, even basic ordinary new cars are in the big interconnected architecture and it is almost certain that internet of things will be the very basic aspect of the technology in the next era. There is one very fundamental requirement that is needed to be complied with for the sustainability of this technological perspective. It is security. Sharing the information securely within the aspects of the technology is the main enabler. Security is not only against adversaries but also to preserve the integrity of the data. Any corrupted data entered into or information leakage from the insecure system, which may be a result of any deliberate action or unintentional error, can end up with a catastrophe in the highly interconnected and cross-dependent technological environment.

"Cryptology" is derived from Greek words "kryptos" and "logos" which means hidden word. Cryptology, as a branch of science, deals with the data and communication security. Importance of the cryptology became evident in World War II when Germany designed and started to use the machine called Enigma. Enigma had a key-space of about 18 bits. With the help of the encryption, Germany could send messages of plans and strategies to distant troop centers ordering the type, place, time of the attacks without being disclosed. This was almost the end of the other part in the war if they could not decrypt the message. Enigma was believed to be so secure that even crucial top-secret military communication was accomplished over the machine. However, Enigma codes were broken in the end and faith of the war had been turned very rapidly.

Cryptology has been a crucial element in the rapidly evolving technology. Without a successful encryption, no one and no system can be secure. However, the advancements in another aspect of technology, namely the quantum computers, have turned out to be life-threatening for the schemes used in crypto-systems, which have been deemed to be safe in the era of classical computers. Techniques used in cryptology are as secure as the hardness of the mathematical problems that they are built-on.

Two main types, namely symmetric and asymmetric, of schemes are used in the current cryptosystems. In symmetric scheme, a single secret key is used between the parties. Same key is used to encrypt the message and to decrypt it back. This key should be shared between the two parties and be kept secret from the third ones. As the number of the related parties and number of the messages to be encrypted increase, so the number of the keys do, which ultimately requires a key management system so that keys are shared between the relevant parties without breaking the confidentiality and stored with respect to the relevant message. Symmetric scheme was the only encryption method until 1976, when asymmetric scheme was introduced by Diffie and Hellman [1]. There are two keys, which are called public key and private key, in asymmetric cryptography. This scheme is also called public key cryptosystem. The two keys are mathematically related to each other but it is not feasible to compute the private key by using the public key alone. Thus, public key can be sent openly as long as the mathematical interrelation between the two keys are kept secret. In this cryptosystem, public key is used to encrypt the message and the private key is needed to decrypt the cyphertext to its original plaintext. In 1978, a practical public key system was designed by three scientists, known then as RSA, which was formed by the first letters of the names of the scientists [2].

RSA, and public key algorithms in general, have been the most widely used cryptosystems. The security of this cryptosystem relies on the complexity of the computations needed to decipher the relation of the two keys used. These high computation cost, hard problems are mostly originated from number theory, such as integer factorization and discrete logarithm problems. For example, RSA is based on integer factorization such that two large prime numbers, which is enlarging with the increase in computation power, are selected to form a product. Basically, this product is the public key while the prime numbers are private key and it is very time consuming with the computation capability of classical computers to find the prime numbers by just knowing the product. Quantum computers are preparing to be the destructive game changer in the future of cryptology. It is anticipated that quantum computers are much more efficient to find solutions to integer factorization and discrete logarithmic problems than classical computers and the cryptosystems such as RSA, Elgamal, Diffie-Hellman Elliptic Curve and Key Exchange are susceptible to be broken by quantum computers with large enough qubits [3]. Shor showed that his algorithm that can be run on a quantum computer can solve integer factorization and discrete logarithm problems in polynomial time [4].

The widely used state of the art public-key cryptoschemes are not resistant to predicted quantum attacks. It seems that it is just a matter of time for large enough quantum computers to be realized. The threat posed by quantum computers gave a way for new research area, which is called postquantum cryptology. In post-quantum cryptology, main aim is to develop a cryptographic technology which stays secure against quantum attacks. Lattice based schemes seem to be the most promising cryptosystems among the other post quantum cryptographic schemes, which are symmetric cipher and hash based schemes, multivariable based schemes, isogeny based schemes, and code based schemes, since lattice based schemes can be efficiently run with adequate bandwidth and they are secure against quantum attacks [5].

Technology has become a serious threat to technology itself. Without secured data, no technology can survive. Total collapse of the technology as we know it today is a real risk if no viable solution to secure the data could be found. To identify the main standards of the post-quantum cryptology, National Institute of Standards and Technology (NIST) of United States kicked-off a worldwide competition for standardization process in 2016. China also started a national competition based on public key post-quantum cryptology algorithms by its Chinese Association for Cryptologic Research (CACR) [6]. In the first round of NIST in 2016, 64 candidates were selected to continue and 26 of them were lattice based. In the second round, 12 lattice based schemes made it with 14 other schemes in 2019. In the third round in 2020, number of lattice based schemes were 5 in 7 candidates. And finally, in 2022, NIST announced the cryptoschemes to be standardized, of which 3 out of 4 are lattice based schemes [5, 6]. IN CACR competition, 26 schemes were lattice based in 38 proposals. 14 schemes were awarded and 11 of them were lattice based.

Lattice based schemes seem to have dominated the competition processes in the end. These lattice based cryptographs are based on mathematically hard problems such as Learning With Errors (LWE) [7], Ring LWE (RLWE) [8], Module LWE (MLWE) [9], Learning With Rounding (LRW) [10], Ring LRW (RLRW) [10], Module LWR (MLWR) [11].

Polynomial multiplications are the bottleneck of lattice based schemes since they are fundamentally used in these schemes and take considerable amount of time and power. The award winning lattice based schemes Dilithium, Kyber and Falcon in NIST use multiplication in polynomial quotient rings with integer cofficients such that $\mathbb{Z}_q[x]/\phi(x)$ where coefficients are smaller than q.

There are various methods for polynomial multiplications such as schoolbook method, Karatsuba method [12], Toom-Cook method [13] and Fourier Transform based methods Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT) [14] and Number Theoretic Transform (NTT) [15]. Simplest and most straightforward of these methods is schoolbook polynomial multiplication where a convolution process is performed between the two polynomials. However, this method requires every element of the polynomials to be multiplied one another so that multiplication of two length n polynomials requires n^2 calculations, thus it has a quadratic computational complexity of $O(n^2)$. Given the lengthy cryptographic schemes, it is not feasible to apply schoolbook method on most of the platforms because of its high computational cost and power requirements. Karatsuba algorithm is based on divide and conquer with a complexity order of $O(n^{1.58})$. Being a generalization of Karatsuba, Toom-Cook method has the complexity of $O(n^{logk(2k-1)})$. Although naïve computation of DFT still has $O(n^2)$, different algorithms based on DFT perspective have decreased the computational effort considerably. In 1965, Cooley and Tukey introduced a divide and conquer based algorithm but with exploiting the symmetry properties of the DFT matrix. This FFT algorithm has had the lowest complexity of $O(nlogn)$ so far.

NTT is just a specific case of FFT with the same complexity of FFT. NTT uses integers over a finite field whereas FFT operates under complex field. Most of the techniques in FFT are used in NTT. Since the polynomial multiplications are performed in a finite field in post-quantum cryptoschemes, NTT is the preferred algorithm in these systems.

In this project paper, a thorough definition of FFT is given in Chapter-2 to systematically cover the basic perspective of FTT to comprehend NTT. Fundamental differences between the FFT and NTT are stated. FTT Radix-2 procedure is explained and NTT Radix-2 algorithm is formed with respect to the FFT algorithm. Then, NTT Radix-3 algorithm is explained in Chapter 3 by using the main perspective gained from NTT Radix-2 procedure. In Chapter 4 a generalized formulation is given for NTT Radix-N algorithm. A brief explanation for the incomplete NTT methodology is stated in Chapter 5.

Chapter 2

Number Theoretic Transform

Discrete Fourier Transform (DFT) can efficiently be used for multiplications in polynomials. Although the standard form of DFT has the complexity of $O(n^2)$, which is the same level of complexity with that of schoolbook polynomial multiplicaftion, number of operations in the multiplication process can be reduced by using FFT methodology.

There had been different FFT algorithms with various levels of complexity reduction with respect to the original DFT until Cooley and Tukey came up with an algorithm which reduces the number of operations to $O(nlogn)$. By using the symmetrical properties of the DFT matrix, Cooley and Tukey reduced the number of computations from quadratic level to almost linear level with their butterfly methodology.

Number Theoretic Transform (NTT) is a special case of DFT and thus FFT techniques can be applied to NTT, so that the computations can be completed faster in a resource friendly way. In DFT, computations are done in complex field with a complex n^{th} root of unity, which makes the computation process prone to error because of the floating point precision. Since post-quantum cryptographic lattice based schemes deal with polynomials that have coefficients of integers, NTT suites well to avoid this round-off error since all procedure is applied in the field of integers while reducing computational complexity considerably.

2.1 Formulation of NTT

Since NTT is a special case of FFT, FFT will be described first, then differences of NTT will be stated as a special case.

Schoolbook multiplication of two polynomials, say $a(x)$ and $b(x)$, requires convolution process of $O(n^2)$ where each coefficient of both polynomials are convoluted with one another and summed up.

Definition 1 *(Schoolbook Linear Convolution).* Let $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$ be two length-n polynomials. Their multiplication is done by performing linear convolution, such that $a(x)b(x) = c(x)$ has $2n-1$ terms and degree of $2n-2$:

$$c(x) = \sum_{i=0}^{2n-2} c_i x^i$$

$$c_i = \sum_{j=0}^{i} a_i b_{i-j}$$

However, element-wise multiplication is enough by using FFT forms of both polynomials $FFTa(x)$ and $FFTb(x)$ to find $FFTc(x)$, where $FFTc = (FFTa) \cdot (FFTb)$ in frequency domain. To be able to get the result back in time domain, inverse FFT process should be completed so that $c(x)$ recovered.

Let P be a length n polynomial with coefficients a_i, where $i \in \{0, 1, \ldots, n-1\}$ such that,

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1.}$$

FFT considers point-value representation of the polynomials during the multiplication process instead of the function itself. In order to describe an n^{th} order polynomial, n distinct points are required. Thus, point-value representation of a polynomial can be given as,

$$\{y_0, y_1, \ldots, y_{n-1}\} = \{P(x_0), P(x_1), \ldots, P(x_{n-1}\}).$$

FFT intentionally uses primitive n^{th} root of unity values for $\{x_0, x_1, \ldots, x_{n-1}\}$ to exploit the symmetric properties of the resulting FFT matrix. Thus, FFT of a polynomial is stated as,

$$FFTp := FFTp[j] \text{ where,}$$

$$FFTp[j] := P(\omega^j) \text{ for } j \in \{0, 1, 2, \ldots, n-1\}.$$

Definition 2 *(Primitive n^{th} root of unity).* $\omega \in \mathbb{C}$ is defined as n^{th} root of unity iff $\omega^n = 1$. $\omega \in \mathbb{C}$ is defined as primitive n^{th} root of unity if ω also holds the inequality $\omega^k \neq 1$ for $k \in \{1, 2, \ldots, n-1\}$.

$\omega = e^{\frac{2\Pi i}{n}}$ is a primitive complex n^{th} root of unity since $\omega^k \neq 1$ for $k < n$, $\omega^n = e^{2\pi i}$ and from Euler's formula,

$$\omega^n = \cos 2\pi + i \sin 2\pi$$

$$\omega^n = 1$$

Lemma 1 *(Periodicity).* $\omega^{k+n} = \omega^k$.
proof. $\omega^{k+n} = (e^{\frac{2\Pi i}{n}})^{k+n} = e^{\frac{2\Pi i}{n}k} e^{\frac{2\Pi i}{n}n} = e^{\frac{2\Pi i}{n}k}(1) = \omega^k$ ■

Lemma 2 *(Symmetry).* $\omega^{k+\frac{n}{2}} = -\omega^k$.
proof. $\omega^{k+\frac{n}{2}} = (e^{\frac{2\Pi i}{n}})^{k+\frac{n}{2}} = e^{\frac{2\Pi i}{n}k} e^{\frac{2\Pi i}{n}\frac{n}{2}} = e^{\frac{2\Pi i}{n}k}(-1) = -\omega^k$ ■

In the FFT method, the polynomial is divided into two sub-polynomials, namely even and odd parts. Then, each even and odd parts are divided again into their even and odd parts such that FFT of length n polynomial is divided into two FFTs with length $n/2$, those two FFTs with length $n/2$ is divided into four FFTs with length $n/4$ and so on until n FFTs are left with degree 0. This method is also called Radix-2 FFT. With the help of the symmetry properties of ω, multiplications are done regarding the half of the FFT.

Definition 3 *(Radix-2 FFT).* Let $P(x)$ be a length n polynomial with the coefficients a_i, $i \in \{0, 1, \dots, n-1\}$, $n \in \mathbb{N}$ be power of 2, $FFTp := [FFTp[0], \dots, FFTp[n-1]]$ be FFT of $P(x)$.

$$FFTp[i] = \sum_{k=0}^{n-1} a_k (\omega^i)^k$$

$$FFTp[i] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^i)^{2k} + \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^i)^{2k+1}$$

$FFTp$ is formed by $[FFTp[i] \quad FFTp[i + \frac{n}{2}]]$, $i \in \{0, 1, \dots, \frac{n}{2} - 1\}$. Then,

$$FFTp[i] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^2)^{ik} + \omega^i \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^2)^{ik}$$

$$FFTp[i + \frac{n}{2}] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^2)^{(i+\frac{n}{2})k} + \omega^{(i+\frac{n}{2})} \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^2)^{(i+\frac{n}{2})k}$$

$$FFTp[i+\frac{n}{2}] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k}(\omega^2)^{ik}(\omega^n)^k - \omega^i \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1}(\omega^2)^{ik}(\omega^n)^k$$

$$FFTp[i+\frac{n}{2}] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k}(\omega^2)^{ik} - \omega^i \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1}(\omega^2)^{ik}$$

Lemma 3. Let ω be n^{th} root of unity. Then ω^2 is $(n/2)^{th}$ root of unity.
proof. $\omega^2 = (e^{\frac{2\pi i}{n}})^2 = e^{\frac{2\pi i}{\frac{n}{2}}}$ ■

Thus, coefficients of the FFT of the regarding polynomial can be found by dividing the polynomial to its odd and even parts and by using symmetry of upper and lower FFT matrix. Regarding odd and even parts are length $n/2$ polynomials, thus FFT computations are performed by using $(n/2)^{th}$ root of unity.

In short, for $i \in \{0, 1, \ldots, \frac{n}{2} - 1\}$,

$$FFTp[i] = FFTp_{even}[i] + \omega^i FFTp_{odd}[i]$$

$$FFTp[i+\frac{n}{2}] = FFTp_{even}[i] - \omega^i FFTp_{odd}[i]$$

This procedure is performed iteratively until even and odd parts are left to be degree 0.

When multiplying two polynomials $a(x)$ and $b(x)$, FFT of both polynomials should be performed by the steps stated above, so that $FFTa$ and $FFTb$ are computed. To find $FFTc$, both FFTs should be multiplied by element-wise such that,

$$FFTc = FFTa \cdot FFTb$$

$$FFTc[i] = FFTa[i] \cdot FFTb[i], i \in \{0, 1, \ldots, n-1\}$$

In order to find $c(x)$, a procedure called inverse FFT (IFFT) should be performed so that coefficients of $c(x)$ is recovered from $FFTc$.

Definition 4 *(Inverse FFT).* Let $FFTc$ with the terms $FFTc[i]$ be the FFT of length n polynomial $c(x)$ with the coefficients $n^{-1}c_i$, $i \in \{0, 1, \ldots, n-1\}$, $n \in \mathbb{N}$ be power of 2.

$$c_i = \sum_{k=0}^{n-1} FFTc[k]\omega^{-ik}$$

$$c_i = \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k](\omega^i)^{-2k} + \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k+1](\omega^i)^{-2k-1}$$

c_i is formed by $[c_i \quad c_{i+\frac{n}{2}}]$, $i \in \{0, 1, \ldots, \frac{n}{2}-1\}$. Then,

$$c_i = \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k]\omega^{-2ik} + (\omega^{-1})^i \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k+1]\omega^{-2ik}$$

$$c_{i+\frac{n}{2}} = \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k](\omega^2)^{-(i+\frac{n}{2})k} + \omega^{-(i+\frac{n}{2})} \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k+1](\omega^2)^{-(i+\frac{n}{2})k}$$

$$c_{i+\frac{n}{2}} = \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k]\omega^{-2ik}(\omega^n)^{-k} - (\omega^{-1})^i \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k+1]\omega^{-2ik}(\omega^n)^{-k}$$

$$c_{i+\frac{n}{2}} = \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k]\omega^{-2ik} - (\omega^{-1})^i \sum_{k=0}^{\frac{n}{2}-1} FFTc[2k+1]\omega^{-2ik}$$

Thus, coefficients of the $c(x)$ can be found by dividing the $FFTc$ to its odd and even parts and by using symmetry of upper and lower FFT matrix. In short, for $i \in \{0, 1, \ldots, \frac{n}{2}-1\}$,

$$c_i = c_{i,even} + \omega^{-i}c_{i,odd}$$
$$c_{i+\frac{n}{2}} = c_{i,even} - \omega^{-i}c_{i,odd}.$$

This procedure is performed iteratively until even and odd parts are left to be single term. Coefficients of the $c(x)$are computed as $n^{-1}c_i$, $i \in \{0, 1, \ldots, n-1\}$.

2.2 NTT Radix-2 Method Using Cooley-Tukey Butterfly

NTT is a special case of FFT, where the computations are done under finite field of integers.

Definition 5 *(polynomial quotient rings with integer cogefficients).* Let $\mathbb{Z}_q[x]$ be a polynomial ring over $\mathbb{Z}_q$ where $\mathbb{Z}$ is an integer ring, q is a prime number and $\mathbb{Z}_q := \{0, 1, \dots, q-1\}$. Let $\phi(x)$ be polynomial with coefficients of integers. $\mathbb{Z}_q[x]/\phi(x)$ is defined as polynomial quotient ring with integer coefficients such that coefficients are smaller than q.

Given a polynomial ring $\mathbb{Z}_q[x]/\phi(x)$, if $\phi(x)$ is a length n polynomial with integer coefficients, a polynomial in this ring is in the form of,

$$a(x) = \sum_{a=1}^{n-1} a_i x^i$$

$$a_i = \{a_0, a_1, \dots, a_{n-1}\}, a_i \in \mathbb{Z}_q$$

Since it is widely applied in post-quantum cryptographic lattice based scheme, the quotient ring form of $\mathbb{Z}_q[x]/x^n - 1$ was studied, i.e. $\phi(x) = x^n - 1$. For standard Radix-2 NTT, n should be power of 2. With the help of being radix-2, original polynomial can be divided into two parts of half length.

Since computations are to be performed under the given integer ring with quotient $x^n - 1$, complex n^{th} root of unity is not a choice any more. Thus, it should be made sure that an integer primitive n^{th} root of unity exists in the finite field bounded by the modulus of q.

Definition 6 *(n^{th} root of unity in $\mathbb{Z}_q[x]/x^n - 1$).* Let n be a power of 2, q be a prime number. n^{th} root of unity ω exists in $\mathbb{Z}_q$ if $q = 1(modn)$. ω_n is defined as primitive n^{th} root of unity if $\omega^n = 1(modq)$ and $\omega^k \neq 1(modq)$ for $k \in \{1, \dots, n-1\}$. All n^{th} roots of unity under $\mathbb{Z}_q[x]/x^n - 1$ are given as ω^j for $j \in \{1, \dots, n-1\}$.

Thus, from the definition, primitive n^{th} roots of unity exist as long as $q = kn + 1$, i.e. n divides $(q-1)$.

Symmetry and periodicity properties stated for complex n^{th} root of unity in FFT also hold for ω_n in NTT.

Proposition 1. *Periodicity of n^{th} root of unity in $\mathbb{Z}_q$: $\omega^{k+n} = \omega^k$.*

Proposition 2. *Symmetry of n^{th} root of unity in $\mathbb{Z}_q$: $\omega^{k+\frac{n}{2}} = -\omega^k$.*

NTT uses n^{th} roots of unity for the roots $\{x_0, x_1, \dots, x_{n-1}\}$ of the polynomial $p(x)$ over the quotient ring $\mathbb{Z}_q[x]/x^n - 1$ to exploit the symmetry and periodicity properties. Thus, NTT of a polynomial is stated as:

$$NTTp := NTTp[j] \text{ where,}$$

$$NTTp[j] := P(\omega^j) \text{ for } j \in \{0, 1, 2, \dots, n-1\}$$

$NTTp$ can be computed by the DFT matrix comprising the n^{th} roots of unity of the regarding polynomials.

Definition 7 *(DFT matrix of n^{th} roots of unity).* Let $p(x)$ be a length n polynomial with the coefficients a_i over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$. $i \in \{0, 1, \dots, n-1\}$, $n \in \mathbb{N}$ be power of 2. $\Omega_n := [\Omega_n[0], \dots, \Omega_n[n-1]]^{nxn}$ *mod* q is the DFT matrix of n^{th} roots of unity, where Ω_n is an n-by-n matrix with elements of $({\omega_n}^j)^k$ and $j, k \in \{0, 1, \dots, n-1\}$.

$$\Omega_n = \begin{bmatrix} ({\omega_n}^0)^0 & ({\omega_n}^0)^1 & .. & ({\omega_n}^0)^{n-1} \\ ({\omega_n}^1)^0 & ({\omega_n}^1)^1 & .. & ({\omega_n}^1)^{n-1} \\ : & : & .. & : \\ ({\omega_n}^{n-1})^0 & ({\omega_n}^{n-1})^1 & .. & ({\omega_n}^{n-1})^{n-1} \end{bmatrix} \quad mod\ q$$

Definition 8 *(NTT using DFT matrix).* Let $p(x)$ be a length n polynomial with the coefficients a_i over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$, $i \in \{0, 1, \dots, n-1\}$, $n \in \mathbb{N}$ be power of 2, $\Omega_n := [\Omega_n[0], \dots, \Omega_n[n-1]]^{nxn}$ be the DFT matrix of n^{th} roots of unity. $NTTp := [NTTp[0], \dots, NTTp[n-1]]$ be NTT of $p(x)$.

$$[NTTp] = [\Omega_n][a_i] \qquad \text{mod } q$$

$$[NTTp] = \begin{bmatrix} ({\omega_n}^0)^0 & ({\omega_n}^0)^1 & .. & ({\omega_n}^0)^{n-1} \\ ({\omega_n}^1)^0 & ({\omega_n}^1)^1 & .. & ({\omega_n}^1)^{n-1} \\ : & : & .. & : \\ ({\omega_n}^{n-1})^0 & ({\omega_n}^{n-1})^1 & .. & ({\omega_n}^{n-1})^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ : \\ a_{n-1} \end{bmatrix} \quad mod\ q$$

The DFT matrix of n^{th} roots of unity is multiplied by the polynomial coefficients to compute the NTT of the polynomial. However, direct use of DFT matrix does not reduce computational complexity. In order to reduce the complexity, the butterfly method introduced by Cooley and Tukey can be used.

With the help of the periodicity and symmetry properties of n^{th} roots of unity, complexity reduction is achieved by just performing the computations regarding only the half of the DFT matrix. Based on the Cooley-Tukey butterfly algorithm, n term NTT is divided into two $n/2$ term NTTs iteratively with divide and conquer perspective until n NTTs are left with single element.

Definition 9 *(Radix-2 NTT using Cooley-Tukey Butterfly).* Let $p(x)$ be a length n polynomial with the coefficients a_i over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$, $i \in \{0, 1, \ldots, n-1\}$, $n \in \mathbb{N}$ be power of 2, $NTTp := [NTTp[0], \ldots, NTTp[n-1]]$ be NTT of $p(x)$.

$$NTTp[i] = \sum_{k=0}^{n-1} a_k (\omega^i)^k mod q$$

$$NTTp[i] = \left(\sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^i)^{2k} + \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^i)^{2k+1} \right) \mod q$$

$NTTp$ is formed by $\left[NTTp[i] \quad NTTp[i + \frac{n}{2}]\right]$, $i \in \{0, 1, \ldots, \frac{n}{2} - 1\}$. By the symmetry and periodicity property of n^{th} roots of unity,

$$NTTp[i] = \left(\sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^2)^{ik} + \omega^i \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^2)^{ik} \right) \mod q.$$

$$NTTp[i + \frac{n}{2}] = \left(\sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^2)^{ik} - \omega^i \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^2)^{ik} \right) \mod q.$$

Proposition 3. *Let ω be n^{th} root of unity in $\mathbb{Z}_q[x]/x^n - 1$. Then ω^2 is $(n/2)^{th}$ root of unity.*

For $i \in \{0, 1, \ldots, \frac{n}{2} - 1\}$,

$$NTTp_{even}[i] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k} (\omega^2)^{ik} mod q$$

$$NTTp_{odd}[i] = \sum_{k=0}^{\frac{n}{2}-1} a_{2k+1} (\omega^2)^{ik} mod q$$

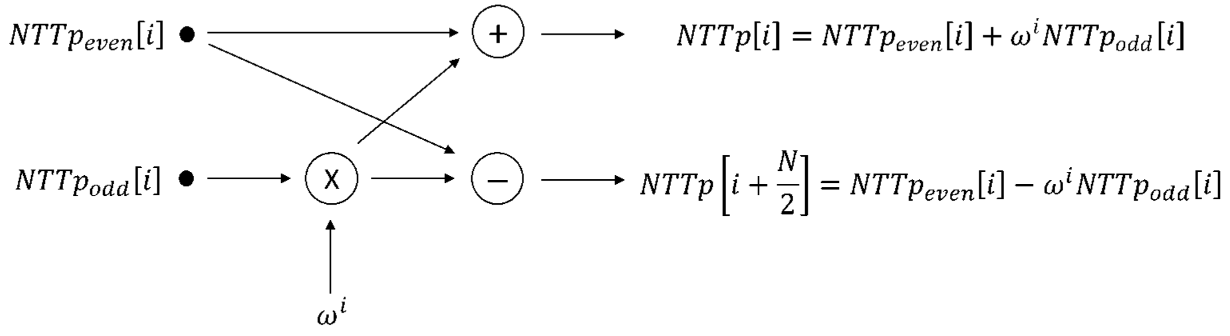
Then,

$$NTTp[i] = NTTp_{even}[i] + \omega^i NTTp_{odd}[i]$$

$$NTTp[i + \frac{n}{2}] = NTTp_{even}[i] - \omega^i NTTp_{odd}[i].$$

Thus, to compute $NTTp$, polynomial was divided into two parts, namely $NTTp_{even}$ and $NTTp_{odd}$. These parts have $n/2$ terms and it is just a simple arithmetic to compute n term $NTTp$ by using n^{th} roots of unity.

This process of forming $NTTp$ with $NTTp[i]$ and $NTTp[i+n/2]$ and computing the whole $NTTp$ by using only the halves $NTTp_{even}$ and $NTTp_{odd}$ is called Cooley-Tukey Butterfly, which reduces the complexity of the computation from quadratic level to quasi-linear manner. A schematic of the forward butterfly methodology can be seen in Figure 2.1.



$NTTp_{even}[i]$
$NTTp_{odd}[i]$
+
X
−
ω^i
$NTTp[i] = NTTp_{even}[i] + \omega^i NTTp_{odd}[i]$
$NTTp\left[i+\frac{N}{2}\right] = NTTp_{even}[i] - \omega^i NTTp_{odd}[i]$

Figure 2.1: Cooley-Tukey Butterfly for NTT calculation

In order to multiply two polynomials $a(x)$ and $b(x)$, NTT of both polynomials should be performed over $\mathbb{Z}_q[x]/x^n - 1$, so that $NTTa$ and $NTTb$ are computed. To find $NTTc$, which is NTT of $c(x)$, both NTTs should be multiplied element-wise such that,

$$NTTc = (NTTa \;\; modq) \cdot (NTTb \mod q)$$
$$NTTc[i] = (NTTa[i] \cdot NTTb[i]) \mod q, i \in \{0, 1, \ldots, n-1\}$$

Getting polynomial $c(x)$ back in time domain requires a procedure called inverse NTT (INTT). With the help of INTT, the coefficients of $c(x)$ is recovered from $NTTc \mod q$.

Definition 10 *(Inverse NTT).* Let $NTTc$ with the terms $NTTc[i]$ be the NTT of length n polynomial $c(x)$ over $\mathbb{Z}_q[x]/x^n - 1$ with the coefficients $n^{-1} \cdot c_i$, $i \in \{0, 1, \ldots, n-1\}$, $n \in \mathbb{N}$ be power of 2.

$$c_i = \sum_{k=0}^{n-1} NTTc[k]\omega^{-ik} modq$$

$$c_i = \left(\sum_{k=0}^{\frac{n}{2}-1} NTTc[2k](\omega^i)^{-2k} + \sum_{k=0}^{\frac{n}{2}-1} NTTc[2k+1](\omega^i)^{-2k-1} \right) \mod q$$

c_i is formed by $\begin{bmatrix} c_i & c_{i+\frac{n}{2}} \end{bmatrix}$, $i \in \{0, 1, \ldots, \frac{n}{2} - 1\}$. Then,

$$c_i = \left(\sum_{k=0}^{\frac{n}{2}-1} NTTc[2k]\omega^{-2ik} + \left(\omega^{-1}\right)^i \sum_{k=0}^{\frac{n}{2}-1} NTTc[2k+1]\omega^{-2ik} \right) \mod q$$

$$c_{i+\frac{n}{2}} = \left(\sum_{k=0}^{\frac{n}{2}-1} NTTc[2k]\omega^{-2ik} - \left(\omega^{-1}\right)^i \sum_{k=0}^{\frac{n}{2}-1} NTTc[2k+1]\omega^{-2ik} \right) \mod q$$

Thus, coefficients of the $c(x)$ can be found by dividing the $NTTc$ to its odd and even parts as in the forward computation using the butterfly methodology.

For $i \in \{0, 1, \ldots, \frac{n}{2} - 1\}$,

$$c_{i,even} = \sum_{k=0}^{\frac{n}{2}-1} NTTc[2k]\omega^{-2ik} modq$$

$$c_{i,odd} = \sum_{k=0}^{\frac{n}{2}-1} NTTc[2k+1]\omega^{-2ik} modq$$

Then,

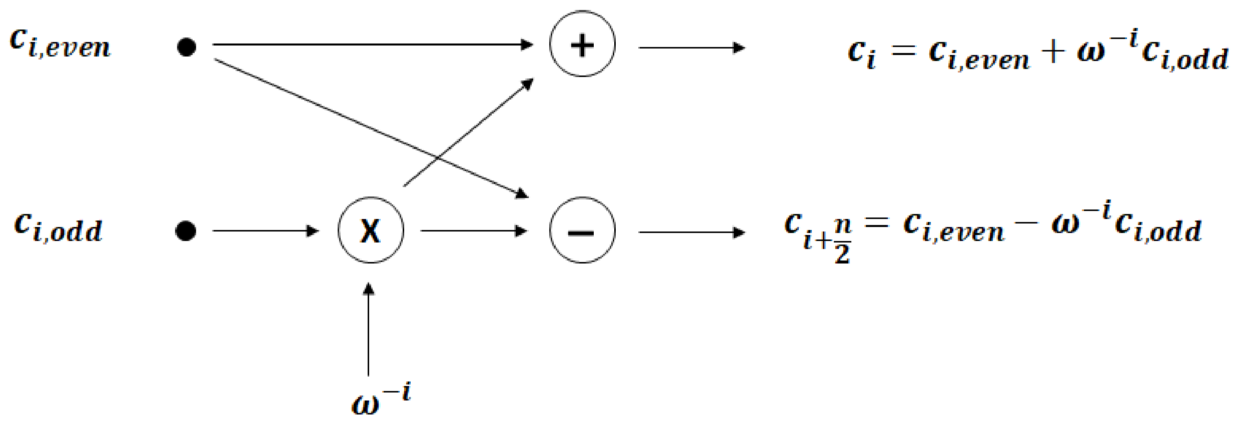
$$c_i = c_{i,even} + \omega^{-i} c_{i,odd}$$

$$c_{i+\frac{n}{2}} = c_{i,even} - \omega^{-i} c_{i,odd}.$$

This procedure is performed iteratively until even and odd parts are left to be single term. Coefficients of the $c(x)$ are computed as $n^{-1} \cdot c_i, i \in \{0, 1, \ldots, n-1\}$.

Proposition 4. Let a be a polynomial over $\mathbb{Z}_q[x]/x^n - 1$. Then $a = INTT(NTT(a))$.

The process of forming coefficients back with c_i and $c_{i+\frac{n}{2}}$ and computing the whole coefficients of the resulting polynomial by using only the halves $c_{i,even}$ and $c_{i,odd}$ is called Cooley-Tukey Butterfly as in the forward case. In the INTT butterfly procedure, inverse of the n^{th} roots of unity ω^{-1} are used in the defined finite field. A schematic of the butterfly methodology in the INTT procedure can be seen in Figure 2.2.



$c_{i,even}$
$c_{i,odd}$
ω^{-i}
$c_i = c_{i,even} + \omega^{-i} c_{i,odd}$
$c_{i+\frac{n}{2}} = c_{i,even} - \omega^{-i} c_{i,odd}$

Figure 2.2: Cooley-Tukey Butterfly for INTT calculation

2.2.1 NTT Radix-2 Cooley-Tukey Butterfly Procedure

Computations are performed over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$.

Input:
$a(x) = \sum_{i=0}^{n-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$

$b(x) = \sum_{i=0}^{n-1} b_i x^i = b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1} x^{n-1}$

Output:
$c(x) = \sum_{i=0}^{n-1} c_i x^i \ \ mod\, q = c_0 \ \ mod\, q + c_1 x \ \ mod\, q + c_2 x^2 \ \ mod\, q + \ldots + c_{n-1} x^{n-1} \ \ mod\, q$

where $c(x)$ is the result of the multiplication of linear convolution of $a(x)$ and $b(x)$, with coefficient of modulus q and polynomial roots of modulus x^{n-1}.

Process:
$a = [a_0, a_1, a_2, \ldots, a_{n-1}]$
$b = [b_0, b_1, b_2, \ldots, b_{n-1}]$
$NTTa = NTT(a)$
$NTTb = NTT(b)$
$NTTc = [NTTa[0] \cdot NTTb[0], \ldots, NTTa[n-1] \cdot NTTb[n-1]]$
$n \cdot c = INTT(NTTc)$
$c := [c_{n-1}, \ldots, c_2, c_1, c_0]$

2.2.2 Complexity of NTT Radix-2 Cooley-Tukey Butterfly Procedure

With the help of the symmetry and periodicity properties of the n^{th} order roots of unity used in NTT Cooley-Tukey Butterfly method, the length n polynomial is iteratively divided into 2 length $n/2$ polynomials and calculations are performed by using the half of the original polynomial. In each iteration, there are n additions(shown as $\oplus$) and $n/2$ multiplications(shown as $\textcircled{x}$) as shown in Figure 2.1 and Figure 2.2. The complexity of the procedure can be calculated as follows:

$$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{2}\textcircled{x} + n\oplus, \quad where\, T(1) = 0$$

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \frac{n}{4}\textcircled{x} + \frac{n}{2}\oplus\right] + \frac{n}{2}\textcircled{x} + n\oplus = 4T\left(\frac{n}{4}\right) + n\textcircled{x} + 2n\oplus$$

$$T(n) = 4\left[2T(\frac{n}{8}) + \frac{n}{8}\textcircled{x} + \frac{n}{4}\oplus\right] + n\textcircled{x} + 2n\oplus = 8T\left(\frac{n}{8}\right) + \frac{3n}{2}\textcircled{x} + 3n\oplus$$

Since $n = 2^h$ and $\log_2 n = h$ where $h \in \mathbb{Z}^+$,

$$T(n) = 2^h T\left(\frac{n}{2^h}\right) + \frac{hn}{2}\textcircled{x} + hn\oplus = nT(1) + \frac{\log_2 n}{2} n\textcircled{x} + n\log_2 n\oplus$$

$$T(n) = O(nlogn)$$

Chapter 3

Formulation for Radix-3 NTT Using Cooley-Tukey Butterfly Methodology

The original standard NTT Cooley-Tukey Butterfly method is designed for Radix-2 polynomials. However, by using a similar methodology, computations can be formulized for Radix-3 case where polynomial length n is power of 3. It should be noted that periodicity property of n^{th} root of unity still holds for Radix-3.

Definition 11 *(Radix-3 NTT based on Cooley-Tukey Butterfly Methodology).* Let $p(x)$ be a length n polynomial with the coefficients a_i over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$. $i \in \{0, 1, \dots, n-1\}$, $n \in \mathbb{N}$ be power of 3. $NTTp := [NTTp[0], \dots, NTTp[n-1]]$ be NTT of $p(x)$.

$$NTTp[i] = \sum_{k=0}^{n-1} a_k(\omega^i)^k \mod q$$

$$NTTp[i] = \left(\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^i)^{3k} + \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^i)^{3k+1} + \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^i)^{3k+2} \right) \mod q$$

$NTTp$ is formed by $\left[NTTp[i] \quad NTTp[i+\frac{n}{3}] \quad NTTp[i+\frac{2n}{3}]\right]$, $i \in \{0, 1, \dots, \frac{n}{3}-1\}$. By the periodicity property of n^{th} roots of unity,

$$NTTp[i] = \Bigg(\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik} + \omega^i \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik} + (\omega^i)^2 \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik} \mod q$$

$$\begin{aligned} NTTp[i+\frac{n}{3}] = \Bigg(&\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{(i+\frac{n}{3})k} + \omega^{(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{(i+\frac{n}{3})k} \\ &+ \omega^{2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{(i+\frac{n}{3})k}\Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{n}{3}] = \Bigg(&\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik}(\omega^n)^k + \omega^{(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik}(\omega^n)^k \\ &+ \omega^{2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik}(\omega^n)^k\Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{n}{3}] = \Bigg(&\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik} + \omega^{(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik} \\ &+ \omega^{2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik}\Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{2n}{3}] = \Bigg(&\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{(i+\frac{2n}{3})k} + \omega^{(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{(i+\frac{2n}{3})k} \\ &+ \omega^{2(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{(i+\frac{2n}{3})k}\Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{2n}{3}] = \Bigg(&\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik}(\omega^n)^{2k} + \omega^{(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik}(\omega^n)^{2k} \\ &+ \omega^{2(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik}(\omega^n)^{2k}\Bigg) \mod q \end{aligned}$$

$$NTTp[i+\frac{2n}{3}] = \Bigg(\sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik} + \omega^{(i+\frac{2n}{3})}\sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik} + \omega^{2(i+\frac{2n}{3})}\sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik}\Bigg) \mod q$$

Proposition 5. *Let ω be n^{th} root of unity in $\mathbb{Z}_q[x]/x^n - 1$. Then ω^3 is $(n/3)^{th}$ root of unity.*

For $i \in \{0, 1, \ldots, \frac{n}{3} - 1\}$, let,

$$NTTp_{null}[i] = \sum_{k=0}^{\frac{n}{3}-1} a_{3k}(\omega^3)^{ik} \mod q$$

$$NTTp_{odd}[i] = \sum_{k=0}^{\frac{n}{3}-1} a_{3k+1}(\omega^3)^{ik} \mod q$$

$$NTTp_{even}[i] = \sum_{k=0}^{\frac{n}{3}-1} a_{3k+2}(\omega^3)^{ik} \mod q$$

Then,

$$NTTp[i] = NTTp_{null}[i] + \omega^i NTTp_{odd}[i] + \omega^{2i} NTTp_{even}[i]$$

$$NTTp[i+\frac{n}{3}] = NTTp_{null}[i] + \omega^{(i+\frac{n}{3})} NTTp_{odd}[i] + \omega^{2(i+\frac{n}{3})} NTTp_{even}[i]$$

$$NTTp[i+\frac{2n}{3}] = NTTp_{null}[i] + \omega^{(i+\frac{2n}{3})} NTTp_{odd}[i] + \omega^{2(i+\frac{2n}{3})} NTTp_{even}[i]$$

Thus, to compute $NTTp$, polynomial was divided into three parts, namely $NTTp_{null}$, $NTTp_{even}$ and $NTTp_{odd}$. These parts have $n/3$ terms and it is just a simple arithmetic to compute n term $NTTp$ by using n^{th} roots of unity.

By using a similar approach based on the Cooley-Tukey Butterfly methodology used in Radix-2 NTT and thus forming $NTTp$ with $NTTp[i]$, $NTTp[i+\frac{n}{3}]$, and $NTTp[i+\frac{2n}{3}]$, whole

$NTTp$ can be computed by using $NTTp_{null}$, $NTTp_{even}$ and $NTTp_{odd}$. This procedure helps to reduce the complexity of the computation from quadratic level to quasi-linear level.

As it is the case in Radix-2 NTT, in order to multiply two polynomials $a(x)$ and $b(x)$, NTT of both polynomials should be performed over $\mathbb{Z}_q[x]/x^n - 1$, so that $NTTa$ and $NTTb$ are computed. To find $NTTc$, which is NTT of $c(x)$, both NTTs should be multiplied element-wise. Getting polynomial $c(x)$ back in time domain requires a procedure called inverse NTT (INTT). With the help of INTT, the coefficients of $c(x)$ is recovered from $NTTc$ mod q.

Definition 12 *(Inverse NTT in Radix 3).* Let $NTTc$ with the terms $NTTc[i]$ be the NTT of length n polynomial $c(x)$ over $\mathbb{Z}_q[x]/x^n - 1$ with the coefficients $n^{-1} \cdot c_i$, $i \in \{0, 1, \dots, n-1\}$, $n \in \mathbb{N}$ be power of 3.

$$c_i = \sum_{k=0}^{n-1} NTTc[k]\omega^{-ik} \mod q$$

$$c_i = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k](\omega^i)^{-3k} + \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1](\omega^i)^{-3k-1} + \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2](\omega^i)^{-3k-2} \right) \mod q$$

c_i is formed by $\begin{bmatrix} c_i & c_{i+\frac{n}{3}} & c_{i+\frac{2n}{3}} \end{bmatrix}$, $i \in \{0, 1, \dots, \frac{n}{3} - 1\}$. Then,

$$\begin{aligned} c_i = \Bigg(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k](\omega^3)^{-ik} + (\omega^{-1})^i \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1](\omega^3)^{-ik} \\ + (\omega^{-1})^{2i} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2](\omega^3)^{-ik} \mod q \end{aligned}$$

$$\begin{aligned} c_{i+\frac{n}{3}} = \Bigg(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k](\omega^3)^{-(i+\frac{n}{3})k} + \omega^{-(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1](\omega^3)^{-(i+\frac{n}{3})k} \\ + \omega^{-2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2](\omega^3)^{-(i+\frac{n}{3})k} \Bigg) \mod q \end{aligned}$$

$$c_{i+\frac{n}{3}} = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k]\omega^{-3ik}(\omega^n)^{-k} + \omega^{-(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1]\omega^{-3ik}(\omega^n)^{-k} \right.$$
$$\left. + \omega^{-2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2]\omega^{-3ik}(\omega^n)^{-k} \right) \mod q$$

$$c_{i+\frac{n}{3}} = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k]\omega^{-3ik} + \omega^{-(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1]\omega^{-3ik} \right.$$
$$\left. + \omega^{-2(i+\frac{n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2]\omega^{-3ik} \right) \mod q$$

$$c_{i+\frac{2n}{3}} = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k](\omega^3)^{-(i+\frac{2n}{3})k} + \omega^{-(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1](\omega^3)^{-(i+\frac{2n}{3})k} \right.$$
$$\left. + \omega^{-2(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2](\omega^3)^{-(i+\frac{2n}{3})k} \right) \mod q$$

$$c_{i+\frac{2n}{3}} = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k]\omega^{-3ik}(\omega^n)^{-2k} + \omega^{-(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1]\omega^{-3ik}(\omega^n)^{-2k} \right.$$
$$\left. + \omega^{-2(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2]\omega^{-3ik}(\omega^n)^{-2k} \right) \mod q$$

$$c_{i+\frac{2n}{3}} = \left(\sum_{k=0}^{\frac{n}{3}-1} NTTc[3k]\omega^{-3ik} + \omega^{-(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1]\omega^{-3ik} \right.$$
$$\left. + \omega^{-2(i+\frac{2n}{3})} \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2]\omega^{-3ik} \right) \mod q$$

Thus, coefficients of the $c(x)$ can be found by dividing the $NTTc$ to its null, odd and even parts as in the forward computation.

For $i \in \{0, 1, \ldots, \frac{n}{3} - 1\}$, let,

$$c_{i,null} = \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k]\omega^{-3ik} \mod q$$

$$c_{i,odd} = \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+1]\omega^{-3ik} \mod q$$

$$c_{i,even} = \sum_{k=0}^{\frac{n}{3}-1} NTTc[3k+2]\omega^{-3ik} \mod q$$

Then,

$$c_i = c_{i,null} + \omega^{-i} c_{i,odd} + \omega^{-2i} c_{i,even}$$

$$c_{i+\frac{n}{3}} = c_{i,null} + \omega^{-(i+\frac{n}{3})} c_{i,odd} + \omega^{-2(i+\frac{n}{3})} c_{i,even}$$

$$c_{i+\frac{2n}{3}} = c_{i,null} + \omega^{-(i+\frac{2n}{3})} c_{i,odd} + \omega^{-2(i+\frac{2n}{3})} c_{i,even}$$

This procedure is performed recursively until null, even and odd parts are left to be single term. Coefficients of the $c(x)$ are computed as $n^{-1} \cdot c_i, i \in \{0, 1, \ldots, n-1\}$.

3.1 Complexity of NTT Radix-3 Cooley-Tukey Butterfly Procedure

With the help of the periodicity properties of the n^{th} order roots of unity used in NTT Cooley-Tukey Butterfly method, the length n polynomial is iteratively divided into 3 length $n/3$ polynomials and calculations are performed by using the 1/3 of the original polynomial. In each iteration, there are $2n$ additions(shown as $\oplus$) and $2n$ multiplications(shown as $\textcircled{x}$). The complexity of the procedure can be calculated as follows:

$$T(n) = 3T\left(\frac{n}{3}\right) + 2n\textcircled{x} + 2n\oplus, \quad where\, T(1) = 0$$

$$T(n) = 3\left[3T\left(\frac{n}{3^2}\right) + \frac{2n}{3}\textcircled{x} + \frac{2n}{3}\oplus\right] + 2n\textcircled{x} + 2n\oplus = 3^2T\left(\frac{n}{3^2}\right) + 4n\textcircled{x} + 4n\oplus$$

$$T(n) = 3^2\left[3T(\frac{n}{3^3}) + \frac{2n}{3^2}\textcircled{x} + \frac{2n}{3^2}\oplus\right] + 4n\textcircled{x} + 4n\oplus = 3^3T\left(\frac{n}{3^3}\right) + 6n\textcircled{x} + 6n\oplus$$

Since $n = 3^h$ and $\log_3 n = h$ where $h \in \mathbb{Z}^+$,

$$T(n) = 3^hT\left(\frac{n}{3^h}\right) + 2hn\textcircled{x} + 2hn\oplus = nT(1) + 2n\log_3 n\textcircled{x} + 2n\log_3 n\oplus$$

$$T(n) = O(nlogn)$$

Chapter 4

Formulation for Radix-N NTT Using Cooley-Tukey Butterfly Methodology

NTT formulization can be generalized to Radix-N by using the insight gained from Radix-2 and Radix-3 cases. Thus, for any polynomial length n, where n is power of N, the generalized NTT approach explained in this chapter can be utilized to perform the polynomial multiplication with $nlogn$ complexity.

It should be noted that periodicity property of n^{th} root of unity still holds for Radix-N.

Definition 13 *(Generalized NTT formulation in Radix-N).* Let $p(x)$ be a length n polynomial with the coefficients a_i over the polynomial quotient ring $\mathbb{Z}_q[x]/x^n - 1$. $i \in \{0, 1, \ldots, n - 1\}$, $n \in \mathbb{N}$ be power of N. $NTTp := [NTTp[0], \ldots, NTTp[n-1]]$ be NTT of $p(x)$.

$$NTTp[i] = \sum_{k=0}^{n-1} a_k (\omega^i)^k \mod q$$

$$NTTp[i] = \left(\sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^i)^{Nk} + \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^i)^{Nk+1} + \ldots + \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^i)^{Nk+(N-1)} \right) \mod q$$

$NTTp$ is formed by $\begin{bmatrix} NTTp[i] & NTTp[i+\frac{n}{N}] & \ldots & NTTp[i+\frac{(N-1)n}{N}] \end{bmatrix}$, $i \in \{0, 1, \ldots, \frac{n}{N} - 1\}$. By the periodicity property of n^{th} roots of unity,

$$NTTp[i] = \left(\sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik} + \omega^i \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik} + \ldots + (\omega^i)^{N-1} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik} \quad \mod q \right.$$

$$\begin{aligned} NTTp[i+\frac{n}{N}] = \Bigg(& \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{(i+\frac{n}{N})k} + \omega^{(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{(i+\frac{n}{N})k} + \ldots \\ & + \omega^{(N-1)(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{(i+\frac{n}{N})k} \Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{n}{N}] = \Bigg(& \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik}(\omega^n)^k + \omega^{(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik}(\omega^n)^k + \ldots \\ & + \omega^{(N-1)(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik}(\omega^n)^k \Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{n}{N}] = \Bigg(& \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik} + \omega^{(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik} + \ldots \\ & + \omega^{(N-1)(i+\frac{n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik} \Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{(N-1)n}{N}] = \Bigg(& \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{(i+\frac{(N-1)n}{N})k} + \omega^{(i+\frac{(N-1)n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{(i+\frac{(N-1)n}{N})k} + \ldots \\ & + \omega^{(N-1)(i+\frac{(N-1)n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{(i+\frac{(N-1)n}{N})k} \Bigg) \mod q \end{aligned}$$

$$\begin{aligned} NTTp[i+\frac{(N-1)n}{N}] = \Bigg(& \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik}(\omega^n)^{(N-1)k} + \omega^{(i+\frac{(N-1)n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik}(\omega^n)^{(N-1)k} + \ldots \\ & + \omega^{(N-1)(i+\frac{(N-1)n}{N})} \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik}(\omega^n)^{(N-1)k} \Bigg) \mod q \end{aligned}$$

$$NTTp[i+\frac{(N-1)n}{N}] = \Big(\sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik} + \omega^{(i+\frac{(N-1)n}{N})}\sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik} + \omega^{(N-1)(i+\frac{(N-1)n}{N})}\sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik}\Big) \mod q$$

Proposition 6 *Let ω be n^{th} root of unity in $\mathbb{Z}_q[x]/x^n-1$. Then ω^N is $(n/N)^{th}$ root of unity.*

For $i \in \{0,1,\ldots,\frac{n}{N}-1\}$, let,

$$NTTp_1[i] = \sum_{k=0}^{\frac{n}{N}-1} a_{Nk}(\omega^N)^{ik} \mod q$$

$$NTTp_2[i] = \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+1}(\omega^N)^{ik} \mod q$$

$$\ldots$$

$$\ldots$$

$$NTTp_N[i] = \sum_{k=0}^{\frac{n}{N}-1} a_{Nk+(N-1)}(\omega^N)^{ik} \mod q$$

Then,

$$NTTp[i] = NTTp_1[i] + \omega^i NTTp_2[i] + \ldots + \omega^{(N-1)i}NTTp_N[i]$$

$$NTTp[i+\frac{n}{N}] = NTTp_1[i] + \omega^{(i+\frac{n}{N})}NTTp_2[i] + \ldots + \omega^{(N-1)(i+\frac{n}{N})}NTTp_N[i]$$

$$\ldots$$

$$\ldots$$

$$NTTp[i+\frac{(N-1)n}{N}] = NTTp_1[i] + \omega^{(i+\frac{(N-1)n}{N})}NTTp_2[i] + \ldots + \omega^{(N-1)(i+\frac{(N-1)n}{N})}NTTp_N[i].$$

Thus, to compute $NTTp$, polynomial was divided into N parts, namely $NTTp_1$, $NTTp_2$ upto $NTTp_N$. These parts have n/N terms and it is just a simple arithmetic to compute n

term $NTTp$ by using n^{th} roots of unity.

By using a similar approach based on the Cooley-Tukey Butterfly methodology used in Radix-2 NTT and thus forming $NTTp$ with $NTTp[i]$, $NTTp[i+\frac{n}{N}]$, and $NTTp[i+\frac{(N-1)n}{N}]$, whole $NTTp$ can be computed by using $NTTp_1$, $NTTp_2$, $\ldots$, $NTTp_N$. This procedure helps to reduce the complexity of the computation from quadratic level to quasi-linear level.

As it is the case in Radix-2 NTT, in order to multiply two polynomials $a(x)$ and $b(x)$, NTT of both polynomials should be performed over $\mathbb{Z}_q[x]/x^n-1$, so that $NTTa$ and $NTTb$ are computed. To find $NTTc$, which is NTT of $c(x)$, both NTTs should be multiplied element-wise. Getting polynomial $c(x)$ back in time domain requires a procedure called inverse NTT (INTT). With the help of INTT, the coefficients of $c(x)$ is recovered from $NTTc \mod q$.

Definition 12 *(Inverse NTT in Radix-N).* Let $NTTc$ with the terms $NTTc[i]$ be the NTT of length n polynomial $c(x)$ over $\mathbb{Z}_q[x]/x^n-1$ with the coefficients $n^{-1}\cdot c_i$, $i \in \{0,1,\ldots,n-1\}$, $n \in \mathbb{N}$ be power of N.

$$c_i = \sum_{k=0}^{n-1} NTTc[k]\omega^{-ik} \mod q$$

$$c_i = \left(\sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk](\omega^i)^{-Nk} + \sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk+1](\omega^i)^{-Nk-1} + \ldots \right.$$
$$\left. + \sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk+(N-1)](\omega^i)^{-Nk-(N-1)} \right) \mod q$$

c_i is formed by $\begin{bmatrix} c_i & c_{i+\frac{n}{N}} & \ldots & c_{i+\frac{(N-1)n}{N}} \end{bmatrix}$, $i \in \{0,1,\ldots,\frac{n}{N}-1\}$. Then, Then, similar approach in the forward NTT can be applied. ω^{-1} should be used instead of ω in the formulations. Thus, coefficients of the $c(x)$ can be found by dividing the $NTTc$ to its N sub-parts as in the forward computation.

For $i \in \{0,1,\ldots,\frac{n}{N}-1\}$,

$$c_{i,1} = \sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk]\omega^{-Nik} \mod q$$

$$c_{i,2} = \sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk+1]\omega^{-Nik} \mod q$$

$$\dots$$

$$\dots$$

$$c_{i,N} = \sum_{k=0}^{\frac{n}{N}-1} NTTc[Nk+(N-1)]\omega^{-Nik} \mod q.$$

Then,

$$c_i = c_{i,1} + \omega^{-i}c_{i,2} + \ldots + \omega^{-(N-1)i}c_{i,N}$$

$$c_{i+\frac{n}{N}} = c_{i,1} + \omega^{-(i+\frac{n}{N})}c_{i,2} + \omega^{-(N-1)(i+\frac{n}{N})}c_{i,N}$$

$$\dots$$

$$\dots$$

$$c_{i+\frac{(N-1)n}{N}} = c_{i,1} + \omega^{-(i+\frac{(N-1)n}{N})}c_{i,2} + \omega^{-(N-1)(i+\frac{(N-1)n}{N})}c_{i,N}$$

This procedure is performed recursively until sub-parts are left to be single term. Coefficients of the c(x) are computed as $n^{-1} \cdot c_i, i \in \{0, 1, \ldots, n-1\}$.

Chapter 5

Incomplete NTT Methodology

Complete NTT methodology relies on dividing the polynomials into sub-parts until single term is left such that $n = 1$ with $FFTa = a$. In the incomplete case, the procedure does not have to be followed until $n = 1$ but can be cut down earlier at higher n value. Since NTT method is advantageous for higher n values, it can be more beneficial to use other polynomial multiplication methods such as schoolbook when n becomes smaller. Thus, NTT can be used at the beginning of the multiplication process for large n values but when n is reduced to a certain limit, methodology can be switched from NTT to any other procedure. This early cut-off of the NTT process is called Incomplete NTT Method.

Incomplete NTT can also be used when the length of the polynomial is not in the neighbourhood of any power of an integer in the finite field. If n can be defined as a result of multiplicative factor with a power of an integer, i.e. $n = mN^j$, then NTT procedure can be used starting at the level $n = mN^j$ and switched off to any other preferable method after a number of j recursions at the level $n = m$. For instance, in the standard use of NTT, n is power of 2. In case polynomial length n is 12, i.e. $n = 3.2^2$, incomplete NTT methodology can be used such that standard Radix-2 NTT procedure is applied from level $n = 12$ for $j = 2$ recursions until each sub-part would have $m = 3$ elements at the level $n = 3$. Unlike the complete NTT case where final sub-parts have single elements, incomplete Radix-2 NTT procedure ends up with $m = 3$ elements in this example. Thus, polynomials are divided into sub-parts of a_{even} and a_{odd} by groups of 3 elements:

$$a_{even} = [(a_8, a_7, a_6), (a_2, a_1, a_0)]$$

$$a_{odd} = [(a_{11}, a_{10}, a_9), (a_5, a_4, a_3)]$$

In this example of incomplete NTT, recursive level reductions are performed in groups of $m = 3$ elements. In order to multiply the two polynomials $a(x)$ and $b(x)$, first, NTT of the polynomials $NTTa$ and $NTTb$ are computed using the standard NTT procedure. Then, $NTTc$ can be found by multiplying $NTTa$ and $NTTb$ element-wise in the integer quotient ring $\mathbb{Z}_q[x]/x^m - \omega^i$ for $i \in \{0, 1, \ldots, m\}$.

$$NTTc[i] = NTTa[i] \cdot NTTb[i]c(q, x^m - \omega^i), \quad i \in \{0, 1, \ldots, m\}$$

$NTTa$ and $NTTb$ are composed of N^j parts with m elements in each part. In our example, $n = 3x2^2$, a_{even} and a_{odd} has 2 parts with 3 elements in each part, $NTTa$ has 4 parts with 3 elements in each part. The same applies fore $NTTb$. $NTTc$ can be computed by schoolbook method since $NTTa$ and $NTTb$ parts are polynomials with m elements.

In order to find $c(x)$ back in continuous domain, standard INTT procedure is applied using ω^{-1}.

Chapter 6

Conclusion

Data and information security is the backbone of the interconnected and interdependent technology of today and tomorrow. With the advancement of quantum computer technology, an existential problem has arisen about the cryptographic security. Thus, technology has become a serious threat to technology itself. Considering that no technology can be implemented sustainably without secured data, the complete collapse of the technology we know today stands before us as a real risk if no viable solution can be found for proper cryto-schemes.

Post-quantum cryptography is a new research area appeared with the development in quantum computation technology. There were held world-wide competitions where the standards of the post-quantum cryptography were determined. The majority of the schemes that were selected for use in standardization process were lattice-based cryptographic systems in these competitions.

Implementation of any cryptographic scheme on different platforms is only possible if adequate amount of resources are provided, given the amount required by the schemes. Polynomial multiplications are intensively used in lattice based post-quantum cryptographic systems and since they consume extensive process power, polynomial multiplications become bottleneck for post-quantum crypto-schemes. In order to reduce the resource need of these schemes, it is a vital requirement to lower the computational complexity of the polynomial multiplications. Although there are particular multiplication methods, the NTT method described in this report enables polynomial multiplications with the lowest known computational complexity of $O(nlogn)$, which are performed in lattice-based schemes using polynomials defined on an integer ring.

The Radix-2, Radix-3 and generalized Radix-N NTT methods described in this report have been formulated in a clear way so that it can be applied on various platforms without much effort. Incomplete NTT method has also been mentioned in the report in case the polynomial quotient ring over which the multiplications are performed is not a power of some integer.

With the help of NTT method, complexity level is reduced from $O(n^2)$ quadratic regime to $O(nlogn)$ quasi-linear quantity. This denotes a substantial reduction of complexity with respect to standard schoolbook method.

Bibliography

[1] Diffie W, Hellman M, 1976, *"Multi-user cryptographic techniques"*, AFIPS Proceedings, 45, 109-112

[2] Rivest RL, Shamir A, Adleman A., 1978, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, 21 (2), 120-126

[3] Lips BAM, 2021, *"The efficiency of polynomial multiplication methods for ring based PQC algorithms of round-3 of the NIST PQC competition"*, MSc Thesis, Eindhoven University of Technology

[4] Shor PW., 1994, *"Algorithms for quantum computation: discrete logarithms and factoring"*, 124–134.

[5] Liang Z, Zhao Y., *"Number Theoretic Transform and Its Applications in Lattice-based cryptosystems: A Survey".*

[6] Wang A, Xiao D, Yu Y, 2022, "Lattice-based cryptosystems in standardisation processes: A survey", IET Inf Secur, 1-17

[7] Regev O., 2009, *"On lattices, learning with errors, random linear codes, and cryptography"*, J ACM, 56(6), 34(1)-34(40)

[8] Lyubashevsky V, Peikert C, Regev O., 2010, *"On ideal lattices and learning with errors over rings"*, In Eurocrypt, 6110, 1–23

[9] Langlois A, Stehlé D., 2015, *Worst-case to average-case reductions for module lattices*, Des Codes Cryptogr, 75(3), 565–599

[10] Banerjee A, Peikert C, Rosen A., 2012, *Pseudorandom functions and lattices*, In Eurocrypt, 7237, 719–737

[11] Alperin-Sheriff J, Apon D., 2016, *Dimension-preserving reductions from LWE to LWR*, IACR Cryptol, 589

[12] Karatsuba A, Ofman Y., 1962, *Multiplication of many-digital numbers by automatic computers*, Doklady Akademii Nauk, 145(2), 293–294

[13] Toom L., 1963, *The complexity of a scheme of functional elements realizing the multiplication of integers*, Soviet Mathematics Doklady, 3(4), 714–716

[14] Cooley JW, Tukey JW., 1965, *An algorithm for the machine calculation of complex fourier series*, Mathematics of Computation, 19(90), 297–301

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