Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

A Generalized Number-Theoretic Transform for Efficient Multiplication in Lattice Cryptography

Ahmad Al Badawi[1*], Sze Ling Yeo[2], Mohd Faizal Bin Yusof[1]

[1] Department of Homeland Security, Rabdan Academy, Abu Dhabi, United Arab Emirates
[2] Department of Cybersecurity, A*STAR-Institute for Infocomm Research, Fusionopolis, Singapore
E-mail: aalbadawi@ra.ac.ae

Received: 18 February 2024; **Revised:** 21 March 2024; **Accepted:** 1 April 2024

Abstract: The Number Theoretic Transform (NTT) has emerged as a powerful tool for efficiently computing convolutions of digital signals, due to its inherent advantages such as numerical stability, reliance on simple integer operations, and proven efficiency. Its applications have extended to accelerating polynomial multiplication in lattice-based cryptography. However, existing NTT multiplication algorithms impose restrictions on the underlying moduli, potentially affecting key and ciphertext sizes as well as computational overhead. Therefore, enabling NTT with small moduli holds significant potential for enhancing the overall system performance. This study introduces a novel reduction framework for NTT computation in cyclotomic rings employing field extensions. Our approach replaces the underlying polynomial ring with a two-dimensional isomorphic ring, effectively relaxing the restrictions imposed on the NTT moduli. The proposed framework is evaluated through two case studies relevant to the LAC and NTTRU lattice-based cryptographic schemes. Comprehensive theoretical analysis is provided, demonstrating the effectiveness of our approach in enabling NTT with small moduli and its potential to improve the efficiency of lattice-based cryptography.

Keywords: lattice cryptography, multiplication methods, number-theoretic transform, NTTRU, LAC

MSC: 11T06, 11T22, 11T55, 11T71, 11R09, 94A60

Abbreviation

NTT	Number Theoretic Transform
PQC	Post-Quantum Cryptography
FHE	Fully Homomorphic Encryption
NIST	National Institute of Standards and Technology
PKE	Public-Key Encryption
KEM	Key Encapsulation Mechanism
DS	Digital Signature
KEE	Key Establishment and Exchange
DWT	Discrete Weighted Transform
IDWT	Inverse DWT

DOI: https://doi.org/10.37256/cm.5420244468

DGT	Discrete Galois Transform
IDFT	Inverse DGT
GDGT	Generalized DGT
DFT	Discrete Fourier Transform
FV	Fan-Vercauteren
CKKS	Cheon-Kim-Kim-Song
FFT	Fast Fourier Transform
GF	Galois Field
CRT	Chinese Remainder Theorem
EIMM	Elementary Integer Modular Multiplications
ECC	Error Correction Code

1. Introduction

Quantum computing and cloud and distributed computing are poised to transform the way we interact with the digital world, fundamentally altering how we store, communicate, and process data. Quantum computers, with their unprecedented computational power, hold the potential to solve complex problems that are intractable for classical computers, leading to breakthroughs in fields like drug discovery, materials science, artificial intelligence, and cryptography. Cloud and distributed computing, on the other hand, offers on-demand access to vast computing resources, enabling seamless collaboration and data sharing across the globe. The convergence of these two technologies promises to usher in an era of unprecedented computational power, data accessibility, and problem-solving capabilities, transforming the way we live, work, and interact in the electronic world. Despite the transformative potential of quantum computing and cloud computing, their widespread adoption necessitates careful consideration of potential security and privacy risks. Ensuring robust data protection and privacy safeguards is paramount to harnessing the full potential of these technologies while mitigating their associated risks.

On the one hand, the recent surge in quantum computing capabilities has necessitated the development of new cryptographic methods that can withstand the potential threats posed by these powerful machines. Quantum computers are rapidly advancing, with existing 1121-qubit machines [1] and roadmaps for developing 2000-qubit machines by 2033 [2]. The emergence of large-scale (multi-million qubits [3]) quantum computers threatens to break most of the classical cryptosystems that rely on integer factorization and discrete logarithm problems, which are the foundation of secure electronic communication on the Internet [4]. This looming threat has driven the search for new cryptographic methods that are resistant to quantum attacks, known as Post-Quantum Cryptography (PQC) [5].

On the other hand, the emergence of cloud and distributed computing paradigms has significantly transformed the computational landscape. Users can now leverage third-party providers for data storage and computation, offering benefits such as scalability and cost-effectiveness. Furthermore, Blockchain technology has emerged as a promising approach for ensuring secure and transparent tracking, data sharing, and authentication at scale. Blockchain's distributed ledger and cryptographic mechanisms can provide tamper-proof audit trails and establish trust between entities without a central authority [6, 7]. However, a critical challenge remains: protecting data during processing. Conventional encryption techniques effectively safeguard data at rest and in transit. However, the very nature of data processing necessitates its unencrypted state, leaving it vulnerable to cyberattacks. This vulnerability is further amplified by the in-memory storage of data during processing, increasing the exposure to unauthorized access. Therefore, robust mechanisms for securing data during processing are paramount. These concerns regarding data security and privacy within cloud computing environments necessitate further research efforts [8].

Potential solutions to address the aforementioned challenges have emerged through the development of lattice-based cryptographic methods. For instance, lattices have been employed to construct post-quantum public-key cryptosystems, ensuring the security of data communication even in the face of quantum attacks [9–11]. Moreover, several Fully Homomorphic Encryption (FHE) schemes that enable computations to be performed on encrypted data have been realized

based on lattices [12–18]. These advancements demonstrate the potential of lattice-based cryptography to address the security and privacy concerns associated with quantum computing and cloud computing. We emphasize that the scope of this work is solely focused on post-quantum lattice-based cryptosystems. Nevertheless, the concepts presented here are applicable to a broader range of lattice-based cryptosystems and any problem involving polynomial multiplication in cyclotomic rings.

In 2016, the U.S. National Institute of Standards and Technology (NIST) initiated a standardization program for Post-Quantum Cryptography, called NIST-PQC [19] targeting Public-Key Encryption (PKE), Key Encapsulation Mechanism (KEM), Digital Signature (DS), and Key Establishment and Exchange (KEE) schemes. From an initial pool of 69 proposals, 26 advanced to Round 2, followed by seven in Round 3. After six years of evaluation, four schemes were selected as the winners: CRYSTALS-Kyber [20] for key establishment, and CRYSTALS-Dilithium [21], FALCON [22], and SPHINCS$^+$ [23] for digital signatures. Note that the first three schemes are based on lattices, which shows the importance of these mathematical structures in modern cryptographic methods.

Lattice-based cryptosystems proposed in the NIST PQC competition fall into two main categories: structure-free lattices, which rely on matrix-vector operations over finite fields, and lattices with algebraic structure, which involve polynomial manipulation in cyclotomic polynomial rings. This work focused on the latter, where polynomial multiplication in a ring is the primary performance bottleneck [24, 25]. Therefore, our focus in this work is solely focused to developing a more relaxed framework to efficiently perform polynomial multiplication.

A cyclotomic ring is denoted as $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where q is the ring coefficient modulus and $f(x)$ is a degree-n polynomial modulus that is irreducible over $\mathbb{Z}$. Cyclotomic rings are advantageous due to the efficient multiplication computation using the Number Theoretic Transform (NTT) or its variants, namely, the Discrete Weighted Transform (DWT) and Discrete Galois Transform (DGT) when n is a power of 2 [26–28].

To use the typical [We use the term typical NTT-based multiplication when $q \equiv 1 \pmod{2n}$.] NTT-based multiplication algorithms, the constraint $q \equiv 1 \pmod{2n}$ must be satisfied so that $f(x)$ factors into linear factors. This implies that $q \geq 2n$, which can impose large key and ciphertext sizes unnecessarily in certain parameter sets. For instance, LAC [LAC is used throughout the text without abbreviation, consistent with the original paper introducing the concept.], Kyber and NTTRU [29–31] use relatively small ring dimensions $n = 512$, $n = 256$ and $n = 768$, respectively. For such rings the smallest moduli that can be used to enable typical NTT-multiplication are $q = 12,289$, $q = 7,681$ [Note that this modulus has been reduced further to $q = 3329$ using a variant of NTT multiplication [31]] and $q = 7681$, respectively, even though smaller moduli can be used with acceptable security level [29].

In this work, our main contribution is a reduction framework that is tailored for NTT computation in cyclotomic polynomial rings. We consider two of the most commonly used classes for the ring dimension n, namely when n is a power of 2 or a product of a power of 2 and a power of 3. Our framework generalizes existing transform computations by allowing for computation in field extensions. The framework replaces the underlying 1-dimensional ring with an isomorphic 2-dimensional ring, as Figure 1 shows. The reduction offers several important benefits. For instance, it relaxes some of the restrictions imposed on the transform moduli, especially their magnitude. Enabling NTT computation with small modulus is of paramount importance due to its effect on the key and ciphertext sizes and computational overhead. It also offers a way to control the transform length, which can be highly beneficial for reusing existing optimized compute data-paths with minimal modifications. Moreover, the reduction can be useful for hardware platforms where fast memory resources are scarce and/or special memory access patterns are preferred such as GPUs [32].

To show the advantage of our reduction framework, we present methods to enable NTT-based ring multiplication for two hypothetical rings that are inspired by the LAC and NTTRU cryptosystems [33, 31]. These two case studies are selected to demonstrate how our framework can be applied to different forms of n and values of q which are either close to, or smaller than the proposed parameters. Note that the original proposal of LAC uses the parameter set $(n, q) = (512, 257)$ that is not friendly with typical NTT algorithms. Likewise, NTTRU employs the parameter set $(n, q) = (768, 7681)$, which requires special treatment to be compatible with typical NTT algorithms [31]. Security analyses are provided to show that the proposed parameters do not affect the security of the original schemes. Moreover, we provide a theoretical analysis to study the computational complexity of the proposed multiplication methods.

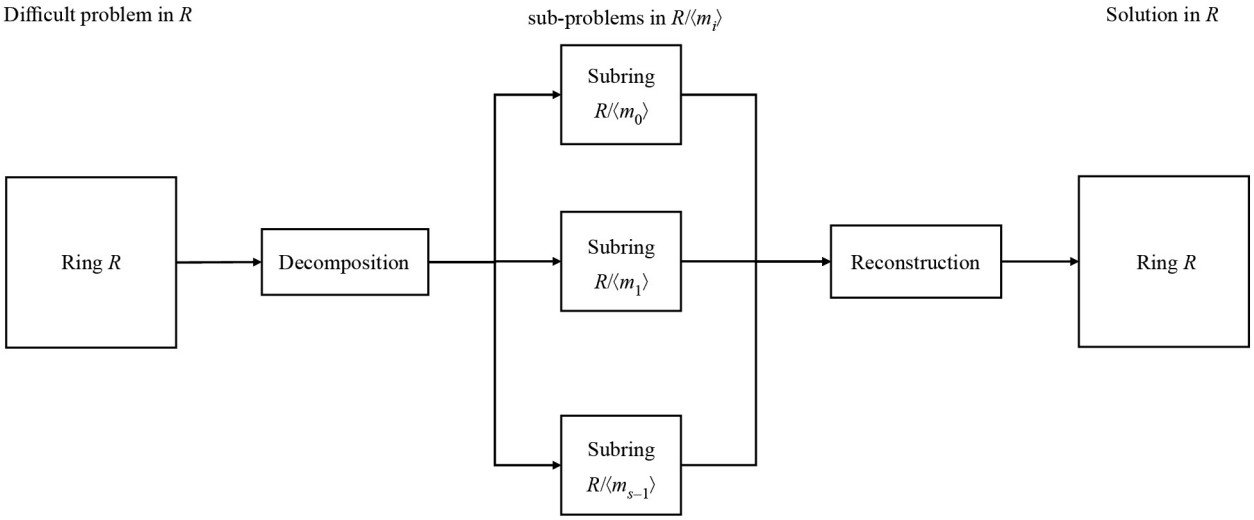

Figure 1. The adopted ring decomposition as a reduction framework to simplify ring multiplication

1.1 *Our contributions*

Precisely, the main contributions and scope of the paper can be summarized as follows:

• We introduce the Generalized Discrete Gaussian Transform (GDGT) framework. This framework offers a reduction technique for efficiently computing the Number Theoretic Transform (NTT) and its variants within power-of-two cyclotomic polynomial rings.

• We demonstrate the applicability of GDGT through a concrete example. Specifically, we show how it can be applied to cyclotomic rings where the modulus n takes the form $n = 3^l \cdot 2^k$, with l and k are positive integers.

• We leverage GDGT to enable NTT-based multiplication for two prominent post-quantum cryptography (PQC) cryptosystems inspired by LAC [29] and NTTRU [31]. This highlights the practical application of GDGT within specific cryptographic contexts.

• We evaluate the performance of our framework by conducting a detailed theoretical complexity analysis focused on ring multiplication. This analysis provides insights into the efficiency gains achieved by employing GDGT.

1.2 *Organization*

The remainder of the paper is organized as follows: In Section 2, we briefly review the state of the art regarding computing the Number Theoretic Transform (NTT) for lattice-based cryptography. Section 3 provides the mathematical background upon which our work is built. Our Generalized Discrete Galois Transform (GDGT) framework is presented in Section 4. Sections 5 and 6 include two GDGT-based ring multiplication algorithms, serving as case studies inspired by the LAC and NTTRU cryptosystems, along with results and discussions. Finally, Section 7 draws conclusions and provides guidelines for potential future work.

2. Literature review

In this section, we review the literature on NTT and its variants which have been used in ring multiplication.

Computing Number theoretic transform in finite fields, was first presented by Pollard [34] who showed how to extend the Discrete Fourier Transform (DFT) to finite fields and applying it for integer convolutions. The NTT methods started to gain more popularity with the rise of digital signal processing. NTT was preferred over DFT due to its numerical stability for computing convolutions of integer signals without rounding errors. In addition, it requires only simple modular integer operations. The main disadvantage of NTT, however, was the strict constraints imposed on the signal length and the

underlying modulus [35]. The literature includes a rich body of work trying to relax these constraints [26, 27, 35–39]. For historical remarks, the reader is referred to [26, 40].

The reduction in the depth of NTT computation data-path was first proposed by Crandall via the DGT algorithm, which can be used to compute the negacyclic integer convolution in power-of-two cyclotomic rings. Given that the input signal (polynomial) has only real values, DGT can be computed via a data-path of length $\frac{n}{2}$ points [27]. Crandall showed how to compute DGT in Gaussian finite fields $GF(q^2)$, where q is a Gaussian prime that is $q \equiv 3 \pmod 4$. In a more recent work, Al Badawi et al. relaxed the condition on the underlying fields and showed how to compute DGT in non-Gaussian finite fields as well [28]. The modified DGT-based multiplication algorithm has been used in GPU implementations of the Fan-Vercauteren (FV) and the Cheon-Kim-Kim-Song (CKKS) Somewhat Homomorphic Encryption schemes [32, 41].

The DGT transform can be viewed as a recursive factoring of the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ into $n/2$ quadratic factors, i.e., factors of the form $x^2 - \omega_j$, where ω_j's are the n-th primitive roots of unity modulo q. The mapping from R_q to the direct product of the quadratic factors can be computed efficiently via a Fast Fourier Transform (FFT)-like data-path of $\log_2 \frac{n}{2} - 1$ stages, each stage comprising $\frac{n}{4}$ butterfly operations (1 multiplication, 1 addition and 1 subtraction) in $GF(q^2)$.

A similar idea has been employed recently by Lyubashevsky and Seiler [31] to implement an NTRU [NTRU stands for the **N**-th degree **T**runcated polynomial **R**ing **U**nits]-based key encapsulation scheme called NTTRU that uses NTT-based ring multiplication. The authors employed a hand-crafted ring that can be split into factors of the form $x^3 - \omega_j$. This non-typical NTT mapping enabled one to use NTT-based multiplication for the NTTRU scheme achieving about one order of magnitude improvement in performance compared to NTRU-HRSS [42]. In addition, Chung et al. [43] proposed non-typical NTT mappings to affect NTT-based ring multiplication for the Saber, NTRU and LAC PQC schemes.

In this work, we follow up on these works and provide a generic reduction framework that can reduce the NTT computation depth by a factor k using an n'-point data-path where $n = kn'$ and $k > 1$. Two relevant use cases are provided to show how the framework can be utilized. This generalization allows for flexibility in choosing k for efficient implementations.

3. Preliminaries

This section includes nomenclature and notations for NTT multiplication, and definitions of the mathematical notions our work builds on.

3.1 *Basic notations*

Conventionally, capital letters denote sets, and lowercase letters represent elements within those sets. The symbol $\mathbb{Z}$ denotes the set of integers. For a positive integer q, we usually represent elements in $\mathbb{Z}_q$ by integers modulo q in the range $\{\lceil -\frac{q}{2} \rceil, ..., \lfloor \frac{q-1}{2} \rfloor\}$ unless stated otherwise. When q is prime, we also use $\mathbb{F}_q$ to denote the Galois field with q elements, and we use $\mathbb{F}_q$ and $\mathbb{Z}_q$ interchangeably. We borrow vector notation to represent polynomials; for instance, polynomial $a(x)$ (sometimes denoted simply as a), with a degree less than n, can be represented as $a = [a_0, a_1, ..., a_{n-1}]$, where a_i is the i-th coefficient of $a(x)$. The notation $[a]_q$, where $a, q \in \mathbb{Z}$, is used to denote the operation of reducing a modulo q in $\mathbb{Z}_q$. If a is a polynomial, the operation is applied to all coefficients. Lastly, matrices are denoted by capital letters.

3.2 *Efficient polynomial multiplication*

Fast polynomial multiplication in cyclotomic rings $R_q = \mathbb{Z}_q[x]/f(x)$ hinges on the idea that R_q can be instantiated with a certain parameter set such that the polynomial modulus $f(x)$ can be factored into smaller co-prime polynomials $f(x) = \prod_{j=1}^{j=l} f_j(x)$ modulo q. The Chinese Remainder Theorem (CRT) establishes an isomorphism: $\mathbb{Z}_q[x]/f(x) \cong \prod_{j=1}^{l} \mathbb{Z}_q[x]/f_j(x)$. Polynomial multiplication can be carried out by first computing the CRT map for the input polynomials, which are assumed to be in $\mathbb{Z}_q[x]/f(x)$, followed by component-wise multiplication of the transformed inputs and finally

computing the inverse CRT map. If the CRT mappings can be computed efficiently, this can provide fast multiplication algorithms.

More concretely, let q be a prime and let n be a positive integer (typically a power of 2). We seek to perform multiplication in R_q efficiently, namely, to compute $a(x)b(x) \mod (x^n+1)$ for any $a(x), b(x) \in R_q$. The typical approach stems from the following notion. Suppose that $2n|(q-1)$, so that there exists a primitive $2n$-th root of unity in the multiplicative cyclic group $\mathbb{Z}_q^\times$, denoted by ω. Then $\omega^{2n} = (\omega^n)^2 = 1$ which yields $\omega^n = \pm 1 \mod q$. Since ω is a primitive $2n$-th root of unity, we must have $\omega^n = -1$. Denote by F the map from R_q to $\mathbb{Z}_q^n$ such that:$F(a) = (a(\omega^0), a(\omega^1), \ldots, a(\omega^{n-1}))$ for all $a \in R_q$. Note that F is invertible since for any $z \in \mathbb{Z}_q^n$, one can find an $a \in R_q$ such that $F(a) = z$ via Lagrange interpolation. The following lemma restates the well-known fact that F is an isomorphism between R_q and $\mathbb{Z}_q^n$.

Lemma 1 For all $a, b \in R_q$, one has: $F^{-1}(F(a) \odot F(b)) = ab$, where $\odot$ denotes the component-wise product in $\mathbb{Z}_q^n$.

The map F is commonly known as the NTT transform. Lemma (1) shows that one can compute the product of any two elements in R_q by performing 2 NTT computations, a product in $\mathbb{Z}_q^n$ and an inverse NTT computation.

3.3 *DWT*

The preceding description of NTT works for any polynomial $f(x) \in R_q$. DWT, also known as negative wrapped convolution, is a popular variant of NTT algorithm for computing the negacyclic convolution in $GF(q)$ where $q \in \mathbb{Z}$ is a prime. Specifically, DWT can be viewed as an efficient variant of NTT when $f(x) = x^n + 1$ over $GF(q)$ when q and n are appropriately chosen as described below.

Let $n \in \mathbb{Z}$ be a power of 2 corresponding to the desired transform length, and ω_n be a primitive n-th root of unity in $GF(q)$, i.e., n is the smallest integer of $e = 1, \ldots, n$ s.t. $\omega^e \equiv 1 \pmod q$. DWT also requires the existence of $\zeta \in GF(q)$ s.t. $\zeta^2 \equiv \omega_n \pmod q$. The algorithm can be adapted to work in any other algebraic structure if it includes an equivalent definition of ω_n, ω_n^{-1}, ζ, ζ^{-1}, and n^{-1}. To guarantee the existence of ω_n and ζ, one may choose q s.t. $q \equiv 1 \pmod{2n}$.

The basic idea behind DWT exploits the following property: for the ring R_q: $\mathbb{Z}_q[x]/(x^n+1)$, substituting x by ζx generates the following isomorphism: $\mathbb{Z}_q[x]/(x^n+1) \to \mathbb{Z}_q[x]/(x^n-1)$ since $\zeta^n = -1$. This is equivalent to computing the negacyclic convolution. The DWT and its inverse (IDWT) for n-point signal are defined below [44]:

$$X_k = \sum_{j=0}^{n-1} \omega_n^{-jk} x_j \bmod q, \;\; x_k = n^{-1} \sum_{j=0}^{n-1} \omega_n^{jk} X_j \bmod q \tag{1}$$

Since we are working in $GF(q)$, the existence of n^{-1}, ω_n^{-1}, and $\zeta^{-1} \pmod q$ are guaranteed.

Remark 1 In the above description, one requires that $2n|(q-1)$. In practical applications of lattice-based cryptography, we typically let n to be a power of 2. This restriction results in only a limited number of possible q.

3.4 *DGT*

DGT [27] is an adapted version of DFT instantiated over the Galois field $GF(q^2)$, where q is a Gaussian prime. Although DGT was shown initially to require $q \equiv 3 \pmod 4$, so-called Gaussian primes, Al Badawi et al. [28] showed how to relax this condition and adapt DGT to work with prime numbers that are $\equiv 1 \pmod 4$, i.e., non-Gaussian primes. In the following paragraphs, we briefly introduce the DGT and refer the reader to the referenced articles for more concrete treatment.

An element $u \in GF(q^2)$ can be represented as $u_r + iu_i$ where $u_r, u_i \in \mathbb{Z}_q$ and $i^2 = -1$ is the imaginary unit. These elements are also known as Gaussian integers. Arithmetic with Gaussian integers is similar to complex number arithmetic with a reduction modulo q for the real and imaginary parts. The DGT and its inverse of signal $x = \{x_0, x_1, \ldots, x_{n-1}\}$ of length n and $x_i \in GF(q^2)$, are shown in equation 2:

$$X_k = \sum_{j=0}^{n-1} h^{-jk} x_j \bmod q,\ x_k = n^{-1} \sum_{j=0}^{n-1} h^{jk} X_j \bmod q \tag{2}$$

where h is a primitive n-th root of unity in $GF(q^2)$.

DGT enjoys several appealing properties, the most notable is that negacyclic convolution of two n-point signals can be computed via $(n/2)$-point DGT/IDGT. This reduces the length of the compute data-path and may lead to improved performance in certain cases [32]. In the case of Gaussian primes, this reduction is particularly helpful as the computations will be performed in the field $GF(q^2)$ instead of the base field $GF(q)$.

3.5 *Computational complexity estimates*

We note that in our theoretical computational complexity analyses of all the algorithms presented in this work, we only consider the number of Elementary Integer Modular Multiplications (EIMM) in a base finite field $GF(q)$, i.e., $a \times b \pmod q$, where a, b, q are all $\in \mathbb{Z}$. Other operations such as modular additions, and subtractions are not included due to their cheap computational overhead compared with modular multiplication.

4. A reduction framework for NTT computation

Let q be a prime, p be a power of q (a prime power), and let n be a power of 2. Let R_q denote the ring $\mathbb{F}_q[x]/(x^n+1)$. Our goal is to compute the product of elements in R_q efficiently via NTT techniques for a wider range of values of q and n that is a power of 2.

We recall some facts from finite field theory.

Lemma 2 There exists an element g of $\mathbb{F}_p^*$ that generates all of $\mathbb{F}_p^*$. Equivalently, g is a primitive $(p-1)$-th root of unity modulo q.

Lemma 3 Suppose that r is a positive integer such that $r|(p-1)$. Then, there exists a primitive r-th root of unity in $\mathbb{F}_p$.

Proof. Let g be a primitive $(p-1)$-th root of unity which exists by Lemma (2). Let $t = (p-1)/r$. Since $r|(p-1)$, t is an integer. Thus, $\omega = g^t$ is a primitive r-th root of unity. □

The ring multiplication can be calculated in at least three different methods as follows:

1. DWT if $2n|(q-1)$.
2. Generalize NTT to field extensions if $2n \not| (q-1)$.
3. Speeding up method 2 via GDGT (2 dimensions).

We describe each method in the following subsections.

4.1 *Using DWT*

Condition required: q is a prime and n a power of 2 such that $2n|(q-1)$. By Lemma (3), there exists an $\zeta \in \mathbb{F}_q$ which is a primitive $2n$-th root of unity modulo q. Further, we have the following:

- $\zeta^n = -1$;
- $\omega_n = \zeta^2$ is a primitive n-th root of unity;
- $i = \zeta^{n/2}$ satisfies $i^2 = -1$. Here, $i \in \mathbb{F}_q$.

Algorithm 1. Polynomial multiplication in R_q via the DWT [26]

Let R_q be the ring $\mathbb{F}_q[x]/(x^n+1)$, q is a prime number and n is a power of 2. Let ω_n be a primitive n-th root of unity in $GF(q)$, and ζ be a square root of ω_n in $GF(q)$. Let $a(x), b(x), c(x) \in R_q$ be polynomials of degrees less than n with integer coefficients $\{\lceil -\frac{q}{2} \rceil, ..., \lfloor \frac{q-1}{2} \rfloor\}$.

Input: $a(x), b(x), \omega_n, \omega_n^{-1}, \zeta, \zeta^{-1}, n, n^{-1}, q$

Output: $c(x) = a(x) \cdot b(x) \pmod{(q, x^n+1)}$

Precompute:

$\omega_n^j, \omega_n^{-j}, \zeta^j, \zeta^{-j} \pmod q$, where $j = 0, \ldots, n-1$

Twist input signals:

$\bar{a}_j = \zeta^j a_j \pmod q$

$\bar{b}_j = \zeta^j b_j \pmod q$

Compute n-DWT:

$A = DWT(\bar{a})$

$B = DWT(\bar{b})$

Coordinate-wise multiplication:

$C_j = A_j.B_j \pmod q$

Compute n-DWT^{-1}:

$\bar{c} = \mathrm{DWT}^{-1}(C)$

Remove twisting factors:

$c_j = \zeta^{-j} \bar{c}_j \pmod q$

return $c(x)$

Here, $DWT(\bar{a}) = \bar{a}W$, where $W_{i,j} = \omega_n^{ij}$, $i, j = 0, 1, \ldots, n-1$. This standard case can be seen as if $q = p^1$.

Computational Complexity: The DWT transform of polynomial $a(x)$ of length n can be calculated in $\mathcal{O}(n \log n)$ EIMM if a FFT-like datapath is used to compute $DWT(\bar{a})$.

4.2 *DWT over finite field extensions*

Next, suppose that $2n \nmid (q-1)$. Let d be an integer such that $2n|(q^d-1)$, and $p = q^d$. We consider the field $\mathbb{F}_p = \mathbb{F}_q[z]/f(z)$, where $f(z)$ is an irreducible polynomial over $\mathbb{F}_q$ of degree d.

We can now apply Algorithm (1), with some modification. In particular, we work over the extension field $\mathbb{F}_p$ instead of over the finite field $\mathbb{F}_q$ as shown in Algorithm (2).

Condition required: q is a prime and n a power of 2, d is an integer such that $2n|(q^d-1)$. By Lemma (3), there exists a $\zeta \in \mathbb{F}_p$ which is a primitive $2n$-th root of unity modulo q. Further, we have the following:

- $\zeta^n = -1$;
- $\omega_n = \zeta^2$ is a primitive n-th root of unity.

From Lemma (3), we further have the following:

If $n|(q-1)$, then $\omega_n \in \mathbb{F}_q$.

Algorithm 2. Polynomial multiplication in R_q via the DWT with extension fields

Let R_q be the ring $\mathbb{F}_q[x]/(x^n+1)$, q is a prime number and n is a power of 2, $p=q^d$ such that $2n|(p-1)$. Let ω_n be a primitive n-th root of unity in $\mathbb{F}_q$, and ζ be a square root of ω_n in $\mathbb{F}_q$. Let $a(x), b(x), c(x) \in R_q$ be polynomials of degrees less than n with integer coefficients $\{\lceil -\frac{q}{2} \rceil, ..., \left\lfloor \frac{q-1}{2} \right\rfloor\}$.

Input: $a(x), b(x), \omega_n, \omega_n^{-1}, \zeta, \zeta^{-1}, n, n^{-1}, q, d, f(z)$

Output: $c(x) = a(x) \cdot b(x) \pmod{(q, x^n+1), f(z)}$

Precompute:

$\omega_n^j, \omega_n^{-j}, \zeta^j, \zeta^{-j} \in \mathbb{F}_p$, where $j = 0, \ldots, n-1$

Twist input signals:

$\bar{a}_j = \zeta^j a_j \pmod{q, f(z)}$

$\bar{b}_j = \zeta^j b_j \pmod{q, f(z)}$

Compute n-DWT:

$A = DWT(\bar{a})$

$B = DWT(\bar{b})$

Coordinate-wise multiplication:

$C_j = A_j.B_j \pmod{q, f(z)}$

Compute n-DWT^{-1}:

$\bar{c} = \mathrm{DWT}^{-1}(C)$

Remove twisting factors:

$c_j = \zeta^{-j} \bar{c}_j \pmod{q, f(z)}$

return $c(x)$

In Algorithm (2), $DWT(\bar{a}) = \bar{a}W$, where $W_{i,j} = \omega_n^{ij}$, $i, j = 0, 1, \ldots, n-1$ and W is a matrix over $\mathbb{F}_q$.

Suppose that q is an odd prime. Since q and n are co-prime, by Euler's theorem, one has $q^{\phi(2n)} = q^n \equiv 1 \pmod{2n}$. It follows that there must exist d such that $2n|(q^d-1)$. Consequently, Algorithm (2) can be applied for any q and n that is a power of 2. However, the extended DWT operations require approximately $n \log n$ products in $\mathbb{F}_p$ or approximately $d^2 n \log n$ products in $\mathbb{F}_q$. If $d > \sqrt{n/\log n}$, this step will require greater than n^2 products in $\mathbb{F}_q$ which is less efficient than performing the product in R_q via schoolbook multiplication.

Remark 2 Suppose that $n|(q-1)$ but $2n \nmid (q-1)$. One can apply Algorithm (2) for a suitable p. In this case, the DWT matrix W will be in $\mathbb{F}_q$ (by Lemma (3)). It follows that the DWT step will require $n \log n$ products of elements in $\mathbb{F}_q$.

4.3 *GDGT*

We demonstrate now how the second method can be sped up under some conditions.

First, we introduce a 2-dimensional ring isomorphic to R_q. Let k be an integer dividing n and write $n = kn'$. Suppose that $2n'|(q-1)$. Consider the ring $\bar{R}_q[x,y] = E[x]/(x^{n'}-y)$, where $E = \mathbb{F}_q[y]/(y^k+1)$. Then, elements in $\bar{R}_q[x,y]$ are of the form $\sum_{i=0}^{n'-1} \sum_{j=0}^{k-1} a_{i,j} x^i y^j$. Clearly, both R_q and $\bar{R}_q[x,y]$ have q^n elements. The next theorem shows that they are in fact isomorphic as rings.

Theorem 4 Define the maps: $\phi: R_q \to \bar{R}_q[x, y]$ with $\phi(a(x)) = a(x) \pmod{x^{n'} - y, y^k + 1}$ and $\psi: \bar{R}_q[x, y] \to R_q$ with $\psi(a(x, y)) = a(x, x^{n'})$. The following hold:

• Both ϕ and ψ are well defined.

• ϕ is a homomorphism.

• ϕ and ψ are inverses of each other. In particular, ϕ is an isomorphism between the rings $\bar{R}_q[x, y]$ and R_q.

Proof.

• ϕ is well defined since $x^n + 1 \pmod{x^{n'} - y, y^k + 1} = 0$. Hence, any two elements in the same class of R_p will map to the same element in $\bar{R}_q[x, y]$ under ϕ. On the other hand, for any $a(x, y) = \sum_{i=0}^{n'-1} \sum_{j=0}^{k-1} a_{i,j} x^i y^j \in \bar{R}_q[x, y]$, $a(x, x^{n'}) = \sum_{i=0}^{n'-1} \sum_{j=0}^{k-1} a_{i,j} x^{i+n'j}$ has degree at most $n-1$ and is thus unique.

• This follows since ϕ is well defined and modulo operation respects both addition and multiplication of polynomials.

• Let $a(x) = \sum_{i=0}^{n-1} a_i x^i$. For $0 \le i \le n-1$, write $i = i_1 + i_2 n'$ with $0 \le i_1 \le n'-1$, $0 \le i_2 \le k-1$. Then,

$$a(x) = \sum_{i=0}^{n-1} a_i x^i$$

$$= \sum_{i_1=0}^{n'-1} \sum_{i_2=0}^{k-1} a_i x^{i_1 (x^{n'})^{i_2}} \tag{3}$$

$$\text{Thus, } \phi(a(x)) = a(x) \pmod{x^{n'} - y, y^k + 1}$$

$$= \sum_{i_1=0}^{n'-1} \sum_{i_2=0}^{k-1} a_i x^{i_1} y^{i_2} \tag{4}$$

It follows that:

$$\psi(\phi(a(x))) = \psi\left(\sum_{i_1=0}^{n'-1} \sum_{i_2=0}^{k-1} a_i x^{i_1} y^{i_2}\right)$$

$$= \sum_{i_1=0}^{n'-1} \sum_{i_2=0}^{k-1} a_i x^{i_1} (x^{n'})^{i_2}$$

$$= \sum_{i=0}^{n-1} a_i x^i = a(x) \tag{5}$$

□

An immediate consequence of Theorem (4) is that for any $a(x), b(x) \in R_q$, it follows that: $a(x)b(x) = \psi(\phi(a(x))\phi(b(x)))$. In other words, instead of directly computing the product of two elements in R_q, one can compute the product of the two corresponding elements in $\bar{R}_q[x, y]$ and find the result via an application of ψ.

Algorithm 3. Polynomial multiplication in R_q via the GDGT

Let R_q be the ring $\mathbb{F}_q[x]/(x^n+1)$, q is a prime number and n is a power of 2. Let d be the smallest integer such that $2n|(q^d-1)$ and let $p=q^d$. Let $f(z)$ be an irreducible polynomial of degree d over $\mathbb{R}_q$. Let $E=\mathbb{F}_q[y]/(y^k+1)$. Let k be the shrinking factor with $1<k<n$ so that $k|(q-1)$ and $n'|(q-1)$. Let $g=\omega_{n'}$ be a primitive n'-th root of unity in E, and $h=(\omega, \omega^3, \ldots, \omega^{2k-1})$ be an n'-th root of y in E. Let $a(x), b(x), c(x) \in R_q$ be polynomials of degrees less than n with integer coefficients $\{\lceil -\frac{q}{2} \rceil, ..., \lfloor \frac{q-1}{2} \rfloor\}$.

Input: $a(x), b(x), g, g^{-1}, h, h^{-1}, n, n^{-1}, p, k, d, q, f(z)$

Output: $c(x)=a(x).b(x) \pmod{(q, x^n+1)}$

Precompute:

$g^j, g^{-j}, h^j, h^{-j} \pmod{q, f(z), y^k+1}$, where $j=0, \ldots, \frac{n}{k}-1$

Initialize:

fold over input signals:

$a'_j = a_j + ya_{j+\frac{n}{k}} + \ldots + y^{k-1}a_{j+\frac{(k-1)n}{k}}$

$b'_j = b_j + yb_{j+\frac{n}{k}} + \ldots + y^{k-1}b_{j+\frac{(k-1)n}{k}}$

Twist the folded signals:

$\widehat{\bar{a'}_j} = DWT(a'_j)$ via Algorithm (2) with n replaced by k).

$\widehat{\bar{b'}_j} = DWT(b'_j)$ via Algorithm (2) with n replaced by k).

$\bar{a'}_j = \widehat{\bar{a'_j}} h^j$

$\bar{b'}_j = \widehat{\bar{b'_j}} h^j$

Compute k parallel NTT: At this point, we have k polynomials $\bar{a'}_l \in \mathbb{F}_p[x]/(x^{n'}-1)$ for $l=0, 1, \ldots, k-1$ and similarly for $\bar{b'}_l$.

$A_l = NTT(\bar{a'}_l)$

$B_l = NTT(\bar{b'}_l)$

Point-wise multiplication:

$C_j = A_j \cdot B_j \pmod{q, f(z)}$. Here, multiplication is done component-wise

Compute k NTT^{-1}:

$\bar{c'} = \mathrm{NTT}^{-1}(C)$

Remove twisting factors:

$c'_j = \bar{c'}_j h^{-j} \pmod{q, f(z)}$. At this point, the c_j's are k-vectors. Convert them back into the polynomial form via inverse DWT

Unfold output signal:

$c_{j+\frac{in}{k}} =$ coefficient of y^i in $c'_j)$

return $c(x)$

Observe that we have reduced the length of the polynomial/vector from n to n' but our operations are now performed in $E = \mathbb{F}_q[y]/(y^k + 1)$ instead of over $\mathbb{R}_q$. For efficient computations, we impose further conditions.

Let k be a power of 2 satisfying:

• $1 < k < n$;

• $k|(q-1)$ so there exists a primitive k-th root of unity in $\mathbb{F}_q$;

Write $n = kn'$. We further require $n'|(q-1)$. It follows that there exists a primitive n'-th root of unity in $\mathbb{F}_q$. Let ζ denote a $2n$-th primitive root of unity. Here, $\zeta \in \mathbb{F}_p$. Let $\omega_n = \zeta^2$, $\omega_k = \zeta^{2n'}$ and $\omega_{n'} = \zeta^{2k}$ be the primitive n, k and n' primitive root of unity. Let $\zeta_k = \zeta^{n'}$ and $\zeta_{n'} = \zeta^k$. Then, $\zeta_k^2 = \omega_k$ and $\zeta_{n'}^2 = \omega_{n'}$. By our assumption, both ω_k and $\omega_{n'}$ are in the base field $\mathbb{F}_q$. By applying Algorithm (2), one can verify that the element $y \in E = \mathbb{F}_q[y]/(y^k + 1)$ can be converted via DWT to the vector $v = (\zeta_k, \zeta_k^3, \ldots, \zeta_k^{2k-1}) = (\zeta^{n'}, \zeta^{3n'}, \ldots, \zeta^{(2k-1)n'}) \in \mathbb{F}_q^k$. We want an h such that $h^{n'} = y = v$. Thus, let $h = (\zeta, \zeta^3, \ldots, \zeta^{2k-1}) \in \mathbb{F}_p$. We may now adapt Algorithm (2) as follows.

We remark that two different matrices are used in Algorithm (3) for the DWT and NTT operations, using the primitive roots ω_k and $\omega_{n'}$, respectively. By our choice of k and n', both these matrices are over $\mathbb{F}_q$. As discussed in Remark (2), converting the $\bar{a_j}$ and $\bar{b_j}$ involve around $k \log k$ products of elements in $\mathbb{F}_q$ and $\mathbb{F}_p$, or around $dk \log k$ products in $\mathbb{F}_q$. Similarly, the main DGT step requires $n' \log n'$ products in $\mathbb{F}_q^k$ and $\mathbb{F}_p$, resulting in around $dkn' \log n' = dn \log n'$ products in $\mathbb{F}_q$. Observe that these operations can be parallelized for great speed-ups.

Remark 3 In the case when n' is not a factor of $(q-1)$, the algorithm can be adapted in a straightforward manner where the computations will be over an extension field of $GF(p)$. In fact, when $k = 2$, this algorithm effectively reduces to the DGT algorithm. We have thus generalized the DGT framework that allows one to vary k and valid for more values of q.

Remark 4 In the case when $2k|(q-1)$, the first set of DGT operations only involve operations in $\mathbb{F}_q$, that is, reduces to DWT operations. It follows that it is generally more efficient to choose k to be as large as possible.

We end this section by summarizing the possible conditions on q and n when n is a power of two and the corresponding algorithm to be applied.

4.4 *Concrete lower bounds on q*

In the design of lattice-based cryptosystems, one may fix n and search for appropriate values of q that balances security, efficiency and accuracy of the scheme. As such, efficient DGDT computations inevitably allows for more values of q to be considered. In this subsection, we provide some lower bounds on the underlying modulus q that can be used with each algorithm for various values of n. These lower bounds are the smallest values of q satisfying the conditions specified in Table (1). In particular, for the GDGT algorithm, the lower bound on q is the smallest q satisfying either of the 3 conditions given.

Table 1. Recommended NTT algorithm to use for different parameters n and q

Condition on q and n	k	d	Algorithm
$2n\|(q-1)$	1 or $2n$	1	DWT
$q \equiv 3 \mod 4$ & $q \equiv -1 \mod n$	2	2	DGT
$q \equiv 3 \mod 4$ & $q \equiv -1 \mod \frac{n}{2}$	2	4	GDGT
$q \equiv 1 \mod n$	n	2	GDGT
$q \equiv 1 \mod \frac{n}{2}$	$\frac{n}{2}$	4	GDGT

Table 2 gives the smallest values of q for a given $n \in \{2^u : 8 \leq u \leq 12\}$ for the 3 algorithms of DWT, DGT and GDGT. The values of n are inspired by common parameters used in lattice-based cryptography. Clearly, GDGT can allow

for smaller moduli compared with DWT and DGT for the same ring dimension n. The parameter set in the second row will be used in the subsequent section to instantiate a ring that is inspired by the LAC cryptoscheme.

Table 2. Smallest values of q for a given n for DWT, DGT and GDGT

n	DWT	DGT	GDGT
256	7681	1279	127
512	12289	3583	257
1024	12289	5119	3583
2048	12289	6143	5119
4096	40961	8191	6143

5. Case study I: LAC

In this section, we show how to utilize the GDGT framework to compute polynomial multiplication in a ring inspired by the LAC NIST-PQC Round 2 candidate scheme [29]. We target the LAC-128 parameter set $n = 512$ with a slight modification to the modulus from $q = 251$ proposed in the original scheme and use instead $q = 257$. Our main purpose of introducing this case study is to illustrate how one can use the GDGT framework in existing schemes where parameters have been optimized. In addition, complete security and theoretical computational analyses are provided for completeness.

5.1 *LAC*

The LAC encryption scheme is a lattice-based cryptosystem that is constructed based on the Ring-LWE problem. LAC is the winner of the Chinese Association for Cryptologic Research competition award. It was also submitted to the NIST PQC competition and advanced as Round 2 candidate. LAC uses a small modulus ($q = 251$) over rings of dimensions $n = 512$ and 1024 [LAC-128 uses the smaller ring, whereas LAC-192 and LAC-256 use the larger ring.].

More formally, LAC is constructed over the ring $R_q = Z_q/(x^n + 1)$ with $n = 512$ for LAC-128 or $n = 1024$ for LAC-192 and LAC-256 and $q = 251$ for all parameter sets. While the small modulus in LAC provides several advantages such as low keys and ciphertext sizes and computing arithmetic with byte-level modulus, it suffers from two main challenges: 1) the small modulus used increases the decryption error rates and 2) the difficulty of utilizing NTT to compute the ring multiplication. The first challenge can be handled by using Error Correction Code (ECC) techniques, while the latter is more involved. In the subsequent section, we present a way to tackle the second challenge and provide an NTT-based method to compute LAC's ring multiplication via the GDGT framework.

5.2 *GDGT-based ring multiplication*

We need to compute the product of two elements $a(x), b(x) \in \mathbb{R}_q[x]/(x^n + 1)$ for $(n, q) = (512, 257)$. Here, $q - 1 = 256 = 2^8$ and $2n = 1024$. Thus, we must have $d \geq 4$. It follows that one can choose $k = 2^u$ and $n' = 2^{9-u}$ for $1 \leq u \leq 8$. Since $d = 4$, it can be checked that the optimal choices of k and n are $k = 2^7 = 128$ and $n' = 4$ (so as to reduce the number of operations involving $\mathbb{F}_{q^4}$. We now simplify the GDGT algorithm to this specific case as follows.

Note that we have $3^{128} \equiv -1 \mod q$ and $3^{256} \equiv 1 \mod q$. Further, we have

$$\mathbb{R}_q[x]/(x^n+1) = \mathbb{R}_q[x]/((x^4)^{n/4}+1)$$

$$\cong \prod_{i=0}^{n/4-1} \mathbb{R}_q[x]/(x^4 - 3^{2i+1}) = E \tag{6}$$

In particular, E is a product of $n/4$ copies of $\mathbb{F}_{q^4}$, where we take the evaluations of x^4 at 3^{2i+1}, $\forall i = 0, 1, \ldots, n/4-1$.

Thus, for every element $a(x) = \sum_{i=0}^{n-1} a_i x^i$, we can group into four parts, according to the remainders when the powers are divided by 4. This gives $a(x) = a^{(0)}(x) + xa^{(1)}(x) + x^2 a^{(2)}(x) + x^3 a^{(3)}(x)$. We can now get the image of this element in the product of fields E by performing 4 DWTs independently. Given two such vectors, one can compute the component-wise product and perform the inverse DWT to obtain the desired product. However, note that the component-wise products are taken over modulo $x^4 - 3^{2i+1}$, that is, with respect to different polynomial moduli for each entry. This may not be desirable for practical computations. Here, we observe that for all even i, say $i = 2j$, the modulus is of the form $x^4 - 3^{4j+1}$. Letting $x = 3^j x$, one gets $\mathbb{F}_p[x]/(x^4 - 3^{4j+1}) \cong \mathbb{F}_p[3z]/(3^{4j}x^4 - 3^{4j+1}) \cong \mathbb{F}_p[z]/(z^4 - 3)$. In other words, by considering appropriately weighted polynomials, one can transform the polynomials where the product is with respect to the modulus $x^4 - 3$ for all the even entries. Similarly, all the moduli in the odd entries can be transformed to $x^4 - 27$.

Concretely, we perform the following steps to compute $c(x) = a(x)b(x) \in \mathbb{R}_q[x]/(x^n+1)$.

Write $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$. Let $k = n/4 = 128$.

• For $i = 0, 1, 2, 3$, let

$$a^{(i)} = \sum_{j=0}^{k-1} a_{4j+i} y^j, \text{ and } b^{(i)} = \sum_{j=0}^{k-1} b_{4j+i} y^j \tag{7}$$

• Compute k-point DWT$(a^{(i)})$ and DWT$(b^{(i)})$ for $i = 0, 1, 2, 3$. Denote the resulting vectors by $a'^{(i)}$ and $b'^{(i)}$, respectively.

• Let

$$a'[0] = (a'_0, a'_2, \ldots, a'_{k-2}), \text{ and } a'[1] = (a'_1, a'_3, \ldots, a'_{k-1}),$$

$$b'[0] = (b'_0, b'_2, \ldots, b'_{k-2}), \text{ and } b'[1] = (b'_1, b'_3, \ldots, b'_{k-1}) \tag{8}$$

In other words, $a'[0]$ and $b'[0]$ are the even entries of a' and b', respectively, while $a'[1]$ and $b'[1]$ are the odd entries.

• For $i = 0, 1, \ldots, k/2-1$, replace

$a'[0]_i$ by $a'[0]_{i0} + 3^i a'[0]_{i1} + 3^{2i} a'[0]_{i2} + 3^{3i} a'[0]_{i3}$,

$a'[1]_i$ by $a'[1]_{i0} + 3^i a'[1]_{i1} + 3^{2i} a'[1]_{i2} + 3^{3i} a'[1]_{i3}$,

and do the same for $b'[0]_i$ and $b'[1]_i$.

• Let $c'[0] = a'[0] \odot b'[0]$, where the product is performed component-wise modulo $x^4 - 3$. Let $c'[1] = a'[1] \odot b'[1]$, where the product is performed component-wise modulo $x^4 - 27$.

• Replace $c'[0]_i$ by $c'[0]_{i0} + 86^i c'[0]_{i1} + 86^{2i} c'[0]_{i2} + 86^{3i} c'[0]_{i3}$. $c'[1]_i$ by $c'[1]_{i0} + 86^i c'[1]_{i1} + 86^{2i} c'[1]_{i2} + 86^{3i} c'[1]_{i3}$.

• Let c' be the vector whose even entries are $c'[0]$ and odd entries are $c'[1]$.

• Perform the inverse DWT with respect to each power of x on c' and recombine to obtain $c(x)$.

Next, we suggest a method to compute the products in $F = \mathbb{F}_p[x]/(x^4 - a)$, where p is small. Let g be a generator of F^*, that is, the powers of g generate all of F^*. Let $h = g^{p^2+1}$. Then, h is a generator of $GF(p^2)$. Compute a look-up table for the discrete log of $c_0 + c_1 x^2$ with respect to h, namely, find $i \in \{0, 1, \ldots, p^2-2\}$ such that $c_0 + c_1 x^2 = h^{L(c_0, c_1)}$.

To compute $a_0+a_1x+a_2x^2+a_3x^3)(b_0+b_1x+b_2x^2+b_3x^3)$, we do as follows:

• Write $a=a_0+a_2x^2+x(a_1+a_3x^2)$ and similarly for b.

• From the look-up table, find $d_0=L(0,1)$, $d_1=L(a_0,a_2)$, $d_2=L(b_0,b_2)$, $d_3=L(a_1,a_3)$, $d_4=L(b_1,b_3)$.

• Compute $e_1=d_1+d_2 \mod p^2-1$, $e_2=d_0+d_3+d_4 \mod p^2-1$, $e_3=d_1+d_4 \mod p^2-1$, $e_4=d_2+d_3 \mod p^2-1$.

• From the look-up table, find the inverse of e_i, $i=1,2,3,4$ and denote it by e_i'.

• Then, $ab=e_1'+e_2'+x(e_3'+e_4')$.

5.3 *Computational complexity*

Analyzing the GDGT-based multiplication algorithm for LAC shows that the number of EIMM required to compute the product of two input polynomial is: $3\cdot(4\cdot 128\cdot\log 128+4\dot{3}\cdot 128)+16\cdot 128$. The first term corresponds to computing DWT of 4 signals, each of length $k=128$. The second term corresponds to the transformation of the DWT results to uniform the polynomial modulus of the extension field. This operation has to be done 3 times, 2 for computing GDGT and 1 for GDGT inverse. The third term corresponds to the point-wise modulo x^4-a. The total number of EIMM without any optimizations such as parallel execution nor using the lookup tables requires 29,312 EIMM, 8.94× faster than the classic schoolbook multiplication that requires $512^2=262,144$ EIMM. Compared with more efficient multiplication methods such as Karatsuba [45] (which runs in $\mathcal{O}(n^{log_2 3})=19{,}683$) and 3-way Toom-Cook [46] (which runs in $\mathcal{O}(n^{\frac{log5}{log3}})=9{,}026$), our GDGT method is slower by 1.49× and 3.25×, respectively. Despite the slowness shown in our algorithm compared with Karatsuba and Toom-Cook, it is clear that our GDGT method is embarrassingly parallel and can benefit from parallel implementations. We believe that it would be very interesting to characterize the performance of our GDGT algorithm experimentally via an optimized parallel implementation.

5.4 *Security level*

To estimate the security of the LAC scheme under the new parameters proposed here, we follow the security analysis in [47]. Using the LWE hardness estimator [48], our chosen parameters do not noticeably affect the security level of LAC as shown in Table 3.

Table 3. Estimated security levels of the vanilla LAC parameters and our proposed parameters for LAC using GDGT

Scheme	Parameters: (n, q)	Security Level
LAC (vanilla)	(512, 251)	134.9 bit
LAC (GDGT)	(512, 257)	134.4 bit

6. Case study II: NTTRU

In this section, we show how to utilize the GDGT framework to propose an efficient compact ring multiplication for an instance that is inspired by the NTTRU cryptosystem [31].

6.1 *NTTRU*

The NTRU-based cryptoscheme we are interested in here is the NTTRU scheme proposed by Lyubashevsky and Seiler in [31]. Following Albrecht et al.'s security analysis [47] and to enable a variant NTT ring multiplication, the authors constructed NTTRU over the cyclotomic ring $\mathbb{Z}_q[X]/(X^n-X^{n/2}+1)$, where $(n,q)=(768,7681)$. It should be noted that, to the best of our knowledge, NTTRU provides the first NTT-based ring multiplication of an NTRU-based scheme. In this case, $n=768$ is not a power of 2 but a small multiple of it. We follow on NTTRU footsteps and provide

an NTT-based ring multiplication algorithm with a more compact parameter set and a more efficient implementation. The purpose of this section is twofold: 1) we demonstrate with a concrete example how our GDGT framework can be extended to work in non-power of two rings, and 2) characterize the performance of a GDGT-based method experimentally to show how it can benefit from parallel implementations. Before presenting our proposed algorithm, we briefly describe the basic idea of NTTRU's NTT-based ring multiplication.

NTTRU is built based on the idea that $\mathbb{Z}_{7681}[X]/(X^{768}-X^{384}+1)$ can be factored into $(X^{384}+684)(X^{384}-685)$. Each of these factors can be factored into two factors of the form (X^i+r') and (X^i-r'). This factoring ends at $(X^3+r)(X^3-r)$. Point-wise multiplication is computed modulo (X^3-r). The process is reversed (via NTT inverse) to get the result back in coefficient representation.

6.2 *GDGT-based ring multiplication*

In this section, we present our GDGT-based ring multiplication algorithm alongside with a hand-tailored parameter set. Specifically, we tweak our general GDGT framework to handle small multiples of powers of 2. While we present a concrete example here, this approach can be applied to other sets of parameters as well.

Recall that in [31], a concrete instantiation was demonstrated on NTTRU for n of the form $3\cdot 2^k$ where the associated polynomial modulus takes the form $f(x)=x^n-x^{\frac{n}{2}}+1$. In fact, $f(x)$ is the cyclotomic polynomial modulo m, where $m=3n=9\cdot 2^k$. In this example, q was chosen so that $n|(q-1)$ and the polynomials are recursively factored into products of cubic polynomials. Essentially, this approach can be viewed as a generalization of DWT computations from n being a power of two to small multiples of powers of two.

By exploiting the power of DGDT computations, we show that the condition imposed on q can be relaxed even when n is of the form $3^l\cdot 2^k$ for some positive integer k, namely, we consider values of q for which n is not a factor of $q-1$. Specifically, we demonstrate the computations with a particular example on n and q satisfying the following conditions:

1. $n=3^l\cdot 2^k$
2. $q\equiv 1 \mod 2\cdot 3^{l+1}$
3. $q\equiv -1 \mod 2^{k-1}$

Here, observe that $3n|(q^2-1)$ but $n \not| (q-1)$. Table 4 gives some possible values of q and n when $l=1$.

Table 4. Possible values of q for a given n for GDGT with $n=3\cdot 2^k$

n	q
384	{127, 1279, 3583, 4159, 7039, 8191, 9343}
768	{127, 1279, 3583, 7039, 8191, 9343}
1536	{1279, 3583, 8191}
3072	{3583, 8191}

In our instantiation for the polynomial multiplication in NTTRU ring, we use the same $n=768$ ($l=1$) and the lowest possible $q=127$ for the sake of illustration. Our parameters should not be used in implementing NTTRU without careful analysis and treatment of the decryption error.

Let $\mathbb{F}=GF(q)$ where $q=127$ and $R=\mathbb{F}[x]/(x^{768}-x^{384}+1)$. We seek to compute the product of two elements $a(x), b(x)\in R$. As in all NTT-like computations, we will compute the vectors $\bar{a}$ and $\bar{b}$, do component-wise multiplication and compute the inverse transform.

We have $q-1=126=2*3^2*7$ and $q^2-1=126*128=2^8*3^2*7$. In particular, $q\equiv 3 \mod 4$. One may be able to perform the usual DGT with $n=128$. Our aim is to perform DGT for $n=768$ which is 6 times bigger.

Note that y^2+1 is irreducible over $\mathbb{F}$. Let i denote a root of y^2+1 so that $E=\mathbb{F}[i]=\{a+bi\colon a\in\mathbb{F}, b\in\mathbb{F}\}$.

We first present some elements in E which are helpful.

• $g = 3$ is a primitive 126-th root of unity modulo q.
• $\zeta = g^7 = 28$ satisfies $\zeta^6 - \zeta^3 + 1 = 0$.
• The six roots that satisfy $x^6 - x^3 + 1$ are $\zeta = 28$, $\zeta^5 = 90$, $\zeta^7 = 75$, $\zeta^{11} = 105$, $\zeta^{13} = 24$ and $\zeta^{17} = 59$.
• $u = g^{35} = 90$ satisfies $u^2 = -28$.
• $z = 59 + i70$ satisfies $z^{64} = i$ and $z^{128} = -1$.
• $h = z^4 = 78 + i34$ satisfies $h^{64} = 1$.

The main steps of the algorithm are:

1) Observe that $R = \mathbb{F}[x]/(x^{768} - x^{384} + 1) \cong \mathbb{F}[x]/(x^{128} - \zeta) \times \mathbb{F}[x]/(x^{128} - \zeta^5) \times \mathbb{F}[x]/(x^{128} - \zeta^7) \times \mathbb{F}[x]/(x^{128} - \zeta^{11}) \times \mathbb{F}[x]/(x^{128} - \zeta^{13}) \times \mathbb{F}[x]/(x^{128} - \zeta^{17})$. We convert a signal of length 768 into six signals, each of length 128. This is done via small NTT computations of length 6.

2) Next, we multiply the signals by appropriate weights to transform the signals to work in the field $\mathbb{F}[x]/(x^{128}+1) \times \ldots \times \mathbb{F}[x]/(x^{128}+1)$.

3) We perform the DGT on the six signals. Note that this can be done in parallel.

4) Merge the six signals to obtain a signal of length 384 in E.

Recall that the first step in DGT computations is to multiply the signal by some weights. In the following, we will merge Step 2 and this step.

Let $a(x) = a_0 + a_1 x + \ldots + a_{127} x^{127}$. Let $\vec{a} = (a_0, \ldots, a_{127})$ represent its coefficients. We now present the concrete steps to perform GDGT on $a(x)$.

1.1 Let A be a 6×128 matrix such that the j-th row of A, for $j = 0, 1, 2, 3, 4, 5$, comprises the elements:

$$A_j = (a_{128*j+0}, a_{128*j+1}, a_{128*j+2}, \ldots, a_{128*j+127}) \tag{9}$$

1.2 Define the 6×6 matrix

$$Z = \begin{pmatrix} 1 & \zeta & \ldots & \zeta^5 \\ 1 & \zeta^5 & \ldots & (\zeta^5)^5 \\ 1 & \zeta^7 & \ldots & (\zeta^7)^5 \\ 1 & \zeta^{11} & \ldots & (\zeta^{11})^5 \\ 1 & \zeta^{13} & \ldots & (\zeta^{13})^5 \\ 1 & \zeta^{17} & \ldots & (\zeta^{17})^5 \end{pmatrix} \tag{10}$$

1.3 Compute the matrix $\bar{A} = Z \cdot A$.

2.1 Define $u_0 = u$, $u_1 = u\zeta^2 = 75$, $u_2 = u\zeta^3 = 68$, $u_3 = u\zeta^5 = 99$, $u_4 = u\zeta^6 = 105$ and $u_5 = u\zeta^8 = 24$.

2.2 Let $\bar{A}_{\text{left}}$ and $\bar{A}_{\text{right}}$ denote the left 64 and the right 64 columns of $\bar{A}$, respectively. Let $D = Diag(u_0,\ u_1,\ -u_2,\ -u_3,\ u_4,\ u_5)$. Compute $\bar{A}' = \bar{A}_{\text{left}} - iD\bar{A}_{\text{right}}$. where $i = \sqrt{-1}$.

2.3 For $s = 0, 1, \ldots, 5$ and $j = 0, 1, \ldots, 63$, let $\bar{A}''_{sj} = (u_s z)^j \bar{A}'_{sj}$.

3.1 Let W' be the 64×64 matrix with $W'_{sj} = h^{sj}$ for $s,\ j = 0, 1, \ldots, 63$.

3.2 Define $\bar{A}''' = \bar{A}'' W'$.

4 Let $\bar{a}'$ be a length 384 vector with $\bar{a}'_{6j+s} = \bar{A}'''_{sj}$. Here, we note that each element in $\bar{a}$ is of the form $a + bi$.

Given two polynomials $a(x)$ and $b(x)$, we perform the above on each of $a(x)$ and $b(x)$ to obtain $\bar{a}'$ and $\bar{b}'$, respectively. Define $\bar{c}' = \bar{a}' \odot \bar{b}'$, where $\odot$ denotes component-wise product in the field E, namely, modulo $y^2 + 1$. Finally, we perform an inverse of the above steps to obtain $c(x)$. The complete procedure can be found in Algorithm 4.

Algorithm 4. Polynomial multiplication for NTTRU via the GDGT

Input: $a(x), b(x)$

Output: $c(x) = a(x).b(x) \pmod{(127, (x^{768} - x^{384} + 1))}$

Precompute:

$g = 3$, $\zeta = 28$, $\zeta^5 = 90$, $\zeta^7 = 75$, $\zeta^{11} = 105$, $\zeta^{13} = 24$, $\zeta^{17} = 59$, $u = g^{35} = 90$, $z = 59 + i70$, $h = z^4 = 78 + i34$, z^j, z^{-j}, h^j, $h^{-j} \pmod{q}$, where $j = 0, \ldots, 63$.

Define $u_0 = u$, $u_1 = u\zeta^2 = 75$, $u_2 = u\zeta^3 = 68$, $u_3 = u\zeta^5 = 99$, $u_4 = u\zeta^6 = 105$ and $u_5 = u\zeta^8 = 24$.

Let $D = Diag(u_0, u_1, -u_2, -u_3, u_4, u_5)$.

Let W' be the 64×64 matrix with $W'_{sj} = h^{sj}$ for $s, j = 0, 1, \ldots, 63$.

Define the 6×6 matrix Z as in Equation (10)

Step 1:

Let A and B be 6×128 matrices such that the j-th rows of A and B, for $j = 0, 1, 2, 3, 4, 5$, comprise the elements:

$A_j = (a_{128*j+1}, a_{128*j+2}, a_{128*j+3}, \ldots, a_{128*j+128})$

$B_j = (b_{128*j+1}, b_{128*j+2}, b_{128*j+3}, \ldots, b_{128*j+128})$

Step 2:

$\bar{A} = Z \cdot A$

$\bar{B} = Z \cdot B$

Step 3:

$\bar{A}' = \bar{A}_{\text{left}} - iD\bar{A}_{\text{right}}$

$\bar{B}' = \bar{B}_{\text{left}} - iD\bar{B}_{\text{right}}$

Step 4:

For $s = 0, 1, \ldots, 5$ and $j = 0, 1, \ldots, 63$,

$\bar{A}''_{sj} = (u_s z)^j \bar{A}'_{sj}$

$\bar{B}''_{sj} = (u_s z)^j \bar{B}'_{sj}$

Step 5:

$\bar{A}''' = \bar{A}'' W'$

$\bar{B}''' = \bar{B}'' W'$

Step 6:

$\bar{C}''' = \bar{A}''' \odot \bar{B}'''$, where $\odot$ is point-wise multiplication in the field E

Step 7:

$\bar{C}'' = \bar{C}''' Winv'$

Step 8:

For $s = 0, 1, \ldots, 5$ and $j = 0, 1, \ldots, 63$,

$\bar{C}'_{sj} = (u_s z)^{-j} \bar{C}''_{sj}$

Step 9:

For $s = 0, 1, \ldots, 5$ and $j = 0, 1, \ldots, 63$,

$\bar{C}'_{\text{s, left}} = Re(\bar{C}_s{}')$

$\bar{C}'_{\text{s, right}} = -Im(\bar{C}_s{}')$

Step 10:

$\bar{C}'_{\text{right}} = Dinv \cdot \bar{C}'_{\text{right}}$

Step 11:

$C = Zinv \cdot Concat(\bar{C}'_{\text{left}}, \bar{C}'_{\text{right}})$

Step 12:

Unfold C. for $j = 0, 1, 2, 3, 4, 5$:

$c_{128*j+1} = C_j$

return $c(x)$

6.3 *Computational complexity*

Analyzing the GDGT-based multiplication algorithm for NTTRU shows that the number of EIMM required to compute the product of two input polynomial is:

$$3 \cdot (6 \cdot 6 \cdot 128 + 1 \cdot 6 \cdot 64 + 6 \cdot 6 \cdot 64 + 4 \cdot 6 \cdot 64 \cdot \log 64) + 6 \cdot 64 \cdot 4 \tag{11}$$

The first term corresponds to computing Step 2 in Algorithm 4. The second anf third terms represent the computational overhead of Step 3 and 4, respectively. The fourth term corresponds to computing DGT for 6 signals (Step 5). The scale by 3 is for computing 2 GDGT and 2 GDGT inverse. The last term corresponds to the point-wise multiplication of the transformed signals. The total number of EIMM without any optimizations such as parallel execution nor using the lookup tables requires 51,072 EIMM, 11.55× faster than the classic schoolbook multiplication that requires 768^2 = 589,824 EIMM. Karatsuba would require $768^{1.585}$ = 37,428 and 3-way Toom-Kook would require $768^{1.46}$ = 16,317. While our GDGT algorithm is slower than both algorithms by 1.36× and 3.13×, respectively, it can benefit largely from a parallel implementation.

6.4 *Security level*

To estimate the security of the NTTRU scheme under the new parameters proposed here, we follow the security analysis in [47]. Using the LWE hardness estimator [48] (commit #fb7deba), our chosen parameters almost doubles the security level of NTTRU as shown in Table 5.

Table 5. Estimated security levels of the vanilla NTTRU parameters and our proposed parameters for NTTRU using GDGT

Scheme	Parameters: (n, q)	Security Level
NTTRU (vanilla)	(768, 7681)	123.5 bit
NTTRU (GDGT)	(768, 127)	243.5 bit

6.5 *Decryption error*

We discuss here the decryption error that might arise from using small moduli as presented in the preceding sections. Lattice-based encryption schemes inherently introduce decryption error that can affect restoring back valid ciphertexts. Luckily, this decryption error can be controlled by an appropriate choice of the cryptosystem parameters. One important parameter that affects the decryption error is the size of the coefficient modulus q. While one desires to minimize q to attain small encryption keys and ciphertext sizes, q cannot be chosen arbitrarily small to maintain the security level and reduce the probability of decryption error. Analyzing the decryption error of the proposed schemes is beyond the scope of this paper, however, we refer the reader to some commonly used techniques to handle this problem.

The first approach is to use ECCs. This approach has been used in the LAC cryptosystem, which suffers from large decryption error due to the employment of a small coefficient modulus. The authors used nested double encoding using BCH and D2 ECC schemes to reduce the decryption error probability to a desirable level [29].

Another approach is to keep the coefficient modulus large enough but composed of the product of smaller prime factors. One can apply the CRT mechanism to compute over the residues modulo each factor and combine the residues via the CRT reconstruction. This approach has been recently used in [43] to compute 32-bit NTT via two 16-bit NTT on Intel processors leading to a faster implementation than computing 32-bit NTT on the same processor. Note also that the two 16-bit NTTs can be computed in parallel leading to a theoretical $2\times$ improvement in performance.

7. Conclusion

In this work, we present a generalized framework for computing NTT in finite fields with small moduli. The framework leverages several algebraic tools, including the CRT, DWT, and DGT, to replace the 1-dimensional polynomial ring with a 2-dimensional one. Specifically, we demonstrate how the GDGT framework can efficiently compute polynomial multiplication in two use cases inspired by the LAC and NTRU post-quantum cryptosystems. To ensure completeness, we provide theoretical security analyses and computational assessments. Our evaluation of the GDGT framework, conducted through complexity analysis, confirms its efficiency. We anticipate that our work will benefit the cryptographic community by offering new parameter settings for compact lattice-based cryptography. By allowing a more relaxed approach to working with NTT, we pave the way for improved cryptographic schemes. In our future endeavors, we will focus on optimizing algorithms for other lattice-based schemes using the techniques introduced in this work. Additionally, we aim to integrate these algorithms with appropriate ECCs to fine-tune decryption error handling. Moreover, We are particularly interested in exploring the potential application of our methods in homomorphic encryption schemes, especially those dealing with small parameters like FHEW and TFHE. This holds promise for advancements in secure computation on encrypted data.

Acknowledgement

This work was supported by the Research and Innovation Department, Rabdan Academy, through a grant for article processing charges. We appreciate their contribution to the dissemination of our research findings.

Disclosure

The authors declare that they have no relevant financial or non-financial conflicts of interest related to the subject matter of this manuscript.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Castelvecchi D. IBM releases first-ever 1,000-qubit quantum chip. *Nature*. 2023; 624(7991): 238-238. Available from: https://www.nature.com/articles/d41586-023-03854-1.

[2] Pasternack A. IBM built the biggest, coolest quantum computer. Now comes the hard part; 2024. Available from: https://www.fastcompany.com/90992708/ibm-quantum-system-two.

[3] Gidney C, Ekerå M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*. 2021; 5: 433. Available from: https://doi.org/10.22331/q-2021-04-15-433.

[4] Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE; 1994. p.124-134. Available from: https://doi.org/10.1109/SFCS.1994.365700.

[5] Bernstein DJ, Lange T. Post-quantum cryptography. *Nature*. 2017; 549(7671): 188-194.

[6] Chinnasamy P, Albakri A, Khan M, Raja AA, Kiran A, Babu JC. Smart contract-enabled secure sharing of health data for a mobile cloud-based e-health system. *Applied Sciences*. 2023; 13(6): 3970. Available from: https://www.mdpi.com/2076-3417/13/6/3970.

[7] Chinnasamy P, Vinodhini B, Praveena V, Vinothini C, Sujitha BB. Blockchain based Access Control and Data Sharing Systems for Smart Devices. *Journal of Physics: Conference Series*. 2021; 1767(1): 012056. Available from: https://dx.doi.org/10.1088/1742-6596/1767/1/012056.

[8] Tari Z. Security and privacy in cloud computing. *IEEE Cloud Computing*. 2014; 1(1): 54-57. Available from: https://doi.org/10.1109/MCC.2014.20.

[9] Hoffstein J, Pipher J, Silverman JH. *Public key cryptosystem method and apparatus*. U.S. Patent 6,081,597. Google Patents; 2000.

[10] Regev O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*. 2009; 56(6): 1-40. Available from: https://doi.org/10.1145/1060590.1060603.

[11] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: *Advances in Cryptology-EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30-June 3, 2010. Proceedings 29*. Springer; 2010. p.1-23. Available from: https://doi.org/10.1007/978-3-642-13190-5_1.

[12] Gentry C. Fully homomorphic encryption using ideal lattices. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC'09. New York, NY, USA: Association for Computing Machinery; 2009. p.169-178. Available from: https://doi.org/10.1145/1536414.1536440.

[13] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*. 2014; 43(2): 831-871. Available from: https://doi.org/10.1109/FOCS.2011.12.

[14] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*. 2012. Available from: https://eprint.iacr.org/2012/144.

[15] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS'12. New York, NY, USA: Association for Computing Machinery; 2012. p.309-325. Available from: https://doi.org/10.1145/2090236.2090262.

[16] Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: *Advances in Cryptology-CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. Springer; 2013. p.75-92. Available from: https://doi.org/10.1007/978-3-642-40041-4_5.

[17] Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In: *Advances in Cryptology-ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer; 2017. p.409-437. Available from: https://doi.org/10.1007/978-3-319-70694-8_15.

[18] Chillotti I, Gama N, Georgieva M, Izabachène M. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*. 2020; 33(1): 34-91. Available from: https://eprint.iacr.org/2018/421.

[19] Chen Lily MD, Yi-Kai L. *Post-Quantum Cryptography Standardization*. Information Technology Laboratory Computer Security Resource Center; 2017. Available from: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[20] Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE; 2018. p.353-367.

[21] Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, et al. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*. 2018; 28(1): 238-268. Available from: https://doi.org/10.13154/tches.v2018.i1.238-268.

[22] Fouque PA, Hoffstein J, Kirchner P, Lyubashevsky V, Pornin T, Prest T, et al. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to the NIST's Post-Quantum Cryptography Standardization Process*. 2018; 36(5): 1-75.

[23] Bernstein DJ, Hopwood D, Hülsing A, Lange T, Niederhagen R, Papachristodoulou L, et al. SPHINCS: practical stateless hash-based signatures. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer; 2015. p.368-397.

[24] Longa P, Naehrig M. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In: *Cryptology and Network Security: 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings 15*. Springer; 2016. p.124-139. Available from: http://eprint.iacr.org/2016/504.

[25] Al Badawi A, Polyakov Y, Aung KMM, Veeravalli B, Rohloff K. Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme. *IEEE Transactions on Emerging Topics in Computing*. 2019; 9(2): 941-956.

[26] Crandall RE, Pomerance C. Prime numbers: a computational perspective *Mathematics and Statistics*. Springer New York, NY; 2005. p.500-597. Available from: https://doi.org/10.1007/0-387-28979-8.

[27] Crandall RE. Integer convolution via split-radix fast Galois transform. *Center for Advanced Computation Reed College*. 1999. Available from: https://www.reed.edu/physics/faculty/crandall/papers/confgt.pdf.

[28] Al Badawi A, Veeravalli B, Aung KMM. Efficient polynomial multiplication via modified discrete galois transform and negacyclic convolution. In: *Future of Information and Communication Conference*. Springer; 2018. p.666-682.

[29] Lu X, Liu Y, Zhang Z, Jia D, Xue H, He J, et al. LAC: Practical Ring-LWE Based Public-Key Encryption with Byte-Level Modulus. *IACR Cryptology ePrint Archive*. 2018; 2018: 1009.

[30] Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE; 2018. p.353-367.

[31] Lyubashevsky V, Seiler G. NTTRU: Truly Fast NTRU Using NTT. *IACR Trans Cryptogr Hardw Embed Syst*. 2019; 2019(3): 180-201. Available from: https://doi.org/10.13154/tches.v2019.i3.180-201.

[32] Al Badawi A, Veeravalli B, Mun CF, Aung KMM. High-performance FV somewhat homomorphic encryption on GPUs: An implementation using CUDA. *IACR Transactions on Cryptographic Hardware and Embedded Systems*. 2018; 2018(2): 70-95. Available from: https://doi.org/10.13154/tches.v2018.i2.70-95.

[33] Lu X, Liu Y, Zhang Z, Jia D, Xue H, He J, et al. LAC: Practical ring-LWE based public-key encryption with byte-level modulus. *Cryptology ePrint Archive*. 2018. Available from: https://eprint.iacr.org/2018/1009.

[34] Pollard JM. The fast Fourier transform in a finite field. *Mathematics of Computation*. 1971; 25(114): 365-374.

[35] Agarwal RC, Burrus CS. Number theoretic transforms to implement fast digital convolution. *Proceedings of the IEEE*. 1975; 63(4): 550-560.

[36] Gold B, Rader CM. Digital processing of signals. *Digital Processing of Signals*. 1969. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1165754.

[37] Agarwal R, Burrus C. Fast convolution using Fermat number transforms with applications to digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1974; 22(2): 87-97.

[38] Creutzburg R, Tasche M. Number-theoretic transforms of prescribed length. *Mathematics of Computation*. 1986; 47(176): 693-701.

[39] Crandall R, Fagin B. Discrete weighted transforms and large-integer arithmetic. *Mathematics of Computation*. 1994; 62(205): 305-324.

[40] Bhattacharya M, Creutzburg R, Astola J. Some historical notes on number theoretic transform. In: *Proc. 2004 Int. TICS Workshop on Spectral Methods and Multirate Signal Processing, vol.2004*; 2004. .

[41] Badawi AA, Hoang L, Mun CF, Laine K, Aung KMM. PrivFT: Private and fast text classification with homomorphic Encryption. *IEEE Access*. 2020; 8: 226544-226556. Available from: https://doi.org/10.1109/ACCESS.2020.3045465.

[42] Hülsing A, Rijneveld J, Schanck J, Schwabe P. High-speed key encapsulation from NTRU. In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer; 2017. p.232-252.

[43] Chung CMM, Hwang V, Kannwischer MJ, Seiler G, Shih CJ, Yang BY. *NTT Multiplication for NTT-unfriendly Rings*. Cryptology ePrint Archive, Report 2020/1397; 2020. Available from: https://eprint.iacr.org/2020/1397.

[44] Winkler F. *Polynomial algorithms in computer algebra*. Springer Science & Business Media; 2012.

[45] Karatsuba AA, Ofman YP; Russian Academy of Sciences. Multiplication of many-digital numbers by automatic computers. *Proceedings of the Academy of Sciences of the USSR*. 1962; 145(2): 293-294.

[46] Bodrato M. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In: *International Workshop on the Arithmetic of Finite Fields*. Springer; 2007. p.116-133.

[47] Albrecht MR, Curtis BR, Deo A, Davidson A, Player R, Postlethwaite EW, et al. Estimate all the {LWE, NTRU} schemes! In: *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*. Springer; 2018. p.351-367. Available from: https://doi.org/10.1007/978-3-319-98113-0_19.

[48] Albrecht MR, Player R, Scott S. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*. 2015; 9(3): 169-203. Available from: https://doi.org/10.1515/jmc-2015-0016.