AKCN-E8: Compact and Flexible KEM from Ideal Lattice*

Zhengzhong Jin
Department of Computer Science, Johns Hopkins University, USA

Yunlei Zhao
Department of Computer Science, Fudan University, China

Abstract

A remarkable breakthrough in mathematics in recent years is the proof of the long-standing conjecture: sphere packing (i.e., packing unit balls) in the E_8 lattice is optimal in the sense of the best density [V17] for sphere packing in $\mathbb{R}^8$. In this work, based on the E_8 lattice code, we design a mechanism for asymmetric key consensus from noise (AKCN), referred to as AKCN-E8, for error correction and key consensus. As a direct application of the AKCN-E8 code, we present highly practical key encapsulation mechanism (KEM) from the ideal lattice based on the ring learning with errors (RLWE) problem. Compared to the RLWE-based NewHope-KEM [NH-NIST], which is a variant of NewHope-Usenix [NH-USENIX] and is now a promising candidate in the second round of NIST post-quantum cryptography (PQC) standardization competition, our AKCN-E8-KEM has the following advantages:

- The size of shared-key is doubled,.
- More compact ciphertexts, at the same or even higher security level.
- More flexible parameter selection for tradeoffs among security, ciphertext size and error probability.

1 Introduction

Advancements in quantum computing have spurred the development of new public-key cryptographic primitives that are conjectured to be secure against quantum attacks. One promising class of these primitives is based on lattices, leading to key encapsulation mechanisms (KEM) based on the *learning with errors* (LWE) problem [NIST]. For cryptographic usage, compared with the classic hard lattice problems such as SVP and CVP, the LWE problem is proven to be much more versatile [Reg09]. Nevertheless, LWE-based cryptosystems are usually less efficient, which was then resolved by the introduction of the ring-LWE (RLWE) problem [LPR10] from ideal lattice. Among RLWE-based asymmetric primitives, NewHope-KEM [NH-NIST] is one of the prominent KEM schemes, which is a variant of NewHope-Usenix [NH-USENIX] (winner of the 2016 Internet Defense Prize), and is now a promising candidate in the second round of NIST post-quantum cryptography (PQC) standardization competition.

In this work, we review the modular and generalized framework, explicitly proposed in [JZ16, JZ19], for designing and analyzing KEM schemes from LWE and its variant (in particular, RLWE). This modular and generalized framework brings us to focus on one key building block for achieving KEMs from LWE and its variants, which is referred to as *asymmetric key consensus* (AKC). Putting into this framework, the underlying (one-dimensional) AKC mechanisms proposed in [Reg09, LPR10, LP11] encode one key bit per polynomial coefficient. One-dimensional AKC was further optimized in [JZ16, JZ19]. The work [PG13] extended one-dimensional reconciliation mechanisms into multi-dimensional ones based on the lattice code in D_2 (resp., D_4), by encoding one key bit into two (resp., four) polynomial coefficients. This multi-dimensional approach can allow to either increase the error and therefore improve the security of the resulting scheme or to decrease the probability of decryption failures. The D_4 code was adapted into key exchange scheme in [NH-USENIX] and later into KEM schemes in [JZ16, NH-NIST] (NewHope-KEM is actually based on the lattice code in $\mathbb{Z}_4$). But KEM schemes based on the code in D_4 or $\mathbb{Z}_4$ has the following disadvantages: Compared to the D_2-based approach, KEM schemes based on the D_4 code improve the ability of error correction but at the cost of halving the key size. In particular, it limits the flexibility in choosing parameters and in balancing security vs. performance.

1.1 Our Contributions

The encoding and decoding algorithms of E_8 were proposed by Conway and Sloane [CS82]. Recently, a remarkable breakthrough in mathematics is the proof of the long-standing con-

*Preliminary version of this work appeared at https://arxiv.org/abs/1611.06150, Version 3, 16 Feb 2017.

jecture: sphere packing in the E_8 lattice is optimal in the sense of the best density [V17] for packing in $\mathbb{R}^8$. However, to apply the algorithms of [CS82] to our KEM setting, we need to specify a one-to-one mapping from binary strings to lattice points in E_8. A natural way to specify such a mapping is to choose a base for the lattice E_8. Then, transforming the lattice points to the binary strings may involve Gaussian elimination. Compared to this method, our encoding and decoding algorithms integrate the coding of E_8 and the mapping from binary strings to E_8 together. This improves the efficiency by avoiding Gaussian elimination. Finally, we adapt the integrated E_8 code into the KEM setting, by combining it with the AKCN scheme of [JZ16]. The resultant code is referred to as AKCN-E8.

As a direct application of the AKCN-E8 code, we present highly practical KEM scheme based on the RLWE assumption, which is referred to as AKCN-E8-KEM. Compared with NewHope-KEM [NH-NIST], our AKCN-E8-KEM has the following advantages:

- The size of shared-key is doubled.
- More compact ciphertexts, at the same or even higher security level.
- More flexible parameter selection for tradeoffs among security, ciphertext size and error probability.

On the importance and desirability of larger shared-key size. The shared-key size of NewHope-512 (resp., NewHope-1024) is 128 (resp., 256) bits. Recently, the variant of NewHope-Compact proposed in [ABC19] has the shared-key of size of 192 bits. In comparison, the shared-key size of AKCN-E8-512 (resp., -768, -1024) is 256 (resp., 384, 512) bits. Here, we would like to highlight the importance and desirability of larger shared-key size.

- Doubling the shared-key size means more powerful and economic ability of key transportation, at about the same level of security and bandwidth.
- Doubling the shared-key size is important for the targeted security level against Grover's search algorithm, and against the possibility of more sophisticated quantum cryptanalysis in the long run.
- Larger key size is indeed needed in many cryptographic standards. For example, according to different security levels (specifically, 128, 192, 256 bit security), in TLS 1.3 [TLS1.3] it mandates three options for the master secrecy size: 256, 384 and 521, by employing the secp256r1, secp384r1 and secp521r1 curves respectively.

1.2 Related Work

Leech lattice is also proven to be the densest for sphere packing in dimension 24 [CKM+17], and has already been used for error correction in communication protocols [CS93, VB93], for example in the IEEE 802.11a WLAN standard https://standards.ieee.org/standard/802_11-2016.html. On the one hand, its encoding and decoding are more complex and less efficient than the AKCN-E8 code. On the other hand, and more importantly, it is difficult to find parameters of RLWE [Pop16], since it is a 24-dimension lattice. For RLWE-based cryptosystems, we usually use number-theoretic transform (NTT) algorithms to speed up the polynomial multiplications. The NTT algorithms can make the most use of the computational resource when the dimension of RLWE is a power of 2. However, one cannot hope for setting the parameter n to be a power of 2 and a multiple of 24 at the same time. The same issue also occurs when setting the key length for Leech lattice, since the key size usually will be a multiple of 12. In comparison, the E_8 lattice doesn't have the aforementioned problems.

The recommended parameter set of NewHope-KEM aims for about 256-bit classic security and about 230-bit post-quantum security (pq-sec). For the KEM proposals in the second round of NIST PQC standardization, the LAC algorithm [LAC-NIST] is another RLWE-based KEM scheme. To our knowledge, NewHope-KEM and LAC are the only two RLWE-based KEM proposals left in the second round of NIST PQC standardization. LAC uses a different approach for building KEM from RLWE: it uses a small $q = 251$ that is not NTT-friendly, and uses error correction code to lower the failure probability. LAC also proposes parameters for about 256-bit classic security but with a relatively higher failure probability. As a consequence, NewHope-KEM and LAC are incomparable in general.

2 Preliminaries

A string or value α means a binary one, and $|\alpha|$ is its binary length. For any real number x, $\lfloor x \rfloor$ denotes the largest integer that less than or equal to x, and $\lceil x \rfloor = \lfloor x + 1/2 \rfloor$. For any positive integers a and b, denote by $\mathsf{lcm}(a,b)$ the least common multiple of them. For any $i, j \in \mathbb{Z}$ such that $i < j$, denote by $[i, j]$ the set of integers $\{i, i+1, \cdots, j-1, j\}$. For any positive integer t, we let $\mathbb{Z}_t$ denote $\mathbb{Z}/t\mathbb{Z}$. The elements of $\mathbb{Z}_t$ are represented, by default, as $[0, t-1]$. Nevertheless, sometimes, $\mathbb{Z}_t$ is explicitly specified to be represented as $[-\lfloor (t-1)/2 \rfloor, \lfloor t/2 \rfloor]$.

If $\mathcal{S}$ is a finite set, then $|\mathcal{S}|$ is its cardinality, and $x \leftarrow \mathcal{S}$ is the operation of picking an element uniformly at random from $\mathcal{S}$. For two sets $A, B \subseteq \mathbb{Z}_q$, define $A + B \triangleq \{a + b | a \in A, b \in B\}$. For an addictive group $(G, +)$, an element $x \in G$ and a subset $S \subseteq G$, denote by $x + S$ the set containing $x + s$ for all $s \in S$. For a set S, denote by $\mathcal{U}(S)$ the uniform distribution over S. For any discrete random variable X over $\mathbb{R}$, denote $\mathsf{Supp}(X) = \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$.

We use standard notations and conventions below for writing probabilistic algorithms, experiments and interactive protocols. If $\mathcal{D}$ denotes a probability distribution, $x \leftarrow \mathcal{D}$ is the

operation of picking an element according to $\mathcal{D}$. If α is neither an algorithm nor a set, $x \leftarrow \alpha$ is a simple assignment statement. If A is a probabilistic polynomial-time (PPT) algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. By $\Pr[R_1; \cdots; R_n : E]$ we denote the probability of event E, after the ordered execution of random processes $R_1, \cdots, R_n$. A function $f(\lambda)$ is *negligible*, if for every $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

2.1 Key Encapsulation Mechanism (KEM)

We review the definition of KEM given in [D02, HHK17]. A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms. On a security parameter κ, the PPT key generation algorithm KeyGen outputs a key pair (pk, sk), where pk also defines a finite key space $\mathcal{K}$. The PPT encapsulation algorithm Encaps, on input pk, outputs a tuple (K, c) where c is said to be an encapsulation of the key K which is contained in key space $\mathcal{K}$. The deterministic polynomial-time decapsulation algorithm Decaps, on input sk and an encapsulation c, outputs either a key $K := \mathsf{Decaps}(sk, c) \in \mathcal{K}$ or a special symbol $\perp \notin \mathcal{K}$ to indicate that c is not a valid encapsulation. We call KEM δ-*correct* if

$$\Pr[\mathsf{Decaps}(sk, c) \neq K | (pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); (K, c) \leftarrow \mathsf{Encaps}(pk)] \leq \delta.$$

The security notion, indistinguishability under chosen ciphertext attacks (CCA), is defined w.r.t. Figure 1. For any PPT adversary $\mathcal{A}$, define its CCA-advantage as $Adv^{CCA}_{KEM}(\mathcal{A}) := |Pr[\textbf{GAME}\ \mathsf{CCA}\ \text{outputs}\ 1]] - 1/2|$. We say the KEM scheme is CCA-secure, if for any sufficiently larger security parameter and any PPT adversary $\mathcal{A}$, $Adv^{CCA}_{KEM}(\mathcal{A})$ is negligible.

GAME IND-CCA	DECAPS($c \neq c^*$)
$(pk, sk) \leftarrow \mathsf{Gen}$	$K := \mathsf{Decaps}(sk, c)$
$b \xleftarrow{\$} \{0,1\}$	**return** K
$(K_0^*, c^*) \leftarrow \mathsf{Encaps}(pk)$	
$K_1^* \xleftarrow{\$} \mathcal{K}$	
$b' \leftarrow \mathsf{A}^{\text{DECAPS}}(c^*, K_b^*)$	
return $[b' = b]$	

Figure 1: CCA game for KEM

2.2 Public-Key Encryption (PKE)

We review the definition of PKE given in [FO13, HHK17]. A public-key encryption scheme is given by a triple of algorithms, $\mathsf{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where for every sufficiently large $\kappa \in \mathbb{N}$.

- KeyGen, the key-generation algorithm, is a probabilistic polynomial-time (in κ) algorithm which on input 1^κ outputs a pair of strings, (pk, sk), called the public and secret keys, respectively. This experiment is written as $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
- $\mathcal{E}$, the encryption algorithm, is a probabilistic polynomial-time (in κ) algorithm that takes public key pk and message M from the message space $\mathtt{MSP}$, draws coins r uniformly from coin space $\mathtt{COIN}$, and produces ciphertext $C := \mathcal{E}_{pk}(M; r)$. This experiment is written as $C \leftarrow \mathcal{E}_{pk}(x)$.
- $\mathcal{D}$, the decryption algorithm, is a deterministic polynomial-time (in κ) algorithm that takes secret key sk and ciphertext $C \in \{0,1\}^*$, and returns message $M \in \mathtt{MSP}$.

We say a PKE scheme is δ-correct, if for every sufficiently large $\kappa \in \mathbb{N}$, every (pk, sk) generated by $\mathsf{KeyGen}(1^\kappa)$ and every $M \in \mathtt{MSP}$, we always have $\mathbf{E}[\max_{M \in \mathtt{MSP}} \Pr[\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) \neq M]] \leq \delta$.

Definition 2.1 (CCA-security)**.** *Let $\mathsf{PKE} = (\mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme, and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary for PKE. For $\kappa \in \mathbb{N}$, define the following CCA-advantage:*

$$\mathbf{Adv}^{\mathsf{CCA}}_{\mathcal{A}}(\kappa) = 2 \cdot \Pr[(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); (M_0, M_1, st) \leftarrow \mathcal{A}_1^{\mathcal{D}_{sk}}(pk); b \leftarrow \{0,1\}; C^* \leftarrow \mathcal{E}_{pk}(M_b) : \mathcal{A}_2^{\mathcal{D}_{sk}}(C^*, st) = b] - 1.$$

We say that the PKE scheme is CCA-secure, if for every sufficiently large security parameter κ, and PPT adversary $\mathcal{A}$, its CCA-advantage $\mathbf{Adv}^{\mathsf{CCA}}_{\mathcal{A}}$ is negligible in κ. We say the PKE scheme is secure against chosen plaintext attacks (CPA-secure, for short), if the advantage of $\mathcal{A}$ is negligible when the access to the decryption oracle $\mathcal{D}_{sk}$ is denied.

2.3 The LWE, and Ring-LWE (RLWE) problems

Given positive *continuous* $\sigma > 0$, define the real Gaussian function $\rho_\sigma(x) \triangleq \exp(-x^2/2\sigma^2)/\sqrt{2\pi\sigma^2}$ for $x \in \mathbb{R}$. Let $D_{\mathbb{Z},\sigma}$ denote the one-dimensional *discrete* Gaussian distribution over $\mathbb{Z}$, which is determined by its probability density function $D_{\mathbb{Z},\sigma}(x) \triangleq \rho_\sigma(x)/\rho_\sigma(\mathbb{Z}), x \in \mathbb{Z}$. Finally, let $D_{\mathbb{Z}^n,\sigma}$ denote the n-dimensional *spherical* discrete Gaussian distribution over $\mathbb{Z}^n$, where each coordinate is drawn *independently* from $D_{\mathbb{Z},\sigma}$.

Given positive integers n and q that are both polynomials in the security parameter λ, an integer vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{q,\mathbf{s},\chi}$ be the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, and an error term $e \leftarrow \chi$, and outputting the pair

$(\mathbf{a}, b = \mathbf{a}^T\mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The error distribution χ is typically taken to be the discrete Gaussian probability distribution $D_{\mathbb{Z},\sigma}$ defined previously; However, as suggested in [BCD+16] and as we shall see in Section 5, other alternative distributions of χ can be taken. Briefly speaking, the (decisional) *learning with errors* (LWE) assumption [Reg09] says that, for sufficiently large security parameter λ, no probabilistic polynomial-time (PPT) algorithm can distinguish, with non-negligible probability, $A_{q,\mathbf{s},\chi}$ from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. This holds even if $\mathcal{A}$ sees polynomially many samples, and even if the secret vector $\mathbf{s}$ is drawn randomly from χ^n [ACPS09].

For the positive integer m that is polynomial in the security parameter λ, let $n \triangleq \varphi(m)$ denote the toties of m, and $\mathcal{K} \triangleq \mathbb{Q}(\zeta_m)$ be the number field obtained by adjoining an abstract element ζ_m satisfying $\Phi_m(\zeta_m) = 0$, where $\Phi_m(x) \in \mathbb{Z}[x]$ is the m-th cyclotomies polynomial of degree n. Moreover, let $\mathcal{R} \triangleq O_\mathcal{K}$ be the ring of integers in $\mathcal{K}$. Finally, given a positive prime $q = \text{poly}(\lambda)$ such that $q \equiv 1 \pmod{m}$, define the quotient ring $\mathcal{R}_q \triangleq \mathcal{R}/q\mathcal{R}$.

We briefly review the RLWE problem, and its hardness result [LPR10, LPR13b, DD12]. In this work, we focus on a *special* case of the RLWE problem defined in [LPR10]. Let $n \geq 16$ be a power-of-two and $q = \text{poly}(\lambda)$ be a positive prime such that $q \equiv 1 \pmod{2n}$. Given $\mathbf{s} \leftarrow \mathcal{R}_q$, a sample drawn from the RLWE distribution $A_{n,q,\sigma,\mathbf{s}}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is generated by first choosing $\mathbf{a} \leftarrow \mathcal{R}_q, \mathbf{e} \leftarrow D_{\mathbb{Z}^n,\sigma}$, and then outputting $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s} + \mathbf{e}) \in \mathcal{R}_q \times \mathcal{R}_q$. Roughly speaking, the (decisional) RLWE assumption says that, for sufficiently large security parameter λ, no PPT algorithm $\mathcal{A}$ can distinguish, with non-negligible probability, $A_{n,q,\sigma,\mathbf{s}}$ from the uniform distribution over $\mathcal{R}_q \times \mathcal{R}_q$. This holds even if $\mathcal{A}$ sees polynomially many samples, and even if the secret $\mathbf{s}$ is drawn randomly from the same distribution of the error polynomial $\mathbf{e}$ [DD12, ACPS09]. Moreover, as suggested in [NH-USENIX], alternative distributions for the error polynomials can be taken for the sake of efficiency while without essentially reducing security.

Recently, a polynomial-time (quantum) reduction from worst-case ideal lattice problems *directly* to the decision version of Ring-LWE is presented in [PRS17]. In particular, the reduction works for any modulus and any number field. Besides the above special version of the RLWE problem [LPR10], another suggested version of the RLWE problem is defined over the polynomial ring $\mathcal{R}_u = \mathbb{Z}[x]/\Phi_{n+1}(x)$, where $n+1$ is a safe prime and $\Phi_{n+1}(x) = x^n + x^{n-1} + \cdots + x + 1$ is the $(n+1)$-th cyclotomic polynomial. This ring has a wider range of n to choose from.

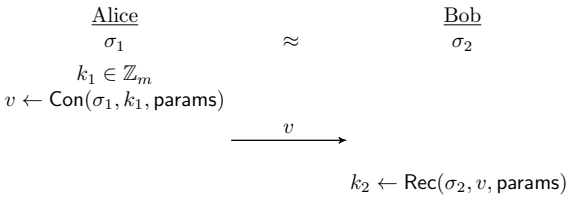

Figure 2: Depiction of AKC

3 A Modular and Generalized Framework for PKE/KEM from Ring-LWE

3.1 Building Block: Asymmetric Key Consensus

Before presenting the definition of asymmetric key consensus (AKC) scheme, we first introduce a new function $|\cdot|_q$ relative to a positive integer $q \geq 1$: $|x|_q = \min\{x \bmod q, q - x \bmod q\}, \quad \forall x \in \mathbb{Z}$, where the result of modular operation is represented in $\{0, ..., (q-1)\}$. For instance, $|-1|_q = \min\{-1 \bmod q, (q+1) \bmod q\} = \min\{q-1, 1\} = 1$. For any $\mathbf{x} = (x_0, x_1, x_2, x_{\mu-1})^T \in \mathbb{Z}_q^\mu$, where μ is a positive integer, denote by $\|\mathbf{x}\|_{q,1}$ the sum $|x_0|_q + |x_1|_q + \cdots + |x_{\mu-1}|_q$.

Definition 3.1. *An asymmetric key consensus scheme AKC = (params, Con, Rec) is specified as follows:*

- $\mathsf{params} = (q, m, g, d, aux)$ *denotes the system parameters, where* q, $2 \leq m, g \leq q, 1 \leq d \leq \lfloor \frac{q}{2} \rfloor$ *are positive integers, and aux denotes some auxiliary values that are usually determined by* (q, m, g, d) *and could be set to be empty.*
- $\mathbf{v} \leftarrow \mathit{Con}(\sigma_1, k_1, \mathsf{params})$*: On input of* $(\sigma_1 \in \mathbb{Z}_q^\mu, \mathbf{k}_1 \in \mathbb{Z}_m^{\mu'}, \mathsf{params})$*, where* μ *is a positive integer, the polynomial-time conciliation algorithm* Con *outputs the public hint* $\mathbf{v} \in \mathbb{Z}_g^\mu$.
- $\mathbf{k}_2 \leftarrow \mathit{Rec}(\sigma_2, \mathbf{v}, \mathsf{params})$*: On input of* $(\sigma_2 \in \mathbb{Z}_q^\mu, \mathbf{v} \in \mathbb{Z}_g^\mu, \mathsf{params})$*, the* deterministic *polynomial-time algorithm* Rec *outputs* $\mathbf{k}_2 \in \mathbb{Z}_m^{\mu'}$.

Correctness: An AKC scheme is correct*, if it holds* $\mathbf{k}_1 = \mathbf{k}_2$ *for any* $\sigma_1, \sigma_2 \in \mathbb{Z}_q^\mu$ *such that* $\|\sigma_1 - \sigma_2\|_{q,1} \leq d$.
Security: An AKC scheme is secure*, if* $\mathbf{v}$ *is independent of* $\mathbf{k}_1$ *whenever* σ_1 *is uniformly distributed over* $\mathbb{Z}_q^\mu$*. Specifically, for arbitrary* $\tilde{\mathbf{v}} \in \mathbb{Z}_g^\mu$ *and arbitrary* $\tilde{\mathbf{k}}_1, \tilde{\mathbf{k}}_1' \in \mathbb{Z}_m^{\mu'}$*, it holds that* $\Pr[\mathbf{v} = \tilde{\mathbf{v}} | \mathbf{k}_1 = \tilde{\mathbf{k}}_1] = \Pr[\mathbf{v} = \tilde{\mathbf{v}} | \mathbf{k}_1 = \tilde{\mathbf{k}}_1']$*, where the probability is taken over* $\sigma_1 \leftarrow \mathbb{Z}_q^\mu$ *and the random coins possibly used by* Con.

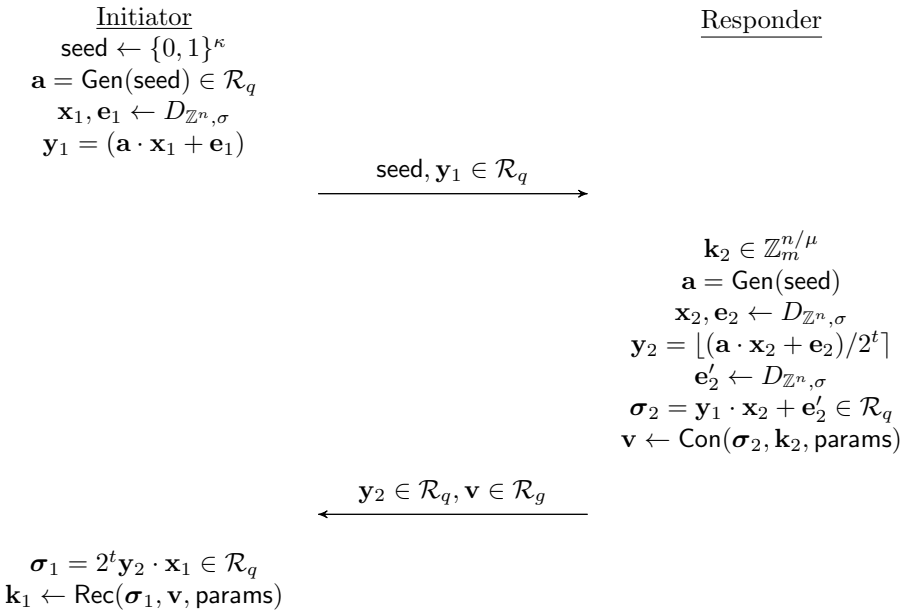

Figure 3: Depiction of RLWE-based CPA-secure PKE from AKC

3.2 CPA-Secure PKE from AKC

Denote by $(\lambda, n, q, \sigma, AKC)$ the system parameters, where λ is the security parameter. $q \geq 2$ is a positive prime number, σ parameterizes the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$, n denotes the degree of polynomials in $\mathcal{R}_q$ where for simplicity we assume $\mu | n$, and Gen is a pseudorandome generator (PRG) generating $\mathbf{a} \in \mathcal{R}_q$ from a small seed $\mathsf{seed} \leftarrow \{0,1\}^\kappa$. Let $AKC = (\mathsf{params}, \mathsf{Con}, \mathsf{Rec})$ be a correct and secure AKC scheme, where $\mathsf{params} = (q, g, m, d)$. In this work, we mainly consider $m = 2$. The AKC-based PKE from RLWE is depicted in Figure 3 (page 5). Here, $(\mathsf{seed}, \mathbf{y}_1)$ serves as the public key, while $(\mathbf{y}_2, \mathbf{v})$ is the ciphertext. In the protocol description, for presentation simplicity, the Con and Rec functions are applied to polynomials, meaning they are applied to each group of μ coefficients respectively. For NewHope $\mu = 4$, while for AKCN-E8 $\mu = 8$. For presentation simplicity, we also referred to $\mathbf{k}_1 = \mathbf{k}_2$ as the shared-key.

It is well established that, under the assumptions that (1) the underlying AKC scheme is both correct and secure, and (2) the (decisional) RLWE is hard, the above modular construction of PKE scheme is CPA-secure [Reg09, LPR10, LP11, JZ16, BCD+16, JZ19]. The above modular and generalized framework for CPA-secure PKE from LWE and its variants was explicitly proposed by Jin and Zhao [JZ16], by explicitly defining and studying the underlying building tool AKC. All the previous works used AKC implicitly in a non-black-box way. The literature should appreciate such an effort of abstraction and generalization. In general, abstraction and generalization are fundamental to natural science (e.g., mathematics, physics), and are particularly important to cryptography. For example, in the area of signature, Schnorr signature is generalized via Fiat-Shamir transformation [FS86], with abstraction of Σ-protocol [CDS94]. The similar abstraction and generalization also plays a fundamental role in CCA-secure PKE, and in many more areas of modern cryptography. Abstraction and generalization are particularly helpful and expected for lattice-based cryptography, as they are usually less easy to understand and evaluate, and are related to the ongoing NIST post-quantum cryptography standardization [NIST].

3.3 Transformation from CPA-PKE to CCA-KEM

There are well-established approaches from CPA-secure PKE to CCA-secure KEM [FO99, FO13, TU16, HHK17, HKSU18, JZM19], with concrete security estimation in the quantum random oracle model (QROM). In this work, for presentation simplicity and ease of comparison, we use the same CCA transformation approach adopted by NewHope-KEM. The reader is referred to [NH-NIST] for more details.

4 Design and Analysis of AKCN-E8

According to the above modular and generalized framework from AKC to RLWE-based CPA and CCA secure KEMs, all left is to develop a practical AKC scheme, which is referred to AKCN-E8 to be developed and analyzed in this section. At the heart of AKCN-E8 is a novel lattice code in E_8.

We divide the coefficients of the polynomial $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ into $\hat{n} = n/8$ groups, where each group is composed of 8 coefficients. In specific, denote $R = \mathbb{Z}[x]/(x^8 + 1), R_q =$

$R/qR, K = \mathbb{Q}[x]/(x^8+1)$ and $K_{\mathbb{R}} = K \otimes \mathbb{R} \simeq \mathbb{R}[x]/(x^8+1)$. Then the polynomial $\boldsymbol{\sigma}_1$ can be represented as $\boldsymbol{\sigma}_1(x) = \sigma_0(x^{\hat{n}}) + \sigma_1(x^{\hat{n}})x + \cdots + \sigma_{\hat{n}-1}(x^{\hat{n}})x^{\hat{n}-1}$, where $\sigma_i(x) \in R_q$ for $i = 0, 1, \ldots \hat{n}$. $\boldsymbol{\sigma}_2$ can be divided in the same way. Then we only need to construct the reconciliation mechanism for each $\sigma_i(x)$, and finally combine the keys together. To do this, we need to first introduce the lattice E_8 and its encoding and decoding.

We construct lattice E_8 from the Extended Hamming Code in dimension 8, which is denoted as H_8 for presentation simplicity. H_8 refers to the 4-dimension linear subspace of 8-dimension linear space $\mathbb{Z}_2^8$.

$$H_8 = \{\mathbf{c} \in \mathbb{Z}_2^8 \mid \mathbf{c} = \mathbf{zH} \bmod 2, \mathbf{z} \in \mathbb{Z}^4\}$$

where

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

The encoding algorithm is straightforward: given a 4-bit string $\mathbf{k}_1$, calculate $\mathbf{k}_1\mathbf{H}$. This operation can be done efficiently by bitwise operations. The complete algorithm is shown in Algorithm 1.[1]

Algorithm 1 AKCN-E8: Con with encoding in E_8

procedure $\mathsf{Con}(\boldsymbol{\sigma}_1 \in \mathbb{Z}_q^8, \mathbf{k}_1 \in \mathbb{Z}_2^4, \mathsf{params})$
  $\mathbf{v} = \left\lfloor \frac{g}{q}\left(\boldsymbol{\sigma}_1 + \frac{q-1}{2}(\mathbf{k}_1\mathbf{H} \bmod 2)\right)\right\rceil \bmod g$[2]
  return $\mathbf{v}$
end procedure

The decoding algorithm finds the solution of the closest vector problem (CVP) for the lattice E_8. For any given $\mathbf{x} \in \mathbb{R}^8$, CVP asks which lattice point in E_8 is closest to $\mathbf{x}$. Based on the structure of E_8, we propose an efficient decoding algorithm.

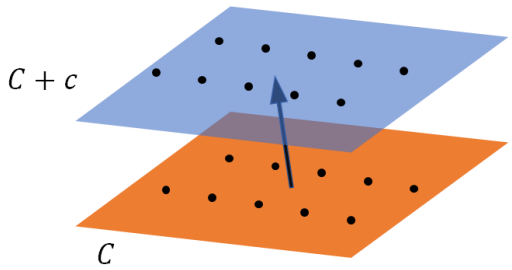

Figure 4: Structure of E_8.

Let $C = \{(x_1,x_1,x_2,x_2,x_3,x_3,x_4,x_4) \in \mathbb{Z}_2^8 \mid x_1+x_2+x_3+x_4 = 0 \bmod 2\}$. In fact, C is spanned by the up most three rows of $\mathbf{H}$. Hence, $E_8 = C \cup (C+\mathbf{c})$, where $\mathbf{c} = (0,1,0,1,0,1,0,1)$ is the last row of $\mathbf{H}$. For a given $\mathbf{x} \in \mathbb{R}^8$, to solve CVP of $\mathbf{x}$ in E_8, we solve CVP of $\mathbf{x}$ and $\mathbf{x}-\mathbf{c}$ in C, and then choose the one that has smaller distance. For a pictorial representation of E_8, refer to Figure 4.

Algorithm 2 AKCN-E8: Rec with decoding in E_8

procedure $\mathsf{Rec}(\boldsymbol{\sigma}_2 \in \mathbb{Z}_q^8, \mathbf{v} \in \mathbb{Z}_g^8, \mathsf{params})$
  $\mathbf{k}_2 = \mathsf{Decode}_{E_8}\left(\left\lfloor \frac{q}{g}\mathbf{v}\right\rceil - \boldsymbol{\sigma}_2\right)$
  return $\mathbf{k}_2$
end procedure

Then we consider how to solve CVP in C. For an $\mathbf{x} \in \mathbb{R}^8$, we choose $(x_1,x_2,x_3,x_4) \in \mathbb{Z}_2^4$, such that $(x_1,x_1,x_2,x_2,x_3,x_3,x_4,x_4)$ is closest to $\mathbf{x}$. However, $x_1+x_2+x_3+x_4 \bmod 2$ may equal to 1. In such cases, we choose the 4-bit string (x_1',x_2',x_3',x_4') such that $(x_1',x_1',x_2',x_2',x_3',x_3',x_4',x_4')$ is secondly closest to $\mathbf{x}$. Note that (x_1',x_2',x_3',x_4') has at most one-bit difference from (x_1,x_2,x_3,x_4). The detailed algorithm is depicted in Algorithm 3. Considering potential timing attack, all the "if" conditional statements can be implemented by constant time bitwise operations. In practice, Decode_C^{00} and Decode_C^{01} are implemented as two subroutines.

For Algorithm 3 (page 7), in Decode_{E_8}, we calculate $\mathsf{cost}_{i,b}$, where $i = 0,1,\ldots,7, b \in \{0,1\}$, which refer to the contribution to the total 2-norm when $x_i = b$. Decode_C^{00} solves the CVP in lattice C, and Decode_C^{01} solves the CVP in lattice $C+\mathbf{c}$. Then we choose the one that has smaller distance. $\mathsf{Decode}_C^{b_0b_1}$ calculates the $k_i, i = 0,1,2,3$ such that $\frac{q-1}{2}(k_0 \oplus b_0, k_0 \oplus b_1, k_1 \oplus b_0, k_1 \oplus b_1, k_2 \oplus b_0, k_2 \oplus b_1, k_3 \oplus b_0, k_3 \oplus b_1)$ is closest to $\mathbf{x}$. We use min_d and min_i to find the second closest vector. Finally, we check the parity to decide which one should be returned.

The following theorem gives a condition of success of the encoding and decoding algorithm in Algorithm 1 and Algorithm 2. For simplicity, for any $\boldsymbol{\sigma} = (x_0,x_1,\ldots,x_7) \in \mathbb{Z}_q^8$, we define $\|\boldsymbol{\sigma}\|_{q,2}^2 = \sum_{i=0}^7 |x_i|_q^2$.

Theorem 4.1. *If* $\|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2\|_{q,2} \le (q-1)/2 - \sqrt{2}\left(\frac{q}{g}+1\right)$, *then* $\mathbf{k}_1$ *and* $\mathbf{k}_2$ *calculated by* Con *and* Rec *are equal.*

Proof. The minimal Hamming distance of the Extended Hamming code H_8 is 4. Hence, the minimal distance in the lattice we used is $\frac{1}{2}\sqrt{\left(\frac{q-1}{2}\right)^2 \times 4} = (q-1)/2$.

We can find $\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}_1 \in [-1/2, 1/2]^8, \boldsymbol{\theta} \in \mathbb{Z}^8$ such that

$$\begin{aligned}\left\lfloor \frac{q}{g}\mathbf{v}\right\rceil - \boldsymbol{\sigma}_2 &= \frac{q}{g}\mathbf{v} + \boldsymbol{\varepsilon} - \boldsymbol{\sigma}_2 \\ &= \frac{q}{g}\left(\frac{g}{q}\left(\boldsymbol{\sigma}_1 + \frac{q-1}{2}\mathbf{k}_1\mathbf{H}\right) + \boldsymbol{\varepsilon} + \boldsymbol{\theta}g\right)\end{aligned}$$

[1] For simplicity, we assume q is a prime and directly use $\frac{q-1}{2}$ in Con (rather than $\lfloor q/2 \rceil$). The construction and analysis can be trivially changed to work with $\frac{q+1}{2}$ in Con. Also, when q is an even number (e.g., power-of-two), it should be $\frac{q}{2}$.

Algorithm 3 Decoding in E_8 and C

procedure $\mathsf{Decode}_{E_8}(\mathbf{x} \in \mathbb{Z}_q^8)$
  for $i = 0 \dots 7$ **do**
    $\mathsf{cost}_{i,0} = |x_i|_q^2$
    $\mathsf{cost}_{i,1} = |x_i - \frac{q-1}{2}|_q^2$
  end for
  $(\mathbf{k}^{00}, \mathsf{TotalCost}^{00}) \leftarrow \mathsf{Decode}_C^{00}(\mathsf{cost}_{i\in 0\dots 7, b\in\{0,1\}})$
  $(\mathbf{k}^{01}, \mathsf{TotalCost}^{01}) \leftarrow \mathsf{Decode}_C^{01}(\mathsf{cost}_{i\in 0\dots 7, b\in\{0,1\}})$
  if $\mathsf{TotalCost}^{00} < \mathsf{TotalCost}^{01}$ **then**
    $b = 0$
  else
    $b = 1$
  end if
  $(k_0, k_1, k_2, k_3) \leftarrow \mathbf{k}^{0b}$
  $\mathbf{k}_2 = (k_0, k_1 \oplus k_0, k_3, b)$
  return $\mathbf{k}_2$
end procedure
procedure $\mathsf{Decode}_C^{b_0 b_1}(\mathsf{cost}_{i\in 0\dots 7, b\in\{0,1\}} \in \mathbb{Z}^{8\times 2})$
  $min_d = +\infty$
  $min_i = 0$
  $\mathsf{TotalCost} = 0$
  for $j = 0 \dots 3$ **do**
    $c_0 \leftarrow \mathsf{cost}_{2j,b_0} + \mathsf{cost}_{2j+1,b_1}$
    $c_1 \leftarrow \mathsf{cost}_{2j,1-b_0} + \mathsf{cost}_{2j+1,1-b_1}$
    if $c_0 < c_1$ **then**
      $k_i \leftarrow 0$
    else
      $k_i \leftarrow 1$
    end if
    $\mathsf{TotalCost} \leftarrow \mathsf{TotalCost} + c_{k_i}$
    if $c_{1-k_i} - c_{k_i} < min_d$ **then**
      $min_d \leftarrow c_{1-k_i} - c_{k_i}$
      $min_i \leftarrow i$
    end if
  end for
  if $k_0 + k_1 + k_2 + k_3 \bmod 2 = 1$ **then**
    $k_{min_i} \leftarrow 1 - k_{min_i}$
    $\mathsf{TotalCost} \leftarrow \mathsf{TotalCost} + min_d$
  end if
  $\mathbf{k} = (k_0, k_1, k_2, k_3)$
  return $(\mathbf{k}, \mathsf{TotalCost})$
end procedure

$$+\boldsymbol{\varepsilon}_1-\boldsymbol{\sigma}_2$$
$$=(\boldsymbol{\sigma}_1-\boldsymbol{\sigma}_2)+\frac{q-1}{2}\mathbf{k}_1\mathbf{H}+\frac{q}{g}\boldsymbol{\varepsilon}+\boldsymbol{\varepsilon}_1+\boldsymbol{\theta}q$$

Hence, the bias from $\frac{q-1}{2}\mathbf{k}_1\mathbf{H}$ is no larger than $\|\boldsymbol{\sigma}_1-\boldsymbol{\sigma}_2\|_{q,2}+\frac{q}{g}\|\boldsymbol{\varepsilon}\|+\sqrt{2}\le\|\boldsymbol{\sigma}_1-\boldsymbol{\sigma}_2\|_{q,2}+\sqrt{2}\left(\frac{q}{g}+1\right)$. If this value is less than the minimal distance $(q-1)/2$, the decoding will be correct, which implies $\mathbf{k}_1=\mathbf{k}_2$. □

Proposition 4.1. *AKCN-E8 is* secure. *Specifically, if* $\boldsymbol{\sigma}_1$ *is subject to uniform distribution over* $\mathbb{Z}_q^8$*, then* $\mathbf{v}$ *and* $\mathbf{k}_1$ *are independent.*

Proof. For arbitrary fixed $\mathbf{k}_1$, $\mathbf{k}_1\mathbf{H}$ mod 2 is fixed. Since $\boldsymbol{\sigma}_1$ is uniform random, $\boldsymbol{\sigma}_1+\frac{q}{2}(\mathbf{k}_1\mathbf{H} \bmod 2)$ is uniform random over $\mathbb{Z}_q$. Thus, $\mathbf{v}$ is subject to the distribution $\lfloor\frac{g}{q}\mathbf{u}\rceil \bmod g$, where $\mathbf{u}$ is uniform random over $\mathbb{Z}_q$. Hence, $\mathbf{v}$ is independent of $\mathbf{k}_1$. □

4.1 Failure Rate Analysis

Now, with respect to the CPA-secure PKE scheme described in Figure 3 with the underlying AKC is replaced with AKCN-E8, we analyze the correctness property by calculating its failure rate.

Denote $\boldsymbol{\varepsilon}=\mathbf{a}\mathbf{x}_2+\mathbf{e}_2-2^t\lfloor(\mathbf{a}\mathbf{x}_2+\mathbf{e}_2)/2^t\rceil$. We have

$$\begin{aligned}\boldsymbol{\sigma}_1-\boldsymbol{\sigma}_2&=\mathbf{x}_1(2^t\mathbf{y}_2)-(\mathbf{y}_1\mathbf{x}_2+\mathbf{e}_2')\\&=2^t\mathbf{x}_1\lfloor(\mathbf{a}\mathbf{x}_2+\mathbf{e}_2)/2^t\rceil-((\mathbf{a}\mathbf{x}_1+\mathbf{e}_1)\mathbf{x}_2+\mathbf{e}_2')\\&=\mathbf{x}_1(\mathbf{a}\mathbf{x}_2+\mathbf{e}_2-\boldsymbol{\varepsilon})-(\mathbf{a}\mathbf{x}_1\mathbf{x}_2+\mathbf{e}_1\mathbf{x}_2+\boldsymbol{e}_2')\\&=\mathbf{x}_1(\mathbf{e}_2-\boldsymbol{\varepsilon})-(\mathbf{e}_1\mathbf{x}_2+\mathbf{e}_2')\end{aligned}$$

From RLWE assumption, $(\mathbf{a},\mathbf{a}\mathbf{x}_2+\mathbf{e}_2)$ is indistinguishable with $(\mathbf{a},\mathbf{u})$, where $\mathbf{u}$ is subject to the uniform distribution. Then, $\boldsymbol{\varepsilon}$ should be closed to $\mathbf{u}-2^t\lfloor\mathbf{u}/2^t\rceil$. We can roughly regard each coefficients of polynomials in $\mathbf{u}-2^t\lfloor\mathbf{u}/2^t\rceil$ as uniform distribution over $[-2^{t-1},2^{t-1}]^n$. Let σ_t be the standard deviation of uniform distribution over $[-2^{t-1},2^{t-1}]^n$. Then we can calculate the standard deviation of each coefficients of polynomials in $\boldsymbol{\sigma}_2-\boldsymbol{\sigma}_1$, denote it as s. We have

$$\begin{aligned}s^2&=n\sigma^2\left(2\sigma^2+\sigma_t^2\right)+\sigma^2\\&=n\sigma^2\left(2\sigma^2+\frac{(1+2^t)^2-1}{12}\right)+\sigma^2\end{aligned}$$

By the Central Limit Theorem, each coefficient of the polynomials in $\boldsymbol{\sigma}_2-\boldsymbol{\sigma}_1$ is close to a Gaussian distribution. From Theorem 4.1, the AKCN-E8 scheme is correct with probability

$$\Pr\left[d'\leftarrow\chi^2(8):\sqrt{d'}\le\left(\frac{q-1}{2}-\sqrt{2}\left(\frac{q}{g}+1\right)\right)/s\right]$$

We provide a script to calculate the concrete failure rate, which is (anonymously) available from http://github.com/AKCN-E8.

5 Parameters and Implementation

The AKCN-E8-KEM scheme resulted from the modular and generalized framework described in Section 3, with the underlying AKC mechanism replaced with the AKCN-E8 scheme presented in Section 4, works on any hard instantiation of the RLWE problem. But if n is power of 2, and prime q satisfies $q \bmod 2n=1$, then number-theoretic transform (NTT) can be used to speed up polynomial multiplication. The performance can be further improved by using the Montgomery arithmetic and AVX2 instruction set [NH-USENIX, NH-NIST]. As in [NH-NIST], the underlying noise distribution is the centered binomial distribution S_η: for some positive integer η, sample $(a_1,\cdots,a_\eta,b_1,\cdots,b_\eta)\leftarrow\{0,1\}^{2\eta}$ and then output $\sum_{i=1}^{\eta}(a_i-b_i)$. For the centered binomial distribution S_η, its standard deviation is $\sigma=\sqrt{\eta/2}$. In NEWHOPE [NH-NIST], $q=12289$, $n=512$ or $n=1024$, $\eta=8$. For ease of comparison, we use the same CCA transformation and the same values of (q,n) of NewHope [NH-NIST] for the construction and implementation of AKCN-E8-KEM.

We use the same script of NewHope-KEM [NH-NIST] for concrete security estimation against the underlying RLWE problem by the best known quantum attacks, and omit the details here for presentation simplicity. The reader is referred to [NH-NIST] for the method and script of concrete security estimation, which is also available from https://newhopecrypto.org/. NewHope-1024 (resp., NewHope-5512) aims for 233-bit (resp., 101-bit) post-quantum security (pq-sec, for short), but gets consensus on the shared-key $\mathbf{k}_1=\mathbf{k}_2$ of size 256 (resp., 128) bits by using a technique first described in [PG13] that encodes one key bit into four polynomial coefficients. We suggest that the shared-key size might not match the target security level in the post-quantum era, in view of the quadratic speedup by Grover's search algorithm and the possibility of more sophisticated quantum cryptanalysis in the long run. Indeed, it is commonly expected that symmetric-key cryptographic primitives like AES need larger key sizes in the post-quantum era. And, in some more critical areas than public commercial usage, larger key sizes actually have already been mandated nowadays. NewHope-KEM is less flexible to increase its shared-key size; for example, if we want a 512-bit shared-key with NewHope-KEM, we have to use a polynomial of degree 2048 that can be significantly less efficient. Thanks to the powerful E_8 lattice code, AKCN-E8-1024 (resp., AKCN-E8-512) reaches the shared-key of size 512 (resp., 256) bits.

The parameters and performance of AKCN-E8-KEM are given in Table 1. For both AKCN-E8-512 and AKCN-E8-1024, we present three sets of parameters: "S" stands for higher *security* level, "E" stands for lower *error* probability, and "C" stands for smaller *ciphertext* size. Compared with NewHope-KEM [NH-NIST], AKCN-E8 always doubles the size of shared-key, which is important to ensure the target security level in the post-quantum era against advanced quantum

	$\|K\|$	n	q	η	g	t	pq-sec	err	pk (B)	cipher (B)
NewHope-512-CPA	128	512	12289	8	2^3	0	101	2^{-213}	928	1088
AKCN-E8-512-S-CPA	256	512	12289	14	2^4	3	110	2^{-224}	928	960
AKCN-E8-512-E-CPA	256	512	12289	8	2^4	4	101	2^{-256}	928	896
AKCN-E8-512-C-CPA	256	512	12289	8	2^3	4	101	2^{-150}	928	832
NewHope-512-CCA	128	512	12289	8	2^3	0	101	2^{-213}	928	1120
AKCN-E8-512-S-CCA	256	512	12289	14	2^4	3	110	2^{-224}	928	992
AKCN-E8-512-E-CCA	256	512	12289	8	2^4	4	101	2^{-256}	928	928
AKCN-E8-512-C-CCA	256	512	12289	8	2^3	4	101	2^{-150}	928	864
NewHope-1024-CPA	256	1024	12289	8	2^3	0	233	2^{-216}	1824	2176
AKCN-E8-1024-S-CPA	512	1024	12289	10	2^4	2	240	2^{-274}	1824	2048
AKCN-E8-1024-E-CPA	512	1024	12289	8	2^4	3	233	2^{-280}	1824	1920
AKCN-E8-1024-C-CPA	512	1024	12289	4	2^3	3	214	2^{-500}	1824	1792
NewHope-1024-CCA	256	1024	12289	8	2^3	0	233	2^{-216}	1824	2208
AKCN-E8-1024-S-CCA	512	1024	12289	10	2^4	2	240	2^{-274}	1824	2080
AKCN-E8-1024-E-CCA	512	1024	12289	8	2^4	3	233	2^{-280}	1824	1952
AKCN-E8-1024-C-CCA	512	1024	12289	4	2^3	3	214	2^{-500}	1824	1824

Table 1: Parameters for AKCN-E8-KEM and comparison with NewHope-KEM [NH-NIST]. $|K|$ refers to the size of shared-key $\mathbf{k}_1 = \mathbf{k}_2$, "pk(B)" refers to the size of $(\mathbf{y}_1, \mathsf{seed})$ in bytes; "cipher(B)" refers to the size of $(\mathbf{y}_2, \mathbf{v})$; "pq-sec" refers to the security of the underlying RLWE problem against the best known quantum attacks.

attacks like Grover algorithms. On the proposed parameters, AKCN-E8 also has more compact ciphertexts than NewHope-KEM. Besides the double of shared-key size, for AKCN-E8-512-S and AKCN-E8-1024-S, they also have stronger security, lower error probability, and smaller ciphertext size *simultaneously*, in comparison with the corresponding versions of NewHope-KEM. For AKCN-E8-512, we recommend to use AKCN-E8-512-C, as its error probability 2^{-150} has already been sufficiently lower than the targeted 101-bit post-quantum security level. The error probability 2^{-213} of NewHope-512 is unnecessarily low for the target security level. The performance advantages of AKCN-E8, as well as its flexibility in parameter selection, are largely enabled by the underlying E_8 lattice code, which is much more dense than the underlying $\mathbb{Z}_4$ lattice code used by NewHope-KEM [NH-NIST, PG13]. Actually, a remarkable breakthrough in mathematics in recent years is that sphere packing (i.e., packing unit balls) in the E_8 lattice is proved to be optimal in the sense of the best density [V17] for packing in $\mathbb{R}^8$.

5.1 Implementation and Benchmark

As we use the same CCA-transformation of NewHope-KEM [NH-NIST], we only present the specifications of CPA-secure AKCN-E8-KEM, which are given in Algorithm 4, 5, 8. Similar to NewHope-KEM [NH-NIST], we also use NTT to speed up the multiplication of the polynomials. The benchmark result for the implementation of AKCN-E8-1024-C-CCA is given in Table 2. The source code is (anonymously) available from http://github.com/AKCN-E8.

In Algorithm 4, the key generation algorithm randomly samples a seed, and then use the seed to deterministically generate seedPublic and seedPrivate. Then the value $\hat{\mathbf{a}}$ is generated honestly using seedPublic. The seed seedPublic is set to be part of the public key pk. The $\mathsf{Encode}(\hat{\mathbf{y}}_1)$ algorithm gathers each 14-bit coefficient in $\hat{\mathbf{y}}_1$ together.

Algorithm 4 Key Generation

function KEYGEN
  $\mathsf{seed} \leftarrow \{0,1\}^{256}$
  $(\mathsf{seedPublic}, \mathsf{seedPrivate}) = \mathsf{H}(\mathsf{seed})$
  $\hat{\mathbf{a}} = \mathsf{GenA}(\mathsf{seedPublic})$
  $\mathbf{x}_1 \leftarrow \mathsf{SampleNoise}(\mathsf{seedPrivate}, 0)$
  $\hat{\mathbf{x}}_1 \leftarrow \mathsf{NTT}(\mathbf{x}_1)$
  $\mathbf{e}_1 \leftarrow \mathsf{SampleNoise}(\mathsf{seedPrivate}, 1)$
  $\hat{\mathbf{e}}_1 \leftarrow \mathsf{NTT}(\mathbf{e}_1)$
  $\hat{\mathbf{y}}_1 \leftarrow \hat{\mathbf{a}} \circ \hat{\mathbf{x}}_1 + \hat{\mathbf{e}}_1$
  return $\mathsf{pk} = (\mathsf{Encode}(\hat{\mathbf{y}}_1), \mathsf{seedPublic}), \mathsf{sk} = \mathsf{Encode}(\hat{\mathbf{x}}_1)$
end function

Algorithm 5 Encryption

function ENCRYPT($\mathsf{pk}, \mathsf{msg}$)
  $(\hat{\mathbf{y}}_1, \mathsf{seedPublic}) = \mathsf{Decode}(\mathsf{pk})$
  $\hat{\mathbf{a}} = \mathsf{GenA}(\mathsf{seedPublic})$
  $\mathbf{x}_2, \mathbf{e}_2, \mathbf{e}'_2 \leftarrow \mathsf{SampleNoise}()$
  $\hat{\mathbf{x}}_2 \leftarrow \mathsf{NTT}(\mathbf{x}_2)$
  $\hat{\mathbf{e}}_2 \leftarrow \mathsf{NTT}(\mathbf{e}_2)$
  $\hat{\mathbf{e}}'_2 \leftarrow \mathsf{NTT}(\mathbf{e}'_2)$
  $\mathbf{y}_2 = \mathsf{NTT}^{-1}(\hat{\mathbf{a}} \circ \hat{\mathbf{x}}_2 + \hat{\mathbf{e}}_2)$
  $\boldsymbol{\sigma}_2 = \mathsf{NTT}^{-1}(\hat{\mathbf{y}}_1 \circ \hat{\mathbf{x}}_2 + \hat{\mathbf{e}}'_2)$
  $\mathbf{v} \leftarrow \mathsf{Con}(\boldsymbol{\sigma}_2, \mathsf{msg})$
  return $\mathsf{ct} = \mathsf{CompressAndEncode}(\mathbf{y}_2, \mathbf{v})$
end function

We use the following Algorithm 6 to encode and compress the ciphertext. In more detail, for each coefficient in $\mathbf{y}_2$, we round it to the range $[0, 2^{11}-1]$. For each coefficient in $\mathbf{v}$, we round it to the range $[0, 2^3-1]$. Then we put the rounded 11-bit coefficients in $\mathbf{y}_2$ in high position, and rounded 3-bit coefficients in $\mathbf{v}$ in low position to get 14-bit integers. To speed up the rounding and other operations, we use bit-operations. Finally we invoke the Encode algorithm to gather the 14-bit integers together. In Algorithm 7, we use a similar algorithm to decompress and decode $\mathbf{y}_2$ and $\mathbf{v}$.

We implement the algorithms on Ubuntu Linux 16.04, GCC version 5.4.0. We run the benchmark on Intel(R) Core(TM) i7-4712MQ CPU @ 2.30GHz, with HyperThreading off. The code is compiled with the option -O3 -fomit-frame-pointer -march=native. The result is in Table 2. We run key generation, encryption and decryption each for 1000 times. The reported time and CPU cycles in Table 2 are the average numbers.

Algorithm 6 Compress and Encode

function COMPRESSANDENCODE($\mathbf{y}_2, \mathbf{v}$)
  $\mathbf{c} = \mathbf{0}$
  for $i = 1 \ldots 1024$ **do**
    $\mathsf{hi} = ((\mathbf{y}_2[i] \ll 11) + 6144)/12289$
    $\mathsf{lo} = ((\mathbf{v}[i] \ll 3) + 6144)/12289$
    $\mathbf{c}[i] = (\mathsf{hi} \ll 3) + \mathsf{lo}$
  end for
  return Encode($\mathbf{c}$)
end function

Algorithm 7 Decode and Decompress

function DECODEANDDECOMPRESS(ct)
  $\mathbf{c} = \mathsf{Decode}(\mathsf{ct})$
  for $i = 1 \ldots 1024$ **do**
    $\mathsf{hi} = (\mathbf{c}[i] \gg 3)$ & 0x7FF
    $\mathsf{lo} = \mathbf{c}[i]$ & 3
    $\mathbf{y}'_2[i] = (\mathsf{hi} * 12289 + 0\text{x}400) \gg 11$
    $\mathbf{v}'[i] = (\mathsf{lo} * 12289 + 0\text{x}4) \gg 3$
  end for
  return $(\mathbf{y}'_2, \mathbf{v}')$
end function

Algorithm 8 Decryption

function DECRYPT(sk, ct)
  $\hat{\mathbf{x}}_1 = \mathsf{Decode}(\mathsf{sk})$
  $(\mathbf{y}'_2, \mathbf{v}') = \mathsf{DecodeAndDecompress}(\mathsf{ct})$
  $\hat{\mathbf{y}}'_2 \leftarrow \mathsf{NTT}(\mathbf{y}'_2)$
  $\boldsymbol{\sigma}_1 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{y}}'_2 \circ \hat{\mathbf{x}}_1)$
  return $\mathsf{Rec}(\boldsymbol{\sigma}_1, \mathbf{v}')$
end function

	AKCN-E8-1024 CCA		NewHope-1024-CCA	
	Time(us)	Cycle	Time(us)	Cycle
Gen	80	185361	91	210020
Enc	128	294398	129	295629
Dec	177	405440	148	338754

Table 2: Benchmark of AKCN-E8

References

[NH-USENIX] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum Key Exchange — A New Hope. *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343. Winner of the 2016 Internet Defense Prize (https://internetdefenseprize.org/)

[ABC19] E. Alkim, Y. A. Bilgin, and M. Cenk. Compact and Simple RLWE Based Key Encapsulation Mechanism. *LATINCRYPT* 2019: 237-256.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. *CRYPTO 2009*: 595-618.

[BCD+16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. *ACM CCS 2016*: 1006-1018.

[CKM+17] H. Cohn, A. Kumar, S. D. Miller, D. Radchenko, M. Viazovska. The Sphere Packing Problem in Dimension 24. *Annuals of Mathematics*, 185 (3): 1017-1033, 2017.

[CS82] J. Conway and N. Sloane. Fast quantizing and decoding algorithm for lattice quantizers and codes. *IEEE Transactions on Information Theory*, 28 (2): 227-232, 1982.

[CS93] J. Conway and N. Sloane. Sphere Packings, Lattices, and Groups. Springer- Verlag, New York, 1993.

[CDS94] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. *CRYPTO* 1994: 174–187.

[D02] A. W. Dent. A Designer's Guide to KEMs. *Cryptology ePrint Archive*, Report 2002/174, 2002.

[DD12] L. Ducas and A. Durmus. Ring-LWE in Polynomial Rings. *PKC 2012*: 34-51.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *CRYPTO* 1986: 186–194.

[FO99] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* Volume 83, Issue 1, pages 24-32, 1999.

[FO13] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, Volume 26, Issue 1, pages 80—101, 2013.

[HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. *TCC* (1) 2017: 341-371.

[HKSU18] K. Hövelmanns, E. Kiltz, S. Schäge, and D. Unruh. Generic Authenticated Key Exchange in the Quantum Random Oracle Model. *Cryptology ePrint Archive*, Report 2018/928.

[JZM19] H. Jiang, Z. Zhang, and Z. Ma. Tighter Security Proofs for Generic Key Encapsulation Mechanism in the Quantum Random Oracle Model. *PQCrypto* 2019: 227-248.

[JZ16] Z. Jin, and Y. Zhao. Optimal Key Consensus in Presence of Noise. CoRR, abs/1611.06150 (2016) https://arxiv.org/abs/1611.06150

[JZ19] Z. Jin, and Y. Zhao. Generic and Practical Key Establishment from Lattice. ACNS 2019: 302-322. (Best Student Paper)

[LP11] R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. *CT-RSA 2011*: 319-339.

[LAC-NIST] X. Lu, Y. Liu, D. Jia, H. Xue, J. He, Z. Zhang, Z. Liu, H. Yang, B. Li, K. Wang. Supporting documentation: LAC. Technical report, National Institute of Standards and Technology. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/round-2-submissions

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. *EUROCRYPT 2010*: 1-23.

[LPR13b] V. Lyubashevsky, C. Peikert, and O. Regev. A Toolkit for Ring-LWE Cryptography. *EUROCRYPT 2013*: 35-54.

[LS19] V. Lyubashevsky and G. Seiler. NTTRU: Truly Fast NTRU Using NTT. *CHES* 2019: 180-201.

[NIST] NIST. Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization

[PRS17] C. Peikert, O. Regev and N. Stephens-Davidowitz. Pseudorandomness of Ring-LWE for Any Ring and Modulus. *STOC* 2017: 461-473.

[Pop16] A.V. Poppelen, Cryptographic Decoding of the Leech Lattice. *Cryptology ePrint Archive*, Report 2016/1050, 2016.

[NH-NIST] T. Pöppelmann, E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. Piedra, P. Schwabe, D. Stebila, M. Albrecht, E. Orsini, V. Osheter, K. Paterson, G. Peer, and N. Smart. Supporting documentation: Newhope. Technical report, National Institute of Standards and Technology. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/round-2-submissions

[PG13] T. Pöppelmann and T. Güneysu. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. *SAC 2013*: 68-85.

[Reg09] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, Volume 56, Issue 6, pages 34, 2009.

[TLS1.3] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, 2018.

[TU16] E. E. Targhi and D. Unruh. Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. *TCC* 2016-B: 192-216.

[VB93] A. Vardy, and Y. Be'ery. Maximum Likelihood Decoding of the Leech Lattice. *IEEE Transactions on Information Theory*, 39(4):1435-1444, 1993.

[V17] M. S. Viazovska. The Sphere Packing Problem in Dimension 8. *Annuals of Mathematics*, 185(3): 991-1015, 2017.

[ZXZ+18] S. Zhou, H. Xue, D. Zhang, K. Wang, X. Lu, B. Li, and J. He. Preprocess-then-NTT Techniques and Its Applications to Kyber and NewHope. *Inscrypt* 2018: 117-137.

[ZPL19] Y. Zhu, Y. Pan and Z. Liu. When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited. *Cryptology ePrint Archive*, 2019/1079.

A More Parameters of AKCN-E8-KEM

The standard NTT technique requires that $q \mod 2n = 1$. Recent advances on the variants of NTT [ZXZ+18, LS19, ABC19, ZPL19] allow us to choose the module q in a more flexible way. For example, we can use $q = 7681$ and $q = 3329$ for AKCN-E8-1024 and AKCN-E8-512. The NTT technique proposed in [LS19] (resp., in [ABC19]) allows us to use $q = 7681$ (resp., $q = 3457$) for AKCN-E8-768. A variant of the NTT technique [LS19] also allows us to use $q = 7681$ for AKCN-E8-640. More parameters of AKCN-E8 enabled by the recent advances of NTT techniques are given in Table 3 and Table 4. We may prefer to the AKCN-E8-7681 parameter sets, as they share the same module $q = 7681$ for AKCN-E8-512, AKCN-E8-640, AKCN-E8-768 and AKCN-E8-1024.

	$\|K\|$	n	q	η	g	t	pq-sec	err	pk (B)	cipher (B)
AKCN-E8-512-CPA-Recom	256	512	7681	4	2^3	4	98	2^{-132}	864	768
AKCN-E8-512-CPA-Option	256	512	7681	6	2^3	3	104	2^{-204}	864	832
AKCN-E8-512-CCA-Recom	256	512	7681	4	2^3	4	98	2^{-125}	864	800
AKCN-E8-512-CCA-Option	256	512	7681	6	2^3	3	104	2^{-204}	864	864
AKCN-E8-640-CPA-Recom	320	640	7681	4	2^3	3	129	2^{-299}	1072	1040
AKCN-E8-640-CPA-Option	320	640	7681	6	2^3	3	137	2^{-159}	1072	1040
AKCN-E8-640-CCA-Recom	320	640	7681	4	2^3	3	129	2^{-299}	1072	1072
AKCN-E8-640-CCA-Option	320	640	7681	6	2^3	3	137	2^{-159}	1072	1072
AKCN-E8-768-CPA-Recom	384	768	7681	4	2^3	3	161	2^{-245}	1280	1248
AKCN-E8-768-CPA-Option	384	768	7681	2	2^3	4	147	2^{-197}	1280	1152
AKCN-E8-768-CCA-Recom	384	768	7681	4	2^3	3	161	2^{-245}	1280	1280
AKCN-E8-768-CCA-Option	384	768	7681	2	2^3	4	147	2^{-197}	1280	1184
AKCN-E8-1024-CPA-Recom	512	1024	7681	4	2^4	3	227	2^{-303}	1696	1792
AKCN-E8-1024-CPA-Option-S	512	1024	7681	6	2^4	2	239	2^{-267}	1696	1960
AKCN-E8-1024-CPA-Option-C	512	1024	7681	2	2^3	3	208	2^{-471}	1696	1664
AKCN-E8-1024-CCA-Recom	512	1024	7681	4	2^4	3	227	2^{-303}	1696	1824
AKCN-E8-1024-CPA-Option-S	512	1024	7681	6	2^4	2	239	2^{-267}	1696	1992
AKCN-E8-1024-CCA-Option-C	512	1024	7681	2	2^3	3	208	2^{-471}	1696	1696

Table 3: Recommended parameters for AKCN-E8-7681. “Recom" (resp., “Option") stands for “Recommended" (resp., “Optional"). We recommend to use the same $q = 7681$ and $\eta = 4$ for all the three sets of parameters: AKCN-E8-512, 768 and 1024.

	$\|K\|$	n	q	η	g	t	pq-sec	err	pk (B)	cipher (B)
AKCN-E8-3329-512-CPA	256	512	3329	2	2^3	3	101	2^{-164}	800	768
AKCN-E8-3329-512-CCA	256	512	3329	2	2^3	3	101	2^{-164}	800	800
AKCN-E8-3329-1024-E-CPA	512	1024	3329	2	2^4	2	230	2^{-303}	1568	1792
AKCN-E8-3329-1024-C-CPA	512	1024	3329	2	2^3	2	230	2^{-178}	1568	1664
AKCN-E8-3329-1024-E-CCA	512	1024	3329	2	2^4	2	230	2^{-303}	1568	1824
AKCN-E8-3329-1024-C-CCA	512	1024	3329	2	2^3	2	230	2^{-178}	1568	1696

Table 4: Parameters for AKCN-E8-3329