Compact and Efficient KEMs over NTRU Lattices

Zhichuang Liang, Boyue Fang, Jieyu Zheng, Yunlei Zhao*

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 in having 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 solve this problem by presenting a new family of NTRU-based KEM schemes, referred to 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 with scalable ciphertext compression via only one *single* ciphertext polynomial, and are the first that could outperform {R,M}LWE-based KEM schemes in integrated performance. For instance, compared to Kyber that is currently the only standardized KEM by NIST, on the recommended parameter set CNTR-768 has about 12% smaller ciphertext size while encapsulating 384-bit keys compared to the fixed 256-bit key size of Kyber, security strengthened by $(8, 7)$ bits for classical and quantum security respectively, and significantly lower error probability (2^{-230} of CNTR-768 vs. 2^{-164} of Kyber-768). In particular, CTRU and CNTR admit more flexible key sizes to be encapsulated, specifically $\frac{n}{2}$ where $n \in \{512, 768, 1024\}$ is the underlying polynomial dimension. In comparison with the state-of-the-art AVX2 implementation of Kyber-768, CNTR-768 is faster by 1.9X in KeyGen, 2.6X in Encaps, and 1.2X in Decaps, respectively. When compared to the NIST Round 3 finalist NTRU-HRSS, our CNTR-768 has about 15% smaller ciphertext size, and the security is strengthened by $(55, 49)$ bits for classical and quantum security respectively. As for the AVX2 implementation, CNTR-768 is faster than NTRU-HRSS by 19X in KeyGen, 2.3X in Encaps, and 1.6X in Decaps, respectively. Along the way, we develop new techniques for more accurate error probability analysis, as well as unified implementations with respect to multiple dimensions with unified NTT methods, for NTRU-based KEM schemes over the polynomial ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, which might be of independent interest.

Index Terms

Post-quantum cryptography, Lattice-based cryptography, Key encapsulation mechanism, NTRU, Lattice codes, Number theoretic transform, Integrated performance.

I. 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 when 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 of quantum computing, post-quantum cryptography (PQC) has become a research focus in recent years. There are five main types of post-quantum cryptographic schemes: hash-based, code-based, lattice-based, multivariable-based, and isogeny-based schemes, among which lattice-based cryptography is commonly viewed as amongst the most promising one due to its outstanding integrated performance in security, communication bandwidth, and computational efficiency.

In the post-quantum cryptography standardization competition held by the U.S. National Institute of Standards and Technology (NIST), lattice-based schemes account for 26 out of 64 schemes in the first round [1], 12 out of 26 in the second round [2], and 7 out of 15 in the third round [3]. Recently, NIST announced 4 candidates to be standardized [4], among which 3 schemes are based on lattices. Most of these lattice-based schemes are based on lattices of the following types: plain lattice and algebraically structured lattice (ideal lattice, NTRU lattice, and module lattice). They are mainly instantiated from the following two categories of hardness assumptions. The first category consists of *Learning With Errors* (LWE) [5] and its variants with algebraic structures such as *Ring-Learning With Errors* (RLWE) [6] and *Module-Learning With Errors* (MLWE) [7], as well as the derandomized version of {R,M}LWE: *Learning With Rounding* (LWR) [8] and its variants such as *Ring-Learning With Rounding* (RLWR) [8] and *Module-Learning With Rounding* (MLWR) [9]. The second category is the *NTRU* assumption [10].

NTRU was first proposed by Hoffstein, Pipher and Silverman at the rump session Crypto96 [11], and it survived a lattice attack in 1997 [12]. With some improvements on security, NTRU was published normally in 1998 [10], which is named as NTRU-HPS for presentation simplicity in this work. NTRU-HPS was the first practical public key cryptosystem based on the lattice hardness assumptions over polynomial rings, and there have been many variants of NTRU-HPS such as those proposed in [13]–[18]. Besides being survived attacks and cryptanalysis over 24 years since its introduction, NTRU-based KEM schemes also enjoy many other desirable features. For example, they admit more flexible key sizes to be encapsulated (corresponding to the message space $\mathcal{M}$ in this work), varying according to the degree of the underlying quotient polynomial. In comparison, the KEM schemes based on MLWE and MLWR like Kyber [19] and Saber [20] in the NIST PQC standardization encapsulate keys of fixed size that is restricted to the underlying quotient polynomial that is of degree 256 for Kyber and Saber.

NTRU has played a basic role in many cryptographic protocols, e.g., [21]–[26]. In particular, NTRU-based schemes have achieved impressive success in the NIST PQC standardization. Specifically, Falcon signature scheme [26], which is based

*Corresponding author: ylzhao@fudan.edu.cn. Zhichuang Liang, Boyue Fang, Jieyu Zheng and Yunlei Zhao are with Department of Computer Science, Fudan University, Shanghai, China.
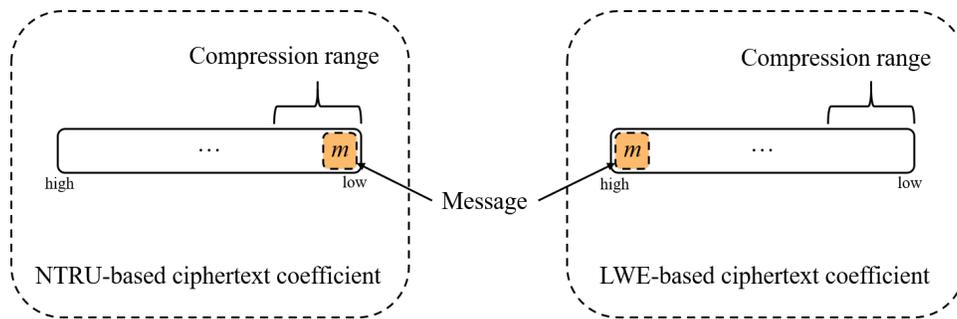
Fig. 1. Differences of message positions between NTRU-based and LWE-based KEMs

on NTRU assumption, is one of the signature candidate to be standardized [4]. NTRU KEM (including NTRU-HRSS and NTRUEncrypt) [16] is one of the seven finalists, and NTRU Prime KEM (including SNTRU Prime and NTRU LPRime) [17] is one of the alternate candidates in the third round of NIST PQC standardization. Although NTRU-based KEM schemes are not chosen to be standardized by NIST, we can not ignore their great potential in PQC research and standardization due to their attractive features. The study and optimization of NTRU-based KEM schemes still deserve further research exploration. Actually, some standardizations have already been including NTRU-based PKE/KEM schemes. The standard IEEE Std 1363.1 [27], which was issued in 2008, standardizes some lattice-based public-key schemes, including NTRUEncrypt. The standard X9.98 [28] standardizes NTRUEncrypt as a part of the X9 standards which are applied to the financial services industry. The European Union's PQCRYPTO project (i.e., Horizon 2020 ICT-645622) [29] is considering another NTRU variant [30] as a potential European standard. In particular, the latest updates of OpenSSH since its 9.0 version released in April 2022 have adopted NTRU Prime, together with X25519 ECDH in a hybrid mode, to prevent "capture now decrypt later" attacks [31].

A. Challenges and Motivations

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

1) Small secret ranges: The first inherent limitation of NTRU-based KEM schemes is that they usually support very narrow secret ranges, typically $\{-1, 0, 1\}$, which inherently limits the security level achievable by NTRU-based KEM schemes [15]–[17]. However, {R,M}LWE-based KEM schemes have the advantage in allowing larger secret ranges for stronger security when using the approximate moduli as in NTRU-based KEM schemes.

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. The importance of reducing bandwidth is self-evident, since low communication bandwidth is friendly to internet protocols (e.g., TLS) and to constrained internet-of-things (IoT) devices. 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. Although {R,M}LWE-based KEM schemes could also choose larger moduli to have zero error probability, they prefer smaller moduli for smaller bandwidth, tolerating negligible error probability instead of insisting on zero error probability. On the other hand, the larger bandwidth is due to the inherent inability to compress ciphertexts of NTRU-based KEM schemes. Below, we briefly explain why ciphertext compression leads to a decryption failure at a high probability for traditional NTRU-based KEM schemes, but the impact of ciphertext compression for {R,M}LWE-based KEM schemes is within some controllable range.

As shown in Figure 1, the initial plaintext message m is encoded into the most significant bits of the second ciphertext term for most {R,M}LWE-based KEM schemes like Kyber, where their first ciphertext term (corresponding to an {R,M}LWE sample) is independent of the second ciphertext term. The randomness and security of the ciphertext are guaranteed by the {R,M}LWE samples. On the contrary, the initial message is encoded into the least significant bits of the ciphertext for traditional NTRU-based KEM schemes. Actually, NTRU-based KEM schemes [10], [16], [30], [32] have ciphertexts of the form $c = phr + m \bmod q$, where p is the message space modulus, h is the public key, r is the randomness, and m is the message to be encrypted. In the decryption process, one could compute $cf \bmod q = pgr + mf$, and clean out the term pgr via reduction modulo p. In order to obtain m, one can multiply the inverse of f modulo p, or directly reduce modulo p if $f = pf' + 1$. This can be viewed as a unidimensional error-correction mechanism.

Compressing the ciphertext of NTRU-based KEM schemes means dropping some least significant bits, which is equivalent to increasing the small error. For {R,M}LWE-based KEM schemes, compressing the first ciphertext term has no impact on the messages. The impact brought by reasonably compressing the second ciphertext term could be eliminated if the total error is within the capacity range of the message-recovering mechanism. However, for traditional NTRU-based KEM schemes,

ciphertext compression will destroy the useful information of the encoded messages in the least significant bits of the ciphertext. Consequently, the initial messages can not be recovered correctly.

3) Weak starting point of security reduction: For most NTRU-based KEM constructions, their chosen ciphertext attack (CCA) security is usually reduced to the one-way (OW-CPA) secure encryption instead of the traditional IND-CPA secure encryption. Above all, IND-CPA security is a strictly stronger security notion than OW-CPA security. Though OW-CPA security can be transformed into IND-CPA security, but at the price of further loosening the reduction bound particularly in the quantum random oracle model (QROM) [32]. One can also have a tight reduction from CCA security to OW-CPA *deterministic* public-key encryption (DPKE), but at the cost of a more complicated decapsulation process [16], [17]. More detailed discussions and clarifications on CCA security reduction of KEM in the ROM and the QROM are presented in Appendix A. As a consequence, it is still desirable for NTRU-based KEM constructions to have security reduction from CCA security to IND-CPA security, as is in {R,M}LWE-based KEM schemes.

4) Complicated key generation: Typically, there are only one or two polynomial multiplications in the encryption process and decryption process of NTRU-based KEM schemes, such that the encryption process and decryption process are as efficient as (and could even be more efficient than) those of {R,M}LWE-based ones. However, for most NTRU-based KEM schemes (with [15], [32] as exceptions), the main efficiency obstacle is from their key generations, since there exits a complicated computation of polynomial inverse for which there does not exit much efficient algorithms for most of the polynomial rings chosen by NTRU-based KEM schemes.

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 have essentially the same or *even better* integrated performance in security, bandwidth, error probability, and computational efficiency *as a whole*, than {R,M}LWE-based KEM schemes?

B. Our Contributions

Our main result shows that NTRU-based KEM schemes can practically have a remarkable integrated performance (in security, bandwidth, error probability, and computational efficiency as a whole), just as and even better than {R,M}LWE-based KEM schemes. In this work, we present such practical constructions, and instantiate such NTRU-based KEM schemes with detailed analysis.

Specifically, in this work, we present new variants of NTRU-based cryptosystem, referred to as CTRU and CNTR for presentation simplicity, which can allow larger secret ranges, achieve scalable ciphertext compression, have CCA provable security reduced directly to IND-CPA security, and have fast implementations. The error probabilities of CTRU and CNTR are low enough, which are usually lower than those of Kyber. They consist of IND-CPA secure public-key encryptions, named CTRU.PKE and CNTR.PKE, and IND-CCA secure key encapsulation mechanisms, named CTRU.KEM and CNTR.KEM constructed through $FO^{\not\perp}_{ID(pk),m}$ that is an enhanced variant of Fujisaki-Okamoto transformation [33], [34] with a short prefix of the public key into the hash function [35].

Our CTRU and CNTR demonstrate novel approaches to constructing NTRU-based schemes. The descriptions of CTRU and CNTR are over NTT-friendly rings of the form $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$. We choose $n \in \{512, 768, 1024\}$ for NIST recommended security levels I, III and V, respectively, with the same modulus $q = 3457$ (that is close to the modulus $q = 3329$ of Kyber) set for all the three dimensions for ease of implementation simplicity and compatability. We may recommend the case of $n = 768$ that could have a moderate balance of post-quantum security and performance for most applications in practice.

1) Efficient constant-time scalable lattice code: Before introducing our proposed schemes, for better capability on recovering message and low enough error probability, we apply and refine the scalable E_8 lattice code based on the works [36]–[39]. 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 $\mathbb{R}^8$ [38]. To avoid the potential timing attacks, we present constant-time encoding and decoding algorithms of the scalable E_8 lattice code. All the conditional statements are implemented by constant-time bitwise operations. Unlike most of other existing error correction codes whose constant-time implementations are inherently difficult, the constant-time implementation of the scalable E_8 lattice code is practical and efficient. We present the scalable E_8 lattice coding algorithms in section III, and give the details about the constant-time implementations in section VII-F.

2) New constructions: The key generation algorithm in CTRU is similar to the exiting NTRU-based KEM schemes such as [10], [16], [17], [32]. CTRU uses $h = g/f$ as its public key and f as its secret key. We develop a novel encryption algorithm which breaks through the limitation of ciphertext compression for NTRU-based KEM, which allows us to compress the ciphertexts in the case of one *single* polynomial. To be specific, we encode every 4-bit message into a scalable E_8 lattice point, and hide its information by adding an RLWE instance, which forms the ciphertext. This way, the message is encoded into the most significant bits of the ciphertext, such that compressing the ciphertext does not destroy the useful information

TABLE I
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 [16]	NTRU	OW-CPA DPKE	$\mathbb{Z}_q[x]/(x^n - 1)$	701	8192	1138	1138	2276	(136,124)	$2^{-\infty}$
SNTRU Prime-761 [17]	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$_{3457}^{768}$ [32]	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 [19]	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 [20]	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}

of the message. As for the decryption algorithm, we multiply the ciphertext polynomial by the secret polynomial, and finally recover the messages correctly with the aid of the decoding algorithm in the scalable E_8 lattice whenever the ℓ_2 norm of the error term is less than the sphere radius of the scalable E_8 lattice. An important point to note here is, different from the most existing NTRU-based KEM schemes such as [10], [16], [32], in CTRU the message space modulus p is removed in the public key h and in the ciphertext c, as it is not needed there to recover the message m with our CTRU construction. The only reserved position for p is the secret key f, which has the form of $f = pf' + 1$. We show that the above steps constitute an IND-CPA secure PKE scheme: CTRU.PKE, based on the NTRU assumption and the RLWE assumption. Finally, we apply the $\text{FO}_{ID(pk),m}^{\not\perp}$ transformation [35] to get the IND-CCA secure CTRU.KEM. The CNTR scheme is a simplified and more efficient variant of CTRU: the noise polynomial is eliminated, and the rounding of the output of the scalable E_8 lattice encoding algorithm is moved. The security of CNTR is based on the NTRU assumption and the RLWR assumption. The detailed construction of CTRU and CNTR are given in section IV. To our knowledge, CTRU and CNTR are the first NTRU-based KEM constructions which bridges 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.

3) Provable security: As for security reduction, our CTRU.PKE (resp., CNTR.PKE) can achieve the IND-CPA security under the NTRU assumption and the RLWE (resp., RLWR) assumption, while most of the existing practical NTRU-based PKEs only achieve OW-CPA security. Note that, the RLWE and RLWR assumptions are only required to achieve IND-CPA security for our schemes, since CTRU.PKE and CNTR.PKE are still OW-CPA secure only based on the NTRU assumption (i.e., without further relying on the RLWE or RLWR assumptions). The reduction advantages of CCA security of our CTRU.KEM and CNTR.KEM are tighter than those of NTTRU [15] and NTRU-C$_{3457}^{768}$ [32]. For example, in the quantum setting, the CCA reduction bound of CTRU.KEM is dominated by $O(\sqrt{q'\epsilon_{CPA}})$, while those of NTTRU and NTRU-C$_{3457}^{768}$ are $O(q'\sqrt{\epsilon_{OW}})$ and $O(q'^{1.5}\sqrt[4]{\epsilon_{OW}})$ respectively, where $\epsilon_{CPA}(\epsilon_{OW})$ is the advantage against the underlying IND-CPA (resp., OW-CPA) secure PKE and q' is the total query number. However, NTRU-HRSS [16], [40] has a tight CCA reduction bound starting from OW-CPA *deterministic* PKE (DPKE), at the cost of more complicated and time-consuming decryption process [16]. In any case, IND-CPA security is a strictly stronger security notion than OW-CPA security.

4) More accurate analysis of error probability: Previously, the work [15] gave a conservative estimation of the error probability, based on the worst case consisting of $\frac{3}{2}n$ terms for each polynomial product coefficient in $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$. In this work, we derive the exact number of the terms of the polynomial product coefficient, and improve the error probability analysis developed in [15], which might be of independent interest. The concrete analysis is provided in section IV-D.

5) Performance and comparisons: By careful evaluation and selection, we provide some parameter sets for CTRU and CNTR, and present the recommended parameter sets in section V-A3. We also make a comprehensive analysis of CTRU and CNTR on the provable security, core-SVP hardness, refined gate-count estimate, dual attack, S-unit attack, BKW attack and side channel attack, etc, in section V. Here, we present brief comparisons between our schemes on the recommended parameters and other prominent practical NTRU-based KEM schemes: NTRU-HRSS [16], [40], SNTRU Prime [17], NTTRU [15] and NTRU-C$_{3457}^{768}$ [32], as well as the NIST standardized candidate Kyber [19] and the NIST Round 3 finalist Saber [20]. The comparisons are summarized in Table I. There, the column "Assumptions" refers to the underlying hardness assumptions. The column "Reduction" means that IND-CCA security is reduced to what kinds of CPA security, where "IND" ("OW") refers to indistinguishability (resp., one-wayness) and "RPKE" ("DPKE") refers to randomized (resp., deterministic) public-

key encryptions. "Rings" refers to the underlying polynomial rings. The column "n" means the total dimension of algebraically structured lattices. "q" is the modulus. The public key sizes $|pk|$, ciphertext sizes $|ct|$, 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 if the two hardness problems, say NTRU and RLWE/RLWR, have different security values. The column "δ" indicates the error probabilities, where the error probabilities of NTTRU and NTRU-C$_{3457}^{768}$ are re-tested according to the accurate measurement methodology discussed in section IV-D.

From the comparisons, CNTR has the smallest bandwidth and the strongest security guarantees among all the practical NTRU-based KEM schemes. For example, when compared to the NIST Round 3 finalist NTRU-HRSS [16], our CNTR-768 has about 15% smaller ciphertext size, and its security is strengthened by $(55, 49)$ bits for classical and quantum security, respectively. The error probabilities of CNTR are set according to the security level targeted by each set of parameters, which can be viewed as negligible in accordance with the security level. When compared to Kyber-768 [19] that is standardized by NIST, CNTR-768 has about 12% smaller ciphertext size, and its security is strengthened by $(8, 7)$ bits for classical and quantum security, respectively. For all the three recommended parameter sets, the error probabilities of CNTR are significantly lower than those of Kyber (e.g., 2^{-230} of CNTR-768 vs. 2^{-164} of Kyber-768). To the best of our knowledge, CNTR is the first NTRU-based KEM that could outperform Kyber in the integrated performance by considering security, bandwidth, error probability, and computational efficiency as a whole. We also would like to stress that we do not know how to have the well balance achieved by CTRU/CNTR by simply adjusting parameters for the existing NTRU-based KEM schemes. Another significant point is that CTRU and CNTR admit more flexible key sizes to be encapsulated, i.e., $n/2$-bit shared keys according to the polynomial rings we used, but Kyber and Saber can only encapsulate fixed 256-bit shared keys.

6) Unified NTT: The NTT-based polynomial operations over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ are very efficient. However, as the dimension n varies with CTRU and CNTR, we have to equip with multiple NTT algorithms with different input/output lengths in accordance with $n \in \{512, 768, 1024\}$. This brings inconvenient issues for software implementations and especially for hardware implementations. In this work, we overcome this problem by presenting the methodology of using a unified NTT technique to compute NTTs over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ for all $n \in \{512, 768, 1024\}$ with $q = 3457$, which might be of independent interest. Technically speaking, we split $f \in \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ into $\alpha \in \{2, 3, 4\}$ sub-polynomials of lower degrees, each of which is in $\mathbb{Z}_q[x]/(x^{256} - x^{128} + 1)$. We then design a 256-point unified NTT based on the ideas from [15], [41], and apply it to each sub-polynomial. Finally, their intermediate NTT results are combined to generate the final results. In this case, in order to obtain the public key (the quotient of two n-dimension polynomials), we need to compute the inversions in the rings of the form $\mathbb{Z}_q[x]/(x^{2\alpha} - \zeta)$, where ζ is some primitive root of unity in $\mathbb{Z}_q$. We use Cramer's Rule [42] to compute the inverse of polynomials of low degree. More details are presented in section VI.

7) Implementation and benchmark: We provide portable C implementation and optimized AVX2 implementation for CTRU-768 and CNTR-768. More details and discussions about the implementation can be seen in section VII. We perform benchmark comparisons with the related lattice-based KEM schemes and some prominent non-lattice-based KEM schemes. The benchmark comparisons show that the encapsulation and decapsulation algorithms of our schemes are among the most efficient. As for the optimized AVX2 implementations, CTRU-768 is faster than NTRU-HRSS by 23X in KeyGen, 2.1X in Encaps, and 1.6X in Decaps, respectively; CNTR-768 is faster than NTRU-HRSS by 19X in KeyGen, 2.3X in Encaps, and 1.6X in Decaps, respectively. When compared to the state-of-the-art AVX2 implementation of Kyber-768, CTRU-768 is faster by 2.3X in KeyGen, 2.3X in Encaps, and 1.2X in Decaps, respectively; CNTR-768 is faster by 1.9X in KeyGen, 2.6X in Encaps, and 1.2X in Decaps, respectively. The benchmark comparisons are referred to section VIII.

C. Related Work

In recent years, many NTRU variants have been proposed. Jarvis and Nevins [13] presented a new variant of NTRU-HPS [10] 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. [14] 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. [40] 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 [16]. Bernstein et al. [43] 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 use some rings with less special structures, i.e., $\mathbb{Z}_q[x]/(x^n - x - 1)$, where both n and q are primes.

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. [32] generalized the rings $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ with various n for flexible parameter selection. But all of them follow the similar structure of NTRU-HPS and do not support ciphertext compression.

Recently, Fouque et al. [18] proposed a new NTRU variant named BAT. It shares many similarities with Falcon signature [26] 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 [32]. 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 $\mathbb{Z}_q[x]/(x^n + 1)$, 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 lacks of parameter set for level III, which, however, is the aimed and recommended security level for most lattice-based KEM schemes like Kyber [19] and our schemes. Although BAT has an advantage of bandwidth, its key generation is 1,000 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.

II. PRELIMINARIES

A. 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 $\mathcal{R}$ (or $\mathcal{R}_q$) can be represented in the form of power series: $f = \sum_{i=0}^{n-1} f_i x^i$, or in the form of vector: $f = (f_0, f_1, \ldots, f_{n-1})$, where $f_i \in \mathbb{Z}$ (or $f_i \in \mathbb{Z}_q$), $i = 0, 1, \ldots, n-1$. A function $\epsilon : \mathbb{N} \to [0, 1]$ is negligible, if $\epsilon(\lambda) < 1/\lambda^c$ holds for any positive c and sufficiently large λ. Denote a negligible function by $negl$.

Cyclotomics. More details about cyclotomics can be found in [44]. Let m be a positive integer, $\xi_m = \exp(\frac{2\pi i}{m})$ be a m-th root of unity. The m-th cyclotomic polynomial $\Phi_m(x)$ is defined as $\Phi_m(x) = \prod_{j=1, \gcd(j,m)=1}^{m} (x - \xi_m^j)$. It is a monic irreducible polynomial of degree $\phi(m)$ in $\mathbb{Z}[x]$, where ϕ is the Euler function. The m-th cyclotomic field is $\mathbb{Q}(\xi_m) \cong \mathbb{Q}[x]/(\Phi_m(x))$ and its corresponding ring of integers is exactly $\mathbb{Z}[\xi_m] \cong \mathbb{Z}[x]/(\Phi_m(x))$. Most of cryptographic schemes based on algebraically structured lattices are defined over power-of-two cyclotomic rings, $\mathbb{Z}[x]/(x^n + 1)$ and $\mathbb{Z}_q[x]/(x^n + 1)$, where $n = 2^e$ is a power of two such that $x^n + 1$ is the 2^{e+1}-th cyclotomic polynomial. We use non-power-of-two cyclotomic rings $\mathbb{Z}[x]/(x^n - x^{n/2} + 1)$ and $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, where $n = 3^l \cdot 2^e, l \geq 0, e \geq 1$ throughout this paper and in this case $x^n - x^{n/2} + 1$ is the $3^{l+1} \cdot 2^e$-th cyclotomic polynomial.

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

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

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.

B. 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 $\text{Enc}(pk, m; coin)$ 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 $\text{E}[\max_{m \in \mathcal{M}} \Pr[\text{Dec}(sk, \text{Enc}(pk, m)) \neq m]] < \delta$ where the expectation is taken over $(pk, sk) \leftarrow \text{KeyGen}$ 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 $\mathbf{Adv}_{\text{PKE}}^{\text{IND-CPA}}(\mathsf{A}) =$

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

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 inputs 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 $\Pr[\text{Decaps}(sk, c) \neq K : (c, K) \leftarrow \text{Encaps}(pk)] < \delta$ where the probability is taken over $(pk, sk) \leftarrow \text{KeyGen}$ 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 $\mathbf{Adv}_{\text{KEM}}^{\text{IND-CCA}}(\mathsf{A}) =$

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

C. Hardness Assumptions

As the lattice cryptography evolved over the decades, the security of NTRU and its variants can be naturally viewed as two assumptions. One is the *NTRU* assumption [10], and the other is the *Ring-Learning with error* (RLWE) assumption [6], which are listed as follows. In some sense, the NTRU assumption can be viewed as a special case of the RLWE assumption. More details about NTRU cryptosystem and its applications can be seen in the excellent survey [45].

Definition 1 (NTRU assumption [10]) *Let Ψ be a distribution over a polynomial ring R. Sample f and g according to Ψ, and f is invertible in R. Let $h = g/f$. The decisional NTRU assumption states that h is indistinguishable from a uniformly-random element in R. More precisely, the decisional NTRU assumption is hard if the advantage $\mathbf{Adv}_{R,\Psi}^{NTRU}(\mathsf{A})$ of any probabilistic polynomial time (PPT) adversary A is negligible, where $\mathbf{Adv}_{R,\Psi}^{NTRU}(\mathsf{A}) =$*

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

Definition 2 (RLWE assumption [6]) *Let Ψ be a distribution over a polynomial ring R. The (decisional) Ring-Learning with error (RLWE) assumption 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 $\mathbf{Adv}_{R,\Psi}^{RLWE}(\mathsf{A})$ of any probabilistic polynomial time adversary A is negligible, where $\mathbf{Adv}_{R,\Psi}^{RLWE}(\mathsf{A}) =$*

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

Definition 3 (RLWR assumption [8]) *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) assumption 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 $\mathbf{Adv}_{R,\Psi}^{RLWR}(\mathsf{A})$ of any probabilistic polynomial time adversary A is negligible, where $\mathbf{Adv}_{R,\Psi}^{RLWR}(\mathsf{A}) =$*

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

D. Number Theoretic Transform

From a computational point of view, the fundamental and also 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 [46]. NTT is the most efficient method for computing polynomial multiplication of high degrees, due to its quasilinear complexity $O(n \log n)$. The complete NTT-based multiplication with respect to f and g is $INTT(NTT(f) \circ NTT(g))$, where NTT is the forward transform, $INTT$ is the inverse transform and "$\circ$" is the point-wise multiplication.

The FFT trick [47] is a fast algorithm to compute NTT, via 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 tirck and inverse FFT tirck can be conducted via Cooley-Tukey butterfly [48] and Gentleman-Sande butterfly [49], 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 more complicated, given 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 step.

III. THE LATTICE CODE

Before introducing our proposed NTRU-based KEM schemes, we present a simple and efficient lattice code. The motivation is that a dense lattice with efficient decoding algorithm is needed in our construction for better efficiency on recovering message and low enough error probability. The coding algorithms should satisfy the following conditions.

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

We note that an 8-dimension lattice, named E_8 lattice (see [37, Chapter 4]) could satisfy the above requirements to some extent. 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 $\mathbb{R}^8$ [38]. As for the efficiency on coding, there have been simple executable encoding and decoding algorithms of the E_8 lattice in [36], [37], [39]. However, the known coding algorithms in [36], [37] cannot be directly applied here. To work in our setting, we need to specify a one-to-one mapping from binary strings to the E_8 lattice points to encode messages. The work [39] specifies such a mapping by choosing a basis, but its implementation of decoding algorithm is not constant-time, since in multiple places there is data flow from secret polynomials into the variables which are then used as lookup indexes. In this work, we provide a scalable version of the E_8 lattice in the spirit of [39] as well as a corresponding constant-time implementation of its decoding algorithm, which can also transform the lattice points to the binary strings without involving Gaussian Elimination.

A. Scalable E_8 Lattice Code

The scalable E_8 lattice is constructed from the Extended Hamming Code with respect to dimension 8, which is defined as $H_8 = \{\mathbf{c} \in \{0,1\}^8 \mid \mathbf{c} = \mathbf{z}\mathbf{H} \bmod 2, \mathbf{z} \in \{0,1\}^4\}$ 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 up most three rows of $\mathbf{H}$. Then the scalable E_8 lattice (named E_8' lattice) is constructed as

$$E_8' = \lambda \cdot [C \cup (C + \mathbf{c})] \subset [0, \lambda]^8,$$

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

1) Encoding algorithm: The encoding algorithm of the E_8' lattice (see Algorithm 1) is to calculate $\lambda \cdot (\mathbf{k}\mathbf{H} \bmod 2)$, given a 4-bit binary string $\mathbf{k}$.

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

1: $\mathbf{v} := \lambda \cdot (\mathbf{k}\mathbf{H} \bmod 2) \in [0, \lambda]^8$
2: **return** $\mathbf{v}$

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

1: Recall that $\mathbf{c} := (0, 1, 0, 1, 0, 1, 0, 1)$
2: $(\mathbf{k}_0, \text{TotalCost}_0) := \text{Decode}_{C'}(\mathbf{x})$
3: $(\mathbf{k}_1, \text{TotalCost}_1) := \text{Decode}_{C'}(\mathbf{x} - \lambda \cdot \mathbf{c})$
4: $b := \arg\min\{\text{TotalCost}_0, \text{TotalCost}_1\}$
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 $\text{Decode}_{C'}(\mathbf{x} \in \mathbb{R}^8)$

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

---

2) Decoding algorithm: Given any $\mathbf{x} \in \mathbb{R}^8$, the decoding algorithm is to find the solution of the closest vector problem (CVP) of $\mathbf{x}$ in the E_8' lattice, which is denoted by $\lambda \cdot \mathbf{k}'\mathbf{H} \bmod 2$, and it outputs the 4-bit string $\mathbf{k}'$. To solve the CVP of $\mathbf{x} \in \mathbb{R}^8$ in the E_8' lattice, we turn to solve the CVP of $\mathbf{x}$ and $\mathbf{x} - \lambda\mathbf{c}$ in the lattice $C' = \lambda \cdot C$. The one that has smaller distance is the final answer.

We briefly introduce the idea of solving the CVP in the lattice C' here. 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. Then we choose the secondly closest vector, $\lambda \cdot (k_0', k_0', k_1', k_1', k_2', k_2', k_3', k_3')$, where there will be at most 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 given in Algorithm 2, along with Algorithm 3 as its subroutines. Note that in Algorithm 3, $mind$ and $mini$ are set to store the minimal difference of the components and the corresponding index, respectively.

Finally, $\text{Decode}_{C'}$ in Algorithm 2 will output 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 E_8' lattice. Since the lattice point has the form of $\lambda \cdot (\mathbf{k}\mathbf{H} \bmod 2)$, the decoding result $\mathbf{k}$ can be obtained by tweaking the solution of the CVP in the E_8' lattice, as in line 5 and line 6 in Algorithm 2. The details about constant-time implementation of decoding algorithms are presented in section VII-F.

B. Bound of Correct Decoding

Theorem 1 gives a bound of correct decoding w.r.t. Algorithm 2. Briefly speaking, for any 8-dimension vector which is close enough to the given E_8' lattice point under the metric of ℓ_2 norm, it can be decoded into the same 4-bit string that generates the lattice point. This theorem is helpful when we try to recover the targeted message from the given lattice point with error terms in our schemes.

Theorem 1 (Correctness bound of the scalable E_8 lattice decoding) *For any given* $\mathbf{k}_1 \in \{0,1\}^4$*, denote* $\mathbf{v}_1 := \text{Encode}_{E_8'}(\mathbf{k}_1)$. *For any* $\mathbf{v}_2 \in \mathbb{R}^8$*, denote* $\mathbf{k}_2 := \text{Decode}_{E_8'}(\mathbf{v}_2)$. *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. Thus, the radius of sphere packing in the* E_8' *lattice we used is* $\frac{1}{2}\sqrt{4 \cdot \lambda^2} = \lambda$*. As shown 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* E_8' *lattice is* $\mathbf{v}_1$*. Since* $\text{Decode}_{E_8'}$ *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* $\mathbf{k}_2$*, i.e.,* $\mathbf{v}_1 = \lambda \cdot (\mathbf{k}_2\mathbf{H} \bmod 2)$*, which indicates that* $\mathbf{k}_1 = \mathbf{k}_2$. □

IV. CONSTRUCTION AND ANALYSIS

In this section, we propose our two new cryptosystems based on NTRU lattice, named CTRU and CNTR, both of which contain an IND-CPA secure public-key encryption and an IND-CCA secure key encapsulation mechanism. CTRU and CNTR have similar forms of public key and secret key to those of the traditional NTRU-based KEM schemes, but the method to recover message in CTRU and CNTR is significantly different from them. With our construction, CTRU and CNTR will achieve integrated performance in security, bandwidth, error probability and computational efficiency as a whole.

A. CTRU: Proposal Description

Our CTRU.PKE scheme is specified in Algorithm 4-6. Restate that $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$, where n and q are the ring parameters. Let q_2 be the modulus, which is usually set to be a power of two that is smaller than q. Let p be the message space modulus, satisfying $\gcd(q, p) = 1$. We mainly focus on $p = 2$ for the odd modulus q in this paper. Let Ψ_1 and Ψ_2 be the distributions over $\mathcal{R}$. For presentation simplicity, the secret terms, f', g are taken from Ψ_1, and r, e are taken from Ψ_2. Actually, Ψ_1 and Ψ_2 can be different distributions. Let $\mathcal{M} = \{0,1\}^{n/2}$ denote the message space, where each $m \in \mathcal{M}$ can be seen as a $\frac{n}{2}$-dimension polynomial with coefficients in $\{0,1\}$.

Algorithm 4 CTRU.PKE.KeyGen()

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: $c := \left\lfloor \frac{q_2}{q}(\sigma + \lfloor \text{PolyEncode}(m) \rceil) \right\rceil \mod q_2$
4: **return** c

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

1: $m := \text{PolyDecode}\,(cf \mod {}^{\pm}q_2)$
2: **return** m

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

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. That is, the encoding algorithm works over $E_8' := \frac{q}{2} \cdot [C \cup (C + \mathbf{c})]$. The PolyEncode algorithm splits each $m \in \mathcal{M}$ into some quadruples, each of which will be encoded via $\text{Encode}_{E_8'}$. As for PolyDecode algorithm, the decoding algorithm works over the lattice $E_8'' := \frac{q_2}{2} \cdot [C \cup (C + \mathbf{c})]$. It splits $v \in \mathcal{R}_{q_2}$ into some octets, each of which will be decoded via $\text{Decode}_{E_8''}$. The final message m can be recovered by combining all the 4-bit binary strings output by $\text{Decode}_{E_8''}$.

We construct our CTRU.KEM=(Keygen, Encaps, Decaps) by applying $\text{FO}_{ID(pk),m}^{\not\perp}$, a variant of Fujisaki-Okamoto (FO) transformation [33], [34] aimed for the strengthened IND-CCA security in multi-user setting [35]. 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 $\mathcal{K}$ is the shared key space of CTRU.KEM and $\mathcal{COINS}$ is the randomness space of CTRU.PKE.Enc. Note that we make explicit 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: $(pk, sk) \leftarrow$ CTRU.PKE.KeyGen()
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: $c :=$ CTRU.PKE.Enc($pk, m; coin$)
4: **return** (c, K)

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

1: $m' :=$ 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 $c =$ CTRU.PKE.Enc($pk, m'; coin'$) **then**
5:     **return** K'
6: **else**
7:     **return** $\tilde{K}$
8: **end if**

B. CNTR: Proposal Description

CNTR is a simple variant of CTRU, which is based on the NTRU assumption [10] and the RLWR assumption [8]. Here, CNTR stands for "Compact NTRu based on RLWR". CNTR is also usually the abbreviation of container, which has the meaning CNTR is an economically concise yet powerful key encapsulation mechanism.

Our CNTR.PKE scheme 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 here are the same as those of CTRU.

Algorithm 12 CNTR.PKE.KeyGen()

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: $c := \left\lfloor \dfrac{q_2}{q}(\sigma + \text{PolyEncode}(m)) \right\rceil \mod q_2$
4: **return** c

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

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

Unlike the encryption algorithm of CTRU (see Algorithm 5), that of CNTR has the following distinctions: (1) the noise polynomial is eliminated; (2) the rounding of the PolyEncode algorithm is moved.

Our CNTR.KEM scheme is constructed in the same way as CTRU.KEM, via the FO transformation $\text{FO}_{ID(pk),m}^{\not{\perp}}$ [35]. The algorithms of CNTR.KEM can be referred to Algorithm 9-11.

C. 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 exits 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*

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

$\square$

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 $\left[\lfloor \frac{q_2}{q}u \rceil - \frac{q_2}{q}u\right] \bmod {}^{\pm}q_2$. Let Err_i be the i-th octet of $gr + ef + \vec{1} \cdot f' + \frac{q}{q_2}\varepsilon f$, where $\vec{1}$ is the polynomial with each coefficient being 1. Denote $1 - \delta = Pr\left[\|Err_i\|_{q,2} < \frac{q}{2} - \sqrt{2}\right]$. 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 have*

$$\begin{aligned} m &= PolyDecode_{E_8''}\left(cf \bmod {}^{\pm}q_2\right) \\ &= PolyDecode_{E_8'}\left((\frac{q}{q_2}c)f \bmod {}^{\pm}q\right), \end{aligned} \quad (1)$$

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 the 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

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

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 i-th 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 indicated 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 $\|(\frac{s}{2})_i\|_{q,2} \leq \sqrt{2}$ 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 $\left(\lfloor \frac{q_2}{q}hr \rceil - \frac{q_2}{q}hr\right) \bmod {}^{\pm}q_2$. Let Err_i be the i-th octet of $gr + \frac{q}{q_2}\varepsilon f$. Denote $1 - \delta = Pr\left[\|Err_i\|_{q,2} < \frac{q}{2}\right]$. Then, the error probability of CNTR is δ.*

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

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

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 $\lfloor \frac{q_2}{q} hr \rceil$ indicates 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

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

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 $\frac{q}{2}$, which we denote by $\frac{q}{2}(\mathbf{k}_i \mathbf{H} \bmod 2)$. From Theorem 1 we know that to recover $\mathbf{k}_i$, it should hold $\|Err_i\|_{q,2} < \frac{q}{4}$, where Err_i is the i-th octet of $gr + \frac{q}{q_2} \varepsilon f$. □

D. More Accurate Form of Polynomial Product and Error Probability Analysis over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$

The more accurate form of polynomial product and more accurate corresponding analysis of error probability over the ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ are presented in detail here. 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 presented via a matrix-vector multiplication

$$h = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{L} - \mathbf{U} & -\mathbf{F} - \mathbf{U} \\ \mathbf{F} + \mathbf{U} & \mathbf{F} + \mathbf{L} \end{bmatrix} \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \end{bmatrix}, \tag{4}$$

where $\mathbf{F}, \mathbf{L}, \mathbf{U}$ are the $n/2$-dimension Toeplitz matrices as follows:

$$\mathbf{F} = \begin{bmatrix} f_{n/2} & f_{n/2-1} & \cdots & f_1 \\ f_{n/2+1} & f_{n/2} & \cdots & f_2 \\ \vdots & \vdots & \ddots & \vdots \\ f_{n-1} & f_{n-2} & \cdots & f_{n/2} \end{bmatrix}, \ \mathbf{L} = \begin{bmatrix} f_0 & 0 & \cdots & 0 \\ f_1 & f_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ f_{n/2-1} & f_{n/2-2} & \cdots & f_0 \end{bmatrix}, \ \mathbf{U} = \begin{bmatrix} 0 & f_{n-1} & \cdots & f_{n/2+1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_{n-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

However, in [15], to bound the error probability, they consider the worst case consisting of the sums of $\frac{3}{2}n$ terms of the form $f_i g_j$ for each coefficient of h, which will give the most conservative estimation result. In the following, we will derive the exact number of the terms of the polynomial product coefficient, and improve the original error probability analysis developed in [15], instead of roughly considering the worst case of using $\frac{3}{2}n$ terms. Firstly, focusing on the arithmetic operations in $\mathbb{Z}_q$, the product of f and g in $\mathbb{Z}_q[x]$ is written 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 have

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

Then consider the form of $\sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le \frac{3n}{2} - 2}} f_i g_j x^k$. Actually, we have

$$\sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le \frac{3n}{2} - 2}} f_i g_j x^k = \sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+\frac{n}{2}, \\ n \le k \le \frac{3n}{2} - 2}} f_i g_j x^k$$

$$= \sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+\frac{3n}{2}, \\ 0 \le k \le \frac{n}{2} - 2}} f_i g_j x^k (x^{\frac{n}{2}} - 1) \tag{6}$$

$$= \sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le n-1}} f_i g_j x^k + \sum_{\substack{i+j=k+n, \\ \frac{n}{2} \le k \le n-2}} f_i g_j x^k - \sum_{\substack{i+j=k+\frac{3n}{2}, \\ 0 \le k \le \frac{n}{2} - 2}} f_i g_j x^k.$$

Therefore, for each coefficient h_k, there will be

$$h_k = \sum_{\substack{i+j=k, \\ 0 \le k \le n-1}} f_i g_j - \sum_{\substack{i+j=k+n, \\ 0 \le k \le n-2}} f_i g_j + \sum_{\substack{i+j=k+\frac{n}{2}, \\ \frac{n}{2} \le k \le n-1}} f_i g_j + \sum_{\substack{i+j=k+n, \\ \frac{n}{2} \le k \le n-2}} f_i g_j - \sum_{\substack{i+j=k+\frac{3n}{2}, \\ 0 \le k \le \frac{n}{2}-2}} f_i g_j. \tag{7}$$

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

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

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

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

Note the symmetry of the distribution, here we only focus on the number of terms of the form $f_i g_j$, which are listed as follows: For $0 \le k \le \frac{n}{2} - 2$, h_k has $\frac{3n}{2} - k - 1$ terms of the form $f_i g_j$; For $\frac{n}{2} - 2 < k \le \frac{n}{2}$, h_k has n terms of the form $f_i g_j$; For $\frac{n}{2} < k \le n - 1$, h_k has $\frac{3n}{2}$ terms of the form $f_i g_j$. According to the results mentioned above, the exact number of terms of the polynomial product coefficient is provided, based on which we can compute more accurate error probabilities of our schemes, as well as NTTRU and NTRU-C$_{3457}^{768}$. Thus, the error probabilities of CTRU, CNTR, NTTRU and NTRU-C$_{3457}^{768}$ in this work are estimated by using a Python script according to our methodology. The whole results of CTRU and CNTR for the selected parameters are given in Table II and Table III.

E. Provable Security

We prove that CTRU.PKE is IND-CPA secure under the NTRU assumption and the RLWE assumption, and CNTR.PKE is IND-CPA secure under the NTRU assumption and the RLWR assumption.

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

Proof *We complete our proof through a sequence of games* G_0*,* G_1 *and* G_2*. Let* A *be the adversary against the IND-CPA security experiment. Denote by* $Succ_i$ *the event that* A *wins in the game* G_i*, that is,* A *outputs* b' *such that* $b' = b$ *in* G_i*.*

Game G_0*. This game is the original IND-CPA security experiment. Thus,* $Adv_{CTRU.PKE}^{IND\text{-}CPA}(A) = |Pr[Succ_0] - 1/2|$*.*

Game G_1*. This game is the same as* G_0*, except that replacing the public key* $h = g/f$ *in the KeyGen by* $h \xleftarrow{\$} \mathcal{R}_q$*. To distinguish* G_1 *from* G_0 *is equivalent to solve an NTRU problem. More precisely, there exits an adversary* B *with the same running time as that of* A *such that* $|Pr[Succ_0] - Pr[Succ_1]| \le Adv_{\mathcal{R}_q, \Psi_1}^{NTRU}(B)$*.*

Game G_2*. This game is the same as* G_1*, except that using uniformly random elements from* $\mathcal{R}_q$ *to replace* σ *in the encryption. Similarly, there exits an adversary* C *with the same running time as that of* A *such that* $|Pr[Succ_1] - Pr[Succ_2]| \le Adv_{\mathcal{R}_q, \Psi_2}^{RLWE}(C)$*.*

In Game G_2*, for any given* m_b*, according to Algorithm 5 and 7,* m_b *is split into* $n/8$ *quadruples. Denote the* i*-th quadruple of* m_b *as* $m_b^{(i)}$*, which will later be operated to output the* i*-th 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* $m_b^{(i)}$*,* $\lfloor Encode_{E_8'}(m_b^{(i)}) \rceil$ *is fixed. Based on the uniform randomness of* σ *in* $\mathcal{R}_q$ *, its* i*-th octet (denoted as* $\sigma^{(i)}$*) is uniformly random in* $\mathbb{Z}_q^8$*, so is* $\sigma^{(i)} + \lfloor Encode_{E_8'}(m_b^{(i)}) \rceil$*. 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* $Pr[Succ_2] = 1/2$*.*

Combining all the probabilities finishes the proof. $\square$

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

Proof *We complete our proof through a sequence of games* G_0*,* G_1 *and* G_2*. Let* A *be the adversary against the IND-CPA security experiment. Denote by* $Succ_i$ *the event that* A *wins in the game* G_i*, that is,* A *outputs* b' *such that* $b' = b$ *in* G_i*.*

Game G_0*. This game is the original IND-CPA security experiment. Thus,* $Adv_{CNTR.PKE}^{IND\text{-}CPA}(A) = |Pr[Succ_0] - 1/2|$*.*

Game G_1*. This game is the same as* G_0*, except that replacing the public key* $h = g/f$ *in the KeyGen by* $h \xleftarrow{\$} \mathcal{R}_q$*. To distinguish* G_1 *from* G_0 *is equivalent to solve an NTRU problem. More precisely, there exits an adversary* B *with the same running time as that of* A *such that* $|Pr[Succ_0] - Pr[Succ_1]| \le Adv_{\mathcal{R}_q, \Psi_1}^{NTRU}(B)$*.*

Game $\boldsymbol{G}_2$. This game is the same as $\boldsymbol{G}_1$, except that using random elements from $\mathcal{R}_{q_2}$ to replace $\lfloor \frac{q_2}{q} hr \rceil$ of $c = \lfloor \frac{q_2}{q} hr \rceil + \frac{q_2}{2} s \mod q_2$ (see the formula (3)) in the encryption where the term $\frac{q_2}{2} s$ implies 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 exits an adversary C with the same running time as that of A such that $|Pr[\boldsymbol{Succ}_1] - Pr[\boldsymbol{Succ}_2]| \le \boldsymbol{Adv}_{\mathcal{R}, \Psi_2}^{RLWR}(\mathsf{C})$.

In Game $\boldsymbol{G}_2$, 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 $\boldsymbol{G}_2$. We have $Pr[\boldsymbol{Succ}_2] = 1/2$.

Combining all the probabilities finishes the proof. $\qquad\square$

By applying the $\mathsf{FO}_{ID(pk),m}^{\not\perp}$ transformation and adapting the results given in [35], we have the following results on CCA security of CTRU.KEM and CNTR.KEM in the random oracle model (ROM) [50] and the quantum random oracle model (QROM) [51].

Theorem 6 (IND-CCA security in the ROM and QROM [35]) *Let ℓ be the min-entropy [33] 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 exits a (quantum) adversary B with roughly the same running time of A, such that:*

- *In the ROM, it holds that $\boldsymbol{Adv}_{CTRU/CNTR.KEM}^{IND\text{-}CCA}(\mathsf{A}) \le$*

$$2\left(\boldsymbol{Adv}_{CTRU/CNTR.PKE}^{IND\text{-}CPA}(\mathsf{B}) + \frac{q_H + 1}{|\mathcal{M}|}\right) + \frac{q_H}{2^\iota} + (q_H + q_D)\delta + \frac{1}{2^\ell};$$

- *In the QROM, it holds that $\boldsymbol{Adv}_{CTRU/CNTR.KEM}^{IND\text{-}CCA}(\mathsf{A}) \le$*

$$2\sqrt{q_{HD}\boldsymbol{Adv}_{CTRU/CNTR.PKE}^{IND\text{-}CPA}(\mathsf{B})} + \frac{4q_{HD}}{\sqrt{|\mathcal{M}|}} + \frac{4(q_H + 1)}{\sqrt{2^\iota}} + 16q_{HD}^2\delta + \frac{1}{|\mathcal{M}|} + \frac{1}{2^\ell},$$

where $q_{HD} := q_H + q_D + 1$.

The detailed discussions and clarifications on CCA security reduction of KEM in the ROM and the QROM are given in Appendix A.

F. Discussions and Comparisons

The rings. As in [15], [32], we can choose non-power-of-two cyclotomics $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ with respect to $n = 3^l \cdot 2^e$ and prime q, This type of ring allows very fast NTT-based polynomial multiplication if the ring moduli are set to be NTT-friendly. Moreover, it also allows very flexible parameter selection, since there are many integral n of the form $3^l \cdot 2^e, l \ge 0, e \ge 1$.

The message modulus. Note that the modulus p is removed in the public key h (i.e., $h = g/f$) and in the ciphertext c of our CTRU and CNTR, for the reason that p is not needed in h and c to recover the message m in our construction. The only reserved position of p is the secret key f, which has the form of $f = pf' + 1$. Since $\gcd(q, p) = 1$ is required for NTRU-based KEM schemes, we can use $p = 2$ instead of $p = 3$. A smaller p can lead to a lower error probability. Note that for other NTRU-based KEM schemes with power-of-two modulus q as in NTRU-HRSS [16], p is set to be 3 because it is the smallest integer co-prime to the power-of-2 modulus.

The decryption mechanism. Technically speaking, the ciphertext of NTRU-based PKE schemes [10], [16], [30], [32] has the form of $c = phr + m \mod q$. One can recover the message m through a unidimensional error-correction mechanism, after computing $cf \mod q$. Instead, we use a multi-dimension coding mechanism. We encode each 4-bit message into a lattice point in the scalable E_8 lattice. They can be recovered correctly with the aid of the scalable E_8 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 the step of compressing ciphertext in CNTR is described in line 3 in Algorithm 13. Both of them can be mathematically written as $y := \lfloor \frac{q_2}{q} x \rceil \mod q_2, x \in \mathbb{Z}_q$ for each component. The sufficient condition of ciphertext compression is that $\lceil \log(q_2) \rceil < \lceil \log(q) \rceil$, such that y occupies less bits than x. But, the capacity of ciphertext compression is invalid under the condition of $\lceil \log(q_2) \rceil \ge \lceil \log(q) \rceil$, especially $q_2 = q$. To the best of our knowledge, CTRU and CNTR are the first NTRU-based KEM schemes with scalable ciphertext compression via a single polynomial. The ciphertext modulus q_2 is adjustable, depending on the bits to be dropped. Recall that most NTRU-based KEM schemes [10], [16], [32] fail to compress ciphertext due to the fact that the message information would be destroyed once the ciphertext is compressed.

The role of noise e and rounding. We remark that the noise e in line 2 in Algorithm 5 is only necessary for basing the IND-CPA security on the RLWE assumption for CTRU.PKE. As we shall see, even without e, CTRU.PKE is still IND-CPA secure under the RLWR assumption if the ciphertext compression exits. In some sense, CTRU.PKE degenerates into tweaking CNTR.PKE with the existing of the rounding of PolyEncode. Moreover, without the noise e and the ciphertext compression, CTRU.PKE is still OW-CPA secure only based on the NTRU assumption (without further relying on the RLWE

TABLE II
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 III
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}

or RLWR assumption). As for CNTR.PKE, once eliminating the ciphertext compression, CNTR.PKE requires the rounding of PolyEncode, such that CNTR.PKE can be OW-CPA secure similarly based on the NTRU assumption.

The trade-off of using lattice code. The scalable E_8 lattice code allows an efficient and constant-time implementation, as described in section III and section VII-F. The error-correction capability of the scalable E_8 lattice code yields a low enough error probability, and allows wider noise distribution and 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 sightly increases extra implementation complexity, this is likely to be a reasonable and acceptable trade-off of the security, ciphertext size and implementation complexity in many contexts.

V. CONCRETE HARDNESS AND PARAMETER SELECTION

In this section, we first estimate and select parameters for CTRU and CNTR, by applying the methodology of core-SVP hardness estimation [52]. Then, we present the refined gate-count estimate, by using the scripts provided by Kyber and NTRU Prime in NIST PQC Round 3. Finally, we overview and discuss some recent attacks beyond the core-SVP hardness.

A. Parameter Selection with Core-SVP

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 [53], Bai-Galbraith embedding [54], etc). The most common lattice reduction algorithm is the BKZ algorithm [55], [56]. 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 [12], while a u-SVP instance can also be constructed from the LWE problem. The dual attack [57] 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.

2) Core-SVP hardness: Following the simple and conservative methodology of the core-SVP hardness developed from [52], the best known cost of running SVP solver on b-dimension sublattice is $2^{0.292b}$ for the classical case and $2^{0.265b}$ for the quantum case. These cost models can be used for conservative estimates of the security of our schemes. Note that the number of samples is set to be $2n$ for NTRU problem (resp., n for LWE problem), since the adversary is given such samples. We estimate the classical and quantum core-SVP hardness security of CTRU and CNTR via the Python script from [19], [52], [58]. The concrete results are given in Table II and Table III.

3) Parameter sets: The parameter sets of CTRU and CNTR are given in Table II and Table III respectively, where those in red are the recommended parameters also given in Table I. Though the parameters in red are marked as recommended, we believe the other parameter sets are still very useful in certain application scenarios. Note that in Table I 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

TABLE IV
GATE-COUNT ESTIMATE OF CTRU PARAMETERS.

Schemes	(Ψ_1, Ψ_2)	d	b	b'	log(gates)	log(memory)	log(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 V
GATE-COUNT ESTIMATE OF CNTR PARAMETERS.

Schemes	q_2	(Ψ_1, Ψ_2)	d	b	b'	log(gates)	log(memory)	log(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	

attack, and was not taken for concrete hardness estimates in many lattice-based cryptosystems including Kyber in NIST PQC Round 3 [19]. For ease of a fair comparison, the security estimate against the LWE dual attack was not listed in Table I.

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 we fix the message space modulus $p = 2$ and the underlying cyclotomic polynomial $\Phi(x) = x^n - x^{n/2} + 1$, which are omitted in Table II and Table III. 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 sizes $|pk|$, ciphertext sizes $|ct|$ and B.W. (bandwidth, $|pk| + |ct|$) are measured in terms of bytes. "Sec.C" and "Sec.Q" mean 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 IV-C.

We stress that our schemes enjoy a flexibility of parameter selections, but selecting these n's in Table II and Table III is only for simplicity. One can also choose n from $\{576, 648, 864, 972, 1152, 1296\}$ which are integers of the form $3^l \cdot 2^e, l \geq 0, e \geq 1$. Note also that the plaintext message space of CTRU and CNTR is $\{0,1\}^{n/2}$, compared to the fixed message space $\{0,1\}^{256}$ of Kyber and Saber. As for CNTR-512, its first parameter set has the smallest ciphertext sizes, and the third parameter set has the strongest hardness of lattice problem (say, NTRU and RLWR). They can be practically applicable in certain application scenarios which are not much sensitive to error probability. In practice, each secret key will not be used for decryption more than 2^{80} times during its lifetime. In this case, the relatively higher error probabilities, e.g., 2^{-99}, does not undermine the actual security of these parameter sets in reality. The third parameter set of CNTR-512 is highly recommended due to its robust security, since the recent improvements on the attacks [59]–[61] 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, which has a gate complexity of $2^{163.9}$ at the memory complexity of $2^{102.7}$.

B. 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 [62] rather than the GSA-intersect model of [52], [63] 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 [64]. It also accounts for the "few dimensions for free" proposed in [65], 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 [65]. The scripts for these refined estimates are provided in a git branch of the leaky-LWE estimator [62][1].

The gate-count estimate results of the parameter sets of CTRU and CNTR are shown in Table IV and Table V, 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

---

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

attack. b is the BKZ blocksize. b' is the sieving dimension accounting for "dimensions for free". Gates and memory are expressed in bits. The last column means the required $\log(\text{gates})$ values by NIST. It is estimated in [19] that the actual cost may not be more than 16 bits away from this estimate in either direction.

C. Attacks Beyond Core-SVP Hardness

1) Hybrid attack: The works [2], [3], [66] consider the hybrid attack as the most powerful against NTRU-based cryptosystems. However, even with many heuristic and theoretical analysis on hybrid attack [66]–[69], 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 [70] that the mixed 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 KEMs, 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 [70].

2) Recent advances on dual attack: There are some recent progress on the dual attack, and we discuss their impacts on CTRU and CNTR. Duc et al. [71] 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 [68], [72], [73]. Albrecht and Martin [72] propose a re-randomization and smaller-dimensional lattice reduction method, and investigate the method for generating coefficients of short vectors in the dual attack. Guo and Thomas [60] show that the current security estimates from the primal attacks are overestimated. Espitau et al. [74] achieve a dual attack that outperforms the primal attack. These attacks can be combined with the hybrid attack proposed in [75] to achieve a further optimized attack under specific parameters [68], [76], [77]. Very recently, MATZOV [61] further optimizes the dual attack, and claims that the impact of its methods is larger than those of Guo and Thomas's work [60]. It is also mentioned in [61] 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.

3) S-unit attack: The basis of the S-unit attack is the unit attack: finding a short generator. On the basis of the constant-degree algorithm proposed in [78], [79], Biasse et al. [80] 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 [81] 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. The S-unit attack is briefly recalled in Appendix B. Campbell et al. [82] 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 means 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. This idea is discovered in [81], [83]. The difference is that the algorithm proposed by Avanzi and Howard [83] can be applied to any lattice, but is limited to the ℓ_2 norm, while the algorithm proposed by Cohen [81] is applicable to more norms. Pellet-Mary et al. [84] analyze the algorithm of Avanzi and Howard [83], and apply it to S-unit. They point out that the S-unit attack could achieve shorter vectors than existing methods, but still with exponential time for an exponentially large approximation factor. Very recently, Bernstein and Tanja [59] 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 $m' \in \{128, 256, 512\}$ [85]. 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}.

4) BKW attack: For cryptographic schemes to which the BKW method can be applied, the combined methods proposed in [86]–[89], 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 [90]–[92]. 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) Side channel attack: Ravi et al. [93] 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.

Recently, Bernstein [94] 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 standrad IND-CCA attack model. However, Bernstein's fault attack is valid for the specific ciphertext form 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.

6) Other attacks: Algebraic attacks [80], [82], [95], [96] and dense sublattice attacks [77] 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.

VI. POLYNOMIAL ARITHMETIC

In this section, some NTT algorithms are introduced to compute and accelerate the polynomial multiplication or division of CTRU and CNTR. In particular, to address the inconvenient issues that multiple NTT algorithms have to be equipped in accordance with each $n \in \{512, 768, 1024\}$, we provide the methodology of using a unified NTT technique to compute NTT algorithms over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ for all $n \in \{512, 768, 1024\}$ with the same q.

A. The Mixed-radix NTT

A type of mixed-radix NTT is utilized to compute the polynomial multiplication and division over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ with respect to $n = 768$ and $q = 3457$. We choose 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 [15], 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 $\zeta_1 \cdot \zeta_2 = 1$. 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, for example, given 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 $\rho = \zeta^{n/2} \bmod q$. Therefore, $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ 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 i-th term and we start the index i 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 obtained by inverting these procedures. In this case, the point-wise multiplication ("$\circ$") is the corresponding linear polynomial multiplication 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 $h = f \cdot g$, it is computed by $h = INTT(NTT(f) \circ NTT(g))$. In addition, 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 $\hat{f}_i^{-1}$ exits, $i = 0, 1, \ldots, \frac{n}{2} - 1$.

1) The pure radix-2 NTT: Similar techniques can be utilized to $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ w.r.t. $(n = 512, q = 3457)$ and $(n = 1024, q = 3457)$. As for both of them, only the steps of the radix-2 FFT trick are required, due to the power-of-two n. Note that for these two n, there only exits the 384-th primitive root of unity ζ in $\mathbb{Z}_q$. Therefore, $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ can be decomposed into $\prod_{i=0}^{n/4-1} \mathbb{Z}_q[x]/(x^4 - \zeta^{\tau(i)})$ for $n = 512$, but into $\prod_{i=0}^{n/8-1} \mathbb{Z}_q[x]/(x^8 - \zeta^{\tau(i)})$ for $n = 1024$. Thus, the point-wise multiplication and the base case inversion are aimed at the corresponding polynomials of degree 3 for $n = 512$ (degree 7 for $n = 1024$).

B. The Unified NTT

As mentioned above, in order to achieve an efficient implementation, we conduct 6 steps of radix-2 FFT trick for $n \in \{512, 768, 1024\}$ and one more step of radix-3 FFT trick for $n = 768$. Therefore, for different n's, the various (mixed-radix and radix-2) NTT algorithms are required for three types of parameter sets. However, for many applications or platforms, all the three types parameter sets may need to be implemented. 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, which is useful for modular and unified implementations when all the three parameter sets are required.

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. In fact, 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 fix $N = 256$ and choose $\alpha \in \{2, 3, 4\}$. With this technique, we only focus on 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 inverting these procedures. The map road is shown in Figure 2.
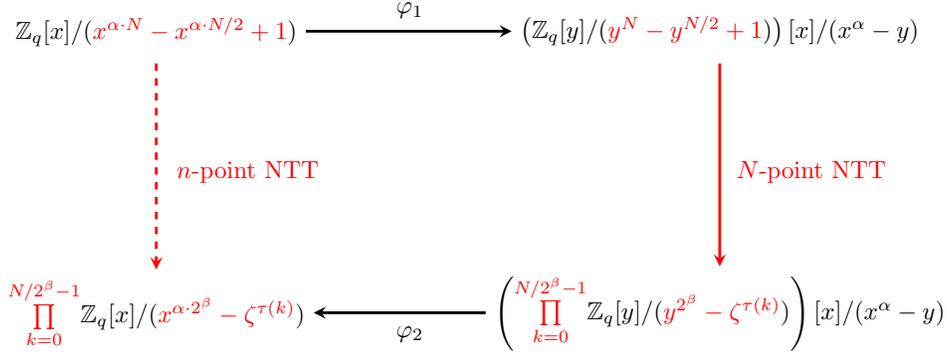
$$\mathbb{Z}_q[x]/(x^{\alpha \cdot N} - x^{\alpha \cdot N/2} + 1) \xrightarrow{\quad \varphi_1 \quad} \left(\mathbb{Z}_q[y]/(y^N - y^{N/2} + 1) \right)[x]/(x^\alpha - y)$$

$$n\text{-point NTT} \qquad\qquad N\text{-point NTT}$$

$$\prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[x]/(x^{\alpha \cdot 2^\beta} - \zeta^{\tau(k)}) \xleftarrow{\quad \varphi_2 \quad} \left(\prod_{k=0}^{N/2^\beta - 1} \mathbb{Z}_q[y]/(y^{2^\beta} - \zeta^{\tau(k)}) \right)[x]/(x^\alpha - y)$$

Fig. 2. Map road for unified NTT

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, inspired by NTTRU [15], 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 satisfying $\frac{3N}{2^\beta} | (q-1)$, where $\beta \in \mathbb{N}$ is called the truncating-parameter, such that it exits 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 $\zeta_2 = \zeta_1^5 = \zeta^{5N/2^{\beta+1}} \bmod q$. Thus, 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 [41]. 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 k-th term and we start the index k from zero. Let $\hat{F}_j$ be the NTT result of F_j and $\hat{F}_{j,l}$ be its l-th coefficient, $l = 0, 1, \ldots, N - 1$. Hence, we can write

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

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 i-th 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:

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

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

C. Discussions

As for the application scenarios w.r.t. $n = 768$, 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 VIII). But, this type of NTT can not 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. Hence, for our CTRU and CNTR, three various

NTT algorithms would be needed for the three recommended parameter sets. The unified NTT can overcome the inconvenient issue. Note that in the base case inversion of the unified NTT-based polynomial division, we need to compute the inverses of degree-3, degree-5 and degree-7 polynomials for $n = 512, 768, 1024$ respectively, which are more complicated than that of linear polynomial in the mixed-radix NTT-based polynomial division. It causes that the unified NTT performs less efficiently in the KeyGen algorithm.

However, most application scenarios and cryptographic devices are usually equipped with three recommended parameter sets ($n = 512, 768, 1024$) for the targeted security levels. In these cases, the unified NTT can lead to modular and simplified software and hardware implementation. Note that in practice the KeyGen algorithm is run once and for all, and its computational cost is less sensitive to most cryptographic applications. As KeyGen is less frequently run, we have taken priority on simple, unified and modular implementations, i.e., unified NTT. In addition, when compared to NTRU-HRSS and SNTRU Prime, the KeyGen algorithm of CTRU and CNTR can be still faster in the context of the unified NTT.

D. Base Case Inversion

The inverse of the polynomials in $\mathbb{Z}_q[x]/(x^i - \zeta^j), i, j \geq 0$ can be computed by Cramer's Rule [42]. Take $\mathbb{Z}_q[x]/(x^4 - \zeta)$ as an example. Let f be a degree-3 polynomial in $\mathbb{Z}_q[x]/(x^4 - \zeta)$, and denote its inverse by f', which implies $f \cdot f' = 1 \mod x^4 - \zeta$. It can be written in the form of matrix-vector multiplication:

$$\begin{bmatrix} f_0 & \zeta f_3 & \zeta f_2 & \zeta f_1 \\ f_1 & f_0 & \zeta f_3 & \zeta f_2 \\ f_2 & f_1 & f_0 & \zeta f_3 \\ f_3 & f_2 & f_1 & f_0 \end{bmatrix} \cdot \begin{bmatrix} f'_0 \\ f'_1 \\ f'_2 \\ f'_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9}$$

Let Δ be the determinant of the coefficient matrix. Hence, the inverse of f exits if and only if $\Delta \neq 0$. In this case, according to Cramer's Rule, there is a unique f', whose individual components are given by

$$f'_i = \frac{\Delta_i}{\Delta}, i = 0, 1, 2, 3 \tag{10}$$

where Δ_i is the determinant of the matrix generated by replacing the $(i+1)$-th column of the coefficient matrix with $(1, 0, 0, 0)^T$. And Δ^{-1} can be computed by using Fermat's Little Theorem, i.e., $\Delta^{-1} \equiv \Delta^{q-2} \mod q$.

E. Multi-moduli NTT

Although directly-using NTT is invalid over $\mathcal{R}_{q_2}$ w.r.t. power-of-two q_2 in the decryption process of CTRU and CNTR, the work [97], [98] show that it is still possible to conduct an efficient multi-moduli NTT over $\mathcal{R}_{q_2}$. Briefly speaking, according to [97], 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. We have the CRT isomorphism: $\mathcal{R}_Q \cong \mathcal{R}_q \times \mathcal{R}_{q'}$. The first NTT algorithm w.r.t (n, q) over $\mathcal{R}_q$ can be instanced as the mixed-radix NTT in section VI-A or the unified NTT in section VI-B. The second NTT algorithm w.r.t. (n, q') over $\mathcal{R}_{q'}$ can be followed from that in NTTRU [15]. 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 recovered via CRT.

VII. IMPLEMENTATION

In this section, the remaining details of the implementations of our schemes are provided, including our portable C implementation, as well as optimized implementations with AVX2 instruction sets. All the implementations are carefully protected against timing attacks.

A. 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 needed to be expanded to the sampling randomness by using **SHAKE-128**. The hash function $\mathcal{H}$ is instantiated with **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 later 32 bytes are used as the secret seed for the encryption algorithm.

B. 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, which are produced from the output of **SHAKE-128** with a secret seed as an input. 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.

C. The Keys and Ciphertexts

The format of the public key. The public key is transmitted in the NTT representation, as in the works like [15], [19], [52], [58]. Specifically, treat the public key 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 $\mathbb{Z}_q$, causing that each coefficient occupies 12 bits. Therefore, 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 $2\eta + 1$, making sure that all the coefficients are in $[0, 4\eta + 1]$. We pack the resulting polynomial 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 needed 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 $n\lceil\lfloor\log(4\eta + 1)\rceil/8 + 3n/2 + 32$ bytes.

The format of the ciphertext. The ciphertexts of our schemes consist of only one (compressed) polynomial c. The polynomial c is in normal representation instead of NTT representation, since the compression through rounding has to work in normal representation. Each coefficient of c occupied $\lceil\log(q_2)\rceil$ bits. Thus, to pack and store such a ciphertext only costs $n\lceil\log(q_2)\rceil/8$ bytes.

The prefix of the public key. As for 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 has 12 bits.

D. Portable C Implementation

Our portable C implementations rely on 16-bit and 32-bit integer arithmetic mainly, 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 methodology in [39] which is based on 32-bit integer arithmetic, but with high developments on constant-time skills for security and simplicity.

NTT implementation. Our C implementations of NTTs do not make use of variable-time operator "%" for the modular reductions, but we turn to use Barrett reduction [99], [100] and Montgomery reduction [100], [101], where the former is applied after additions and the later is applied for multiplication between coefficients with primitive roots. But we only use the signed variants of these two reductions as described in [100]. Lazy reduction strategy [100] 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 q. However, NTT is invalid in $\mathcal{R}_{q_2}$ w.r.t. 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 w.r.t. power-of-two q_2 can be implemented by logical AND operations efficiently.

E. 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 [52], [102]. As for the computation of $cf \bmod {}^{\pm} q_2$ w.r.t. power-of-two q_2 in the decryption process of CTRU and CNTR, we choose the multi-moduli NTT (see section VI-E), instead of the schoolbook algorithm, which is different from that in the portable C implementation. Besides, 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 which NTT is suitable for vectorized implementation, especially AVX2.

NTT optimizations. Our AVX2-based NTT implementation handles 16-bit signed integer coefficients, every 16 values of which are loaded into one vector register. Load and store instructions are time-consuming in AVX2 instruction set. To accelerate AVX2 implementation, we need to reduce the memory access operations. 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 is no extra load or store operations. We achieve this by using different pair of vector registers and permutating coefficients order. The instructions we use for permutation task are vperm2i128,

vpunpcklqdq, vpblendd and vpblendw. The coefficients are permutated in levels 3-5. After the radix-2 FFT trick, we do not choose to store the coefficients immediately, instead we use the vector register to complete the radix-3 FFT trick if necessary. It's because after the last level of the radix-2 FFT trick, the order of coefficients in the vector register is naturally the order we need in the radix-3 FFT trick. To reduce total permutation time, we propose to store coefficients in a shuffled order. For polynomial point-wise multiplication and polynomial inversion, pairwise modular multiplications are involved, since we store coefficients in the shuffled order. The two continual coefficients are stored in two different vector registers, therefore, we can easily implement polynomial point-wise multiplication and polynomial inversion. We encapsulate some functional codes which are used multiple times during the whole process into macro, making the codes more concise and readable, as well as avoiding repetitive codes.

F. 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], [100], both Barrett reduction [99], [100] and Montgomery reduction [15], [101] 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 we have implemented with 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 exit 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 [103] 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 [104]. 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 [104]. 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 [37]. However, unlike those error correction codes, our scalable E_8 lattice code features constant-time encoding and decoding algorithms, enabling safe implementations against timing attacks.

VIII. 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 under the applications of the mixed-radix NTT. All the benchmark tests are run on an Intel(R) Core(TM) i7-10510U CPU at 2.3GHz (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*. We run the corresponding KEM algorithms for 10,000 times and calculate the average CPU cycles. The benchmark results are shown in Table VI, along with comparisons with other schemes. Concretely, we re-run the C source codes of parts 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 3.0GHz Intel Xeon E3-1220 v6) [102]. Regarding the benchmark results in this section, we stress 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.

A. 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_{3457}^{768} are not online available in [32], and the AVX2 benchmark results of NTRU-C_{3457}^{768} are absent in [32], so all of its benchmark results are omitted here.

Note that the work [97] 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 [97] is not obvious (in

TABLE VI
CPU CYCLES OF KEMs (IN KILO CYCLES).

Schemes	C			AVX2		
	KeyGen	Encaps	Decaps	KeyGen	Encaps	Decaps
CTRU-768, Mixed-radix NTT (Ours)	117.5	63.4	134.6	10.6	11.7	35.7
CNTR-768, Mixed-radix NTT (Ours)	118.4	64.9	133.1	12.8	10.6	35.4
CTRU-768, Unified NTT (Ours)	6.2×10^3	75.3	135.9	–	–	–
CNTR-768, Unified NTT (Ours)	6.3×10^3	78.0	137.0	–	–	–
NTRU-HRSS	127.6×10^3	3.2×10^3	9.4×10^3	254.0	24.9	59.2
SNTRU Prime-761	17.1×10^3	9.0×10^3	23.7×10^3	156.3	46.9	56.2
Kyber-768	140.3	159.0	205.9	25.3	27.6	43.4
Saber-768	94.7	109.7	138.9	64.2	69.3	95.3
CTRU-768, **SHA-2** variant (Ours)	101.4	58.6	116.9	7.0	6.0	21.8
CNTR-768, **SHA-2** variant (Ours)	104.6	60.0	114.2	8.4	6.0	23.6
NTTRU	157.4	98.9	142.4	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

fact, the speed-up is $\pm 0\%$). Hence, we omit the benchmark results about NTRU-HRSS provided in [97]. Consequently, the benchmark results of the state-of-the-art AVX2 implementation of NTRU-HRSS are reported in [102]. As for SNTRU Prime-761, its state-of-the-art AVX2 implementation is presented in [105]. Thus, we take the AVX2 benchmark results of NTRU-HRSS and SNTRU Prime-761 from [102] and [105], respectively. 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 are benefited from the applications of NTT in polynomial operations. For example, as for portable C implementation, CTRU-768 is faster than NTRU-HRSS by 1,000X in KeyGen, 50X in Encaps, and 69X in Decaps, respectively. As for optimized AVX2 implementation, CTRU-768 is faster than NTRU-HRSS by 23X in KeyGen, 2.1X in Encaps, and 1.6X in Decaps, respectively; CNTR-768 is faster than NTRU-HRSS by 19X in KeyGen, 2.3X in Encaps, and 1.6X in Decaps, respectively.

When compared to NTTRU fairly, we do the following modifications to present vaianrts of CTRU and CNTR, which have the same hash functions, symmetric primitives and FO transformations as NTTRU: (1) use **SHA-2** family to instantiate hash functions; (2) use **AES** to expand seeds; (3) change the FO transformation into $FO_m^\perp$. The AVX2 benchmark results of the KeyGen of CTRU/CNTR variants are close to those of NTTRU, whereas the Encaps results of CTRU/CNTR variants are faster than that of NTTRU. However, the Decaps results of CTRU/CNTR variants are slightly slower than that of NTTRU, on the following grounds: (1) the decoding algorithm of the scalable E_8 lattice code costs extra time; (2) multi-moduli NTT is more time-consuming than the NTT algorithm of NTTRU, since multi-moduli NTT essentially consists of two routines of NTT algorithms.

B. Comparison with Other Lattice-based KEM Schemes

The C source codes of Kyber-768 and Saber-768 are taken from their Round 3 supporting documentations. We modify their FO transformation into $FO_{ID(pk),m}^{\not\perp}$, and re-run their C source codes. The work [35] has reported the state-of-the-art AVX2 implementation of Kyber and Saber with $FO_{ID(pk),m}^{\not\perp}$. Thus, we take their AVX2 benchmark results from [35] directly.

As shown in Table VI, our CTRU-768 and CNTR-768 outperform both Kyber-768 and Saber-768. For example, when compared to the state-of-the-art AVX2 implementation of Kyber-768, CTRU-768 is faster by 2.3X in KeyGen, 2.3X in Encaps, and 1.2X in Decaps, respectively; CNTR-768 is faster by 1.9X in KeyGen, 2.6X in Encaps, and 1.2X in Decaps, respectively. It is due to the following reasons: (1) in Kyber, there are a rejection sampling to generate the matrix **A** and a complicated polynomial matrix-vector multiplication in the key generation process and the encryption process of Kyber-768 (which are also re-run with the Decaps); (2) but there is only one polynomial multiplication in the encryption process of CTRU-768 and CNTR-768.

C. Comparison with Other Non-lattice-based KEM Schemes

We present a rough comparison with other non-lattice-based KEM schemes, i.e., BIKE [106], Classic McEliece [107], HQC [108] and SIKE [109], which are candidates advancing to the fourth round of NIST PQC [110]. 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 [110]. 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 [102]. As shown in Table VI, our schemes are much faster than these non-lattice-based KEM schemes. For example, CTRU-768 is faster by 160X in KeyGen, 6.6X in Encaps and 7.1X in Decaps than Classic McEliece460896.

D. Benchmark Results with Unified NTT

The C implementations of CTRU-768 and CNTR-768 with our unified NTT are also provided, whose benchmark results could be found in Table VI. Although the overall performances of CTRU-768 and CNTR-768 with our unified NTT are inferior than those of CTRU-768 and CNTR-768 with mixed-radix NTT, we stress that the primary goal of the unified NTT is to provide a modular and convenient implementation, instead of a faster implementation. Note that an optimized AVX2 implementation of the unified NTT could be more precise to present benchmark results. But the AVX2 implementation is still a work in progress and we left as a future work.

APPENDIX A
ON CCA SECURITY REDUCTION OF KEM IN THE ROM AND THE QROM

Generic constructions of an efficient IND-CCA secure KEM are well studied in [34], [111], which are essentially various KEM variants of Fujisaki-Okamoto (FO) transformation [33] and GEM/REACT transformation [112], [113]. The work [34] gives a modular analysis of various FO transformations in the ROM and the QROM, and summarizes some practical FO transformations that are widely used to construct an IND-CCA secure KEM from a passive secure PKE (e.g., OW-CPA and IND-CPA), including the following transformations $\text{FO}^\perp$, $\text{FO}_m^\perp$, $\text{FO}^{\not\perp}$, $\text{FO}_m^{\not\perp}$, $\text{U}^{\not\perp}$ and $\text{U}_m^{\not\perp}$, etc, where m (without m) means $K = H(m)$ ($K = H(m, c)$), $\not\perp$ ($\perp$) means implicit (explicit) rejection.

$\text{FO}^\perp$, $\text{FO}_m^\perp$, $\text{FO}^{\not\perp}$ and $\text{FO}_m^{\not\perp}$ are the most common transformations used in NIST PQC. According to [34], in the ROM, the reduction bound of these four transformations are all $\epsilon' \leq \epsilon_{CPA} + q'\delta$ and $\epsilon' \leq q'\epsilon_{OW} + q'\delta$, where ϵ' is the advantage of an adversary against IND-CCA security of KEM, ϵ_{CPA} (ϵ_{OW}) is the the advantage of an adversary against IND-CPA (OW-CPA) security of the underlying PKE, q' is the total number of hash queries, and δ is the error probability. Notice that in order to keep the comparison lucid, we ignore the small constant factors and additional inherent summands. The reduction is tight for IND-CPA secure PKE, but it has a loss factor q' for OW-CPA secure PKE in the ROM. However, all of their reduction bounds in the QROM suffer from a quartic loss, i.e., $\epsilon' \leq q'\sqrt{q'\sqrt{\epsilon_{OW}} + q'^2\delta}$ with an additional hash in [34]. Later, the bound of $\text{FO}^{\not\perp}$ is improved as follows: $\epsilon' \leq q'\sqrt{\epsilon_{OW}} + q'\sqrt{\delta}$ without additional hash in [114], $\epsilon' \leq \sqrt{q'\epsilon_{CPA}} + q'\sqrt{\delta}$ with semi-classical oracles [115] in [116], $\epsilon' \leq \sqrt{q'\epsilon_{CPA}} + q'^2\delta$ with double-sided OW2H lemma in [117], and $\epsilon' \leq q'^2\epsilon_{CPA} + q'^2\delta$ with measure-rewind-measure technique in [118]. The bound of $\text{FO}_m^{\not\perp}$ is improved as follows: $\epsilon' \leq q'\sqrt{\epsilon_{OW}} + q'\sqrt{\delta}$ without additional hash in [114], $\epsilon' \leq \sqrt{q'\epsilon_{CPA}} + q'^2\delta$ with disjoint simulatability in [119], $\epsilon' \leq \sqrt{q'\epsilon_{CPA}} + q'^2\delta$ with prefix hashing in [35]. The bound of $\text{FO}_m^\perp$ is improved as follows: $\epsilon' \leq q'\sqrt{\epsilon_{OW}} + q'\sqrt{\delta}$ and $\epsilon' \leq \sqrt{q'\epsilon_{CPA}} + q'\sqrt{\delta}$ with extra hash in [120], $\epsilon' \leq q'\sqrt{\epsilon_{OW}} + q'^2\sqrt{\delta}$ without extra hash in [121].

There also exist some transformations with tight reduction for deterministic PKE (DPKE) with disjoint simulatability and perfect correctness, for example, a variant of $\text{U}_m^{\not\perp}$ proposed in [122]. In the case that the underlying PKE is non-deterministic, all known bounds are of the form $O(\sqrt{q'\epsilon_{CPA}})$ and $O(q'\sqrt{\epsilon_{OW}})$ as we introduce above, with the exception of [118]. The work [123] shows that the measurement-based reduction involving no rewinding will inevitably incur a quadratic loss of the security in the QROM. In another word, as for the underlying PKE, the IND-CPA secure PKE has a tighter reduction bound than the OW-CPA secure PKE. It also significantly leads us to construct an IND-CPA secure PKE for tighter reduction bound of the resulting IND-CCA secure KEM.

Some discussions are presented here for comparing the reduction bounds of CTRU and CNTR and other NTRU-based KEM schemes. Most of the existing NTRU-based encryption schemes can only achieve OW-CPA security. NTRU-HRSS and SNTRU Prime construct the KEM schemes from OW-CPA DPKEs via $\text{U}_m^{\not\perp}$ variants. Although they can reach tight CCA reductions with extra assumptions in the (Q)ROM [16], [17], there is a disadvantage that some extra computation is needed to recover the randomness in the decryption algorithms.

Determinism is a much stricter condition, thus some NTRU-based PKEs prefer to be non-deterministic (i.e., randomized). NTTRU applies $\text{FO}_m^\perp$ to build an IND-CCA KEM from an OW-CPA randomized PKE [15]. According to [34], [121], its IND-CCA reduction bounds are not-tight in both the ROM ($O(q'\epsilon_{OW})$) and the QROM ($O(q'\sqrt{\epsilon_{OW}})$).

NTRU-C is the general form of NTRU-C$_{3457}^{768}$. NTRU-C uses a slightly different way that it first constructs an IND-CPA PKE from an OW-CPA NTRU-based PKE via ACWC_0 transformation [32], and then transforms it into an IND-CCA KEM via $\text{FO}_m^\perp$. Note that ACWC_0 brings two terms of ciphertexts, where the extra term of ciphertexts costs 32 bytes. The IND-CPA security of the resulting after-ACWC_0 PKE can be tightly reduced to the OW-CPA security of the underlying before-ACWC_0 PKE in the ROM. However, there is a quadratic loss advantage in the QROM, i.e., $\epsilon_{CPA} \leq q'\sqrt{\epsilon_{OW}}$. In the ROM, the advantage of the adversary against IND-CCA security of KEM is tightly reduced to that of the adversary against IND-CPA security of after-ACWC_0 PKE, and consequently is tightly reduced to that of the adversary against OW-CPA security of before-ACWC_0 PKE. However, in the QROM, there is no known direct reduction proof about $\text{FO}_m^\perp$ from IND-CPA PKE to IND-CCA KEM without additional hash. The reduction bound of $\text{FO}_m^\perp$ in the QROM in [121] only aims at the underlying OW-CPA PKE.

Since the IND-CPA security implies OW-CPA security [34], the reduction bound of IND-CCA KEM to before-ACWC$_0$ OW-CPA PKE will suffer from the quartic advantage loss in the QROM. That is, if the adversary has ϵ_{OW} advantage against the before-ACWC$_0$ OW-CPA PKE, then it has $O(q'^{1.5}\sqrt[4]{\epsilon_{OW}})$ advantage against the resulting IND-CCA KEM in the QROM. On the other hand, with an additional hash, a better bound of FO$_m^\perp$ for after-ACWC$_0$ IND-CPA PKE can be achieved, i.e., $O(\sqrt{q'\epsilon_{CPA}})$ advantage against the resulting IND-CCA KEM in the QROM [120] at the cost of some extra ciphertext burden. ACWC$_0$ also has an effect on the efficiency, since an extra transformation from OW-CPA PKE to IND-CPA PKE is also relatively time-consuming.

Our CTRU and CNTR seem to be more simple, compact, efficient and memory-saving than other NTRU-based KEM schemes, along with a tight bound in the ROM and a tighter bound in the QROM for IND-CCA security. When compared to NTRU-HRSS, SNTRU Prime and NTRU-C, an obvious efficiency improvement of our CTRU and CNTR is due to the fact that there is no extra requirement of recovering randomness in decryption algorithm or reinforced transformation to obtain IND-CPA security. CTRU/CNTR.PKE can achieve IND-CPA security in the case that its ciphertext can be only represented by a single polynomial, without any extra ciphertext term like NTRU-C. Starting from our IND-CPA PKE to construct KEM with FO$_{ID(pk),m}^{\not\perp}$, the reduction bound of IND-CCA security is tightly reduced to IND-CPA security in the ROM ($\epsilon' \leq O(\epsilon_{CPA})$, restated), so it is tightly reduced to the underlying hardness assumptions. We also have the known best bound in the QROM ($\epsilon' \leq O(\sqrt{q'\epsilon_{CPA}})$, restated) according to [35], which is better than those in NTTRU and NTRU-C.

A. CCA Security in Multi-User Setting

We remark that, the work [35] originally gives the multi-user/challenge IND-CCA reduction bound of FO$_{ID(pk),m}^{\not\perp}$ in the ROM and the QROM. We adapt the results from Theorem 3.1 and Theorem 3.2 in [35] into the single-user/challenge setting of CTRU and CNTR, which is only for ease of fair comparisons as other KEM schemes only utilize single-user/challenge FO transformations. As CTRU/CNTR.PKE is IND-CPA secure, another advantage of using FO$_{ID(pk),m}^{\not\perp}$ is that CTRU and CNTR can be improved to enjoy the multi-user/challenge IND-CCA security as well. To address this issue, some adjustments are needed as follows. Unlike the single-user/challenge setting, the adversary (against the n'-user/q_C-challenge IND-CPA security of the underlying PKE) is given the public keys of n' users, and is allowed to make at most q_C challenge queries w.r.t. the same challenge plaintext m_b chosen by the challenger. According to [35], based on the single-user/challenge IND-CPA security of the underlying PKE, the formal multi-user/challenge IND-CCA security of the resulting KEM is given in Theorem 7.

Theorem 7 (n'-user/q_C-challenge IND-CCA security in the ROM and the QROM [35]) *Following [35], we will use (or recall) the following terms in the concrete security statements.*

- n'-user error probability $\delta(n')$ [35].
- *Min-entropy ℓ [33] of $ID(pk)$, i.e., $\ell = H_\infty(ID(pk))$, where $(pk, sk) \leftarrow CTRU/CNTR.PKE.KeyGen$.*
- *Bit-length ι of the secret seed $z \in \{0,1\}^\iota$.*
- *Maximal number of (Q)RO queries q_H.*
- *Maximal number of decapsulation queries q_D.*
- *Maximal number of challenge queries q_C.*

For any (quantum) adversary A *against the (n', q_C)-IND-CCA security of CTRU/CNTR.KEM, there exits a (quantum) adversary* B *against the (n', q_C)-IND-CPA security of CTRU/CNTR.PKE with roughly the same running time of* A, *such that:*

- *In the ROM, it holds that* $\boldsymbol{Adv}_{CTRU/CNTR.KEM}^{(n',q_C)\text{-}IND\text{-}CCA}(\mathsf{A}) \leq$

$$2\left(\boldsymbol{Adv}_{CTRU/CNTR.PKE}^{(n',q_C)\text{-}IND\text{-}CPA}(\mathsf{B}) + \frac{(q_H + q_C)q_C}{|\mathcal{M}|}\right) + \frac{q_H}{2^\iota} + (q_H + q_D)\delta(n') + \frac{n'^2}{2^\ell};$$

- *In the QROM, it holds that* $\boldsymbol{Adv}_{CTRU/CNTR.KEM}^{(n',q_C)\text{-}IND\text{-}CCA}(\mathsf{A}) \leq$

$$2\sqrt{q_{HD}\boldsymbol{Adv}_{CTRU/CNTR.PKE}^{(n',q_C)\text{-}IND\text{-}CPA}(\mathsf{B})} + 4q_{HD}\sqrt{\frac{q_C \cdot n'}{|\mathcal{M}|}} + 4(q_H + 1)\sqrt{\frac{n'}{2^\iota}} + 16q_{HD}^2\delta(n') + \frac{q_C^2}{|\mathcal{M}|} + \frac{n'^2}{2^\ell},$$

where $q_{HD} := q_H + q_D + 1$.

APPENDIX B
S-UNIT ATTACK

Here we refer to [59] to briefly introduce S-unit attack.

S-unit attack begins with a nonzero $v \in I$ and outputs v/u, but now u is allowed to range over a larger subset of K^*, specifically the group of S-units.

Here S is a finite set of places, a subset of the set V mentioned above. There are two types of places:

- The "infinite places" are labeled $1, 3, 5, \ldots, n-1$, except that for $n = 1$ there is one infinite place labeled 1. The entry at place j in $\log \alpha$ is defined as $2\log|\sigma_j(\alpha)|$, except that the factor 2 is omitted for $n = 1$. The set of all infinite places is denoted ∞, and is required to be a subset of S.

- For each nonzero prime ideal P of R, there is a "finite place" which is labeled as P. The entry at place P in $\log \alpha$ is defined as $-(ord_P \alpha) \log |(R/P)|$, where $ord_P \alpha$ is the exponent of P in the factorization of α as the product of powers of prime ideals. There are many choices of S here. It focuses on the following form of S: choose a parameter y, and take $P \in S$ if and only if $|(R/P)| \le y$.

The group U_S of S-units of K is, by definition, the set of elements $u \in K^*$ such that the vector $\log u$ is supported on S, i.e., it is 0 at every place outside S. The S-unit lattice is the lattice $\log U_S$, which has rank $|S-1|$.

Short v/u again corresponds to short $\log v - \log u$, but it is required to ensure that $v/u \in I$, i.e., $ord_P(v/u) \ge ord_P I$ for each finite place P. This was automatic for unit attacks but is not automatic for general S-unit attacks. One thus wants to find a vector $\log u$ in the S-unit lattice $\log U_S$ that is close to $\log v$ in the following sense: $\log u$ is close to $\log v$ at the infinite places, and $ord_P u$ is close to but no greater than $ord_P v - ord_P I$.

As for closeness, as a preliminary step, if $ord_P v < ord_P I$ for some P, update v by multiplying it by a generator of $P\hat{P}$ (or, if possible, of P) as explained above, and repeat this step. Then $v \in I$. Next, if some u in the list has v/u shorter than v and $v/u \in I$, replace v with v/u, and repeat this step. Output the final v.

As an extreme case, if $S = \infty$ (the smallest possible choice, not including any P), then $U_S = R^*$: the S-units of K are the units of R, the S-unit lattice is the unit lattice, and S-unit attacks are the same as unit attacks. Extending S to include more and more prime ideals P gives S-unit attacks the ability to modify more and more places in $\log v$.

APPENDIX C
GENERALIZATION AND MORE VARIANTS

Finally, to demonstrate the flexibility of our framework, we present and discuss some generalization approaches and more variants. The following approaches are applicable to both CTRU and CNTR.

A. Compressing the Public Key h

In general, let $q_1 \le q$ be an integer, we set the public key to be $\hat{h} = \lfloor \frac{q_1}{q} h \rceil \in \mathcal{R}_{q_1}$ in the KeyGen. This not only shortens the public key size, but also can strengthen the security of the NTRU assumption in general. Then, there are two approaches to deal with this change in PKE.Enc.

- $\sigma = \hat{h} r \in \mathcal{R}_{q_1}$, and now PolyEncode needs to work in $\mathcal{R}_{q_1}$ rather than $\mathcal{R}_q$ (i.e., the parameter q is replaced with q_1).
- $\sigma = \lfloor \frac{q}{q_1} \hat{h} \rceil r \in \mathcal{R}_q$. That is, we lift $\hat{h}$ from $\mathcal{R}_{q_1}$ to $\mathcal{R}_q$. With this approach, PolyEncode remains unchanged.

These approaches can reduce the size of public key, but at the cost of larger error probability or lower security (as we may need to narrow the space of secret polynomials for reducing error probability). With experiments, when $q_1 = 2^{11}$ (i.e., cutting off one bit from each dimension) we still can achieve reasonable balance between security and performance.

B. Masking the Public Key h

Similarly, we would like also to strengthen the NTRU assumption, by setting $h = g/f + x$, where x is an n-dimension small noise polynomial with each coefficient typically taken from B_1 or U_1, i.e., the uniform distribution over $\{0, \pm 1\}$. In this case, h is analogous to an RLWE sample, except that f^{-1} is not publicly accessible. Intuitively, it makes the NTRU problem harder than its standard form. In this case, one extra error term xr will be introduced. Thanks to the powerful error correction ability of the E_8 lattice code, our experiments show that we can still achieve good balance between security and performance, with x taken from B_1 or U_1. Note that the above approach to compressing the public key can also strengthen the hardness of the NTRU problem.

C. More Possibilities of PolyEncode and PolyDecode

We choose the E_8 lattice code within our framework because: (1) the error correction ability of the E_8 lattice code is powerful and almost optimal; and (2) it is simple, very efficient, and well fits our framework combining NTRU and RLWE/RLWR. However, in general, we can use other error correction codes (ECC) within our framework. Also, an extreme choice is to not use any extra ECC mechanism, i.e., letting $PolyEncode(m) = \frac{q}{2} m$. The corresponding decryption process is $PolyDecode(cf \bmod {}^{\pm}q_2) = \lfloor \frac{2}{q_2}(cf \bmod {}^{\pm}q_2) \rceil \bmod 2$.

D. More Possibilities of the Underlying Rings

The modulus q is set to be a prime number that allows efficient NTT algorithms over $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ in this paper. One can also choose power-of-two q for the flexibility of parameter selection. Upon setting power-of-two q and q_2, the distribution χ in Theorem 3 can be simplified as follows: Sample $u \xleftarrow{\$} [-\frac{q}{2q_2}, \frac{q}{2q_2}) \cap \mathbb{Z}$ and output $-\frac{q_2}{q} u$. However, in this case the computation of polynomial division is slightly slower, since NTT is invalid and we need to turn to other less efficient algorithms to compute polynomial divisions.

We can naturally generalize the underlying polynomial rings of the form $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ to power-of-two cyclotomic rings of the form $\mathbb{Z}_q[x]/(x^n + 1)$. To consider the dimension n, one can pick $n = 512, 1024$ for NIST recommended security levels I and V. When the modulus q is NTT-friendly (i.e., $2^e | (q - 1)$ for some integer e in this case), the efficient polynomial multiplications and divisions are possible via some NTT algorithms (or their variants). For the flexibility of ring selection, one can also consider the rings of the form $\mathbb{Z}_q[x]/(x^n - x - 1)$ w.r.t. prime n and q like those in NTRU Prime [17], [43].

E. Variants without RLWE or RLWR

If we insist on a purely NTRU-based KEM, we can simply set $c := \sigma + \lfloor \mathrm{PolyEncode}(m) \rceil \bmod q$ where we can set $\sigma := hr$. In this case, the resulting KEM scheme is OW-CPA secure based on the NTRU assumption. With this variant, at about the same error probabilities of CTRU and CNTR, we can choose much larger ranges for the secret key and the ephemeral secrecy r leading to stronger NTRU hardness.

REFERENCES

[1] NIST, "Post-quantum cryptography, round 1 submissions," https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions, 2016.

[2] ——, "Post-quantum cryptography, round 2 submissions," https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions, 2019.

[3] ——, "Post-quantum cryptography, round 3 submissions," https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions, 2020.

[4] ——, "Pqc standardization process: Announcing four candidates to be standardized, plus fourth round candidates," https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4, 2022.

[5] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 34:1–34:40, 2009.

[6] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *EUROCRYPT 2010*, vol. 6110, 2010, pp. 1–23.

[7] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des. Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, 2015.

[8] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *EUROCRYPT 2012*, vol. 7237, 2012, pp. 719–737.

[9] J. Alperin-Sheriff and D. Apon, "Dimension-preserving reductions from LWE to LWR," *IACR Cryptol. ePrint Arch.*, p. 589, 2016.

[10] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *ANTS*, vol. 1423, 1998, pp. 267–288.

[11] ——, "Ntru: a new high speed public key cryptosystem," *presented at the rump session of Crypto 96*, 1996.

[12] D. Coppersmith and A. Shamir, "Lattice attacks on NTRU," in *EUROCRYPT '97*, vol. 1233, 1997, pp. 52–61.

[13] K. Jarvis and M. Nevins, "ETRU: NTRU over the eisenstein integers," *Des. Codes Cryptogr.*, vol. 74, no. 1, pp. 219–242, 2015.

[14] K. Bagheri, M. Sadeghi, and D. Panario, "A non-commutative cryptosystem based on quaternion algebras," *Des. Codes Cryptogr.*, vol. 86, no. 10, pp. 2345–2377, 2018.

[15] V. Lyubashevsky and G. Seiler, "NTTRU: truly fast NTRU using NTT," *IACR TCHES*, vol. 2019, no. 3, pp. 180–201, 2019.

[16] C. Chen, O. Danba, J. Hoffstein, and A. Hulsing, "Ntru submission," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[17] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, "Ntru prime: round 3," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[18] P.-A. Fouque, P. Kirchner, T. Pornin, and Y. Yu, "Bat: Small and fast kem over ntru lattices," *IACR TCHES*, vol. 2022, no. 2, pp. 240–265, 2022.

[19] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber - algorithm specifications and supporting documentation (version 3.01)," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[20] A. Basso, J. M. B. Mera, and J.-P. D'Anvers, "Supporting documentation: Saber: Mod-lwr based kem (round 3 submission)," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[21] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSIGN: digital signatures using the NTRU lattice," in *CT-RSA 2003*, vol. 2612, 2003, pp. 122–140.

[22] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC 2012*, 2012, pp. 1219–1234.

[23] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices," in *EUROCRYPT 2013*, vol. 7881, 2013, pp. 1–17.

[24] A. Langlois, D. Stehlé, and R. Steinfeld, "Gghlite: More efficient multilinear maps from ideal lattices," in *EUROCRYPT 2014*, vol. 8441, 2014, pp. 239–256.

[25] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *ASIACRYPT 2014*, vol. 8874, 2014, pp. 22–41.

[26] P.-A. Fouque, J. Hoffstein, P. Kirchner, and V. Lyubashevsky, "Falcon: Fast-fourier lattice-based compact signatures over ntru," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[27] D. Jablon, "Ieee p1363 standard specifications for public-key cryptography," in *CTO Phoenix Technologies Treasurer, IEEE P1363 NIST Key Management Workshop*, 2008.

[28] B. Wire, "Security innovation's ntruencrypt adopted as x9 standard for data protection," https://www.businesswire.com/news/home/20110411005309/en/Security-Innovations-NTRUEncrypt-Adopted-X9-Standard-Data, 2011.

[29] D. Augot, L. Batina, D. J. Bernstein, and J. Bos, "Initial recommendations of long-term secure post-quantum systems," *PQCRYPTO. EU. Horizon*, vol. 2020, 2015.

[30] D. Stehlé and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *EUROCRYPT 2011*, vol. 6632, 2011, pp. 27–47.

[31] OpenSSH, "Openssh release notes," https://www.openssh.com/releasenotes.html, 2022.

[32] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, G. Seiler, and D. Unruh, "A thorough treatment of highly-efficient NTRU instantiations," *IACR Cryptol. ePrint Arch.*, p. 1352, 2021.

[33] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *CRYPTO' 99*, vol. 1666, 1999, pp. 537–554.

[34] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the fujisaki-okamoto transformation," in *TCC 2017*, vol. 10677, 2017, pp. 341–371.

[35] J. Duman, K. Hövelmanns, E. Kiltz, V. Lyubashevsky, and G. Seiler, "Faster lattice-based kems via a generic fujisaki-okamoto transform using prefix hashing," 2021, pp. 2722–2737.

[36] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 227–231, 1982.

[37] ——, *Sphere packings, lattices and groups*. Springer Science & Business Media, 2013, vol. 290.

[38] M. Viazovska, "The sphere packing problem in dimension 8," *Annals of Mathematics*, pp. 991–1015, 2017.

[39] Z. Jin, S. Shen, and Y. Zhao, "Compact and flexible KEM from ideal lattice," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 3829–3840, 2022.

[40] A. Hülsing, J. Rijneveld, J. M. Schanck, and P. Schwabe, "High-speed key encapsulation from NTRU," in *CHES 2017*, vol. 10529, 2017, pp. 232–252.

[41] R. T. Moenck, "Practical fast polynomial multiplication," in *SYMSAC 1976*. ACM, 1976, pp. 136–148.
[42] W. H. Greub, *Linear algebra*. Springer Science & Business Media, 2012, vol. 23.
[43] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, "NTRU prime: Reducing attack surface at low cost," in *SAC 2017*, vol. 10719, 2017, pp. 235–260.
[44] L. C. Washington, *Introduction to cyclotomic fields*. Graduate Texts in Mathematics 83, Springer-Verlag, 1997.
[45] R. Steinfeld, "Ntru cryptosystem: Recent developments and emerging mathematical problems in finite polynomial rings," *Algebraic Curves and Finite Fields*, pp. 179–212, 2014.
[46] J. M. Pollard, "The fast fourier transform in a finite field," *Mathematics of computation*, vol. 25, no. 114, pp. 365–374, 1971.
[47] D. J. Bernstein, "Multidigit multiplication for mathematicians," http://cr.yp.to/papers.html#m3, 2001.
[48] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.
[49] W. M. Gentleman and G. Sande, "Fast fourier transforms: for fun and profit," in *AFIPS '66*, ser. AFIPS Conference Proceedings, vol. 29, 1966, pp. 563–578.
[50] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *CCS '93*, 1993, pp. 62–73.
[51] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, "Random oracles in a quantum world," in *ASIACRYPT 2011*, vol. 7073, 2011, pp. 41–69.
[52] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange - A new hope," in *USENIX 2016*, 2016, pp. 327–343.
[53] R. Kannan, "Minkowski's convex body theorem and integer programming," *Math. Oper. Res.*, vol. 12, no. 3, pp. 415–440, 1987.
[54] S. Bai and S. D. Galbraith, "Lattice decoding attacks on binary LWE," in *ACISP 2014*, vol. 8544, 2014, pp. 322–337.
[55] C. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, pp. 181–199, 1994.
[56] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT 2011*, vol. 7073, 2011, pp. 1–20.
[57] D. Micciancio and O. Regev, "Post-quantum cryptography, chapter lattice-based cryptography," *Computing*, vol. 85, no. 1-2, pp. 105–125, 2008.
[58] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - kyber: A cca-secure module-lattice-based KEM," in *IEEE EuroS&P 2018*, 2018, pp. 353–367.
[59] D. J. Bernstein and T. Lange, "Non-randomness of s-unit lattices," *IACR Cryptol. ePrint Arch.*, p. 1428, 2021.
[60] Q. Guo and T. Johansson, "Faster dual lattice attacks for solving LWE with applications to CRYSTALS," in *ASIACRYPT 2021*, vol. 13093, 2021, pp. 33–62.
[61] MATZOV, "Report on the Security of LWE: Improved Dual Lattice Attack," Apr. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6412487
[62] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi, "LWE with side information: Attacks and concrete security estimation," in *CRYPTO 2020*, vol. 12171, 2020, pp. 329–358.
[63] M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the expected cost of solving usvp and applications to LWE," in *ASIACRYPT 2017*, vol. 10624, 2017, pp. 297–322.
[64] M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, "Estimating quantum speedups for lattice sieves," in *ASIACRYPT 2020*, vol. 12492, 2020, pp. 583–613.
[65] L. Ducas, "Shortest vector from lattice sieving: A few dimensions for free," in *EUROCRYPT 2018*, vol. 10820, 2018, pp. 125–145.
[66] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte, "Choosing ntruencrypt parameters in light of combined lattice reduction and MITM approaches," in *ACNS 2009,*, vol. 5536, 2009, pp. 437–455.
[67] T. Wunderer, "A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack," *J. Math. Cryptol.*, vol. 13, no. 1, pp. 1–26, 2019.
[68] J. Buchmann, F. Göpfert, R. Player, and 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.
[69] N. Howgrave-Graham, "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU," in *CRYPTO 2007*, vol. 4622, 2007, pp. 150–169.
[70] P. Nguyen, "Boosting the hybrid attack on ntru: torus lsh, permuted hnf and boxed sphere," in *NIST Third PQC Standardization Conference*, 2021.
[71] A. Duc, F. Tramèr, and S. Vaudenay, "Better algorithms for LWE and LWR," in *EUROCRYPT 2015*, vol. 9056, 2015, pp. 173–202.
[72] 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.
[73] J. H. Cheon, M. Hhan, S. Hong, and Y. Son, "A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE," *IEEE Access*, vol. 7, pp. 89 497–89 506, 2019.
[74] T. Espitau, A. Joux, and N. Kharchenko, "On a hybrid approach to solve small secret lwe," *Cryptology ePrint Archive*, 2020.
[75] N. Howgrave-Graham and A. Menezes, "A hybrid meet-in-the-middle and lattice reduction attack on ntru," in *CRYPTO*, 2007, pp. 150–169.
[76] Y. Son and J. H. Cheon, "Revisiting the hybrid attack on sparse and ternary secret LWE," *IACR Cryptol. ePrint Arch.*, p. 1019, 2019.
[77] P. Kirchner and P. Fouque, "Revisiting lattice attacks on overstretched NTRU parameters," in *EUROCRYPT 2017*, vol. 10210, 2017, pp. 3–26.
[78] S. Hallgren, "Fast quantum algorithms for computing the unit group and class group of a number field," in *STOC 2005*, 2005, pp. 468–474.
[79] K. Eisenträger, S. Hallgren, A. Y. Kitaev, and F. Song, "A quantum algorithm for computing the unit group of an arbitrary degree number field," in *STOC 2014*, 2014, pp. 293–302.
[80] J. Biasse and 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.
[81] H. Cohen, *Advanced topics in computational number theory*. Springer Science & Business Media, 2012, vol. 193.
[82] P. Campbell, M. Groves, and D. Shepherd, "Soliloquy: A cautionary tale," in *ETSI 2nd Quantum-Safe Crypto Workshop*, vol. 3, no. 9, 2014, pp. 1–9.
[83] R. Avanzi and H. M. Heys, Eds., *SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, vol. 10532, 2017.
[84] A. Pellet-Mary, G. Hanrot, and D. Stehlé, "Approx-svp in ideal lattices with pre-processing," in *EUROCRYPT 2019*, vol. 11477, 2019, pp. 685–716.
[85] D. J. Bernstein, "S-unit attacks," Aug. 2016. [Online]. Available: https://groups.google.com/g/cryptanalytic-algorithms/c/mCMdsFemzQk/m/3cewE8Q5BwAJ
[86] A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant learning, the parity problem, and the statistical query model," *J. ACM*, vol. 50, no. 4, pp. 506–519, 2003.
[87] M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret, "On the complexity of the BKW algorithm on LWE," *Des. Codes Cryptogr.*, vol. 74, no. 2, pp. 325–354, 2015.
[88] Q. Guo, T. Johansson, and P. Stankovski, "Coded-bkw: Solving LWE using lattice codes," in *CRYPTO 2015*, vol. 9215, 2015, pp. 23–42.
[89] P. Kirchner and P. Fouque, "An improved BKW algorithm for LWE with applications to cryptography and lattices," in *CRYPTO 2015*, vol. 9215, 2015, pp. 43–62.
[90] M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, "On the efficacy of solving LWE by reduction to unique-svp," in *ICISC 2013*, vol. 8565, 2013, pp. 293–310.
[91] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption," in *CT-RSA 2011*, vol. 6558, 2011, pp. 319–339.
[92] M. Liu and P. Q. Nguyen, "Solving BDD by enumeration: An update," in *CT-RSA 2013*, vol. 7779, 2013, pp. 293–309.
[93] P. Ravi, M. F. Ezerman, S. Bhasin, A. Chattopadhyay, and S. S. Roy, "Will you cross the threshold for me? generic side-channel assisted chosen-ciphertext attacks on ntru-based kems," *IACR TCHES*, vol. 2022, no. 1, pp. 722–761, 2022.

[94] D. J. Bernstein, "A one-time single-bit fault leaks all previous ntru-hrss session keys to a chosen-ciphertext attack," *IACR Cryptol. ePrint Arch.*, p. 1125, 2022. [Online]. Available: https://eprint.iacr.org/2022/1125

[95] R. Cramer, L. Ducas, C. Peikert, and O. Regev, "Recovering short generators of principal ideals in cyclotomic rings," in *EUROCRYPT 2016*, vol. 9666, 2016, pp. 559–585.

[96] R. Cramer, L. Ducas, and B. Wesolowski, "Short stickelberger class relations and application to ideal-svp," in *EUROCRYPT 2017*, vol. 10210, 2017, pp. 324–348.

[97] C. M. Chung, V. Hwang, M. J. Kannwischer, G. Seiler, C. Shih, and B. Yang, "NTT multiplication for ntt-unfriendly rings new speed records for saber and NTRU on cortex-m4 and AVX2," *IACR TCHES*, vol. 2021, no. 2, pp. 159–188, 2021.

[98] A. Abdulrahman, J. Chen, Y. Chen, V. Hwang, M. J. Kannwischer, and B. Yang, "Multi-moduli ntts for saber on cortex-m3 and cortex-m4," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 1, pp. 127–151, 2022.

[99] 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.

[100] G. Seiler, "Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography," *IACR Cryptol. ePrint Arch.*, p. 39, 2018. [Online]. Available: http://eprint.iacr.org/2018/039

[101] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[102] D. J. Bernstein and T. Lange, "ebacs: Ecrypt benchmarking of cryptographic systems," https://bench.cr.yp.to/, month = August, year = 2022,.

[103] T. Fritzmann, T. Pöppelmann, and J. Sepúlveda, "Analysis of error-correcting codes for lattice-based key exchange," in *SAC 2018*, vol. 11349, 2018, pp. 369–390.

[104] J. Wang and C. Ling, "How to construct polar codes for ring-lwe-based public key encryption," *Entropy*, vol. 23, no. 8, p. 938, 2021.

[105] D. J. Bernstein, B. B. Brumley, M. Chen, and N. Tuveri, "Opensslntru: Faster post-quantum TLS key exchange," in *USENIX Security 2022 (to appear)*, 2022.

[106] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, and O. Blazy, "Bike: Bit flipping key encapsulation (round 3 submission)," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[107] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, and J. Gilcher, "Classic mceliece: conservative code-based cryptography," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[108] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, and O. Blazy, "Hamming quasi-cyclic (hqc): Third round version," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[109] D. Jao, C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, and O. Blazy, "Supersingular isogeny key encapsulation," *NIST Post-Quantum Cryptography Standardization Process*, 2020.

[110] NIST, "Post-quantum cryptography, round 4 submissions," https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions, 2022.

[111] A. W. Dent, "A designer's guide to kems," in *IMACC 2003*, vol. 2898, 2003, pp. 133–151.

[112] J. Coron, H. Handschuh, M. Joye, P. Paillier, D. Pointcheval, and C. Tymen, "GEM: A generic chosen-ciphertext secure encryption method," in *CT-RSA 2002*, vol. 2271, 2002, pp. 263–276.

[113] T. Okamoto and D. Pointcheval, "React: Rapid enhanced-security asymmetric cryptosystem transform," in *CT-RSA 2001*, 2001, pp. 159–174.

[114] H. Jiang, Z. Zhang, L. Chen, H. Wang, and Z. Ma, "Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited," in *CRYPTO 2018*, vol. 10993, 2018, pp. 96–125.

[115] A. Ambainis, M. Hamburg, and D. Unruh, "Quantum security proofs using semi-classical oracles," in *CRYPTO 2019*, vol. 11693, 2019, pp. 269–295.

[116] H. Jiang, Z. Zhang, and Z. Ma, "Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model," in *PQCrypto 2019*, vol. 11505, 2019, pp. 227–248.

[117] N. Bindel, M. Hamburg, K. Hövelmanns, A. Hülsing, and E. Persichetti, "Tighter proofs of CCA security in the quantum random oracle model," in *TCC 2019*, vol. 11892, 2019, pp. 61–90.

[118] V. Kuchta, A. Sakzad, D. Stehlé, R. Steinfeld, and S. Sun, "Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security," in *EUROCRYPT 2020*, vol. 12107, 2020, pp. 703–728.

[119] K. Hövelmanns, E. Kiltz, S. Schäge, and D. Unruh, "Generic authenticated key exchange in the quantum random oracle model," in *PKC 2020*, vol. 12111, 2020, pp. 389–422.

[120] H. Jiang, Z. Zhang, and Z. Ma, "Key encapsulation mechanism with explicit rejection in the quantum random oracle model," in *PKC 2019*, vol. 11443, 2019, pp. 618–645.

[121] J. Don, S. Fehr, C. Majenz, and C. Schaffner, "Online-extractability in the quantum random-oracle model," *IACR Cryptol. ePrint Arch.*, p. 280, 2021.

[122] T. Saito, K. Xagawa, and T. Yamakawa, "Tightly-secure key-encapsulation mechanism in the quantum random oracle model," in *EUROCRYPT 2018*, vol. 10822, 2018, pp. 520–551.

[123] H. Jiang, Z. Zhang, and Z. Ma, "On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model," in *ASIACRYPT 2021*, vol. 13090, 2021, pp. 487–517.