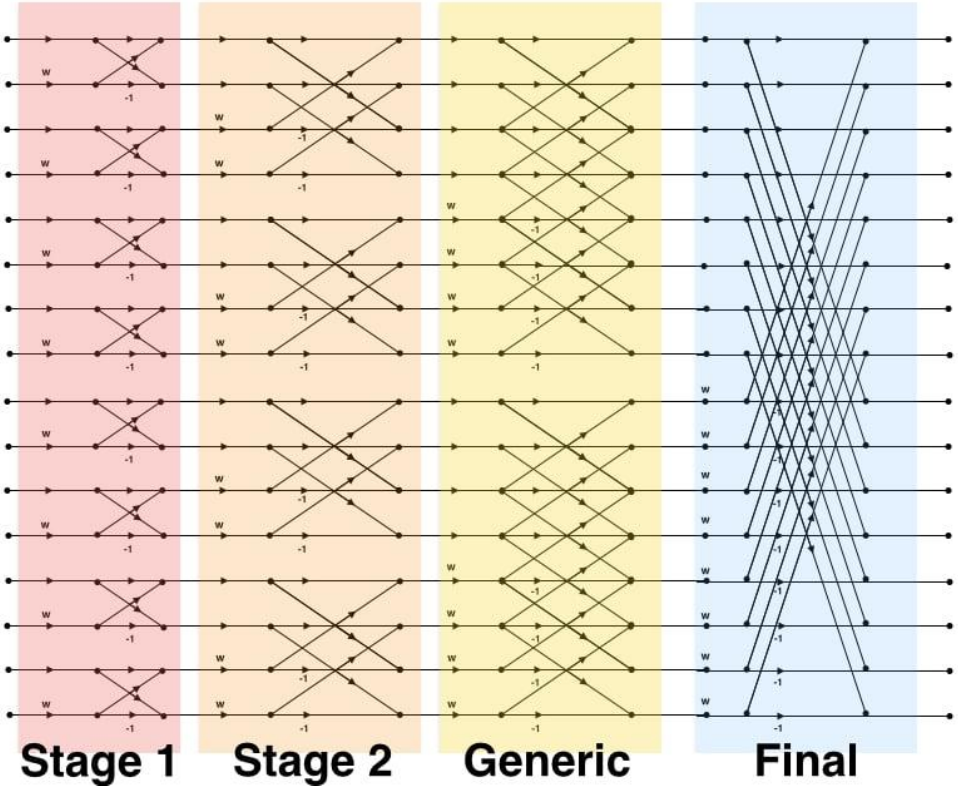
Number Theoretic Transform

What this sharing will ***not*** be about



Instead, this sharing will be about

1) Why NTTs
2) How NTTs work, from theory to practice

NTTs == FFTs?

Fast fourier transforms are just fast algorithms to calculate DFTs

The terms FFT and NTT are therefore used interchangeably in literature, but they (usually) refer to the same thing

Motivation

NTTs and why we need them

cryptography. For the lattice-based PQC, modular polynomial multiplication dominates the computations across key-generation, encryption, and decryption steps in the prior works [3], [20]. Similarly, the most expensive operation for homomorphic encryption schemes is also modular polynomial multiplication. Therefore, improving the efficiency of modular polynomial multiplication is critical to the practical deployment of lattice-based PQC schemes and homomorphic encryption.

Reference:
High-Speed VLSI Architectures for Modular Polynomial Multiplication via Fast Filtering and Applications to Lattice-Based Cryptography

Besides quantum security and better privacy, lattices also facilitate parallelism and scalability across different security levels [16]–[18]. However, implementing PQC and FHE on constrained devices, like microcontrollers in Internet of Things (IoT) applications, requires computing, bandwidth, and memory efficiency. This often demands trade-offs and optimizations of polynomial multiplications in quotient rings, operations that account for 30-50% of computation in lattice-based PQC and FHE [19]. For poly-

Reference:
Incompleteness in Number-Theoretic Transforms: New Tradeoffs and Faster Lattice-Based Cryptographic Applications

Polynomial multiplication

Schoolbook multiplication

$$G(x) = 1 + 2x + 3x^2 + 4x^3, \ and$$
$$H(x) = 5 + 6x + 7x^2 + 8x^3$$

or in vector notation:

$$\boldsymbol{g} = [1, 2, 3, 4], \ and$$
$$\boldsymbol{h} = [5, 6, 7, 8].$$

Schoolbook multiplication

$$
\begin{array}{r}
1 + 2x + 3x^2 + 4x^3 \\
5 + 6x + 7x^2 + 8x^3 \\
\hline
8x^3 + 16x^4 + 24x^5 + 32x^6 \\
7x^2 + 14x^3 + 21x^4 + 28x^5 \quad\quad\quad \\
6x + 12x^2 + 18x^3 + 24x^4 \quad\quad\quad\quad\quad\quad \\
5 + 10x + 15x^2 + 20x^3 \quad\quad\quad\quad\quad\quad\quad\quad\quad \\
\hline
5 + 16x + 34x^2 + 60x^3 + 61x^4 + 52x^5 + 32x^6
\end{array}
$$

Fig. 1: Schoolbook method for polynomial multiplication or linear convolution.

Schoolbook multiplication

This is a discrete **linear convolution** between **g** and **h**

$$\boldsymbol{y}[k] = (\boldsymbol{g} * \boldsymbol{h})[k] = \sum_{i=0}^{k} \boldsymbol{g}[i]\boldsymbol{h}[k-i]$$

Schoolbook multiplication - problems

1. Polynomial result is of degree 2d - 2
2. O(n^2) complexity - too slow!

Schoolbook multiplication - problems

1. **Polynomial result is of degree 2d - 2**
2. O(n^2) complexity - too slow!

Polynomial rings

After every polynomial operation, take the result and modulo some polynomial $\phi(x)$

Final polynomial's degree would be at most as large as the polynomial ϕ's degree

In the context of lattice cryptography, this polynomial is chosen to be something like $x^n - 1$ or $x^n + 1$

Cyclic convolutions

A convolution that 'wraps around' to be kept within a certain bound

- **Cyclic convolution.** Consider $\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b} \in \mathbb{Z}_q[x]/(x^n - 1)$, then $\boldsymbol{c} = \sum_{k=0}^{n-1} c_k x^k$, where $c_k = \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{n-1} a_i b_{k+n-i} \bmod q$, $k = 0, 1, \ldots, n-1$. And $\boldsymbol{c}$ is referred to as the cyclic convolution (CC for short)[1] of $\boldsymbol{a}$ and $\boldsymbol{b}$.

'Wrapping around' by (x^n - 1) means we can keep our resulting polynomial to be bounded by a maximum degree *n*

Cyclic convolutions

In most literature, the term ***cyclic convolution*** refers to multiplication done over the ring $\mathbb{Z}_q[x]/(x^n - 1)$, while the term ***negacyclic convolution*** refers to multiplication done over the ring $\mathbb{Z}_q[x]/(x^n + 1)$.

Alternatively, some papers suggest using terms like ***positive wrapped convolution*** vs ***negative wrapped convolution*** for clarity.

Cyclic convolutions

For PWC, the quotient ring we use is $\mathbb{Z}_q[x]/(x^n - 1)$ where $q \in \mathbb{Z}$ and c_k is defined as

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{n-1} a_i b_{k+n-i} \bmod q$$

Conversely, the quotient ring for NWC is $\mathbb{Z}_q[x]/(x^n + 1)$, and c_k can be defined as

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} - \sum_{i=k+1}^{n-1} a_i b_{k+n-i} \bmod q$$

Schoolbook multiplication - problems

1. Polynomial result is of degree 2d - 2
2. **O(n^2) complexity - too slow!**

Convolution theory

Under convolution theory, we know that convolution in one domain equals point-wise multiplication in the other domain

We can therefore transform 2 vectors of coefficients of polynomials into their NTT forms, multiply them pointwise and apply iNTT to get the result of a polynomial multiplication

$$c = INTT(NTT(a) \cdot NTT(b))$$

Nice properties of NTTs

they preserve randomness - we can directly generate a random polynomial and view it as a random polynomial already in the transformed domain

they preserve dimension and bit length - result of NTT(a) can be stored where a was in memory. The result can also be stored for use in multiple computations to save cpu cycles

NTT and iNTT

The NTT of a sequence of values (in this case, a vector of polynomial coefficients), is defined as $\hat{a} = NTT(a)$, where

$$\hat{a_j} = \sum_{i=0}^{n-1} \omega^{ij} a_i \bmod q, i = 0, 1, 2, \ldots, n-1$$

Conversely, the iNTT of a sequence of values is defined as

$$a_i = n^{-1} \sum_{j=0}^{n-1} \omega^{-ij} \hat{a}_j \bmod q, j = 0, 1, 2, \ldots, n-1$$

The difference here is that the ω is replaced by its inverse in $\mathbb{Z}_q$ and we scale the result by a factor of n^{-1} at the end.

ω here is the primitive n-th root of unity in Z_q iff

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

and

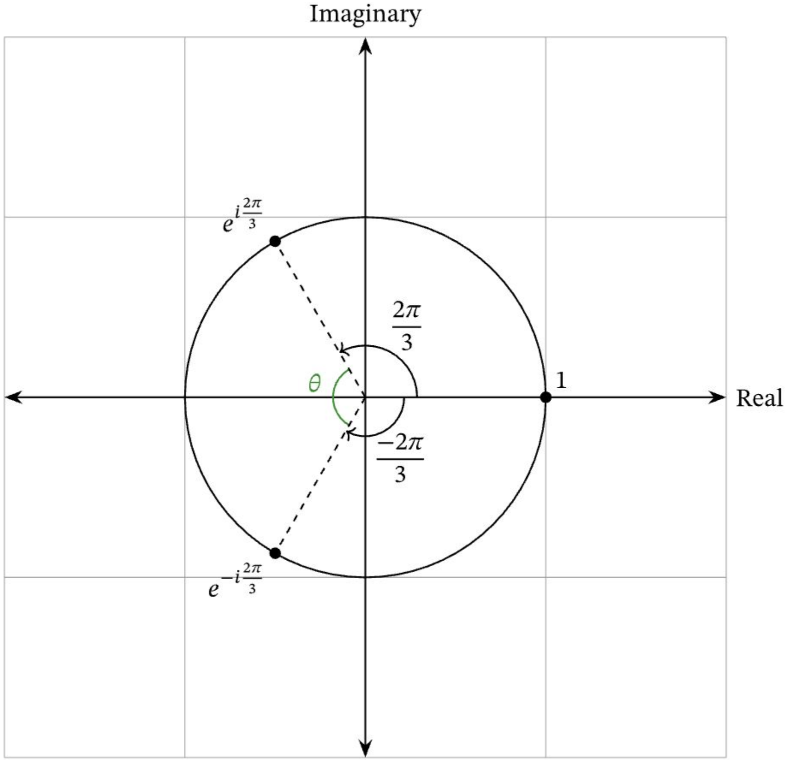
$$\omega^k \not\equiv 1 \bmod q$$

Roots of unity

A root of unity is any complex number that yields 1 when raised to some positive integer n, otherwise known as the n-th root of unity

Roots of unity form the roots of the cyclotomic polynomials which factorize the polynomial x^n - 1

Roots of unity



Example: primitive 8th roots of unity in Z_{17}

$2^4 \equiv -1 \mod 17$

$2^8 \equiv 1 \mod 17$

2 is a primitive **8th** root of unity

FFT trick

FFT trick

THEOREM 5.1 (CHINESE REMAINDER THEOREM IN RING FORM [BER01]). *Let R be a commutative ring with multiplicative identity, $I_1, I_2, \ldots, I_k$ be ideals in R that are pairwise co-prime, and I be their intersection. Then there is a ring isomorphism:*

$$\Phi : R/I \cong R/I_1 \times R/I_2 \times \cdots \times R/I_k. \tag{9}$$

In the work [Ber01], FFT trick means that according to Theorem 5.1, for polynomial rings $\mathbb{Z}_q[x]/(x^{2m} - \omega^2)$ where $m > 0$ and invertible $\omega \in \mathbb{Z}_q$, we have the following isomorphism:

$$\begin{aligned} \Phi : \mathbb{Z}_q[x]/(x^{2m} - \omega^2) \ &\cong\ \mathbb{Z}_q[x]/(x^m - \omega) \times \mathbb{Z}_q[x]/(x^m + \omega) \\ &\boldsymbol{a} \mapsto \left(\boldsymbol{a}' = \boldsymbol{a} \bmod x^m - \omega, \boldsymbol{a}'' = \boldsymbol{a} \bmod x^m + \omega\right) \end{aligned}$$

The FFT is just the FFT trick applied recursively from x^2^k - 1 all the way down to linear polynomials!

FFT trick

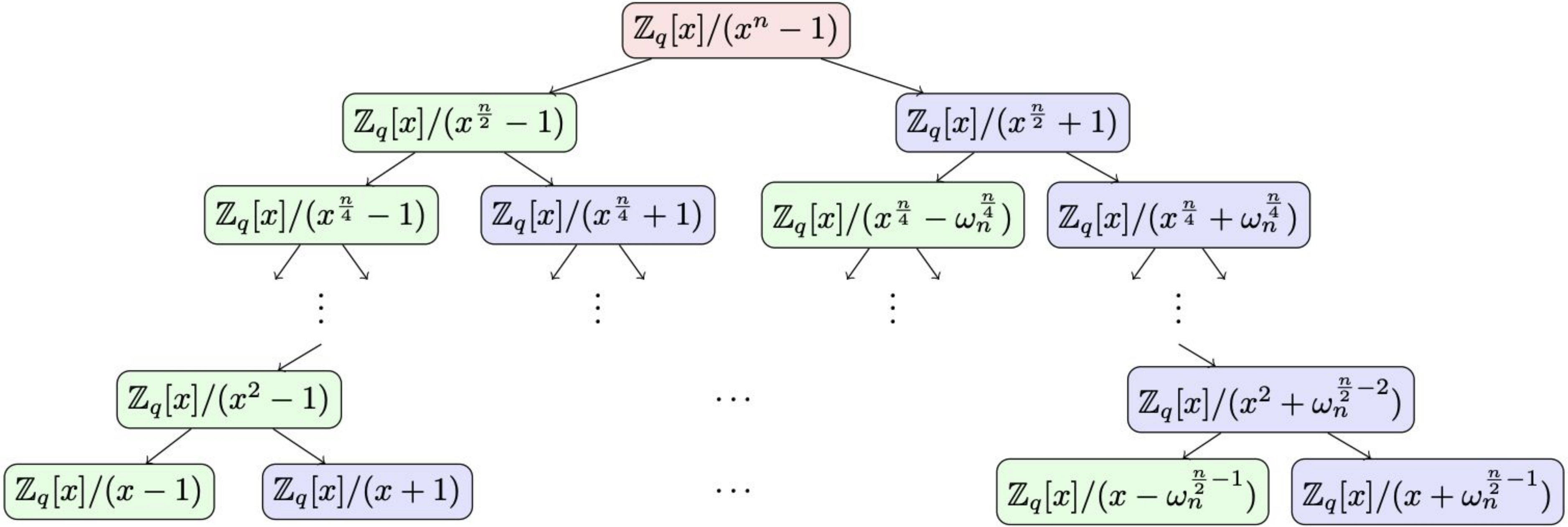


Fig. 2. CRT map of FFT trick over $\mathbb{Z}_q[x]/(x^n - 1)$

FFT trick

In the work [Ber01], FFT trick means that according to Theorem 5.1, for polynomial rings $\mathbb{Z}_q[x]/(x^{2m}-\omega^2)$ where $m>0$ and invertible $\omega \in \mathbb{Z}_q$, we have the following isomorphism:

$$\begin{aligned}\Phi : \mathbb{Z}_q[x]/(x^{2m}-\omega^2) \; &\cong \; \mathbb{Z}_q[x]/(x^m-\omega) \times \mathbb{Z}_q[x]/(x^m+\omega) \\ \boldsymbol{a} &\mapsto \left(\boldsymbol{a}' = \boldsymbol{a} \bmod x^m-\omega, \boldsymbol{a}'' = \boldsymbol{a} \bmod x^m+\omega\right)\end{aligned}$$

and the detailed mapping process:

$$\Phi\left(\sum_{i=0}^{2m-1} a_i x^i\right) = \left(\sum_{i=0}^{m-1} (a_i + \omega \cdot a_{i+m})x^i, \sum_{i=0}^{m-1} (a_i - \omega \cdot a_{i+m})x^i\right) \tag{10}$$

$$\Phi^{-1}\left(\sum_{i=0}^{m-1} a_i' x^i, \sum_{i=0}^{m-1} a_i'' x^i\right) = \sum_{i=0}^{m-1} \frac{1}{2}(a_i' + a_i'')x^i + \sum_{i=0}^{m-1} \frac{\omega^{-1}}{2}(a_i' - a_i'')x^{i+m}. \tag{11}$$

FFT trick

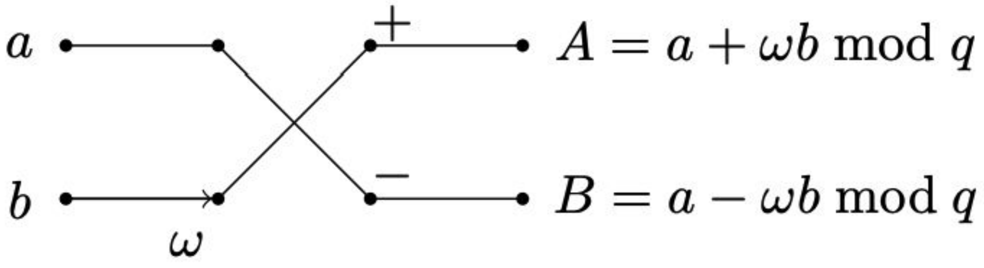
$$a_i^{'} = a_i + \omega \cdot a_{i+m}$$

$$a_i^{''} = a_i - \omega \cdot a_{i+m}$$

FFT trick

$$a_i' = a_i + \omega \cdot a_{i+m}$$

$$a_i'' = a_i - \omega \cdot a_{i+m}$$

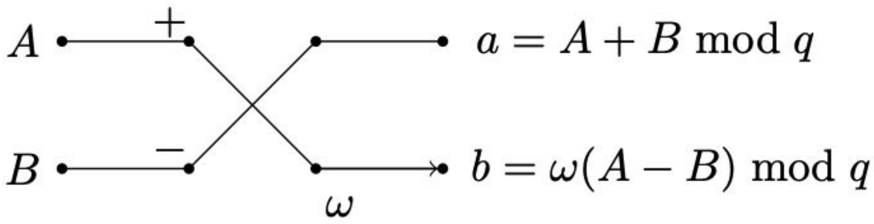


(a) Cooley-Tukey butterfly

FFT trick

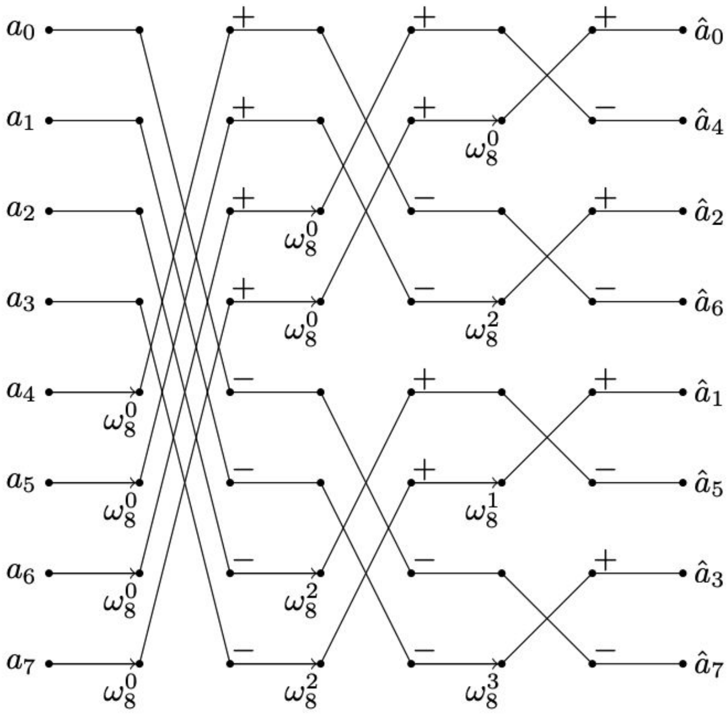
$$\boldsymbol{a_i} = (\boldsymbol{a_i'} + \boldsymbol{a_i''})/\mathbf{2}$$

$$\boldsymbol{a_{i+m}} = \boldsymbol{\omega^{-1}}(\boldsymbol{a_i'} - \boldsymbol{a_i''})/\mathbf{2}$$

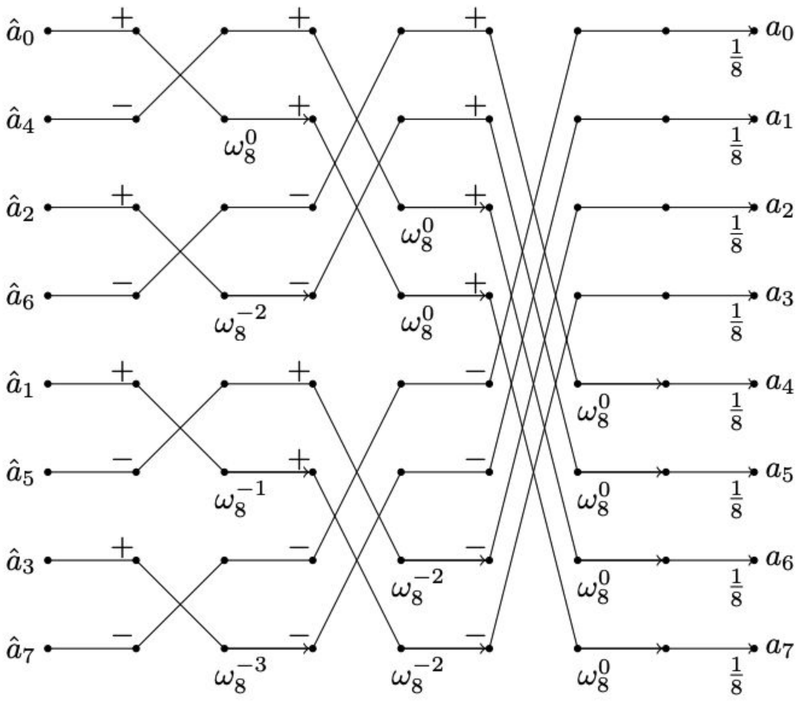


(b) Gentleman-Sande butterfly

Complete picture



(c) $\mathsf{NTT}^{CT}_{no \to bo}$



(b) $\mathsf{INTT}^{GS}_{bo \to no}$

Types of NTT - Overview

Table 3. Parameter sets of algebraically-structural lattice-based schemes in NIST PQC. Recommended parameter sets of NTRU Prime are given here. Kyber KEM, Dilithium signature and Falcon signature are standardized by NIST [NIS22]. Saber KEM and NTRU KEM were NIST PQC Round 3 finalists. NTRU Prime KEM was a alternate candidate in NIST PQC Round 3.

Schemes		n	q	Rings	Types	Methods & Algorithms
Kyber	Round 1 [ABD+17]	256	7681	$\mathbb{Z}_q[x]/(x^n+1)$	NTT-friendly $q \equiv 1 \pmod{2n}$	n-point full NWC-based NTT [ABD+17, BDK+18]
	Round 2 [ABD+19] Round 3 [ABD+20]	256	3329		NTT-friendly $q \equiv 1 \pmod{n}$	Incomplete FFT trick [ABD+19, ABD+20] Splitting polynomial ring [LSS+20, ZXZ+18]
Dilithium	Round 3 [BDK+20]	256	8380417	$\mathbb{Z}_q[x]/(x^n+1)$	NTT-friendly $q \equiv 1 \pmod{2n}$	n-point full NWC-based NTT [BDK+20]
Falcon	Round 3 [FHK+20]	512 1024	12289	$\mathbb{Z}_q[x]/(x^n+1)$	NTT-friendly $q \equiv 1 \pmod{2n}$	n-point full NWC-based NTT [FHK+20]
Saber	Round 3 [BMD+20]	256	8192	$\mathbb{Z}_q[x]/(x^n+1)$	NTT-unfriendly power-of-two q	Method based on large modulus [CHK+21, FSS20, FBR+22, ACC+22]
NTRU	Round 3 [CDH+20]	509 677	2048	$\mathbb{Z}_q[x]/(x^n-1)$	NTT-unfriendly prime n	Power-of-two n' + Method based on large modulus [FBR+22] Good's trick (+ Method based on large modulus) [CHK+21]
		701	8192			
		821	4096			
NTRU Prime	Round 3 [BBC+20]	653	4621	$\mathbb{Z}_q[x]/(x^n-x-1)$	NTT-unfriendly prime n and q	Power-of-two n' + Method based on large modulus [ACC+21] Good's trick (+ Method based on large modulus) [ACC+21, PMT+21] Schönhage's trick + Nussbaumer's trick [BBCT22]
		761	4591			
		857	5167			

References

[Number Theoretic Transform and Its Applications in Lattice-based Cryptosystems: A Survey]

[A Complete Beginner Guide to the Number Theoretic Transform (NTT)]