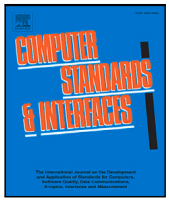
Computer Standards & Interfaces 89 (2024) 103828

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi



Compact and efficient KEMs over NTRU lattices

Zhichuang Liang [a], Boyue Fang [a], Jieyu Zheng [a], Yunlei Zhao [a,b,*]

[a] Department of Computer Science, Fudan University, Shanghai, China
[b] State Key Laboratory of Cryptology, Beijing, China

ARTICLE INFO

Keywords:
Lattice-based cryptography
Key encapsulation mechanism
NTRU
Number theoretic transform
Integrated performance

ABSTRACT

The NTRU lattice is a promising candidate to construct practical cryptosystems, in particular key encapsulation mechanism (KEM), resistant to quantum computing attacks. Nevertheless, there are still some inherent obstacles to NTRU-based KEM schemes when considering integrated performance, taking security, bandwidth, error probability, and computational efficiency *as a whole*, that is as good as and even better than their {R,M}LWE-based counterparts. In this work, we address the challenges by presenting a new family of NTRU-based KEM schemes, denoted as CTRU and CNTR. By bridging low-dimensional lattice codes and high-dimensional NTRU-lattice-based cryptography with careful design and analysis, to the best of our knowledge, CTRU and CNTR are the first NTRU-based KEM schemes featuring scalable ciphertext compression via only one *single* ciphertext polynomial, and are the first that can outperform {R,M}LWE-based KEM schemes in terms of integrated performance. For instance, when compared to Kyber, the only KEM scheme currently standardized by NIST, our recommended parameter set CNTR-768 exhibits approximately 12% smaller ciphertext size, when its security is strengthened by $(8, 7)$ bits for classical and quantum security respectively, with a significantly lower error probability (2^{-230} for CNTR-768 vs. 2^{-164} for Kyber-768). In terms of the state-of-the-art AVX2 implementation of Kyber-768, CNTR-768,achieves a speedup of 2.7X in KeyGen, 3.3X in Encaps, and 1.6X in Decaps, respectively. When compared to the NIST Round 3 finalist NTRU-HRSS, CNTR-768,features 15% smaller ciphertext size, coupled with an improvement of $(55, 49)$ bits for classical and quantum security respectively. As for the AVX2 implementation, CNTR-768,outperforms NTRU-HRSS by 26X in KeyGen, 3.0X in Encaps, and 2.2X in Decaps, respectively. Along the way, we develop new techniques for more accurate error probability analysis, and a unified number theoretic transform (NTT) implementation for multiple parameter sets, which may be of independent interest.

1. Introduction

Most current public-key cryptographic schemes in use, which are based on the hardness assumptions of factoring large integers and solving (elliptic curve) discrete logarithms, will suffer from quantum attacks once practical quantum computers are built. These cryptosystems play an important role in ensuring the confidentiality and authenticity of communications on the Internet. With the increasing cryptographic security risks posed by quantum computing, post-quantum cryptography (PQC) has garnered significant research attention in recent years. Post-quantum cryptographic schemes can be broadly categorized into five main types: hash-based, code-based, lattice-based, multivariable-based, and isogeny-based schemes, among which lattice-based cryptography is commonly regarded as one of the most promising, owing to its outstanding integrated performance in terms of security, communication bandwidth, and computational efficiency.

In the post-quantum cryptography standardization competition organized by the U.S. National Institute of Standards and Technology (NIST), lattice-based schemes occupy a prominent position. NIST announced four candidates to be standardized [1], with three of them being based on lattices. The majority of lattice-based schemes are based on algebraically structured lattices (ideal lattice, NTRU lattice, and module lattice). They are predominantly instantiated from the following two categories of hardness assumptions: *{R,M}LWE/LWR* [2–6] and *NTRU* [7].

NTRU was initially introduced by Hoffstein, Pipher, and Silverman during the rump session of Crypto96 [8], and it survived a lattice attack in 1997 [9]. With subsequent enhancements to its security, NTRU was formally published in 1998 [7], referred to as NTRU-HPS in this work. NTRU-HPS was the first practical public key cryptosystem based on the lattice hardness assumptions over polynomial rings, and over the years, various variants of NTRU-HPS have emerged, including those proposed in [10–15]. Since its introduction, the NTRU cryptosystem has demonstrated resilience against attacks and cryptanalysis for over 26 years.

* Corresponding author at: Department of Computer Science, Fudan University, Shanghai, China.
E-mail address: ylzhao@fudan.edu.cn (Y. Zhao).

https://doi.org/10.1016/j.csi.2023.103828
Received 9 October 2023; Received in revised form 15 November 2023; Accepted 25 December 2023
Available online 2 January 2024

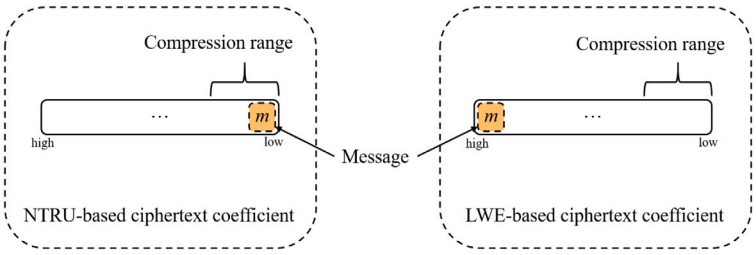


Fig. 1. Differences of message positions between NTRU-based and LWE-based KEM schemes.

NTRU has played a foundational role in numerous cryptographic protocols, as evidenced by its inclusion in notable references such as [16–21]. Particularly, NTRU-based schemes have achieved impressive success in the NIST PQC standardization. Notably, the Falcon signature scheme [17], based on the NTRU assumption, is one of the signature candidates standardized by NIST [1]. NTRU KEM (including NTRU-HRSS and NTRUEncrypt) [12] is one of the seven finalists, and NTRU Prime KEM (including SNTRU Prime and NTRU LPRime) [11] is one of the alternate candidates in the third round of NIST PQC standardization. Although NTRU-based KEM schemes were not chosen to be standardized by NIST, it is crucial not to ignore their considerable potential in PQC research and practical deployment due to their appealing features. Further research exploration and optimization of NTRU-based KEM schemes remain warranted.

Actually, some standardizations and Internet protocols have already incorporated NTRU-based PKE/KEM schemes. The standard IEEE Std 1363.1 [22], which was issued in 2008, standardizes some lattice-based public-key schemes, including NTRUEncrypt. The standard X9.98 [23] includes NTRUEncrypt as a part of the X9 standards applied to the financial services industry. In 2016, NTRUEncrypt was used to generate additional temporary keys alongside temporary Elliptic Curve Diffie–Hellman (ECDH) keys in the Tor protocol, in order to enhance forward secrecy against quantum adversaries [24]. The European Union's PQCRYPTO project (Horizon 2020 ICT-645622) [25] explores another NTRU variant [26]. In particular, the OpenSSH standard, starting from its 9.0 version released in April 2022, has adopted NTRU Prime *by default*, along with X25519 ECDH in a hybrid mode, in order to prevent "capture now decrypt later" attacks [27].

1.1. Challenges and motivations

When considering *integrated performance* (in terms of security, bandwidth, error probability, and computational efficiency *as a whole*, instead of individual metrics), up to now NTRU-based KEM schemes are, *in general*, inferior to their {R,M}LWE-based counterparts. It might be the partial reason that NTRU-based KEM schemes were not finally standardized by NIST. In the following, some obstacles and challenges faced by current NTRU-based KEM schemes are summarized, which also provides the motivations of this work.

1.1.1. Small secret ranges

The first common limitation of some NTRU-based KEM schemes like [11,12,15] is that they usually support a narrow secret range, typically $\{-1, 0, 1\}$, which inherently limits the security level achievable by NTRU-based KEM schemes like [11,12,15]. Although larger secret ranges are possible, at the same security level they lead to larger bandwidth [28, Figures 13 and 14]. However, {R,M}LWE-based KEM schemes possess advantages in accommodating larger secret ranges to enhance security and maintaining tolerable bandwidth when using the approximate moduli as in NTRU-based KEM schemes.

1.1.2. Large bandwidth

The next limitation is that traditional NTRU-based KEM schemes commonly have larger bandwidth compared to their {R,M}LWE-based counterparts. On the one hand, traditional NTRU-based KEM schemes set relatively large moduli (together with relatively small secret ranges) in order to achieve perfect correctness. On the other hand, the larger bandwidth is due to the inherent inability to compress the ciphertext in NTRU-based KEM schemes. As shown in Fig. 1, the initial message is encoded into the least significant bits of the ciphertext in traditional NTRU-based KEM schemes.

Compressing the ciphertext means dropping some of the least significant bits, thereby equivalently increasing the small error. For {R,M}LWE-based KEM schemes, the consequences of reasonably compressing the ciphertext can be mitigated if the total error falls within the capacity range of the message-recovering mechanism. However, in the context of traditional NTRU-based KEM schemes, compressing the ciphertext results in the loss of valuable information encoded in the least significant bits. Consequently, the original messages cannot be accurately recovered.

1.1.3. Weak starting point of security reduction

For most NTRU-based KEM schemes, their security against chosen ciphertext attacks (CCA) is typically reduced to the security of one-way (OW-CPA) secure encryption, as opposed to the traditional security of indistinguishability under chosen plaintext attack (IND-CPA). However, it is important to note that IND-CPA security represents a more robust security concept than OW-CPA security. Although OW-CPA security can be transformed into IND-CPA security, this transformation usually comes at the cost of further relaxing the reduction bound, especially in the context of the quantum random oracle model (QROM) [29]. Alternatively, there exists a possibility of achieving a tight reduction from CCA security to OW-CPA security in *deterministic* public-key encryption (DPKE). This, however, necessitates a more intricate decapsulation process and imposes additional requirements such as zero error probability and certain extra assumptions (e.g., NTRU-HRSS [12] or SNTRU Prime [11]). As a consequence, it is still desirable for NTRU-based KEM schemes to attain a security reduction from CCA security to IND-CPA security, as in {R,M}LWE-based KEM schemes.

1.1.4. Complicated key generation

Typically, there is only one or two polynomial multiplications in the encryption and decryption algorithms of NTRU-based KEM schemes, making the encryption and decryption algorithms very fast. However, for most NTRU-based KEM schemes (with the works [15,29] as exceptions), the primary efficiency obstacle arises during key generation algorithm since there exists a complicated computation involved in polynomial inversion, and for many of the polynomial rings selected by NTRU-based KEM schemes, there are not many efficient algorithms available for this task.

Unfortunately, there are no literatures to propose such NTRU-based KEM schemes which can overcome all the obstacles mentioned above. This leads us to the following motivating question.

Motivating question

Is it possible to construct NTRU-based KEM schemes that exhibit essentially the same, or *even superior*, integrated performance in terms of security, bandwidth, error probability, and computational efficiency *as a whole*, compared to {R,M}LWE-based KEM schemes?

1.2. Our contributions

Our main result demonstrates that NTRU-based KEM schemes can practically achieve remarkable integrated performance, comparable to or even surpassing {R,M}LWE-based KEM schemes.

Specifically, we introduce new variants of the NTRU-based cryptosystem, denoted as CTRU and CNTR, which enable larger secret ranges, achieve scalable ciphertext compression, demonstrate CCA provable security directly reduced to IND-CPA security, and feature efficient implementations. The error probabilities of CTRU and CNTR are sufficiently low, which are typically lower than those of Kyber.

1.2.1. Efficient constant-time scalable lattice code

For enhanced capability in message recovery and the achievement of negligible error probabilities, we refine and apply the scalable E_8 lattice code, building upon the research works [30–33]. To avoid potential timing attacks, we present efficient and constant-time encoding and decoding algorithms for the scalable E_8 lattice code in Sections 3 and 7.6.

1.2.2. New constructions

Our CTRU and CNTR demonstrate novel approaches to constructing NTRU-based KEM schemes. Regarding CTRU.PKE, the key generation algorithm is similar to the existing NTRU-based KEM schemes such as [7,11,12,29]. We develop a novel encryption algorithm which breaks through the limitation of ciphertext compression in NTRU-based KEM schemes, such that each ciphertext can be compressed in the case of one *single* polynomial. The decryption algorithm correctly recovers messages with the assistance of the decoding algorithm of the scalable E8 lattice code. CTRU.KEM is constructed via $\mathrm{FO}^{\not\perp}_{ID(pk),m}$, a variant of the Fujisaki-Okamoto transformation proposed in [34]. CNTR is a simplified and more efficient variant of CTRU. To the best of our knowledge, CTRU and CNTR are the first NTRU-based KEM constructions which bridge high-dimensional NTRU-lattice-based cryptography and low-dimensional lattice codes, and are the first NTRU-based KEM schemes with scalable ciphertext compression via only one single ciphertext polynomial. For detailed constructions of CTRU and CNTR, refer to Section 4.

1.2.3. Provable security

CTRU.PKE (resp., CNTR.PKE) can achieve IND-CPA security under the NTRU assumption and RLWE (resp., RLWR) assumption, while most of the existing practical NTRU-based PKE schemes only achieve OW-CPA security. Note that, the RLWE and RLWR assumptions are only required to guarantee IND-CPA security in our schemes, since CTRU.PKE and CNTR.PKE are still OW-CPA secure merely based on the NTRU assumption. The reduction advantages for achieving CCA security in our CTRU.KEM and CNTR.KEM are tighter than those of NTTRU [15] and NTRU-C [29]. For example, in the quantum setting, the CCA reduction bound of CTRU.KEM is dominated by $O(\sqrt{q'\epsilon_{CPA}})$, whereas those of NTTRU and NTRU-C are dominated by $O(q'\sqrt{\epsilon_{OW}})$ and $O(q'^{1.5}\sqrt[4]{\epsilon_{OW}})$ respectively, where ϵ_{CPA} (resp., ϵ_{OW}) represents the advantage against the underlying IND-CPA (resp., OW-CPA) secure PKE and q' is the total query number.

1.2.4. More accurate analysis of error probability

In a prior study [15], the error probability was conservatively estimated based on a worst-case setting, involving $\frac{3}{2}n$ terms for each coefficient in the polynomial product in $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$. In this work, we precisely derive the number of terms in the polynomial product coefficients and improve the error probability analysis presented in [15], which may be of independent interest. The detailed analysis is provided in Section 4.4.

1.2.5. Parameter sets and comparisons

As illustrated in Table 1, we provide concise comparisons between our schemes and other prominent practical NTRU-based KEM schemes: NTRU-HRSS [12], SNTRU Prime [11], NTTRU [15] and NTRU-C^{768}_{3457} [29], as well as the NIST standardized candidate Kyber [35] and the NIST Round 3 finalist Saber [36]. "Assumptions" refers to the underlying hardness assumptions. "Reduction" means that IND-CCA security is reduced to what kinds of CPA security, where "IND" (resp., "OW") refers to indistinguishability (resp., one-wayness) and "RPKE" (resp., "DPKE") refers to randomized (resp., deterministic) public-key encryptions. "Rings" refers to the underlying polynomial rings. "n" means the total dimension of algebraically structured lattices. "q" stands for the ring modulus. The public key size $|pk|$, ciphertext size $|ct|$, and B.W. (bandwidth, $|pk| + |ct|$) are measured in bytes. "Sec.C" and "Sec.Q" mean the estimated security expressed in bits in the classical and quantum setting respectively, which are gotten by the same methodology and scripts provided by Kyber, Saber, and NTRU KEM in NIST PQC Round 3, where we minimize the target values of the two hardness problems (saying, NTRU and RLWE/RLWR) when they have different security values. "δ" indicates the error probabilities, where the error probabilities of NTTRU and NTRU-C^{768}_{3457} are re-tested according to our accurate methodology in Section 4.4.

Upon comparison, CNTR stands out with the smallest bandwidth and the strongest security guarantees among all practical NTRU-based KEM schemes. For example, when compared to the NIST Round 3 finalist NTRU-HRSS, our CNTR-768 exhibits approximately 15% smaller ciphertext size, with security strengthened by $(55, 49)$ bits for classical and quantum security, respectively. The error probabilities of CNTR are meticulously set based on the targeted security level for each parameter set, making them negligible in accordance with the specified security level. When compared to Kyber-768, CNTR-768 exhibits approximately 12% smaller ciphertext size, accompanied by security enhancements of $(8, 7)$ bits for classical and quantum security, respectively. Across all three recommended parameter sets, CNTR demonstrates significantly lower error probabilities compared to Kyber (e.g., 2^{-230} for CNTR-768 vs. 2^{-164} for Kyber-768). To the best of our knowledge, CNTR is the first NTRU-based KEM scheme that can outperform Kyber in the integrated performance when considering security, bandwidth, error probability, and computational efficiency as a whole. Another significant point is that CTRU and CNTR theoretically admit more flexible key sizes to be encapsulated, i.e., $n/2$-bit shared keys, whereas Kyber and Saber are limited to encapsulating fixed 256-bit shared keys.

1.2.6. Unified NTT

Regarding NTT-based polynomial multiplications in NTRU-based KEM schemes, a common challenge arises from the necessity of employing various NTT algorithms for multiple parameter sets, which is often perceived as a significant drawback. In particular, this issue brings inconvenient issues for software implementation and especially for hardware implementation. In this work, we overcome this problem by presenting a unified NTT technique capable of computing various NTT algorithms over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ for all $n \in \{512, 768, 1024\}$, which may be of independent interest. Further details are provided in Section 6.

Table 1
Comparisons between CTRU, CNTR and other practical lattice-based KEM schemes.

Schemes	Assumptions	Reduction	Rings	n	q	$\|pk\|$	$\|ct\|$	B.W.	(Sec.C, Sec.Q)	δ
CTRU (Ours)	NTRU, RLWE	IND-CPA RPKE	$\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$	512	3457	768	640	1408	(118,107)	2^{-143}
				768	3457	1152	960	2112	(181,164)	2^{-184}
				1024	3457	1536	1408	2944	(255,231)	2^{-195}
CNTR (Ours)	NTRU, RLWR	IND-CPA RPKE	$\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$	512	3457	768	640	1408	(127,115)	2^{-170}
				768	3457	1152	960	2112	(191,173)	2^{-230}
				1024	3457	1536	1280	2816	(253,230)	2^{-291}
NTRU-HRSS [12]	NTRU	OW-CPA DPKE	$\mathbb{Z}_q[x]/(x^n - 1)$	701	8192	1138	1138	2276	(136,124)	$2^{-\infty}$
SNTRU Prime-761 [11]	NTRU	OW-CPA DPKE	$\mathbb{Z}_q[x]/(x^n - x - 1)$	761	4591	1158	1039	2197	(153,137)	$2^{-\infty}$
NTTRU [15]	NTRU	OW-CPA RPKE	$\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$	768	7681	1248	1248	2496	(153,140)	2^{-1352}
NTRU-C^{768}_{3457} [29]	NTRU, RLWE	IND-CPA RPKE	$\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$	768	3457	1152	1184	2336	(171,155)	2^{-281}
Kyber [35]	MLWE	IND-CPA RPKE	$\mathbb{Z}_q[x]/(x^{n/k} + 1)$ $k = 2, 3, 4$	512	3329	800	768	1568	(118,107)	2^{-139}
				768	3329	1184	1088	2272	(183,166)	2^{-164}
				1024	3329	1568	1568	3136	(256,232)	2^{-174}
Saber [36]	MLWR	IND-CPA RPKE	$\mathbb{Z}_q[x]/(x^{n/k} + 1)$ $k = 2, 3, 4$	512	8192	672	736	1408	(118,107)	2^{-120}
				768	8192	992	1088	2080	(189,172)	2^{-136}
				1024	8192	1312	1472	2784	(260,236)	2^{-165}

1.2.7. Implementation and benchmark

The portable C implementation and optimized AVX2 implementation for CTRU-768 and CNTR-768 are presented in Section 7. We conduct benchmark comparisons with related lattice-based KEM schemes and prominent non-lattice-based KEM schemes. The experimental results demonstrate that the encapsulation and decapsulation algorithms of our schemes are among the most efficient. In terms of the AVX2 implementation, CTRU-768 outperforms NTRU-HRSS by 33X in KeyGen, 2.7X in Encaps, and 2.1X in Decaps, respectively; CNTR-768 is faster than NTRU-HRSS by 26X in KeyGen, 3.0X in Encaps, and 2.2X in Decaps, respectively. When compared to the state-of-the-art AVX2 implementation of Kyber-768, CTRU-768 achieves higher speeds by 3.4X in KeyGen, 3.0X in Encaps, and 1.5X in Decaps, respectively; CNTR-768 outperforms Kyber-768 by 2.7X in KeyGen, 3.3X in Encaps, and 1.6X in Decaps, respectively.

1.3. Related works

In recent years, many NTRU variants have been proposed. Jarvis and Nevins [14] presented a new variant of NTRU-HPS [7] over the ring of Eisenstein integers $\mathbb{Z}[\omega]/(x^n - 1)$ where $\omega = e^{2\pi i/3}$, which has smaller key sizes and faster performance than NTRU-HPS. Bagheri et al. [10] generalized NTRU-HPS over bivariate polynomial rings of the form $(-1, -1)/(\mathbb{Z}[x, y]/(x^n - 1, y^n - 1))$ for stronger security and smaller public key sizes. Hülsing et al. [37] improved NTRU-HPS in terms of speed, key size, and ciphertext size, and presented NTRU-HRSS, which was one of the finalists in NIST PQC Round 3 [12]. Bernstein et al. [38] proposed NTRU Prime, which aims for "an efficient implementation of high security prime-degree large-Galois-group inert-modulus ideal-lattice-based cryptography". It tweaks the textbook NTRU scheme to certain rings with less special structures, i.e., $\mathbb{Z}_q[x]/(x^n - x - 1)$ where both n and q are prime numbers.

In order to obtain better performance of NTRU encryption, Lyubashevsky and Seiler [15] instantiated it over $\mathbb{Z}_{7681}[x]/(x^{768} - x^{384} + 1)$. Then Duman et al. [29] generalized the rings $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ with various n's for flexible parameter selection. However, all of these schemes follow the similar structure of NTRU-HPS and do not provide support for ciphertext compression.

Fouque et al. [13] proposed a new NTRU variant named BAT. It shares many similarities with Falcon signature [17] where a trapdoor basis is required in the secret key, which makes its key generation complicated. BAT uses two linear equations in two unknowns to recover the secret and error, without introducing the modulus p to extract message. It reduces the ciphertext sizes by constructing its intermediate value as an RLWR instance (with binary secrets), and encrypts the message via ACWC_0 transformation [29]. However, ACWC_0 transformation consists of two terms, causing that there are some dozens of bytes in the second ciphertext term. Another disadvantage is about the inflexibility of selecting parameters. Since BAT applies power-of-two cyclotomics $\mathbb{Z}_q[x]/(x^{2^k} + 1)$, it is inconvenient to find an underlying cyclotomic polynomial of some particular degree up to the next power of two. For example, BAT chooses $\mathbb{Z}_q[x]/(x^{512} + 1)$ and $\mathbb{Z}_q[x]/(x^{1024} + 1)$ for NIST recommended security levels I and V, but it lacks of parameter set for level III, which, however, is the aimed and recommended security level for most lattice-based KEM schemes like Kyber [35] and our schemes. Although BAT has an advantage of bandwidth, its key generation is thousands times slower than other NTRU-based KEM schemes, and there are some worries about its provable security based on the RLWR assumption with binary secrets which is quite a new assumption tailored for BAT. For the above reasons, we do not make a direct comparison between our schemes and BAT.

2. Preliminaries

2.1. Notations and definitions

Let $\mathbb{Z}$ and $\mathbb{R}$ be the set of rational integers and real numbers, respectively. Let n and q be some positive integers. Denote $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z} \cong \{0, 1, \ldots, q-1\}$ and $\mathbb{R}_q = \mathbb{R}/q\mathbb{R}$. Let $\mathbb{Z}_q^\times$ be the group of invertible elements of $\mathbb{Z}_q$. For any $x \in \mathbb{R}$, $\lfloor x \rceil$ denotes the closest integer to x. We denote $\mathbb{Z}[x]/(x^n - x^{n/2} + 1)$ and $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ by $\mathcal{R}$ and $\mathcal{R}_q$ respectively in this work. The elements in $\mathcal{R}$ or $\mathcal{R}_q$ are polynomials, which are denoted by regular font letters such as f, g. The polynomial, e.g., f, in $\mathcal{R}$ (resp., $\mathcal{R}_q$) can be represented as: $f = \sum_{i=0}^{n-1} f_i x^i$, where $f_i \in \mathbb{Z}$ (resp., $f_i \in \mathbb{Z}_q$).

Cyclotomics. Let m be a positive integer, and $\xi_m = \exp(\frac{2\pi i}{m})$ be a mth root of unity. The mth cyclotomic polynomial $\Phi_m(x)$ is defined as $\Phi_m(x) = \prod_{j \in \mathbb{Z}_m^\times} (x - \xi_m^j)$. It is a monic irreducible polynomial of degree $\phi(m)$ in $\mathbb{Z}[x]$, where ϕ is the Euler function. We focus on the trinomial cyclotomic polynomial: $\Phi_m(x) = x^n - x^{n/2} + 1$ where $m = 2^e 3^l$, $l, e \geq 1$, and $n = \phi(m) = m/3$.

Modular reductions. In this work, we expand the definition of modular reduction from $\mathbb{Z}$ to $\mathbb{R}$. For a positive number q, $r' = r \bmod^{\pm} q$ means that r' is the representative element of r in $[-\frac{q}{2}, \frac{q}{2})$. Let $r' = r \bmod q$ denote as the representative element of r in $[0, q)$.

Sizes of elements. Let q be a positive number. For any $w \in \mathbb{R}$, denote by $\|w\|_{q,\infty} = |w \bmod^{\pm} q|$ its ℓ_∞ norm. If w is an n-dimension vector, then its ℓ_2 norm is defined as $\|w\|_{q,2} = \sqrt{\|w_0\|_{q,\infty}^2 + \cdots + \|w_{n-1}\|_{q,\infty}^2}$. Note that $\|w\|_{q,2} = \|w\|_{q,\infty}$ holds for any $w \in \mathbb{R}$.

Sets and Distributions. For a set D, we denote by $x \xleftarrow{\$} D$ sampling x from D uniformly at random. If D is a probability distribution, $x \leftarrow D$ means that x is chosen according to the distribution D. The centered binomial distribution B_η with respect to a positive integer η is defined as follows: Sample $(a_1, \ldots, a_\eta, b_1, \ldots, b_\eta) \xleftarrow{\$} \{0,1\}^{2\eta}$, and output $\sum_{i=1}^{\eta}(a_i - b_i)$. Sampling a polynomial $f \leftarrow B_\eta$ means sampling each coefficient according to B_η individually.

2.2. Cryptographic primitives

A public-key encryption scheme contains PKE = (KeyGen, Enc, Dec), with a message space $\mathcal{M}$. The key generation algorithm KeyGen returns a pair of public key and secret key (pk, sk). The encryption algorithm Enc takes a public key pk and a message $m \in \mathcal{M}$ to produce a ciphertext c. Denote by $\mathrm{Enc}(pk, m; coin)$ the encryption algorithm with an explicit randomness $coin$ if necessary. The deterministic decryption algorithm Dec takes a secret key sk and a ciphertext c, and outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$ to indicate a rejection. The decryption error δ of PKE is defined as $\mathrm{E}[\max_{m\in\mathcal{M}}\Pr[\mathrm{Dec}(sk,\mathrm{Enc}(pk,m)) \neq m]] < \delta$ where the expectation is taken over $(pk, sk) \leftarrow$ KeyGen and the probability is taken over the random coins of Enc. The advantage of an adversary A against *indistinguishability under chosen-plaintext attacks* (IND-CPA) for public-key encryption is defined as $\mathbf{Adv}^{\text{IND-CPA}}_{\text{PKE}}(\mathsf{A}) =$

$$\left|\Pr\left[b' = b : \begin{array}{c}(pk, sk) \leftarrow \mathrm{KeyGen}();\\ (m_0, m_1, s) \leftarrow \mathsf{A}(pk);\\ b \xleftarrow{\$} \{0,1\}; c^* \leftarrow \mathrm{Enc}(pk, m_b);\\ b' \leftarrow \mathsf{A}(s, c^*)\end{array}\right] - \frac{1}{2}\right|.$$

A key encapsulation mechanism contains KEM = (KeyGen, Encaps, Decaps) with a key space $\mathcal{K}$. The key generation algorithm KeyGen returns a pair of public key and secret key (pk, sk). The encapsulation algorithm Encaps takes a public key pk to produce a ciphertext c and a key $K \in \mathcal{K}$. The deterministic decapsulation algorithm Decaps takes a secret key sk and a ciphertext c, and outputs either a key $K \in \mathcal{K}$ or a special symbol $\perp$ indicating a rejection. The error probability δ of KEM is defined as $\Pr[\mathrm{Decaps}(sk, c) \neq K : (c, K) \leftarrow \mathrm{Encaps}(pk)] < \delta$ where the probability is taken over $(pk, sk) \leftarrow$ KeyGen and the random coins of Encaps. The advantage of an adversary A against *indistinguishability under chosen-ciphertext attacks* (IND-CCA) for KEM is defined as $\mathbf{Adv}^{\text{IND-CCA}}_{\text{KEM}}(\mathsf{A}) =$

$$\left|\Pr\left[b' = b : \begin{array}{c}(pk, sk) \leftarrow \mathrm{KeyGen}();\\ b \xleftarrow{\$} \{0,1\};\\ (c^*, K_0^*) \leftarrow \mathrm{Encaps}(pk);\\ K_1^* \xleftarrow{\$} \mathcal{K};\\ b' \leftarrow \mathsf{A}^{\mathrm{Decaps}(\cdot)}(pk, c^*, K_b^*)\end{array}\right] - \frac{1}{2}\right|.$$

2.3. Hardness assumptions

The NTRU assumption [7], RLWE assumption [5] and RLWR assumption [3] are introduced as follows.

Definition 1 (*NTRU Assumption [7]*)**.** Let Ψ be a distribution over a polynomial ring R. Sample f' and g according to Ψ, and $f = pf' + 1$ is invertible in R where $p \in \mathbb{N}$. Let $h = g/f$. The (decisional) NTRU problem is to distinguish h from a uniformly-random element in R. It is hard if the advantage $\mathbf{Adv}^{\text{NTRU}}_{R,\Psi}(\mathsf{A})$ of any probabilistic polynomial time (PPT) adversary A is negligible, where $\mathbf{Adv}^{\text{NTRU}}_{R,\Psi}(\mathsf{A}) =$

$$\left|\Pr\left[b' = 1 : \begin{array}{c}f', g \leftarrow \Psi; f = pf' + 1 \wedge f^{-1} \in R\\ h = g/f \in R; b' \leftarrow \mathsf{A}(h)\end{array}\right] - \Pr\left[b' = 1 : h \xleftarrow{\$} R; b' \leftarrow \mathsf{A}(h)\right]\right|.$$

Definition 2 (*RLWE Assumption [5]*)**.** Let Ψ be a distribution over a polynomial ring R. The (decisional) Ring-Learning with errors (RLWE) problem over R is to distinguish uniform samples $(h, c) \xleftarrow{\$} R \times R$ from samples $(h, c) \in R \times R$ where $h \xleftarrow{\$} R$ and $c = hr + e$ with $r, e \leftarrow \Psi$. It is hard if the advantage $\mathbf{Adv}^{\text{RLWE}}_{R,\Psi}(\mathsf{A})$ of any probabilistic polynomial time adversary A is negligible, where $\mathbf{Adv}^{\text{RLWE}}_{R,\Psi}(\mathsf{A}) =$

$$\left|\Pr\left[b' = 1 : \begin{array}{c}h \xleftarrow{\$} R; r, e \leftarrow \Psi;\\ c = hr + e \in R; b' \leftarrow \mathsf{A}(h, c)\end{array}\right] - \Pr\left[b' = 1 : h \xleftarrow{\$} R; c \xleftarrow{\$} R; b' \leftarrow \mathsf{A}(h, c)\right]\right|.$$

Definition 3 (*RLWR Assumption [3]*)**.** Let $q > p \geq 2$ be integers. Let Ψ be a distribution over a polynomial ring R. Let $R_q = R/qR$ and $R_p = R/pR$ be the quotient rings. The (decisional) Ring-Learning with rounding (RLWR) problem is to distinguish uniform samples $(h, c) \xleftarrow{\$} R_q \times R_p$ from samples $(h, c) \in R_q \times R_p$ where $h \xleftarrow{\$} R_q$ and $c = \lfloor \frac{p}{q} hr \rceil \bmod p$ with $r \leftarrow \Psi$. It is hard if the advantage $\mathbf{Adv}^{\text{RLWR}}_{R,\Psi}(\mathsf{A})$ of any probabilistic polynomial time adversary A is negligible, where $\mathbf{Adv}^{\text{RLWR}}_{R,\Psi}(\mathsf{A}) =$

$$\left|\Pr\left[b' = 1 : \begin{array}{c}h \xleftarrow{\$} R_q; r \leftarrow \Psi;\\ c = \lfloor \frac{p}{q} hr \rceil \bmod p \in R_p; b' \leftarrow \mathsf{A}(h, c)\end{array}\right] - \Pr\left[b' = 1 : h \xleftarrow{\$} R_q; c \xleftarrow{\$} R_p; b' \leftarrow \mathsf{A}(h, c)\right]\right|.$$

2.4. Number theoretic transform

From a computational point of view, the fundamental and time-consuming operations in NTRU-based schemes are the multiplications and divisions of the elements in the rings $\mathbb{Z}_q[x]/(\Phi(x))$. Number theoretic transform (NTT) is a special case of fast Fourier transform (FFT) over a finite field [39]. NTT is the most efficient method for computing polynomial multiplication of high degrees, due to its quasilinear complexity $O(n \log n)$. The concrete NTT-based multiplication with respect to f and g is $INTT(NTT(f) \circ NTT(g))$, where NTT represents the forward transform, $INTT$ represents the inverse transform, and "$\circ$" represents the point-wise multiplication.

The FFT trick [40] is a fast algorithm to compute NTT based on the Chinese Remainder Theorem (CRT) in the ring form. Briefly speaking, given pairwise co-prime polynomials $g_1, g_2, \ldots, g_k$, the CRT isomorphism is that φ :

$$\mathbb{Z}_q[x]/(g_1 g_2 \cdots g_k) \cong \mathbb{Z}_q[x]/(g_1) \times \mathbb{Z}_q[x]/(g_2) \times \cdots \times \mathbb{Z}_q[x]/(g_k)$$

along with $\varphi(f) = (f \bmod g_1, f \bmod g_2, \ldots, f \bmod g_k)$. In the case of the classical radix-2 FFT trick step, given the isomorphism $\mathbb{Z}_q[x]/(x^{2m} - \zeta^2) \cong \mathbb{Z}_q[x]/(x^m - \zeta) \times \mathbb{Z}_q[x]/(x^m + \zeta)$ where ζ is invertible in $\mathbb{Z}_q$, the computation of the forward FFT trick and inverse FFT trick can be conducted via Cooley–Tukey butterfly [41] and Gentleman-Sande butterfly [42], respectively. The former indicates the computation from (f_i, f_j) to $(f_i + \zeta \cdot f_j, f_i - \zeta \cdot f_j)$, while the later indicates the computation from (f_i', f_j') to $(f_i' + f_j', (f_i' - f_j') \cdot \zeta^{-1})$. As for the classical radix-3 FFT trick step, it is based on the isomorphism $\mathbb{Z}_q[x]/(x^{3m} - \zeta^3) \cong \mathbb{Z}_q[x]/(x^m - \zeta) \times \mathbb{Z}_q[x]/(x^m - \rho\zeta) \times \mathbb{Z}_q[x]/(x^m - \rho^2\zeta)$ where ζ is invertible in $\mathbb{Z}_q$ and ρ is the third root of unity. The mixed-radix NTT means that there are more than one type of FFT trick.

3. The lattice code

Before introducing our NTRU-based KEM schemes, we present a simple and efficient lattice code in this section. The rationale behind this is the necessity of a dense lattice with an efficient decoding algorithm in our schemes, aiming to enhance the efficiency of message recovery and achieve negligible error probabilities. The coding algorithms must satisfy the following conditions.

- The operations should be simple enough, and can be implemented by efficient arithmetic (better for integer-only arithmetic).
- The implementations of the coding algorithms are constant-time in order to avoid timing attacks.
- The decoding bound is large enough such that it leads to a highly fault-tolerant mechanism.

It is noteworthy that an 8-dimension lattice, referred to as the E_8 lattice [31, Chapter 4], can partially fulfill the above requirements. As for its density, there is a remarkable mathematical breakthrough that sphere packing in the E_8 lattice is proved to be optimal in the sense of the best density when packing in $\mathbb{R}^8$ [33]. As for the efficiency on coding, there have been simple executable encoding and decoding algorithms of the E_8 lattice code in [30–32]. However, the known coding algorithms in [30,31] cannot be directly utilized in our schemes. In our context, a prerequisite is to establish a one-to-one mapping from binary strings to the E_8 lattice points for message encoding. While the work [32] defines such a mapping through the selection of a basis, its decoding algorithm implementation is not constant-time, since it is attributed to instances where data flow from secret polynomials into variables, subsequently utilized as lookup indices. In this work, we provide a scalable version of the E_8 lattice in the spirit of [32], as well as a corresponding constant-time implementation of its decoding algorithm, which is capable of transforming the lattice points to the binary strings without involving Gaussian Elimination.

3.1. Scalable E_8 lattice code

The scalable E_8 lattice is constructed from the Extended Hamming Code in dimension 8, which is defined as $H_8 = \{\mathbf{c} \in \{0,1\}^8 \mid \mathbf{c} = \mathbf{zH} \bmod 2, \mathbf{z} \in \{0,1\}^4\}$ where the binary matrix $\mathbf{H}$ is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Let $C = \{(x_1, x_1, x_2, x_2, x_3, x_3, x_4, x_4) \in \{0,1\}^8 \mid \sum x_i \equiv 0 \bmod 2\}$, where C is spanned by the uppermost three rows of $\mathbf{H}$. Then the scalable E_8 lattice is constructed as

$$E_8 = \lambda \cdot [C \cup (C + \mathbf{c})],$$

where $\mathbf{c} = (0,1,0,1,0,1,0,1)$ represents the last row of $\mathbf{H}$, $\lambda \in \mathbb{R}^+$ is the scale factor and $\lambda \cdot C$ denotes the multiplication of all elements in C by λ.

3.1.1. Encoding algorithm

The encoding algorithm of the scalable E_8 lattice code is presented in Algorithm 1, which is to compute $\lambda \cdot (\mathbf{kH} \bmod 2)$, where $\mathbf{k}$ represents a 4-bit binary string.

Algorithm 1 $\mathsf{Encode}_{E_8}(\mathbf{k} \in \{0,1\}^4)$

1: $\mathbf{v} := \lambda \cdot (\mathbf{kH} \bmod 2)$
2: **return** $\mathbf{v}$

Algorithm 2 $\mathsf{Decode}_{E_8}(\mathbf{x} = (x_0, \ldots, x_7) \in \mathbb{R}^8)$

1: Recall that $\mathbf{c} := (0,1,0,1,0,1,0,1)$
2: $(\mathbf{k}_0, \mathsf{TotalCost}_0) := \mathsf{Decode}_{C'}(\mathbf{x})$
3: $(\mathbf{k}_1, \mathsf{TotalCost}_1) := \mathsf{Decode}_{C'}(\mathbf{x} - \lambda \cdot \mathbf{c})$
4: $b := \arg\min\{\mathsf{TotalCost}_0, \mathsf{TotalCost}_1\}$
5: $(k_0, k_1, k_2, k_3) := \mathbf{k}_b$
6: $\mathbf{k} := (k_0, k_1 \oplus k_0, k_3, b) \in \{0,1\}^4$
7: **return** $\mathbf{k}$

Algorithm 3 $\mathsf{Decode}_{C'}(\mathbf{x} \in \mathbb{R}^8)$

1: $mind := +\infty$
2: $mini := 0$
3: $\mathsf{TotalCost} := 0$
4: **for** $i = 0 \ldots 3$ **do**
5: $\quad c_0 := \|x_{2i}\|^2_{2\lambda,2} + \|x_{2i+1}\|^2_{2\lambda,2}$
6: $\quad c_1 := \|x_{2i} - \lambda\|^2_{2\lambda,2} + \|x_{2i+1} - \lambda\|^2_{2\lambda,2}$
7: $\quad k_i := \arg\min\{c_0, c_1\}$
8: $\quad \mathsf{TotalCost} := \mathsf{TotalCost} + c_{k_i}$
9: $\quad$ **if** $c_{1-k_i} - c_{k_i} < mind$ **then**
10: $\quad\quad mind := c_{1-k_i} - c_{k_i}$
11: $\quad\quad mini := i$
12: $\quad$ **end if**
13: **end for**
14: **if** $k_0 + k_1 + k_2 + k_3 \bmod 2 = 1$ **then**
15: $\quad k_{mini} := 1 - k_{mini}$
16: $\quad \mathsf{TotalCost} := \mathsf{TotalCost} + mind$
17: **end if**
18: $\mathbf{k} := (k_0, k_1, k_2, k_3) \in \{0,1\}^4$
19: **return** $(\mathbf{k}, \mathsf{TotalCost})$

3.1.2. Decoding algorithm

Given any $\mathbf{x} \in \mathbb{R}^8$, the decoding algorithm aims to find the solution of the closest vector problem (CVP) of $\mathbf{x}$ in the scalable E_8 lattice, denoted as $\lambda \cdot \mathbf{k}'\mathbf{H} \bmod 2$, and it outputs the 4-bit string $\mathbf{k}'$. Solving the CVP of $\mathbf{x} \in \mathbb{R}^8$ in the scalable E_8 lattice involves addressing the CVP for both $\mathbf{x}$ and $\mathbf{x} - \lambda\mathbf{c}$ in $C' = \lambda \cdot C$. The one with the smaller distance is considered the final answer.

We briefly introduce the idea of solving the CVP in C' as follows. Given $\mathbf{x} \in \mathbb{R}^8$, for every two components in $\mathbf{x}$, determine whether they are close to $(0,0)$ or (λ, λ). Assign the corresponding component of $\mathbf{k}$ to 0 if the former is true, and 1 otherwise. If $\sum k_i \bmod 2 = 0$ holds, it indicates that $\lambda \cdot (k_0, k_0, k_1, k_1, k_2, k_2, k_3, k_3)$ is the solution. However, $\sum k_i \bmod 2$ might be equal to 1. Next, we select the second-closest vector $\lambda \cdot (k'_0, k'_0, k'_1, k'_1, k'_2, k'_2, k'_3, k'_3)$, ensuring that there is at most a one-bit difference between (k_0, k_1, k_2, k_3) and (k'_0, k'_1, k'_2, k'_3). The detailed algorithm is provided in Algorithm 2, with Algorithm 3 serving as its subroutine. Note that in Algorithm 3, *mind* and *mini* are utilized to store the minimal difference of the components and the corresponding index, respectively.

In the final step, $\mathsf{Decode}_{C'}$ in Algorithm 2 will produce the 4-bit string (k_0, k_1, k_2, k_3) such that the lattice point $\lambda \cdot (k_0, k_0 \oplus b, k_1, k_1 \oplus b, k_2, k_2 \oplus b, k_3, k_3 \oplus b)$ is closest to $\mathbf{x}$ in the scalable E_8 lattice. Since the lattice point follows the form of $\lambda \cdot (\mathbf{kH} \bmod 2)$, the decoding result $\mathbf{k}$ can be obtained by tweaking the solution of the CVP in the scalable E_8 lattice, as indicated in line 5 and line 6 in Algorithm 2. Further insights into the constant-time implementation of decoding algorithms are provided in Section 7.6.

3.2. Correct decoding bound

Theorem 1 provides an upper bound on the correctness of decoding with respect to Algorithm 2. In essence, for any 8-dimension vector sufficiently close to the specified scalable E_8 lattice point under the ℓ_2 norm, it can be decoded into the same 4-bit string that generated the lattice point. This theorem proves beneficial when attempting to recover the targeted message from a specified lattice point with error terms in our schemes.

Theorem 1 (*Correct Decoding Bound*). *For any given* $\mathbf{k}_1 \in \{0,1\}^4$, *let* $\mathbf{v}_1 := Encode_{E_8}(\mathbf{k}_1)$. *For any* $\mathbf{v}_2 \in \mathbb{R}^8$, *let* $\mathbf{k}_2 := Decode_{E_8}(\mathbf{v}_2)$. *If* $\|\mathbf{v}_2 - \mathbf{v}_1\|_{2\lambda,2} < \lambda$, *then* $\mathbf{k}_1 = \mathbf{k}_2$.

Proof. According to the construction of the Extended Hamming Code H_8, we know that its minimal Hamming distance is 4. Consequently, the radius of sphere packing in the scalable E_8 lattice is $\frac{1}{2}\sqrt{4\cdot\lambda^2}=\lambda$. As illustrated in Algorithm 1, $\mathbf{v}_1$ is the lattice point generated from $\mathbf{k}_1$. As for $\mathbf{v}_2\in\mathbb{R}^8$, if $\|\mathbf{v}_2-\mathbf{v}_1\|_{2\lambda,2}<\lambda$, the solution of the CVP about $\mathbf{v}_2$ in the scalable E_8 lattice is $\mathbf{v}_1$. Since Decode_{E_8} in Algorithm 2 will output the 4-bit string finally, instead of the intermediate solution of the CVP, $\mathbf{v}_1$ is also generated from $\mathbf{k}_2$, specifically as $\mathbf{v}_1=\lambda\cdot(\mathbf{k}_2\mathbf{H} \bmod 2)$, which implies that $\mathbf{k}_1=\mathbf{k}_2$. □

4. Construction and analysis

In this section, we propose two new schemes based on NTRU lattices, denotes as CTRU and CNTR, each of which consists of an IND-CPA secure public-key encryption and an IND-CCA secure key encapsulation mechanism. While CTRU and CNTR share a similar structure for their public key and secret key with traditional NTRU-based KEM schemes, the method employed for message recovery in CTRU and CNTR significantly diverges from these conventional approaches. Our construction ensures that CTRU and CNTR attain integrated performance when considering security, bandwidth, error probability, and computational efficiency as a whole.

4.1. CTRU: Proposal description

Our PKE scheme CTRU.PKE consists of the following three algorithms (KeyGen, Enc, Dec), which are specified in Algorithm 4–6. Let n and q be the ring parameters of $\mathcal{R}_q$. Let q_2 be the ciphertext modulus, which is usually set to be a power of two that is smaller than q. In this paper, we set p as the modulus for the message space, with a primary focus on the case where $p=2$ for an odd modulus q. Let Ψ_1 and Ψ_2 be the distributions over $\mathcal{R}$. For presentation simplicity, the secret polynomials, f', g, are sampled according to Ψ_1, and r, e are sampled according to Ψ_2. We stress that Ψ_1 and Ψ_2 can be different distributions. Let $\mathcal{M}$ represent the message space, which is, however, fixed to $\mathcal{M}=\{0,1\}^{n/2}$ in this paper, where each $m\in\mathcal{M}$ can be conceptualized as a binary polynomial of dimension $\frac{n}{2}$.

Algorithm 4 CTRU.PKE.KeyGen(1^κ)

1: $f', g \leftarrow \Psi_1$
2: $f := pf'+1$
3: If f is not invertible in $\mathcal{R}_q$, restart.
4: $h := g/f$
5: **return** $(pk := h, sk := f)$

Algorithm 5 CTRU.PKE.Enc($pk=h$, $m\in\mathcal{M}$)

1: $r, e \leftarrow \Psi_2$
2: $\sigma := hr+e$
3: $c := \left\lfloor \frac{q_2}{q}(\sigma + \lfloor \text{PolyEncode}(m) \rceil) \right\rfloor \bmod q_2$
4: **return** c

Algorithm 6 CTRU.PKE.Dec($sk=f$, c)

1: $m := \text{PolyDecode}\left(cf \bmod^{\pm} q_2\right)$
2: **return** m

The PolyEncode algorithm and PolyDecode algorithm are described in Algorithm 7 and 8, respectively. Specifically, we construct the E_8' lattice with the scale factor $\frac{q}{2}$ in Algorithm 7. This implies that the encoding algorithm operates over $E_8' := \frac{q}{2}\cdot[C\cup(C+\mathbf{c})]$. The PolyEncode algorithm partitions each $m\in\mathcal{M}$ into quadruples, each of which will be encoded via $\text{Encode}_{E_8'}$. Concerning the PolyDecode algorithm, the decoding process is performed over the lattice $E_8'' := \frac{q_2}{2}\cdot[C\cup(C+\mathbf{c})]$. It partitions $v\in\mathcal{R}_{q_2}$ into octets, each of which will be decoded via $\text{Decode}_{E_8''}$. The final message m can be reconstructed by amalgamating all the 4-bit binary strings output by $\text{Decode}E_8''$.

Algorithm 7 PolyEncode($m=\sum_{i=0}^{n/2-1} m_i x^i \in \mathcal{M}$)

1: $E_8' := \frac{q}{2}\cdot[C\cup(C+\mathbf{c})] \subset [0,\frac{q}{2}]^8$
2: **for** $i=0\ldots n/8-1$ **do**
3: $\quad \mathbf{k}_i := (m_{4i}, m_{4i+1}, m_{4i+2}, m_{4i+3}) \in \{0,1\}^4$
4: $\quad (v_{8i}, v_{8i+1}, \ldots, v_{8i+7}) := \text{Encode}_{E_8'}(\mathbf{k}_i) \in [0,\frac{q}{2}]^8$
5: **end for**
6: $v := \sum_{i=0}^{n-1} v_i x^i$
7: **return** v

Algorithm 8 PolyDecode($v=\sum_{i=0}^{n-1} v_i x^i \in \mathcal{R}_{q_2}$)

1: $E_8'' := \frac{q_2}{2}\cdot[C\cup(C+\mathbf{c})] \subset [0,\frac{q_2}{2}]^8$
2: **for** $i=0\ldots n/8-1$ **do**
3: $\quad \mathbf{x}_i := (v_{8i}, v_{8i+1}, \ldots, v_{8i+7}) \in \mathbb{R}^8$
4: $\quad (m_{4i}, m_{4i+1}, m_{4i+2}, m_{4i+3}) := \text{Decode}_{E_8''}(\mathbf{x}_i) \in \{0,1\}^4$
5: **end for**
6: $m := \sum_{i=0}^{n/2-1} m_i x^i \in \mathcal{M}$
7: **return** m

Our KEM scheme CTRU.KEM = (Keygen, Encaps, Decaps) is constructed by applying $\text{FO}^{\not\perp}_{ID(pk),m}$, a variant of the Fujisaki-Okamoto (FO) transformation [43,44] designed to enhance IND-CCA security in a multi-user setting [34]. Let ι, γ be positive integers. We prefer to choose $\iota, \gamma \geq 256$ for strong security. Let $\mathcal{H}: \{0,1\}^* \to \mathcal{K}\times\mathcal{COINS}$ be a hash function, where $\mathcal{K}$ represents the shared key space of CTRU.KEM and $\mathcal{COINS}$ is the randomness space of CTRU.PKE.Enc. It is noteworthy that we explicitly specify the randomness in CTRU.PKE.Enc here. Define $\mathcal{H}_1(\cdot)$ as $\mathcal{H}(\cdot)$'s partial output that is mapped into $\mathcal{K}$. Let $\mathcal{PK}$ be the public key space of CTRU.PKE. Let $ID: \mathcal{PK}\to\{0,1\}^\gamma$ be a fixed-output length function. The algorithms of CTRU.KEM are described in Algorithm 9–11.

Algorithm 9 CTRU.KEM.KeyGen(1^κ)

1: $(pk, sk) \leftarrow \text{CTRU.PKE.KeyGen}(1^\kappa)$
2: $z \xleftarrow{\$} \{0,1\}^\iota$
3: **return** $(pk' := pk, sk' := (sk, z))$

Algorithm 10 CTRU.KEM.Encaps(pk)

1: $m \xleftarrow{\$} \mathcal{M}$
2: $(K, coin) := \mathcal{H}(ID(pk), m)$
3: $c := \text{CTRU.PKE.Enc}(pk, m; coin)$
4: **return** (c, K)

Algorithm 11 CTRU.KEM.Decaps($(sk, z), c$)

1: $m' := \text{CTRU.PKE.Dec}(sk, c)$
2: $(K', coin') := \mathcal{H}(ID(pk), m')$
3: $\tilde{K} := \mathcal{H}_1(ID(pk), z, c)$
4: **if** $m' \neq \perp$ and $c = \text{CTRU.PKE.Enc}(pk, m'; coin')$ **then**
5: $\quad$ **return** K'
6: **else**
7: $\quad$ **return** $\tilde{K}$
8: **end if**

4.2. CNTR: Proposal description

CNTR is a streamlined variant of CTRU, which is based on the NTRU assumption [7] and RLWR assumption [3]. In this context, CNTR is also commonly the abbreviation of "container", reflecting its character as an economically concise yet powerful key encapsulation mechanism.

Our PKE scheme CNTR.PKE is specified in Algorithm 12–14. The PolyEncode algorithm and PolyDecode algorithm are the same as Algorithm 7 and 8, respectively. The symbols and definitions used in the following are consistent with those used in CTRU.

Algorithm 12 CNTR.PKE.KeyGen(1^κ)

1: $f', g \leftarrow \Psi_1$
2: $f := pf' + 1$
3: If f is not invertible in $\mathcal{R}_q$, restart.
4: $h := g/f$
5: **return** $(pk := h, sk := f)$

Algorithm 13 CNTR.PKE.Enc($pk = h$, $m \in \mathcal{M}$)

1: $r \leftarrow \Psi_2$
2: $\sigma := hr$
3: $c := \left\lfloor \frac{q_2}{q}(\sigma + \text{PolyEncode}(m)) \right\rceil \bmod q_2$
4: **return** c

Algorithm 14 CNTR.PKE.Dec($sk = f$, c)

1: $m := \text{PolyDecode}\left(cf \bmod^{\pm} q_2\right)$
2: **return** m

In contrast to the encryption algorithm of CTRU (refer to Algorithm 5), that of CNTR exhibits the following distinctions: (1) the elimination of the noise polynomial; (2) the cancellation of the rounding step beyond the PolyEncode algorithm.

Our KEM scheme CNTR.KEM is constructed via the same approach as CTRU.KEM, i.e., applying $\text{FO}^{\not\perp}_{ID(pk),m}$[34]. The algorithms of CNTR.KEM can be referred to in Algorithm 9–11.

4.3. Correctness analysis

Lemma 1. *It holds that* $cf \bmod^{\pm} q_2 = \frac{q_2}{q}((\frac{q}{q_2}c)f \bmod^{\pm} q)$.

Proof. Since polynomial multiplication can be described as matrix–vector multiplication, which keeps the linearity, it holds that $(\frac{q}{q_2}c)f = \frac{q}{q_2}(cf)$. There exists an integral vector $\theta \in \mathbb{Z}^n$ such that $\frac{q}{q_2}cf \bmod^{\pm} q = \frac{q}{q_2}cf + q\theta$ where each component of $\frac{q}{q_2}cf + q\theta$ is in $[-\frac{q}{2}, \frac{q}{2})$. Thus, each component of $cf + q_2\theta$ is in $[-\frac{q_2}{2}, \frac{q_2}{2})$. Hence, we obtain

$$cf \bmod^{\pm} q_2 = cf + q_2\theta = \frac{q_2}{q}(\frac{q}{q_2}cf + q\theta) = \frac{q_2}{q}((\frac{q}{q_2}c)f \bmod^{\pm} q). \quad \square$$

Theorem 2 (*Correctness of CTRU*). *Let Ψ_1 and Ψ_2 be the distributions over the ring $\mathcal{R}$, and q, q_2 be positive integers. Let $f', g \leftarrow \Psi_1$ and $r, e \leftarrow \Psi_2$. Let $\varepsilon \leftarrow \chi$, where χ is the distribution over $\mathcal{R}$ defined as follows: Sample $u \xleftarrow{\$} \mathcal{R}_q$ and output $\left\lfloor \frac{q_2}{q}u \right\rceil - \frac{q_2}{q}u \bmod^{\pm} q_2$. Let Err_i be the ith octet of $gr + ef + \vec{1} \cdot f' + \frac{q}{q_2}\varepsilon f$, where $\vec{1}$ is the polynomial with all ones. Denote $1 - \delta = \Pr\left[\|Err_i\|_{q,2} < \frac{q}{2} - \sqrt{2}, \forall i\right]$. Then, the error probability of CTRU is δ.*

Proof. Scale the E_8'' lattice and $cf \bmod^{\pm} q_2$ by the factor q/q_2. According to Lemma 1, we obtain

$$m = \text{PolyDecode}_{E_8''}\left(cf \bmod^{\pm} q_2\right) = \text{PolyDecode}_{E_8'}\left((\frac{q}{q_2}c)f \bmod^{\pm} q\right) \quad (1)$$

in Algorithm 6. For any $m \in \mathcal{M}$, the result of PolyEncode(m) in Algorithm 5 can be denoted by $\frac{q}{2}s$ where $s \in \mathcal{R}_2$. Based on the hardness of the NTRU assumption and RLWE assumption, σ in line 2 in Algorithm 5 is pseudo-random in $\mathcal{R}_q$, so is $\sigma + \lfloor \frac{q}{2}s \rceil$ for any given $s \in \mathcal{R}_2$. We mainly consider the case of odd q, since an even q leads to a simpler proof due to $\lfloor \frac{q}{2}s \rceil = \frac{q}{2}s$. Therefore, the value of c in line 3 in Algorithm 5 is

$$c = \left\lfloor \frac{q_2}{q}(\sigma + \lfloor \frac{q}{2}s \rceil) \right\rceil \bmod q_2 = \frac{q_2}{q}(\sigma + \frac{q+1}{2}s) + \varepsilon \bmod q_2.$$

Along with $\sigma = hr + e$, $h = g/f$ and $f = 2f' + 1$, for the formula (1) we get

$$\begin{aligned}(\frac{q}{q_2}c)f \bmod^{\pm} q &= \frac{q}{q_2}[\frac{q_2}{q}(\sigma + \frac{q+1}{2}s) + \varepsilon] \cdot f \bmod^{\pm} q \\ &= \frac{q+1}{2}s(2f'+1) + \sigma f + \frac{q}{q_2}\varepsilon f \bmod^{\pm} q \\ &= \frac{q}{2}s + gr + ef + sf' + \frac{s}{2} + \frac{q}{q_2}\varepsilon f \bmod^{\pm} q.\end{aligned} \quad (2)$$

Each octet of $\frac{q}{2}s$ in (2) is essentially a lattice point in the E_8' lattice, which we denoted by $\frac{q}{2}(\mathbf{k}_i\mathbf{H} \bmod 2)$. Denote the ith octet of the polynomial X by $(X)_i$. From Theorem 1 we know that to recover $\mathbf{k}_i$, one could hold the probability condition $\|(gr + ef + sf' + \frac{s}{2} + \frac{q}{q_2}\varepsilon f)_i\|_{q,2} < \frac{q}{2}$ which can be reflected by the condition $\|(gr + ef + \vec{1} \cdot f' + \frac{q}{q_2}\varepsilon f)_i\|_{q,2} + \sqrt{2} < \frac{q}{2}$, since $\vec{1} \cdot f'$ has a wider distribution than $s \cdot f'$ and $\|(\frac{s}{2})_i\|_{q,2} \le \sqrt{2}$ for any $s \in \mathcal{R}_2$. Similarly, for an even q, it can be simplified to the condition $\|(gr + ef + \frac{q}{q_2}\varepsilon f)_i\|_{q,2} < \frac{q}{2}$ directly which can be implied by the inequality of the case of odd q. Therefore, we consider the bound of the case of odd q as a general bound. $\square$

Theorem 3 (*Correctness of CNTR*). *Let Ψ_1 and Ψ_2 be the distributions over the ring $\mathcal{R}$, and q, q_2 be positive integers. q_2 is an even number that is smaller than q. Let $f', g \leftarrow \Psi_1$ and $r \leftarrow \Psi_2$. Let $\varepsilon \leftarrow \chi$, where χ is the distribution over $\mathcal{R}$ defined as follows: Sample $h \xleftarrow{\$} \mathcal{R}_q$ and $r \leftarrow \Psi_2$, and output $\left(\left\lfloor \frac{q_2}{q}hr \right\rceil - \frac{q_2}{q}hr\right) \bmod^{\pm} q_2$. Let Err_i be the ith octet of $gr + \frac{q}{q_2}\varepsilon f$. Denote $1 - \delta = \Pr\left[\|Err_i\|_{q,2} < \frac{q}{2}, \forall i\right]$. Then, the error probability of CNTR is δ.*

Proof. The main observation is that the computation of the ciphertext c is equivalent to

$$\begin{aligned}c &= \left\lfloor \frac{q_2}{q}(\sigma + \text{PolyEncode}(m)) \right\rceil \bmod q_2 = \left\lfloor \frac{q_2}{q}hr + \frac{q_2}{q} \cdot \frac{q}{2}s \right\rceil \bmod q_2 \\ &= \left\lfloor \frac{q_2}{q}hr \right\rceil + \frac{q_2}{2}s \bmod q_2 = \frac{q_2}{q}hr + \varepsilon + \frac{q_2}{2}s \bmod q_2\end{aligned} \quad (3)$$

for even $q_2 < q$, where $s \in \mathcal{R}_2$. Based on the hardness of the NTRU assumption, h is pseudo-random in $\mathcal{R}_q$. The term $\lfloor \frac{q_2}{q}hr \rceil$ represents an RLWR sample, and the term $\frac{q_2}{2}s$ implies the encoding output of m via the scalable E_8 lattice w.r.t. the scale factor $\frac{q_2}{2}$. Similarly, we have

$$m = \text{PolyDecode}_{E_8''}\left(cf \bmod^{\pm} q_2\right) = \text{PolyDecode}_{E_8'}\left((\frac{q}{q_2}c)f \bmod^{\pm} q\right),$$

thereby $(\frac{q}{q_2}c)f \bmod^{\pm} q = \frac{q}{2}s + gr + \frac{q}{q_2}\varepsilon f \bmod^{\pm} q$. Each octet of $\frac{q}{2}s$ is essentially a lattice point in the scalable E_8 lattice w.r.t. the scale factor $\frac{q}{2}$, denoted as $\frac{q}{2}(\mathbf{k}_i\mathbf{H} \bmod 2)$. From Theorem 1 it is known that to recover $\mathbf{k}_i$, it should hold $\|\text{Err}_i\|_{q,2} < \frac{q}{2}$, where Err_i is the ith octet of $gr + \frac{q}{q_2}\varepsilon f$. $\square$

4.4. More accurate form of polynomial product and error probability analysis

As previously described in [15], the general form of the polynomial product of $f = \sum_{i=0}^{n-1} f_i x^i$ and $g = \sum_{i=0}^{n-1} g_i x^i$ in the ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ is expressed through matrix–vector multiplication. However,

the work [15] bounds the error probability by considering the worst-case setting, which involves the sums of $\frac{3}{2}n$ terms of the form $f_i g_j$ for each coefficient of h. This approach results in a highly conservative estimation. In the subsequent discussion, we aim to accurately derive the number of terms in the polynomial product coefficient, and seek to improve the original error probability analysis presented in [15], moving away from a rough consideration of the worst case involving $\frac{3}{2}n$ terms. Firstly, focusing on the arithmetic operations in $\mathbb{Z}_q$, the product of f and g in $\mathbb{Z}_q[x]$ is expressed as

$$\sum_{\substack{i+j=k,\\ 0\le k\le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k,\\ n\le k\le 2n-2}} f_i g_j x^k.$$

To obtain the result in $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, we consider the second summation $\sum_{\substack{i+j=k,\\ n\le k\le 2n-2}} f_i g_j x^k$, and obtain

$$\begin{aligned}\sum_{\substack{i+j=k,\\ n\le k\le 2n-2}} f_i g_j x^k &= \sum_{\substack{i+j=k+n,\\ 0\le k\le n-2}} f_i g_j x^k (x^{\frac{n}{2}} - 1)\\ &= \sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le \frac{3n}{2}-2}} f_i g_j x^k - \sum_{\substack{i+j=k+n,\\ 0\le k\le n-2}} f_i g_j x^k.\end{aligned} \tag{4}$$

Then, we continue to consider the expression $\sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le \frac{3n}{2}-2}} f_i g_j x^k$ and get

$$\begin{aligned}&\sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le \frac{3n}{2}-2}} f_i g_j x^k = \sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+\frac{n}{2},\\ n\le k\le \frac{3n}{2}-2}} f_i g_j x^k\\ &= \sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+\frac{3n}{2},\\ 0\le k\le \frac{n}{2}-2}} f_i g_j x^k (x^{\frac{n}{2}} - 1)\\ &= \sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+n,\\ \frac{n}{2}\le k\le n-2}} f_i g_j x^k - \sum_{\substack{i+j=k+\frac{3n}{2},\\ 0\le k\le \frac{n}{2}-2}} f_i g_j x^k.\end{aligned} \tag{5}$$

Therefore, for each coefficient h_k, there will be

$$\begin{aligned}h_k = &\sum_{\substack{i+j=k,\\ 0\le k\le n-1}} f_i g_j - \sum_{\substack{i+j=k+n,\\ 0\le k\le n-2}} f_i g_j + \sum_{\substack{i+j=k+\frac{n}{2},\\ \frac{n}{2}\le k\le n-1}} f_i g_j + \sum_{\substack{i+j=k+n,\\ \frac{n}{2}\le k\le n-2}} f_i g_j\\ &- \sum_{\substack{i+j=k+\frac{3n}{2},\\ 0\le k\le \frac{n}{2}-2}} f_i g_j.\end{aligned} \tag{6}$$

Concretely, for $0 \le k \le \frac{n}{2} - 2$, we have

$$h_k = \sum_{i+j=k} f_i g_j - \sum_{i+j=k+n} f_i g_j - \sum_{i+j=k+\frac{3n}{2}} f_i g_j.$$

For $k = \frac{n}{2} - 1$, we have

$$h_k = \sum_{i+j=k} f_i g_j - \sum_{i+j=k+n} f_i g_j.$$

For $\frac{n}{2} \le k \le n - 1$, we have

$$h_k = \sum_{i+j=k} f_i g_j + \sum_{i+j=k+\frac{n}{2}} f_i g_j.$$

According to the symmetry of the distribution, we only focus on the number of terms of the form $f_i g_j$, listed as follows: For $0 \le k \le \frac{n}{2} - 1$, h_k consists of $\frac{3n}{2} - k - 1$ terms of the form $f_i g_j$; For $\frac{n}{2} \le k \le n - 1$, h_k consists of $\frac{3n}{2}$ terms of the form $f_i g_j$. Leveraging the results mentioned above, the exact number of terms of the polynomial product coefficient is obtained, which enables the computation of more accurate error probabilities for our schemes, as well as NTTRU and NTRU-C^{768}_{3457}. Consequently, the error probabilities of CTRU, CNTR, NTTRU and NTRU-C^{768}_{3457} in this work are estimated by using a Python script according to our methodology. The specific results of CTRU and CNTR for the selected parameters are presented in Tables 2 and 3, respectively.

4.5. Provable security

The following theorems establish the IND-CPA security of CTRU.PKE under the NTRU assumption and RLWE assumption, while affirming the IND-CPA security of CNTR.PKE under the NTRU assumption and RLWR assumption.

Theorem 4 (*IND-CPA Security of CTRU.PKE*)**.** *For any adversary* A*, there exist adversaries* B *and* C *such that* $\mathbf{Adv}^{IND\text{-}CPA}_{CTRU.PKE}(\mathsf{A}) \le \mathbf{Adv}^{NTRU}_{\mathcal{R}_q,\Psi_1}(\mathsf{B}) + \mathbf{Adv}^{RLWE}_{\mathcal{R}_q,\Psi_2}(\mathsf{C})$.

Proof. We complete our proof through a sequence of games $\mathbf{G}_0$, $\mathbf{G}_1$ and $\mathbf{G}_2$. Let A be the adversary against the IND-CPA security experiment. Denote by $\mathbf{Succ}_i$ the event that A wins in the game $\mathbf{G}_i$, that is, A outputs b' such that $b' = b$ in $\mathbf{G}_i$.

Game $\mathbf{G}_0$. This game is the original IND-CPA security experiment. Thus, $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTRU.PKE}}(\mathsf{A}) = |\Pr[\mathbf{Succ}_0] - 1/2|$.

Game $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$, except that replacing the public key $h = g/f$ in the key generation by $h \xleftarrow{\$} \mathcal{R}_q$. To distinguish $\mathbf{G}_1$ from $\mathbf{G}_0$ is equivalent to solve an NTRU problem. More precisely, there exists an adversary B with the same running time as that of A such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \le \mathbf{Adv}^{\text{NTRU}}_{\mathcal{R}_q,\Psi_1}(\mathsf{B})$.

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$, except that using uniformly random elements from $\mathcal{R}_q$ to replace σ in the encryption. Similarly, there exists an adversary C with the same running time as that of A such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| \le \mathbf{Adv}^{\text{RLWE}}_{\mathcal{R}_q,\Psi_2}(\mathsf{C})$.

In Game $\mathbf{G}_2$, for any given m_b, according to Algorithm 5 and 7, m_b is split into $n/8$ quadruples. Denote the ith quadruple of m_b as $m_b^{(i)}$, which will later be operated to output the ith octet of the ciphertext c that is denoted as $c^{(i)}$, $i = 0, 1, \ldots, n/8 - 1$. Since $c^{(i)}$ is only dependent on $m_b^{(i)}$ and other parts of m_b do not interfere with $c^{(i)}$, our aim is to prove that $c^{(i)}$ is independent of $m_b^{(i)}$, $i = 0, 1, \ldots, n/8 - 1$. For any i and any given $m_b^{(i)}$, $\lfloor \text{Encode}_{E'_8}(m_b^{(i)}) \rceil$ is fixed. Based on the uniform randomness of σ in $\mathcal{R}_q$, its ith octet (denoted as $\sigma^{(i)}$) is uniformly random in $\mathbb{Z}_q^8$, so is $\sigma^{(i)} + \lfloor \text{Encode}_{E'_8}(m_b^{(i)}) \rceil$. Therefore, the resulting $c^{(i)}$ is subject to the distribution $\lfloor \frac{q_2}{q} u \rceil \bmod q_2$ where u is uniformly random in $\mathbb{Z}_q^8$, which implies that $c^{(i)}$ is independent of $m_b^{(i)}$. Hence, each $c^{(i)}$ leaks no information of the corresponding $m_b^{(i)}$, $i = 0, 1, \ldots, n/8 - 1$. We have $\Pr[\mathbf{Succ}_2] = 1/2$.

Combining all the probabilities finishes the proof. □

Theorem 5 (*IND-CPA Security of CNTR.PKE*)**.** *For any adversary* A*, there exist adversaries* B *and* C *such that* $\mathbf{Adv}^{IND\text{-}CPA}_{CNTR.PKE}(\mathsf{A}) \le \mathbf{Adv}^{NTRU}_{\mathcal{R}_q,\Psi_1}(\mathsf{B}) + \mathbf{Adv}^{RLWR}_{\mathcal{R},\Psi_2}(\mathsf{C})$.

Proof. We complete our proof through a sequence of games $\mathbf{G}_0$, $\mathbf{G}_1$ and $\mathbf{G}_2$. Let A be the adversary against the IND-CPA security experiment. Denote by $\mathbf{Succ}_i$ the event that A wins in the game $\mathbf{G}_i$, that is, A outputs b' such that $b' = b$ in $\mathbf{G}_i$.

Game $\mathbf{G}_0$. This game is the original IND-CPA security experiment. Thus, $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CNTR.PKE}}(\mathsf{A}) = |\Pr[\mathbf{Succ}_0] - 1/2|$.

Game $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$, except that replacing the public key $h = g/f$ in the key generation by $h \xleftarrow{\$} \mathcal{R}_q$. To distinguish $\mathbf{G}_1$ from $\mathbf{G}_0$ is equivalent to solve an NTRU problem. More precisely, there exists an adversary B with the same running time as that of A such that $|\Pr[\mathbf{Succ}_0] - \Pr[\mathbf{Succ}_1]| \le \mathbf{Adv}^{\text{NTRU}}_{\mathcal{R}_q,\Psi_1}(\mathsf{B})$.

Game $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$, except that using random elements from $\mathcal{R}_{q_2}$ to replace $\lfloor \frac{q_2}{q} hr \rceil$ of $c = \lfloor \frac{q_2}{q} hr \rceil + \frac{q_2}{2} s \bmod q_2$ (see the formula (3)) in the encryption where the term $\frac{q_2}{2} s$ represents the encoding output of the given challenge plaintext m_b via the scalable E_8 lattice w.r.t. the scale factor $\frac{q_2}{2}$. Similarly, there exists an adversary C with the same running time as that of A such that $|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_2]| \le \mathbf{Adv}^{\text{RLWR}}_{\mathcal{R},\Psi_2}(\mathsf{C})$.

In Game $\mathbf{G}_2$, the information of the challenge plaintext m_b is perfectly hidden by the uniformly random element from $\mathcal{R}_{q_2}$. Hence, the advantage of the adversary is zero in $\mathbf{G}_2$. We have $\Pr[\mathbf{Succ}_2] = 1/2$.

Combining all the probabilities finishes the proof. □

Since CTRU.KEM and CNTR.KEM are constructed by applying $\mathrm{FO}^{\not\perp}_{ID(pk),m}$ [34], according to the studies in [34] we obtain the following security theorem concerning CCA security of CTRU.KEM and CNTR.KEM in the random oracle model (ROM) [45] and the quantum random oracle model (QROM) [46].

Theorem 6 (*IND-CCA Security in the (Q)ROM [34]*). *Let ℓ be the min-entropy of $ID(pk)$, i.e., $\ell = H_\infty(ID(pk))$, where $(pk, sk) \leftarrow$CTRU/CNTR.PKE.KeyGen. For any (quantum) adversary A, making at most q_D decapsulation queries, q_H (Q)RO queries, against the IND-CCA security of CTRU/CNTR.KEM, there exists a (quantum) adversary B with roughly the same running time of A, such that:*

- *in the ROM, it holds that* $\mathbf{Adv}^{\text{IND-CCA}}_{\text{CTRU/CNTR.KEM}}(\mathsf{A}) \le$
$$2\left(\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTRU/CNTR.PKE}}(\mathsf{B}) + \frac{q_H+1}{|\mathcal{M}|}\right) + \frac{q_H}{2^t} + (q_H+q_D)\delta + \frac{1}{2^\ell};$$
- *in the QROM, it holds that* $\mathbf{Adv}^{\text{IND-CCA}}_{\text{CTRU/CNTR.KEM}}(\mathsf{A}) \le$
$$2\sqrt{q_{HD}\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTRU/CNTR.PKE}}(\mathsf{B})} + \frac{4q_{HD}}{\sqrt{|\mathcal{M}|}} + \frac{4(q_H+1)}{\sqrt{2^t}} + 16q_{HD}^2\delta + \frac{1}{|\mathcal{M}|} + \frac{1}{2^\ell}, \quad (7)$$
where $q_{HD} := q_H + q_D + 1$.

4.6. Discussions and comparisons

The rings. As specified in [15,29], our underlying algebraic structure is chosen to be the trinomial cyclotomic ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ where $n = 3^l \cdot 2^e$ and q is prime. Utilizing this type of cyclotomic ring facilitates very fast NTT-based polynomial multiplication when the ring moduli are chosen to be NTT-friendly. Furthermore, it also offers very flexible parameter selection, as there exist numerous integral values of n in the form $3^l \cdot 2^e$, $l \ge 0$, $e \ge 1$.

The message modulus. Note that the modulus p is removed in both the public key h (i.e., $h = g/f$) and the ciphertext c in our CTRU and CNTR, for the reason that p is unnecessary in h and c for the recovery of the message m in our construction. The only reserved position of p is in the secret key f, expressed as $f = pf' + 1$. Since $\gcd(q, p) = 1$ is required for NTRU-based KEM schemes, we can use a smaller modulus $p = 2$ instead of $p = 3$. The use of a smaller p offers the advantage of reducing the error probability. It is worth noting that for other NTRU-based KEM schemes with power-of-two modulus q, as in NTRU-HRSS [12], the modulus p is set to 3 due to its status as the smallest integer coprime to the power-of-2 modulus.

The decryption mechanism. Technically speaking, the ciphertexts of NTRU-based PKE schemes [7,12,26,29] take the form of $c = phr + m \bmod q$. One can recover the message m through a unidimensional error-correction mechanism, after computing $cf \bmod q$. Instead, our approach utilizes a multi-dimension coding mechanism. We encode each 4-bit message into a lattice point in the scalable E_8 lattice. These messages can be recovered correctly with the aid of the scalable E_8 lattice decoding algorithm if the ℓ_2 norm of the error term is less than the sphere radius of the scalable E_8 lattice.

The ciphertext compression. The step of compressing ciphertext in CTRU is described in line 3 in Algorithm 5, while that in CNTR is described in line 3 in Algorithm 13. Both of them can be mathematically written as $y := \lfloor \frac{q_2}{q} x \rceil \bmod q_2, x \in [0, q)$ for each component. The sufficient condition for ciphertext compression is $\lceil \log(q_2) \rceil < \lceil \log(q) \rceil$, ensuring that y requires fewer bits than x. However, the capacity of ciphertext compression becomes invalid when $\lceil \log(q_2) \rceil \ge \lceil \log(q) \rceil$, particularly when $q_2 = q$. To the best of our knowledge, CTRU and CNTR are the first NTRU-based KEM schemes featuring scalable ciphertext compression via a single polynomial. The ciphertext modulus q_2 is adjustable based on the bits to be dropped. Recall that most NTRU-based KEM schemes [7,12,15,29] fail to compress ciphertext due to the fact that the message information would be destroyed once the ciphertext is compressed.

The role of noise polynomial and rounding. We remark that the noise polynomial e in line 2 in Algorithm 5 is only necessary for establish the IND-CPA security on the RLWE assumption for CTRU.PKE. Even in the absence of e, CTRU.PKE is still IND-CPA secure under the NTRU assumption and RLWR assumption if the ciphertext compression exists. In essence, CTRU.PKE degenerates into tweaking CNTR.PKE with the existing of the rounding of PolyEncode. Additionally, in the absence of both noise e and ciphertext compression, CTRU.PKE is still OW-CPA secure only based on the NTRU assumption (without further relying on the RLWE or RLWR assumption). Regarding CNTR.PKE, once eliminating the ciphertext compression, CNTR.PKE requires the rounding of PolyEncode, enabling CNTR.PKE to maintain OW-CPA security similarly based on the NTRU assumption.

The trade-off of using lattice code. The implementation of the scalable E_8 lattice code is both efficient and constant-time, as elaborated in Sections 3 and 7.6. The error-correction capability of the scalable E_8 lattice code ensures a sufficiently low error probability, enabling a wider noise distribution and scalable ciphertext compression, such that the security of our schemes can be strengthened by about 40 bits and the ciphertext size is reduced by 15% at least, with the mostly the same (even faster) overall running time when compared to other NTRU-based KEM schemes. Although the scalable E_8 lattice code introduces a slight increase in implementation complexity, this trade-off between security, ciphertext size, and implementation complexity is likely reasonable and acceptable in many contexts.

5. Concrete hardness and parameter selection

In this section, we first estimate and select parameters for CTRU and CNTR through the application of the core-SVP hardness methodology [47]. Subsequently, we present the refined gate-count estimate using the scripts provided by Kyber and NTRU Prime in NIST PQC Round 3. Finally, we provide an overview and discussion of recent attacks beyond the core SVP hardness.

5.1. Parameter selection with core-SVP

5.1.1. Primal attack and dual attack

Currently, for the parameters selected for most practical lattice-based cryptosystems, the dominant attacks considered are the lattice-based primal and dual attacks. The primal attack is to solve the *unique-Short Vector Problem* (u-SVP) in the lattice by constructing an integer *embedding lattice* (Kannan embedding [48], Bai–Galbraith embedding [49], etc.). The most common lattice reduction algorithm is the BKZ algorithm [50,51]. Given a lattice basis, the *blocksize*, which we denote by b, is necessarily chosen to recover the short vector while running the BKZ algorithm. NTRU problem can be treated as a u-SVP instance in the NTRU lattice [9], while a u-SVP instance can also be constructed from the LWE problem. The dual attack [52] is to solve the *decisional* LWE problem, consisting of using the BKZ algorithm in the dual lattice, so as to recover part of the secret and infer the final secret vector.

5.1.2. Core-SVP hardness

Following the simple and conservative methodology of the core-SVP hardness developed from [47], the best known cost of running SVP solver on b-dimension sublattice is $2^{0.292b}$ (resp., $2^{0.265b}$) for the classical (resp., quantum) case. These cost models can be used for conservative estimates of the security of our schemes. We estimate the classical and quantum core-SVP hardness security of CTRU and CNTR via the Python scripts from [35,47]. The concrete results are included in Tables 2 and 3.

Table 2
Parameter sets of CTRU.

Schemes	n	q	q_2	(Ψ_1, Ψ_2)	$\|pk\|$	$\|ct\|$	B.W.	NTRU (Sec.C, Sec.Q)	LWE, primal (Sec.C, Sec.Q)	LWE, dual (Sec.C, Sec.Q)	δ
CTRU-512	512	3457	2^9	(B_2, B_2)	768	576	1344	(111,100)	(111,100)	(110,100)	2^{-122}
	512*	3457	2^{10}	(B_3, B_3)	768	640	1408	(118,107)	(118,107)	(117,106)	2^{-143}
CTRU-768	768*	3457	2^{10}	(B_2, B_2)	1152	960	2112	(181,164)	(181,164)	(180,163)	2^{-184}
	768	3457	3457	(B_3, B_3)	1152	1152	2304	(192,174)	(192,174)	(190,173)	2^{-136}
	768	3457	2^{11}	(B_3, B_3)	1152	1056	2208	(192,174)	(192,174)	(190,173)	2^{-121}
CTRU-1024	1024*	3457	2^{11}	(B_2, B_2)	1536	1408	2944	(255,231)	(255,231)	(252,229)	2^{-195}
	1024	3457	2^{10}	(B_2, B_2)	1536	1280	2816	(255,231)	(255,231)	(252,229)	2^{-132}
	1024	3457	3457	(B_3, B_3)	1536	1536	3072	(269,244)	(269,244)	(266,241)	2^{-96}

Table 3
Parameter sets of CNTR.

Schemes	n	q	q_2	(Ψ_1, Ψ_2)	$\|pk\|$	$\|ct\|$	B.W.	NTRU (Sec.C, Sec.Q)	RLWR (Sec.C, Sec.Q)	δ
CNTR-512	512	3457	2^9	(B_3, B_3)	768	576	1344	(118,107)	(117,106)	2^{-99}
	512*	3457	2^{10}	(B_5, B_5)	768	640	1408	(127,115)	(127,115)	2^{-170}
	512	3457	2^{10}	(B_6, B_6)	768	640	1408	(131,119)	(131,119)	2^{-126}
CNTR-768	768*	3457	2^{10}	(B_3, B_3)	1152	960	2112	(192,174)	(191,173)	2^{-230}
	768	3457	2^{10}	(B_4, B_4)	1152	960	2112	(200,181)	(199,181)	2^{-151}
CNTR-1024	1024*	3457	2^{10}	(B_2, B_2)	1536	1280	2816	(255,231)	(253,230)	2^{-291}
	1024	3457	2^{10}	(B_3, B_3)	1536	1280	2816	(269,244)	(267,243)	2^{-167}

5.1.3. Parameter sets

The parameter sets of CTRU and CNTR are given in Tables 2 and 3 respectively, where those marked with "*" are the recommended parameters also given in Table 1. Though the parameters marked with "*" are recommended, we believe the other parameter sets are still very useful in certain application scenarios. Note that in Table 1 we did not list the security against the LWE dual attack. The reason is that the LWE dual attack was considered less realistic than the primal attack, and was not taken for concrete hardness estimates in many lattice-based cryptosystems including Kyber in NIST PQC Round 3 [35]. For ease of a fair comparison, the security estimate against the LWE dual attack was not listed in Table 1.

The ring dimension n is chosen from $\{512, 768, 1024\}$, corresponding to the targeted security levels I, III and V recommended by NIST. The ring modulus q is set to 3457, and q_2 is the ciphertext modulus (also the RLWR modulus for CNTR). Recall that p has been set to be the message space modulus $p = 2$ and the underlying cyclotomic polynomial $\Phi(x) = x^n - x^{n/2} + 1$, which are omitted in Tables 2 and 3. Ψ_1 and Ψ_2 are the probability distributions which are set to be B_η, where B_η is the centered binomial distribution w.r.t. the integer η. The public key size $|pk|$, ciphertext size $|ct|$ and B.W. (bandwidth, $|pk|+|ct|$) are measured in terms of bytes. "Sec.C" and "Sec.Q" represent the estimated security level expressed in bits in the classical and quantum settings respectively, where the types of NTRU attack, LWE primal attack, LWE dual attack, and RLWR attack are considered. The last column "δ" indicates the error probability, which is evaluated by a script according to the analysis given in Section 4.3.

It is essential to emphasize that our schemes offer flexibility in parameter selection. However, the choice of the parameter n presented in Tables 2 and 3 is for the sake of simplicity. Alternatively, one can choose n from the set $\{576, 648, 864, 972, 1152, 1296\}$, where these values are integers in the form of $3^l \cdot 2^e$, $l \geq 0$, $e \geq 1$. It is worth noting that the plaintext message space for CTRU and CNTR is $\{0,1\}^{n/2}$, in contrast to the fixed message space of $\{0,1\}^{256}$ for Kyber and Saber. Therefore, CTRU and CNTR theoretically admit more flexible key sizes to be encapsulated, i.e., $n/2$-bit shared keys, whereas Kyber and Saber can only encapsulate fixed 256-bit shared keys. But we fix the key sizes of CTRU and CNTR to be 256 bits in this paper for convenience. Regarding CNTR-512, its first parameter set has the smallest ciphertext size, and the third parameter set demonstrates the strongest hardness of lattice problem (saying, NTRU and RLWR). These parameter sets find practical applicability in certain scenarios where sensitivity to error probability is not a predominant concern. In real-world usage, each secret key is anticipated to be employed for decryption no more than 2^{80} times throughout its lifetime. In such cases, the relatively higher error probabilities, for instance, 2^{-99}, do not compromise the actual security of these parameter sets. The third parameter set of CNTR-512 is also recommended due to its robust security, since the recent improvements on the attacks [53–55] might cause the worry that other lattice-based schemes like Kyber and Saber do not achieve the claimed security goals, especially on the dimension of 512. However, it is seen that the third parameter set of CNTR-512 is possible, featuring a gate complexity of $2^{163.9}$ at the memory complexity of $2^{102.7}$.

5.2. Refined gate-count estimate

As for the quantum gates and space complexity related to the LWE and LWR problems, we use the same gate number estimation method as Kyber, Saber, NTRU KEM, and SNTRU Prime in NIST PQC Round 3. Briefly speaking, it uses the probabilistic simulation of [56] rather than the GSA-intersect model of [47,57] to determine the BKZ blocksize b for a successful attack. And it relies on the concrete estimation for the cost of sieving in gates from [58]. It also accounts for the "few dimensions for free" proposed in [59], which permits to solve SVP in dimension b by sieving in a somewhat smaller dimension $b_0 = b - O(b)$. Finally, it dismisses the dual attack as realistically more expensive than the primal attack. In particular, in the dual attack, exploiting the short vectors generated by the Nearest Neighbor Search used in lattice sieving is not compatible with the "dimension for free" trick [59]. The scripts for these refined estimates are provided in a git branch of the leaky-LWE estimator [56].[1]

The gate-count estimate results of the parameter sets of CTRU and CNTR are presented in Tables 4 and 5, respectively. Ψ_1 and Ψ_2 are the probability distributions. q_2 is the RLWR modulus for CNTR. d is the optimal lattice dimension for the attacks. b is the BKZ blocksize. b' is the sieving dimension accounting for "dimension for free". Gate and memory are expressed in bits. The last column means the required gates values by NIST. It is estimated in [35] that the actual cost may not be more than 16 bits away from this estimate in either direction.

[1] https://github.com/lducas/leaky-LWE-Estimator/tree/NIST-round3.

Table 4
Gate-count estimate of CTRU parameters.

Schemes	(Ψ_1, Ψ_2)	d	b	b'	Gates	Memory	Gates by NIST
CTRU-512	(B_2, B_2)	1007	386	350	144.1	88.4	143
	(B_3, B_3)	1025	411	373	150.9	93.3	
CTRU-768	(B_2, B_2)	1467	634	583	214.2	137.9	207
	(B_3, B_3)	1498	671	618	224.6	145.3	
CTRU-1024	(B_2, B_2)	1919	890	825	286.1	188.9	272
	(B_3, B_3)	1958	939	871	299.7	198.5	

Table 5
Gate-count estimate of CNTR parameters.

Schemes	q_2	(Ψ_1, Ψ_2)	d	b	b'	Gates	Memory	Gates by NIST
CNTR-512	2^9	(B_3, B_3)	1025	411	373	150.9	93.3	143
	2^{10}	(B_5, B_5)	1025	444	404	160.1	100.0	
	2^{10}	(B_6, B_6)	1025	457	417	163.9	102.7	
CNTR-768	2^{10}	(B_3, B_3)	1498	671	618	224.6	145.3	207
	2^{10}	(B_4, B_4)	1521	699	644	232.3	150.8	
CNTR-1024	2^{10}	(B_2, B_2)	1919	890	825	286.1	188.9	272
	2^{10}	(B_3, B_3)	1958	939	871	299.7	198.5	

5.3. Attacks beyond core-SVP hardness

5.3.1. Hybrid attack

The works [60–62] consider the hybrid attack as the most powerful against NTRU-based cryptosystems. However, even with many heuristic and theoretical analysis on hybrid attack [60,63–65], so far it still fails to make significant security impact on NTRU-based cryptosystems partially due to the memory constraints. By improving the collision attack on NTRU problem, it is suggested in [66] that the hybrid attack complexity estimate used for NTRU problem is unreliable, and there are both overestimation and underestimation. Judging from the current hybrid and meet-in-the-middle (MITM) attacks on NTRU problem, there is an estimation bias in the security estimates of NTRU-based KEM schemes, but this bias does not make a big difference to the claimed security. For example, under the MITM search, the security of NTRU KEM in NIST PQC Round 3 may be 2^{-8} less than the acclaimed value in the worst situation [66].

5.3.2. Recent advances on dual attack

There are some recent progress on the dual attack, whose impacts on CTRU and CNTR will be discussed in the following. Duc et al. [67] propose that fast Fourier transform (FFT) can be useful to the dual attack. As for the small coefficients of the secrets, various improvements can also be achieved [63,68,69]. Albrecht [68] proposes a re-randomization and smaller-dimensional lattice reduction method, and investigates the method for generating coefficients of short vectors in the dual attack. Guo and Thomas [54] show that the current security estimates from the primal attacks are overestimated. Espitau et al. [70] achieve a dual attack that outperforms the primal attack. These attacks can be combined with the hybrid attack proposed in [64] to achieve a further optimized attack under specific parameters [63,71,72]. MATZOV [55] further optimizes the dual attack, and claims that the impact of its method is larger than that of Guo and Thomas's work [54]. It is also mentioned in [55] that the newly developed methods might also be applicable to NTRU-based cryptosystems (e.g., by improving the hybrid attack). The improvements of dual attacks mentioned above have potential threats to the security of CTRU and CNTR (as well as to other cryptosystems based on algebraically structured lattices). This line of research is still actively ongoing, and there is still no mature and convincing estimate method up to now.

5.3.3. S-unit attack

The basis of the S-unit attack is the unit attack, aiming at finding a short generator. On the basis of the constant-degree algorithm proposed in [73,74], Biasse et al. [75] present a quantum polynomial time algorithm, which is the basis for generating the generator used in the unit attack and S-unit attack. Then, the unit attack is to shorten the generator by reducing the modulus of the unit, and the idea is based on the variant of the LLL algorithm [76] to reduce the size of the generator in the S-unit group. That is, it replaces y_i with y_i/ϵ, thereby reducing the size of y_i, where y_i refers to the size of the generator and ϵ is the reduction factor of the modulus of the unit. Campbell et al. [77] consider the application of the cycloid structure to the unit attack, which mainly depends on the simple generator of the cycloid unit. Under the cycloid structure, the determinant is easy to determine, and is larger than the logarithmic length of the private key, which implies that the private key can be recovered through the LLL algorithm.

After establishing a set of short vectors, the simple reduction repeatedly uses $v - u$ to replace v, thereby reducing the modulus of vector v, where u belongs to the set of short vectors, the idea of which is discovered in [76,78]. The difference is that the algorithm proposed by Avanzi and Howard [78] can be applied to any lattice, but it is limited to the ℓ_2 norm, while the algorithm proposed by Cohen [76] is applicable to more norms. Pellet-Mary et al. [79] analyze the algorithm of Avanzi and Howard [78], and apply it to S-unit. They point out that the S-unit attack could achieve shorter vectors than other existing methods, but still with exponential time for an exponentially large approximation factor. Then, Bernstein and Tanja [53] further improve the S-unit attack.

Up to now, it is still an open problem to predict the effectiveness of the reduction inside the unit attacks. The statistical experiments on various m'th cyclotomics (with respect to power-of-two m') show that the efficiency of the S-unit attack is much higher than a spherical model of the same lattice for $m' \in \{128, 256, 512\}$ [80]. The effect is about a factor of 2^{-3}, 2^{-6} and 2^{-11}, respectively. Therefore, even with a conservative estimate, the security impact on CTRU and CNTR may not exceed a factor of 2^{-11}.

5.3.4. BKW attack

As for cryptographic schemes to which the BKW method can be applied, the combined methods proposed in [81–84], which extend the BKW method, can be the most efficient method for specific parameters. These methods require a large number of samples, and their security estimates are based on the analysis of lattice basis reduction, either by solving the encoding problem in the lattice or by converting to a u-SVP problem [85–87]. These attacks do not affect the security of CTRU and CNTR, because the parameters chosen for CTRU and CNTR do not meet the conditions of BKW method.

5.3.5. Side channel attack

Ravi et al. [88] construct some ciphertexts with specific structures where the key information exists in the intermediate variables, so as to recover the key through side channel attack (SCA). They apply this attack to NTRU KEM and NTRU Prime in NIST PQC Round 3, which can recover the full secret keys through a few thousands of chosen ciphertext queries. This type of SCA-aided chosen ciphertext attack is not directly applicable to CTRU and CNTR, but might be possible to be improved against CTRU and CNTR.

Bernstein [89] proposes an efficient fault attack with a one-time single-bit fault in the random string stored inside the secret key, such that this attack can recover all the previous NTRU-HRSS session keys with the aid of about a thousand of modified ciphertexts in the standard IND-CCA attack model. However, Bernstein's fault attack is valid for the specific ciphertext structures of NTRU-HRSS, and is invalid for compressed ciphertext (as in CTRU and CNTR). Thus, this type of fault attack does not threaten CTRU and CNTR yet.

5.3.6. Other attacks

Algebraic attacks [75,77,90,91] and dense sublattice attacks [71] also provide new ideas for LWE-based cryptographic analysis. However, these attacks do not currently affect the acclaimed security of the proposed parameters of CTRU and CNTR.

6. Polynomial arithmetic

In this section, some NTT algorithms are introduced to accelerate the polynomial multiplication and division of our schemes. In particular, we provide the methodology of using a unified NTT technique applicable to multiple parameter sets.

6.1. The mixed-radix NTT

A type of mixed-radix NTT is utilized to compute the polynomial multiplication and division over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ for recommended parameter sets of $n = 768$ and $q = 3457$. There exists the $\frac{3}{2}n$th primitive root of unity $\zeta = 5$ in $\mathbb{Z}_q$ due to $\frac{3}{2}n|(q-1)$. As for the forward NTT transform (NTT), inspired by NTTRU, there is a mapping such that $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1) \cong \mathbb{Z}_q[x]/(x^{n/2} - \zeta_1) \times \mathbb{Z}_q[x]/(x^{n/2} - \zeta_2)$ where $\zeta_1 + \zeta_2 = 1$ and $\zeta_1 \cdot \zeta_2 = 1$. In order to apply the mixed-radix NTT, we choose $\zeta_1 = \zeta^{n/4} \bmod q$ and $\zeta_2 = \zeta_1^5 = \zeta^{5n/4} \bmod q$. Thus, both $x^{n/2} - \zeta_1$ and $x^{n/2} - \zeta_2$ can be recursively split down into degree-6 terms like $x^6 \pm \zeta^3$ through 6 steps of radix-2 FFT trick for $n = 768$. Then the steps of radix-3 FFT trick can be utilized based on the isomorphism $\mathbb{Z}_q[x]/(x^6 - \zeta^3) \cong \mathbb{Z}_q[x]/(x^2 - \zeta) \times \mathbb{Z}_q[x]/(x^2 - \rho\zeta) \times \mathbb{Z}_q[x]/(x^3 - \rho^2\zeta)$ where $\rho = \zeta^{n/2} \bmod q$. Consequently, $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ can be decomposed into $\prod_{i=0}^{n/2-1} \mathbb{Z}_q[x]/(x^2 - \zeta^{\tau(i)})$, where $\tau(i)$ is the power of ζ of the ith term and the index i starts from zero. Upon receiving the polynomial f, its result of the forward NTT transform is $\hat{f} = (\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_{\frac{n}{2}-1})$ where $\hat{f}_i \in \mathbb{Z}_q[x]/(x^2 - \zeta^{\tau(i)})$ is a linear polynomial, $i = 0, 1, \ldots, \frac{n}{2} - 1$.

The inverse NTT transform ($INTT$) can be derived by reversing these procedures. In this case, the point-wise multiplication ("$\circ$") involves the corresponding linear polynomial multiplications in $\mathbb{Z}_q[x]/(x^2 - \zeta^{\tau(i)}), i = 0, 1, \ldots, \frac{n}{2} - 1$.

As for the mixed-radix NTT-based polynomial multiplication with respect to $h = f \cdot g$, the entire computing process is expressed as $h = INTT(NTT(f) \circ NTT(g))$. Similarly, as for the mixed-radix NTT-based polynomial division with respect to $h = g/f$ (i.e., computing the public key in this paper), it is essentially to compute $h = INTT(\hat{g} \circ \hat{f}^{-1})$. Here, $\hat{g} = NTT(g)$, $\hat{f} = NTT(f)$, and $\hat{f}^{-1} = (\hat{f}_0^{-1}, \hat{f}_1^{-1}, \ldots, \hat{f}_{\frac{n}{2}-1}^{-1})$ where $\hat{f}_i^{-1}$ is the inverse of $\hat{f}_i$ in $\mathbb{Z}_q[x]/(x^2 - \zeta^{\tau(i)})$, if each $\hat{f}_i^{-1}$ exists, $i = 0, 1, \ldots, \frac{n}{2} - 1$.

6.1.1. The pure radix-2 NTT

Similar techniques can be utilized to compute the polynomial multiplication and division over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ for other parameter sets, such as $(n = 512, q = 3457)$ and $(n = 1024, q = 3457)$. Note that only the steps of the radix-2 FFT trick are required since both n's are power-of-two. Another observation is that for these two n's, there only exists the 384th primitive root of unity ζ in $\mathbb{Z}_q$, such that $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ can be decomposed into $\prod_{i=0}^{n/4-1} \mathbb{Z}_q[x]/(x^4 - \zeta^{\tau(i)})$ when $n = 512$ and into $\prod_{i=0}^{n/8-1} \mathbb{Z}_q[x]/(x^8 - \zeta^{\tau(i)})$ when $n = 1024$. Therefore, the point-wise multiplication and the base case inversion involve the corresponding polynomials of degree 3 (resp., degree 7) when $n = 512$ (resp., $n = 1024$).

6.2. The unified NTT

As previously discussed, in order to achieve an efficient implementation, we conduct 6 steps of radix-2 FFT trick for $n \in \{512, 768, 1024\}$ and an additional step of radix-3 FFT trick for $n = 768$. Consequently, varying NTT algorithms (i.e., mixed-radix and radix-2) are required for the three types of parameter sets corresponding to different n's. However, in many applications or platforms, there may be a necessity to implement all three types of parameter sets. In order to deal with the inconvenient issue, we present a unified NTT methodology over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, $n \in \{512, 768, 1024\}$, such that only one type of NTT computation is required for different n's, facilitating modular and unified implementations when all the three parameter sets are considered.

In this work, we consider $n = \alpha \cdot N$, where $\alpha \in \{2, 3, 4\}$ is called the splitting-parameter and N is a power of two. Notably, α can be chosen more freely as arbitrary values of the form $2^i 3^j, i \geq 0, j \geq 0$. With the traditional NTT technique, when the dimension n changes we need to use different NTT algorithms of various input/output lengths to compute polynomial multiplications over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$. This causes much inconvenience to software and particularly hardware implementations. To address this issue, we unify the various n-point NTTs through an N-point NTT, which is referred to as the unified NTT technique. For $n \in \{512, 768, 1024\}$, we set $N = 256$ and choose $\alpha \in \{2, 3, 4\}$. With this technique, our focus narrows to the implementation of the N-point NTT, which serves as the unified procedure to be invoked for different n's. Specifically, the computation of NTT over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ is divided into three steps. For presentation simplicity, we only give the procedures of the forward transform as follows, since the inverse transform can be obtained by reversing these procedures. The map road is illustrated in Fig. 2.

Step 1. Construct a splitting-polynomial map φ_1 :

$$\mathbb{Z}_q[x]/(x^{\alpha \cdot N} - x^{\alpha \cdot N/2} + 1) \to \left(\mathbb{Z}_q[y]/(y^N - y^{N/2} + 1)\right)[x]/(x^\alpha - y)$$
$$f = \sum_{i=0}^{\alpha \cdot N - 1} f_i x^i \mapsto \sum_{j=0}^{\alpha - 1} F_j x^j$$

where $F_j = \sum_{i=0}^{N-1} f_{\alpha \cdot i + j} y^i \in \mathbb{Z}_q[y]/(y^N - y^{N/2} + 1)$. Namely, the n-dimension polynomial is split into α N-dimension sub-polynomials.

Step 2. Apply the unified N-point NTT to F_j over $\mathbb{Z}_q[y]/(y^N - y^{N/2} + 1)$, $j = 0, 1, \ldots, \alpha - 1$. Specifically, there is a mapping such that $\mathbb{Z}_q[y]/(y^N - y^{N/2} + 1) \cong \mathbb{Z}_q[y]/(y^{N/2} - \zeta_1) \times \mathbb{Z}_q[y]/(y^{N/2} - \zeta_2)$ where $\zeta_1 + \zeta_2 = 1$ and $\zeta_1 \cdot \zeta_2 = 1$. Let q be the prime number satisfying $\frac{3N}{2^\beta}|(q-1)$, where $\beta \in \mathbb{N}$ is called the truncating-parameter, ensuring that there exists the primitive $\frac{3N}{2^\beta}$th root of unity ζ in $\mathbb{Z}_q$. To apply the radix-2 FFT trick, we choose $\zeta_1 = \zeta^{N/2^{\beta+1}} \bmod q$ and $\zeta_2 = \zeta_1^5 = \zeta^{5N/2^{\beta+1}} \bmod q$. Consequently, both $y^{N/2} - \zeta_1$ and $y^{N/2} - \zeta_2$ can be recursively split down into degree-2^β terms like $y^{2^\beta} \pm \zeta$. The idea of truncating FFT trick originates from [92]. Therefore, $\mathbb{Z}_q[y]/(y^N - y^{N/2} + 1)$ can be decomposed into $\prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[y]/(y^{2^\beta} - \zeta^{\tau(k)})$, where $\tau(k)$ is the power of ζ of the kth term. Let $\hat{F}_j$ be the NTT result of F_j and $\hat{F}_{j,l}$ be its lth coefficient, $l = 0, 1, \ldots, N - 1$. Hence, the output of Step 2 can be expressed as:

$$\hat{F}_j = \left(\sum_{l=0}^{2^\beta - 1} \hat{F}_{j,l} y^l, \sum_{l=0}^{2^\beta - 1} \hat{F}_{j,l+2^\beta} y^l, \ldots, \sum_{l=0}^{2^\beta - 1} \hat{F}_{j,l+N-2^\beta} y^l\right)$$
$$\in \prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[y]/(y^{2^\beta} - \zeta^{\tau(k)}).$$

Step 3. Combine the intermediate values and obtain the final result by the map φ_2 :

$$\left(\prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[y]/(y^{2^\beta} - \zeta^{\tau(k)})\right)[x]/(x^\alpha - y) \to \prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[x]/(x^{\alpha \cdot 2^\beta} - \zeta^{\tau(k)})$$
$$\sum_{j=0}^{\alpha - 1} \hat{F}_j x^j \mapsto \hat{f}$$

where $\hat{f} = \sum_{i=0}^{\alpha \cdot N - 1} \hat{f}_i x^i$ is the NTT result of f. Its ith coefficient is $\hat{f}_i = \hat{F}_{j,l}$, where $j = i \bmod \alpha$ and $l = \lfloor \frac{i}{\alpha} \rfloor$. It can be rewritten as:

$$\hat{f} = \left(\sum_{i=0}^{\alpha \cdot 2^\beta - 1} \hat{f}_i x^i, \sum_{i=0}^{\alpha \cdot 2^\beta - 1} \hat{f}_{i+\alpha \cdot 2^\beta} x^i, \ldots, \sum_{i=0}^{\alpha \cdot 2^\beta - 1} \hat{f}_{i+n-\alpha \cdot 2^\beta} x^i\right) \quad (8)$$
$$\in \prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[x]/(x^{\alpha \cdot 2^\beta} - \zeta^{\tau(k)}).$$

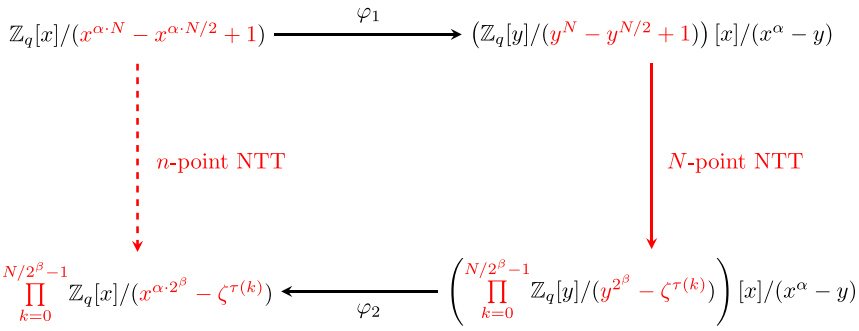


Fig. 2. Map road for unified NTT.

In this work, we choose $\beta = 1$ and $q = 3457$, where the primitive 384th root of unity $\zeta = 55$ exists in $\mathbb{Z}_{3457}$. In this case, the point-wise multiplication involves the corresponding 2α-dimension polynomial multiplication in $\mathbb{Z}_q[x]/(x^{2\alpha} - \zeta^{\tau(k)}), \alpha \in \{2, 3, 4\}, k = 0, 1, \ldots, N/2 - 1$.

6.3. Discussions

Regarding application scenarios for $n = 768$, the polynomial multiplication and division over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ can be efficiently improved with the aid of the mixed-radix NTT (the benchmark results are shown in Section 8). However, this type of NTT cannot be implemented universally and modularly for more general n's, since it can be only applied in the case of $n = 3\cdot 2^e$ for some integer e, instead of power-of-two n like 512 and 1024. Since the common application scenarios and cryptographic devices typically involve three recommended parameter sets ($n \in \{512, 768, 1024\}$) for targeted security levels, three distinct NTT algorithms would be required for CTRU and CNTR corresponding to these parameter sets. The implementation of a unified NTT serves to address this inconvenience, leading to modular and simplified software and hardware implementations in such cases.

6.4. Base case inversion

As for the case of the mixed-radix NTT w.r.t $n = 768$, the inverses of the polynomials in $\mathbb{Z}_q[x]/(x^2 - \zeta^j), j \geq 0$ can be computed directly. For example, given $a = a_0 + a_1 x \in \mathbb{Z}_q[x]/(x^2 - \zeta^j)$, its inversion can be written explicitly as $a^{-1} := (a_0 - a_1 x)/(a_0^2 - \zeta^j \cdot a_1^2)$ when $a_0^2 - \zeta^j \cdot a_1^2 \neq 0$. As for the case of the unified NTT w.r.t $n \in \{512, 768, 1024\}$, we can efficiently compute $\hat{f}^{-1}$ through the "divide and conquer" strategy used in the work [13].

6.5. Multi-moduli NTT

Although directly using NTT is invalid over $\mathcal{R}_{q_2}$ for power-of-two q_2 in the decryption of CTRU and CNTR, the works [93,94] demonstrate the feasibility of efficiently applying a multi-moduli NTT over $\mathcal{R}_{q_2}$. Briefly speaking, according to [94], the polynomial multiplication over $\mathcal{R}_{q_2}$ can be lifted to that over $\mathcal{R}_Q = \mathbb{Z}_Q[x]/(x^n - x^{n/2} + 1)$, where Q is a positive integer and larger than the maximum absolute value of the coefficients during the computation over $\mathbb{Z}$. Then, one can recover the targeted product polynomial through reduction modulo q_2. We choose $Q = qq'$ where $q = 3457$ and $q' = 7681$ in this paper, leading to the CRT isomorphism: $\mathcal{R}_Q \cong \mathcal{R}_q \times \mathcal{R}_{q'}$. The first NTT algorithm for (n, q) over $\mathcal{R}_q$ can be instanced as the mixed-radix NTT in Section 6.1 or the unified NTT in Section 6.2. The second NTT algorithm for (n, q') over $\mathcal{R}_{q'}$ can be followed from that in NTTRU. After using NTTs to compute two intermediate products in $\mathcal{R}_q$ and $\mathcal{R}_{q'}$ respectively, the targeted product in $\mathcal{R}_Q$ can be reconstructed through CRT.

7. Implementation and benchmark

In this section, the remaining implementation details of our schemes are provided, including our portable C implementation and optimized implementation with AVX2 instruction sets.

7.1. Symmetric primitives

All the hash functions are instantiated with functions from **SHA-3** family. To generate the secret polynomials, i.e., f', g, r, e, the secret seeds are required to be expanded to the sampling randomness by using **SHAKE-256**. The hash function $\mathcal{H}$ is instantiated based on **SHA3-512**, aiming to hash the short prefix of public key $ID(pk)$ and message m into the 64 bytes where the first 32 bytes are used to generate the shared keys and the latter 32 bytes are used as the secret seed for the encryption algorithm.

7.2. Generation of secret polynomials

All the secret polynomials in our schemes are sampled according to the centered binomial distribution B_η. Each of them totally requires $2n\eta$ bits, or saying $2n\eta/8$ bytes, as sampling randomness. To generate each coefficient of an secret polynomial, we arrange the adjacent independent 2η random bits and subtract the Hamming weight of the most significant η bits from the Hamming weight of the least significant η bits.

7.3. The keys and ciphertexts

The format of the public key. The public key is transmitted in the NTT representation, following the approach utilized in previous works such as [15,35,47,95]. Specifically, the public key is treated as $\hat{h} = \hat{g} \circ \hat{f}^{-1}$, which saves an inverse transform in the key generation and a forward transform in the encryption (re-run in the decapsulation). The coefficients of $\hat{h}$ are reduced modulo q into $\mathbb{Z}_q$, resulting in each coefficient occupying 12 bits. Consequently, the public key is packed into an array of $12n$ bits, i.e., $3n/2$ bytes in total.

The format of the secret key. Note that the polynomial f has coefficients in normal representation of $[-2\eta, 2\eta + 1]$ where η is the parameter of the centered binomial distribution B_η. Instead of directly packing the polynomial f into bytes array, we subtract each coefficient from $2\eta + 1$, ensuring that all the coefficients fall within the interval $[0, 4\eta + 1]$. The resulting polynomial is then packed into $n\lceil \log(4\eta + 1) \rceil / 8$ bytes. The initial coefficient can be recovered by being subtracted from $2\eta + 1$ in the unpack step in decryption. Since the public key is required in the re-encryption during the decapsulation, we simply concatenate and store the packed public key as part of the secret key. An extra 32-byte z is also concatenated, since z is used to derive a pseudo-random key as output of implicit rejection if re-encryption does not succeed.

The total size of a decapsulation secret key contains $n\lceil\lfloor\log(4\eta+1)\rfloor/8+3n/2+32$ bytes.

The format of the ciphertext. The ciphertext of our schemes consists of only one (compressed) polynomial c. The polynomial c is presented in normal representation instead of NTT representation, as the compression process involving rounding necessitates the use of normal representation. Each coefficient of c occupied $\lceil\log(q_2)\rceil$ bits. Consequently, the packing and storage of such a ciphertext incur a cost of only $n\lceil\log(q_2)\rceil/8$ bytes.

The prefix of the public key. Regarding the prefix $ID(pk)$ of the public key h in CTRU and CNTR, we use the first 33 bytes of the bit-packed NTT representation of h. It is reasonable, since h is computationally indistinguishable from a uniformly random polynomial in $\mathcal{R}_q$ and the forward NTT transform keeps the randomness property (i.e., h is uniformly random, so is $\hat{h} = NTT(h)$). Thus, the first 22 coefficients of the public key have the min-entropy of more than 256 bits and occupy 33 bytes in the bit-packed NTT representation since each coefficient occupies 12 bits.

7.4. Portable C implementation

Our portable C implementations predominantly depend on 16-bit and 32-bit integer arithmetic, excluding any floating-point arithmetic. The polynomials are represented as arrays of 16-bit signed integers. It is reasonable since we use a 12-bit prime. Our implementation of decoding algorithm of the scalable E_8 lattice follows the approach in [32] which is based on 32-bit integer arithmetic, but we make progress in achieving a constant-time implementation.

NTT implementation. Our C implementations of NTTs do not make use of variable-time operator "%" for the modular reductions. Instead, we apply Barrett reduction [96,97] and Montgomery reduction [97,98], where the former is applied after additions and the later is applied for multiplication between coefficients with primitive roots. Lazy reduction strategy [97] is suitable for the forward NTT transform. Note that the output range of Montgomery reduction is in $[-q, q]$. For 12-bit coefficients of the input polynomial, after 7-level FFT tricks the forward NTT transform outputs the polynomials with coefficients in $[-8q, 8q]$, which does not overflow the valid representation of a 16-bit signed integer in the context of 12-bit modulus q. However, NTT is invalid in $\mathcal{R}_{q_2}$ for power-of-two q_2 in the decryption process, so we turn to the schoolbook algorithm to compute $cf \bmod^{\pm} q_2$ for efficiency and simplicity in the portable C implementation, where the modular reduction concerning power-of-two q_2 can be efficiently implemented by using logical AND operations.

7.5. Optimized AVX2 implementation

The optimized implementations of our schemes for CPUs which support the AVX2 instruction sets are provided. The main optimized targets are polynomial arithmetic, sampling secrets and modular reduction algorithms in NTT, all of which are the time-consuming operations. However, as for **SHA-3** hash functions, we do not have any AVX2-based optimization. Consistently, we use the same source codes as in portable C implementation. This is because the vectorized implementations of **SHA-3** hash functions are not very helpful for accelerating, and the fastest implementation is based on C language [47,99]. As for the computation of $cf \bmod^{\pm} q_2$ for power-of-two q_2 in the decryption process of CTRU and CNTR, we choose the multi-moduli NTT, instead of the schoolbook algorithm, deviating from the choice made in the portable C implementation. Additionally, according to our experiments on a full polynomial multiplication over $\mathcal{R}_{q_2}$, compared to the schoolbook algorithm, the multi-moduli NTT is slower in the context of C implementation, but is faster in the context of AVX2 implementation, for the reason that NTT is suitable for vectorized implementation, especially AVX2.

NTT optimizations. Our AVX2-based NTT implementation handles 16-bit signed integer coefficients, with every 16 values loaded into one vector register. Load and store instructions are time-consuming in AVX2 instruction set. To accelerate AVX2 implementation, reducing memory access operations is crucial. We present some implementation strategies to fully utilize vector registers and minimize total CPU cycles. For the radix-2 FFT trick, we merge the first three levels and the following three levels. During the merging levels there are no extra load or store operations. This optimization is achieved by using different pair of vector registers and permutating coefficients order. The instructions used for permutation task are `vperm2i128`, `vpunpcklqdq`, `vpblendd` and `vpblendw`. The coefficients are permutated in levels 3–5. After the radix-2 FFT trick, the coefficients are not stored immediately, instead we use the vector register to complete the radix-3 FFT trick if necessary. It is because after the last level of the radix-2 FFT trick, the order of coefficients in the vector register is naturally the order required in the radix-3 FFT trick. To minimize total permutation time, we propose storing coefficients in a shuffled order. This proves advantageous for polynomial point-wise multiplication and polynomial inversion, both of which involve pairwise modular multiplications. Storing two contiguous coefficients in two separate vector registers facilitates the straightforward implementation of polynomial point-wise multiplication and polynomial inversion.

7.6. Constant-time implementation

We report on our constant-time implementation to avoid the potential timing attacks. Specifically, our implementations do not use any variable-time instructions to operate the secret data, do not use any branch depending on the secret data, and do not access any memory at addresses depending on the secret data.

As for the modular reductions used in the NTTs, as described in [15,97], both Barrett reduction and Montgomery reduction used in our implementations are constant-time algorithms. Furthermore, the reduction algorithms are not specific to the modulus q.

As for the scalable E_8 lattice code, in our encoding algorithm $\mathbf{kH} \bmod 2$ can be computed efficiently by simple bitwise operations, which have been implemented by constant-time steps. For the implementation of the scalable E_8 decoding algorithms in Algorithm 2 and Algorithm 3, we present branching-free implementations. All the "arg min" statements and "if" conditional statements are implemented by constant-time bitwise operations. In essence, these can be summarized as choosing the minimal value of two secrets in signed integer representation, which are defined as `a, b`. This can be implemented without timing leakage of the secret data flow as: `c = ((-(r XOR 1)) AND a) XOR ((-(r AND 1)) AND b)`, where `r` $\in \{0, 1\}$ is the sign bit of the value `b − a`, XOR is the logical Exclusive OR operator, and AND is the logical AND operator. We emphasize that, although there exist a variety of error correction codes, including lattice codes, there are indeed inherent difficulties on constant-time implementations for most existing error correction codes. Take BCH code and LDPC code [100] as examples, which are widely used in reality. BCH code does not enable a constant-time implementation for its decoding process, since it needs to locate the error bits by computing the syndrome and correct the error bits within its range of error correction capability, but these proceeds are not constant-time [101]. LDPC code also does not have a constant-time decoding process, since its decoding process works under iterative steps which stop unless the errors are corrected or the iterations reach the maximum number [101]. A similar situation happens to some lattice codes. For example, none has found constant-time implementations of decoding algorithms with respect to BW_{32} lattice and BW_{64} lattice [31]. However, in contrast to those error correction codes, our scalable E_8 lattice code features constant-time encoding and decoding algorithms, enabling implementations protected against timing attacks.

Table 6
CPU cycles of KEMs (in kilo cycles).

Schemes	C			AVX2		
	KeyGen	Encaps	Decaps	KeyGen	Encaps	Decaps
CTRU-768, Mixed-radix NTT (Ours)	90.9	58.9	123.1	7.5	9.2	28.6
CNTR-768, Mixed-radix NTT (Ours)	93.6	56.2	119.0	9.5	8.4	26.9
CTRU-768, Unified NTT (Ours)	102.0	63.5	126.1	–	–	–
CNTR-768, Unified NTT (Ours)	103.3	62.2	125.5	–	–	–
NTRU-HRSS	110.9×10^3	3.0×10^3	9.1×10^3	254.0	24.9	59.2
SNTRU Prime-761	138.1×10^3	15.0×10^3	46.4×10^3	156.3	46.9	56.2
Kyber-768	104.1	115.4	155.7	25.3	27.6	43.4
Saber-768	70.2	74.2	110.9	64.2	69.3	95.3
CTRU-768, **SHA-2** variant (Ours)	87.9	54.1	109.1	4.3	4.0	14.8
CNTR-768, **SHA-2** variant (Ours)	88.9	52.4	106.0	4.8	3.9	14.6
NTTRU	116.4	77.7	112.3	6.4	6.1	7.9
BIKE (Level 3)	–	–	–	1.7×10^3	267.1	5.3×10^3
Classic McEliece460896	–	–	–	150.0×10^3	77.3	253.9
HQC-192	–	–	–	417.8	719.7	1.2×10^3
SIKEp610	–	–	–	15.0×10^3	27.2×10^3	27.7×10^3

8. Benchmark and comparison

In this section, we provide the benchmark results of our CTRU and CNTR where we focus on the recommended parameter set of dimension 768, i.e., $(n = 768, q = 3457, q_2 = 2^{10}, \Psi_1 = \Psi_2 = B_2)$ for CTRU-768 and $(n = 768, q = 3457, q_2 = 2^{10}, \Psi_1 = \Psi_2 = B_3)$ for CNTR-768, with the applications of mixed-radix NTT. All the benchmark tests are run on an Intel(R) Core(TM) i7-10510U CPU at 2.3 GHz (16 GB memory) with Turbo Boost and Hyperthreading disabled. The operating system is Ubuntu 20.04 LTS with Linux Kernel 4.4.0 and the gcc version is 9.4.0. The compiler flag of our schemes is listed as follows: *-Wall -march=native -mtune=native -O3 -fomit-frame-pointer -Wno-unknown-pragmas*. Median cycle counts of 10,000 executions of the corresponding KEM algorithms are obtained. The benchmark results are provided in Table 6, along with comparisons with other schemes. Concretely, we re-run the C source codes of other schemes on the exact same system as CTRU and obtain the corresponding benchmark results for providing reasonable reference comparisons, but their state-of-the-art AVX2 benchmark results are directly taken from the literatures or SUPERCUP (the supercop-20220506 benchmarking run on a 3.0 GHz Intel Xeon E3-1220 v6) [99]. Regarding the benchmark results in this section, we emphasize that this may be not an exhaustive benchmark ranking but serves as optional illustration that our schemes might perform reasonably well when compared to other schemes.

8.1. Comparison with other NTRU-based KEM schemes

The C source codes of NTRU-HRSS and SNTRU-Prime are taken from their Round 3 supporting documentations, while those of NTTRU are taken from [15]. One regret is that the source codes of NTRU-C^{768}_{3457} are not online available in [29], and the AVX2 benchmark results of NTRU-C^{768}_{3457} are absent in [29], so all of its benchmark results are omitted here.

Note that the work [94] shows how to apply multi-moduli NTT to accelerate the polynomial multiplications in NTRU-HRSS, but the polynomial divisions remain unchanged. However, the resulting speed-up for NTRU-HRSS in [94] is not obvious (in fact, the speed-up is $\pm 0\%$). Hence, we omit the benchmark results about NTRU-HRSS provided in [94]. Consequently, the benchmark results of the state-of-the-art AVX2 implementation of NTRU-HRSS are reported in [99]. As for SNTRU Prime-761, its state-of-the-art AVX2 implementation is presented in [102]. Thus, we take the AVX2 benchmark results of NTRU-HRSS and SNTRU Prime-761 from [99,102], respectively. Additionally, the AVX2 benchmark results of NTTRU are taken from [15].

When compared to NTRU-HRSS and SNTRU Prime-761, the efficiency improvements of CTRU-768 and CNTR-768 benefit from the applications of NTT in polynomial operations. Concretely, as for portable C implementation, CTRU-768 is faster than NTRU-HRSS by more than 1,200X in KeyGen, 50X in Encaps, and 73X in Decaps, respectively. As for the optimized AVX2 implementation, CTRU-768 is faster than NTRU-HRSS by 33X in KeyGen, 2.7X in Encaps, and 2.1X in Decaps, respectively; CNTR-768 is faster than NTRU-HRSS by 26X in KeyGen, 3.0X in Encaps, and 2.2X in Decaps, respectively.

In a fair comparison with NTTRU, we implement certain modifications to present variants of CTRU and CNTR, ensuring consistency with NTTRU in terms of hash functions, symmetric primitives, and FO transformations: (1) use **SHA-2** family to instantiate hash functions; (2) use **AES** to expand seeds; (3) change the FO transformation into $\mathrm{FO}^{\perp}_{m}$. For C implementation, both CTRU-768 and CNTR-768 are faster than NTTRU in all the three processes of KeyGen, Encaps and Decaps. When compared to the AVX2 benchmark results of NTTRU, CTRU-768 is faster by 1.5X in KeyGen and 1.5X in Encaps, respectively; CNTR-768 is faster than NTRU-HRSS by 1.3X in KeyGen and 1.6X in Encaps, respectively. However, the AVX2 implementations of Decaps in CTRU/CNTR-768 are slower than that of NTTRU, on the following grounds: (1) the decoding algorithm of the scalable E_8 lattice code is less amenable to vectorization implementation; (2) the multi-moduli NTT is more time-consuming than the NTT algorithm of NTTRU, since the multi-moduli NTT essentially consists of two routines of NTT algorithms.

8.2. Comparison with other lattice-based KEM schemes

The C source codes of Kyber-768 and Saber-768 are obtained from their Round 3 supporting documentations. We modify their FO transformation into $\mathrm{FO}^{\not\perp}_{ID(pk),m}$, and re-run their C source codes. The state-of-the-art AVX2 implementation of Kyber and Saber with $\mathrm{FO}^{\not\perp}_{ID(pk),m}$ has been previously reported in [34]. Therefore, we directly extracted their AVX2 benchmark results from [34].

As depicted in Table 6, both CTRU-768 and CNTR-768 exhibit superior performance compared to Kyber-768 and Saber-768. Specifically, when compared to the state-of-the-art AVX2 implementation of Kyber-768, CTRU-768 is faster by 3.4X in KeyGen, 3.0X in Encaps, and 1.5X in Decaps, respectively; CNTR-768 is faster by 2.7X in KeyGen, 3.3X in Encaps, and 1.6X in Decaps, respectively. This notable advantage can be attributed to the following reasons: (1) Kyber involves rejection sampling to generate the matrix $\mathbf{A}$ and a complicated polynomial matrix–vector multiplication in both the key generation and

encryption processes (which are also re-run with the Decaps); (2) in contrast, CTRU-768 and CNTR-768 require only a single polynomial multiplication in the encryption process.

8.3. Comparison with other non-lattice-based KEM schemes

We conduct a rough comparison with other non-lattice-based KEM schemes, i.e., BIKE [103], Classic McEliece [104], HQC [105] and SIKE [106], which are candidates advancing to the fourth round of NIST PQC [107]. The first three KEM schemes are code-based, and the last one is isogeny-based. However, the SIKE team acknowledges that SIKE is insecure and should not be used [107]. Nevertheless, the benchmark results of SIKE are still presented and only used for intuitive comparisons. We only present their state-of-the-art AVX2 benchmark results, which can be also found in SUPERCUP [99]. As illustrated in Table 6, our schemes are much faster than these non-lattice-based KEM schemes. For example, CTRU-768 outperforms Classic McEliece460896 by more than 20,000X in KeyGen, 8.4X in Encaps and 8.9X in Decaps.

8.4. Benchmark results with unified NTT

The benchmark results of C implementations of CTRU-768 and CNTR-768 with our unified NTT are also provided in Table 6. Despite being slightly inferior to the performance of CTRU-768 and CNTR-768 with mixed-radix NTT, their overall performance is still remarkably efficient. We emphasize that the primary goal of the unified NTT is to provide a modular and convenient implementation, instead of a faster implementation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61877011), the National Key Research and Development Program of China (Grant No. 2022YFB2701600), Shanghai Science and Technology Innovation Action Plan, China (Grant No. 21DZ2200500), and Shandong Provincial Key Research and Development Program of China (Grant No. 2017CXG0701, 2018CXGC0701).

References

[1] NIST, PQC standardization process: Announcing four candidates to be standardized, plus fourth round candidates, 2022, https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.
[2] J. Alperin-Sheriff, D. Apon, Dimension-preserving reductions from LWE to LWR, IACR Cryptol. ePrint Arch. (2016) 589.
[3] A. Banerjee, C. Peikert, A. Rosen, Pseudorandom functions and lattices, in: EUROCRYPT 2012, Vol. 7237, 2012, pp. 719–737.
[4] A. Langlois, D. Stehlé, Worst-case to average-case reductions for module lattices, Des. Codes Cryptogr. 75 (3) (2015) 565–599.
[5] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, in: EUROCRYPT 2010, Vol. 6110, 2010, pp. 1–23.
[6] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, J. ACM 56 (6) (2009) 34:1–34:40.
[7] J. Hoffstein, J. Pipher, J.H. Silverman, NTRU: a ring-based public key cryptosystem, in: ANTS, Vol. 1423, 1998, pp. 267–288.
[8] J. Hoffstein, J. Pipher, J.H. Silverman, NTRU: a new high speed public key cryptosystem, 1996, presented at the rump session of Crypto 96.
[9] D. Coppersmith, A. Shamir, Lattice attacks on NTRU, in: EUROCRYPT '97, Vol. 1233, 1997, pp. 52–61.
[10] K. Bagheri, M. Sadeghi, D. Panario, A non-commutative cryptosystem based on quaternion algebras, Des. Codes Cryptogr. 86 (10) (2018) 2345–2377.
[11] D.J. Bernstein, B.B. Brumley, M.-S. Chen, C. Chuengsatiansup, T. Lange, A. Marotzke, B.-Y. Peng, N. Tuveri, C. van Vredendaal, B.-Y. Yang, NTRU prime: round 3, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[12] C. Chen, O. Danba, J. Hoffstein, A. Hulsing, J. Rijneveld, J.M. Schanck, T. Saito, NTRU submission, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[13] P.-A. Fouque, P. Kirchner, T. Pornin, Y. Yu, BAT: Small and fast KEM over NTRU lattices, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2022 (2) (2022) 240–265.
[14] K. Jarvis, M. Nevins, ETRU: NTRU over the eisenstein integers, Des. Codes Cryptogr. 74 (1) (2015) 219–242.
[15] V. Lyubashevsky, G. Seiler, NTTRU: truly fast NTRU using NTT, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019 (3) (2019) 180–201.
[16] L. Ducas, V. Lyubashevsky, T. Prest, Efficient identity-based encryption over NTRU lattices, in: ASIACRYPT 2014, Vol. 8874, 2014, pp. 22–41.
[17] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang, Falcon: Fast-Fourier lattice-based compact signatures over NTRU, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[18] S. Garg, C. Gentry, S. Halevi, Candidate multilinear maps from ideal lattices, in: EUROCRYPT 2013, Vol. 7881, 2013, pp. 1–17.
[19] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman, W. Whyte, NTRUSIGN: digital signatures using the NTRU lattice, in: CT-RSA 2003, Vol. 2612, 2003, pp. 122–140.
[20] A. Langlois, D. Stehlé, R. Steinfeld, GGHLite: More efficient multilinear maps from ideal lattices, in: EUROCRYPT 2014, Vol. 8441, 2014, pp. 239–256.
[21] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: STOC 2012, 2012, pp. 1219–1234.
[22] D. Jablon, IEEE P1363 standard specifications for public-key cryptography, in: CTO Phoenix Technologies Treasurer, IEEE P1363 NIST Key Management Workshop, 2008.
[23] B. Wire, Security innovation's NTRUEncrypt adopted as X9 standard for data protection, 2011, https://www.businesswire.com/news/home/20110411005309/en/Security-Innovations-NTRUEncrypt-Adopted-X9-Standard-Data.
[24] J.M. Schanck, W. Whyte, Z. Zhang, Circuit-extension handshakes for tor achieving forward secrecy in a quantum world, Proc. Privacy Enhanc. Technol. 2016 (4) (2016) 219–236.
[25] D. Augot, L. Batina, D.J. Bernstein, J. Bos, Initial recommendations of long-term secure post-quantum systems, in: PQCRYPTO. EU. Horizon, Vol. 2020, 2015.
[26] D. Stehlé, R. Steinfeld, Making NTRU as secure as worst-case problems over ideal lattices, in: EUROCRYPT 2011, Vol. 6632, 2011, pp. 27–47.
[27] OpenSSH, OpenSSH release notes, 2022, https://www.openssh.com/releasenotes.html.
[28] J.M. Schanck, A comparison of NTRU variants, IACR Cryptol. ePrint Arch. (2018) 1174, URL: https://eprint.iacr.org/2018/1174.
[29] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, G. Seiler, D. Unruh, A thorough treatment of highly-efficient NTRU instantiations, in: PKC 2023, Vol. 13940, 2023, pp. 65–94.
[30] J.H. Conway, N.J.A. Sloane, Fast quantizing and decoding and algorithms for lattice quantizers and codes, IEEE Trans. Inf. Theory 28 (2) (1982) 227–231.
[31] J.H. Conway, N.J.A. Sloane, Sphere Packings, Lattices and Groups, Vol. 290, Springer Science & Business Media, 2013.
[32] Z. Jin, S. Shen, Y. Zhao, Compact and flexible KEM from ideal lattice, IEEE Trans. Inf. Theory 68 (6) (2022) 3829–3840.
[33] M. Viazovska, The sphere packing problem in dimension 8, Ann. of Math. (2017) 991–1015.
[34] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, G. Seiler, Faster lattice-based KEMs via a generic Fujisaki–Okamoto transform using prefix hashing, in: ACM CCS 2021, 2021, pp. 2722–2737.
[35] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS-kyber - algorithm specifications and supporting documentation (version 3.0), in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[36] A. Basso, J.M.B. Mera, J.-P. D'Anvers, A. Karmakar, S.S. Roy, M.V. Beirendonck, F. Vercauteren, Supporting documentation: SABER: Mod-LWR based KEM (round 3 submission), in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[37] A. Hülsing, J. Rijneveld, J.M. Schanck, P. Schwabe, High-speed key encapsulation from NTRU, in: CHES 2017, Vol. 10529, 2017, pp. 232–252.
[38] D.J. Bernstein, C. Chuengsatiansup, T. Lange, C. van Vredendaal, NTRU prime: Reducing attack surface at low cost, in: SAC 2017, Vol. 10719, 2017, pp. 235–260.
[39] J.M. Pollard, The fast Fourier transform in a finite field, Math. Comput. 25 (114) (1971) 365–374.
[40] D.J. Bernstein, Multidigit multiplication for mathematicians, 2001, http://cr.yp.to/papers.html#m3.

[41] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (90) (1965) 297–301.
[42] W.M. Gentleman, G. Sande, Fast Fourier transforms: for fun and profit, in: AFIPS '66, in: AFIPS Conference Proceedings, vol. 29, 1966, pp. 563–578.
[43] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: CRYPTO' 99, Vol. 1666, 1999, pp. 537–554.
[44] D. Hofheinz, K. Hövelmanns, E. Kiltz, A modular analysis of the fujisaki-okamoto transformation, in: TCC 2017, Vol. 10677, 2017, pp. 341–371.
[45] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: CCS '93, 1993, pp. 62–73.
[46] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, M. Zhandry, Random oracles in a quantum world, in: ASIACRYPT 2011, Vol. 7073, 2011, pp. 41–69.
[47] E. Alkim, L. Ducas, T. Pöppelmann, P. Schwabe, Post-quantum key exchange - a new hope, in: USENIX 2016, 2016, pp. 327–343.
[48] R. Kannan, Minkowski's convex body theorem and integer programming, Math. Oper. Res. 12 (3) (1987) 415–440.
[49] S. Bai, S.D. Galbraith, Lattice decoding attacks on binary LWE, in: ACISP 2014, Vol. 8544, 2014, pp. 322–337.
[50] Y. Chen, P.Q. Nguyen, BKZ 2.0: Better lattice security estimates, in: ASIACRYPT 2011, Vol. 7073, 2011, pp. 1–20.
[51] C. Schnorr, M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, Math. Program. 66 (1994) 181–199.
[52] D. Micciancio, O. Regev, Post-quantum cryptography, chapter lattice-based cryptography, Computing 85 (1–2) (2008) 105–125.
[53] D.J. Bernstein, T. Lange, Non-randomness of S-unit lattices, IACR Cryptol. ePrint Arch. (2021) 1428.
[54] Q. Guo, T. Johansson, Faster dual lattice attacks for solving LWE with applications to CRYSTALS, in: ASIACRYPT 2021, Vol. 13093, 2021, pp. 33–62.
[55] MATZOV, Report on the security of LWE: Improved dual lattice attack, 2022, URL: https://zenodo.org/record/6412487.
[56] D. Dachman-Soled, L. Ducas, H. Gong, M. Rossi, LWE with side information: Attacks and concrete security estimation, in: CRYPTO 2020, Vol. 12171, 2020, pp. 329–358.
[57] M.R. Albrecht, F. Göpfert, F. Virdia, T. Wunderer, Revisiting the expected cost of solving uSVP and applications to LWE, in: ASIACRYPT 2017, Vol. 10624, 2017, pp. 297–322.
[58] M.R. Albrecht, V. Gheorghiu, E.W. Postlethwaite, J.M. Schanck, Estimating quantum speedups for lattice sieves, in: ASIACRYPT 2020, Vol. 12492, 2020, pp. 583–613.
[59] L. Ducas, Shortest vector from lattice sieving: A few dimensions for free, in: EUROCRYPT 2018, Vol. 10820, 2018, pp. 125–145.
[60] P.S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, W. Whyte, Choosing ntruencrypt parameters in light of combined lattice reduction and MITM approaches, in: ACNS 2009, Vol. 5536, 2009, pp. 437–455.
[61] NIST, Post-quantum cryptography, round 2 submissions, 2019, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.
[62] NIST, Post-quantum cryptography, round 3 submissions, 2020, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.
[63] J. Buchmann, F. Göpfert, R. Player, T. Wunderer, On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack, in: AFRICACRYPT 2016, Vol. 9646, 2016, pp. 24–43.
[64] N. Howgrave-Graham, A hybrid lattice-reduction and meet-in-the-middle attack against NTRU, in: CRYPTO 2007, Vol. 4622, 2007, pp. 150–169.
[65] T. Wunderer, A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack, J. Math. Cryptol. 13 (1) (2019) 1–26.
[66] P. Nguyen, Boosting the hybrid attack on NTRU: torus LSH, permuted HNF and boxed sphere, in: NIST Third PQC Standardization Conference, 2021.
[67] A. Duc, F. Tramèr, S. Vaudenay, Better algorithms for LWE and LWR, in: EUROCRYPT 2015, Vol. 9056, 2015, pp. 173–202.
[68] M.R. Albrecht, On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL, in: EUROCRYPT 2017, Vol. 10211, 2017, pp. 103–129.
[69] J.H. Cheon, M. Hhan, S. Hong, Y. Son, A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE, IEEE Access 7 (2019) 89497–89506.
[70] T. Espitau, A. Joux, N. Kharchenko, On a hybrid approach to solve small secret LWE, Cryptol. ePrint Arch. (2020).
[71] P. Kirchner, P. Fouque, Revisiting lattice attacks on overstretched NTRU parameters, in: EUROCRYPT 2017, Vol. 10210, 2017, pp. 3–26.
[72] Y. Son, J.H. Cheon, Revisiting the hybrid attack on sparse and ternary secret LWE, IACR Cryptol. ePrint Arch. (2019) 1019.
[73] K. Eisenträger, S. Hallgren, A.Y. Kitaev, F. Song, A quantum algorithm for computing the unit group of an arbitrary degree number field, in: STOC 2014, 2014, pp. 293–302.
[74] S. Hallgren, Fast quantum algorithms for computing the unit group and class group of a number field, in: STOC 2005, 2005, pp. 468–474.
[75] J. Biasse, F. Song, Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields, in: SODA 2016, 2016, pp. 893–902.
[76] H. Cohen, Advanced Topics in Computational Number Theory, Vol. 193, Springer Science & Business Media, 2012.
[77] P. Campbell, M. Groves, D. Shepherd, Soliloquy: A cautionary tale, in: ETSI 2nd Quantum-Safe Crypto Workshop, Vol. 3, 2014, pp. 1–9.
[78] R. Avanzi, H.M. Heys (Eds.), SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers, Vol. 10532, 2017.
[79] A. Pellet-Mary, G. Hanrot, D. Stehlé, Approx-SVP in ideal lattices with pre-processing, in: EUROCRYPT 2019, Vol. 11477, 2019, pp. 685–716.
[80] D.J. Bernstein, S-unit attacks, 2016, URL: https://groups.google.com/g/cryptanalytic-algorithms/c/mCMdsFemzQk/m/3cewE8Q5BwAJ.
[81] M.R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, L. Perret, On the complexity of the BKW algorithm on LWE, Des. Codes Cryptogr. 74 (2) (2015) 325–354.
[82] A. Blum, A. Kalai, H. Wasserman, Noise-tolerant learning, the parity problem, and the statistical query model, J. ACM 50 (4) (2003) 506–519.
[83] Q. Guo, T. Johansson, P. Stankovski, Coded-BKW: Solving LWE using lattice codes, in: CRYPTO 2015, Vol. 9215, 2015, pp. 23–42.
[84] P. Kirchner, P. Fouque, An improved BKW algorithm for LWE with applications to cryptography and lattices, in: CRYPTO 2015, Vol. 9215, 2015, pp. 43–62.
[85] M.R. Albrecht, R. Fitzpatrick, F. Göpfert, On the efficacy of solving LWE by reduction to unique-SVP, in: ICISC 2013, Vol. 8565, 2013, pp. 293–310.
[86] R. Lindner, C. Peikert, Better key sizes (and attacks) for LWE-based encryption, in: CT-RSA 2011, Vol. 6558, 2011, pp. 319–339.
[87] M. Liu, P.Q. Nguyen, Solving BDD by enumeration: An update, in: CT-RSA 2013, Vol. 7779, 2013, pp. 293–309.
[88] P. Ravi, M.F. Ezerman, S. Bhasin, A. Chattopadhyay, S.S. Roy, Will you cross the threshold for me? Generic side-channel assisted chosen-ciphertext attacks on NTRU-based KEMs, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2022 (1) (2022) 722–761.
[89] D.J. Bernstein, A one-time single-bit fault leaks all previous NTRU-HRSS session keys to a chosen-ciphertext attack, in: INDOCRYPT 2022, Vol. 13774, 2022, pp. 617–643.
[90] R. Cramer, L. Ducas, C. Peikert, O. Regev, Recovering short generators of principal ideals in cyclotomic rings, in: EUROCRYPT 2016, Vol. 9666, 2016, pp. 559–585.
[91] R. Cramer, L. Ducas, B. Wesolowski, Short stickelberger class relations and application to ideal-SVP, in: EUROCRYPT 2017, Vol. 10210, 2017, pp. 324–348.
[92] R.T. Moenck, Practical fast polynomial multiplication, in: SYMSAC 1976, ACM, 1976, pp. 136–148.
[93] A. Abdulrahman, J. Chen, Y. Chen, V. Hwang, M.J. Kannwischer, B. Yang, Multi-moduli NTTs for saber on cortex-M3 and cortex-M4, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2022 (1) (2022) 127–151.
[94] C.M. Chung, V. Hwang, M.J. Kannwischer, G. Seiler, C. Shih, B. Yang, NTT multiplication for NTT-unfriendly rings new speed records for saber and NTRU on cortex-M4 and AVX2, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2021 (2) (2021) 159–188.
[95] J.W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - kyber: A CCA-secure module-lattice-based KEM, in: IEEE EuroS&P 2018, 2018, pp. 353–367.
[96] P. Barrett, Implementing the rivest Shamir and adleman public key encryption algorithm on a standard digital signal processor, in: CRYPTO '86, Vol. 263, 1986, pp. 311–323.
[97] G. Seiler, Faster AVX2 optimized NTT multiplication for ring-LWE lattice cryptography, IACR Cryptol. ePrint Arch. (2018) 39, URL: http://eprint.iacr.org/2018/039.
[98] P.L. Montgomery, Modular multiplication without trial division, Math. Comput. 44 (170) (1985) 519–521.
[99] D.J. Bernstein, T. Lange, eBACS: ECRYPT benchmarking of cryptographic systems, 2023, https://bench.cr.yp.to/.
[100] T. Fritzmann, T. Pöppelmann, J. Sepúlveda, Analysis of error-correcting codes for lattice-based key exchange, in: SAC 2018, Vol. 11349, 2018, pp. 369–390.
[101] J. Wang, C. Ling, How to construct polar codes for ring-LWE-based public key encryption, Entropy 23 (8) (2021) 938.
[102] D.J. Bernstein, B.B. Brumley, M. Chen, N. Tuveri, OpenSSLNTRU: Faster post-quantum TLS key exchange, in: USENIX Security 2022, 2022.
[103] N. Aragon, P.S.L.M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, BIKE: Bit flipping key encapsulation (round 3 submission), in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[104] M.R. Albrecht, D.J. Bernstein, T. Chou, C. Cid, J. Gilcher, Classic McEliece: conservative code-based cryptography, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[105] C.A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, Hamming quasi-cyclic (HQC): Third round version, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[106] D. Jao, C.A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, Supersingular isogeny key encapsulation, in: NIST Post-Quantum Cryptography Standardization Process, 2020.
[107] NIST, Post-quantum cryptography, round 4 submissions, 2022, https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.