Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts

Hao Chen[1], Wei Dai[1], Miran Kim[2], and Yongsoo Song[1]

[1] Microsoft Research, Redmond, USA
{haoche,wei.dai,yongsoo.song}@microsoft.com
[2] UT Health Science Center at Houston, USA
miran.kim@uth.tmc.edu

Abstract. In the past few years, significant progresses on homomorphic encryption (HE) have been made toward both theory and practice. The most promising HE schemes are based on the hardness of the Learning With Errors (LWE) problem or its ring variant (RLWE). In this work, we present new conversion algorithms which switch between different (R)LWE-based HE schemes to take advantages of them. Specifically, we present and combine three ideas to improve the key-switching procedure between LWE ciphertexts, transformation from LWE to RLWE, as well as packing of multiple LWE ciphertexts in a single RLWE encryption. Finally, we demonstrate an application of building a secure channel between a client and a cloud server with lightweight encryption, low communication cost, and capability of homomorphic computation.

Keywords: Homomorphic encryption · Learning with Errors · Key switching.

1 Introduction

In recent years, there have been remarkable advances in cryptographic primitives for secure computation without compromising data privacy. Specifically, homomorphic encryption (HE) [21] has been considered as one of the most attractive solutions due to its conceptual simplicity and efficiency. HE is a cryptosystem which supports arithmetic operation on encrypted data, so that any computational task can be outsourced to a public cloud while data provider does not need to either perform a large amount of work or stay online during the protocol execution. In addition, the concrete efficiency of HE has been improved rapidly by theoretic and engineering optimizations [3, 10, 31]. Recent studies demonstrated that this technology show reasonable performance in real-world tasks such as machine learning [24, 25, 15].

Currently all the best performing HE schemes, such as BGV [6], BFV [5, 17], TFHE [13] and CKKS [11], are based on the hardness of Learning with Errors (LWE) or its ring variant (RLWE). In particular, ring-based HE systems have shown remarkable performance in real-world applications due to efficient use of the ciphertext packing technique [33]. Each HE scheme has its own pros and cons, but it has been relatively less studied how to take advantages of various HE schemes by building conversion algorithms between ciphertexts of different types.

Our Contribution. We provide a toolkit to transform (R)LWE-based ciphertexts and generate another ciphertext under a new key or of a different structure. We present several use cases along with experimental results to show that our techniques can be widely used in applications to obtain better asymptotic and concrete performance. Specifically, we describe a new key-switching (KS) method between LWE ciphertexts, a transformation from an LWE ciphertext into an RLWE-based ciphertext, and a packing algorithm which merges multiple LWE encryptions into a single RLWE ciphertext.

The first two conversions (from LWE to LWE/RLWE) have quasi-linear complexity $\tilde{O}(N)$ where N denotes the dimension of (R)LWE. The last packing algorithm is a generalization of LWE-to-RLWE conversion which achieves a better amortized complexity. Our algorithms are almost optimal in the sense that their complexities are quasi-linear with respect to the size

	LWE to LWE	n LWEs to RLWE	
		Total	Amortized
Previous work [29, 12]	N	N	N/n
Ours	1	$(n-1)+\log(N/n)$	$< 1 + n^{-1} \cdot \log N$

Table 1: Complexity of conversion algorithms, measured by the number of KS operations (or homomorphic automorphisms). N denotes the dimension of (R)LWE and n denotes the number of input LWE ciphertexts to be packed in an RLWE ciphertext. Our second algorithm (LWE to RLWE) is the case of $n = 1$.

of inputs. In addition, all building blocks of our algorithms are depth-free (e.g. homomorphic automorphism), so our solutions do not consume any ciphertext level or modulus during conversion.

Finally, based on a conversion algorithm, we demonstrate a secure communication between a client and a public cloud. The client encrypts data via an LWE-based symmetric encryption on a lightweight device. On receiving LWE ciphertexts, the cloud transforms or packs them into RLWE encryptions to give better functionality for homomorphic arithmetic. Compared to prior works based on block or stream ciphers [20, 2, 28, 16], our approach has various advantages in flexibility, functionality and performance.

Technical Overview. Let N be the dimension and q the modulus of an LWE problem. An LWE ciphertext with secret $\mathbf{s} \in \mathbb{Z}^N$ is of the form $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ and its *phase* is defined as $\mu = b + \langle \mathbf{a}, \mathbf{s} \rangle \pmod q$. It is also called 'half decryption' since the phase of a ciphertext is a randomized encoding of the encrypted plaintext with Gaussian error. Similarly, in the case of RLWE over $R = \mathbb{Z}[X]/(X^N + 1)$ and its residue ring $R_q = R/(q \cdot R)$, the phase of an RLWE ciphertext $(b, a) \in R_q^2$ of secret s is defined as $\mu = b + as \pmod q$.

Suppose that we are given some ciphertexts of a cryptosystem (which is not necessarily to be an HE scheme) and wish to publicly transform them into ciphertexts of another HE scheme for secure computation. In general, this task can be done by evaluating the decryption circuit of initial cryptosystem using an HE system if homomorphically encrypted secret key is given together. Fortunately, the conversion can be more efficient if input ciphertexts are encrypted by an LWE-based cryptosystem because it suffices to homomorphically evaluate the phase, instead of the full decryption which usually includes expensive (non-arithmetic) operations such as rounding or bit extraction.

We remark that this approach can be still inefficient in some cases. For example, if we aim to convert an LWE encryption $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ under secret $\mathbf{s} \in \mathbb{Z}^N$ into an RLWE ciphertext, the secret key owner should generate and publish an RLWE 'encryption' of $\mathbf{s}$ as the evaluation key, and the conversion can be done by computing the phase $\mu = b + \langle \mathbf{a}, \mathbf{s} \rangle$ over RLWE system. In fact, the evaluation key consists of N key-switching (KS) keys from individual s_i to the RLWE secret and the conversion requires N KS operations. Consequently, the total complexity grows quadratically with the security parameter. The techniques we present in this work do not follow the existing framework of phase evaluation.

Our first idea is to embed elements of $\mathbb{Z}_q^N$ or $\mathbb{Z}_q$ into R_q. Given an LWE ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ of the phase $\mu_0 = b + \langle \mathbf{a}, \mathbf{s} \rangle$, we consider the RLWE ciphertext $\mathsf{ct} = (b, a) \in R_q^2$ for $a = \sum_{i \in [N]} \mathbf{a}[i] \cdot X^i$ and the secret $s = \sum_{i \in [N]} \mathbf{s}[i] \cdot X^{-i} \in R$. The ciphertext ct is not a completely valid RLWE ciphertext but its phase $\mu = b + as \pmod q$ contains $\mu_0 = \mu[0]$ in its constant term. We use this idea to accelerate the KS procedure between LWE ciphertexts. For another LWE secret $\mathbf{s}'$, if we take ct and apply the RLWE KS procedure from s to $s' = \sum_{i \in [N]} \mathbf{s}'[i] \cdot X^{-i}$, then the resulting ciphertext $\mathsf{ct}' = (b', a')$ has a phase $\mu' = b' + a's'$ which is approximately equal to μ in R. Therefore, we can extract the LWE ciphertext

$(b'[0], \mathbf{a}')$ where $\mathbf{a}' = (a'[0], \ldots, a'[N-1])$ whose phase is $b'[0] + \langle \mathbf{a}', \mathbf{s}' \rangle = \mu'[0] \approx \mu[0] = \mu_0$, as desired. Its complexity is the same as that of RLWE KS, which grows quasi-linear with the LWE dimension (security parameter).

Secondly, we provide an efficient LWE-to-RLWE conversion. In the above example, the RLWE ciphertext ct cannot be directly used for homomorphic arithmetic because the phase μ contains invalid values in its coefficients except the constant term. We observe that the field trace function $\mathrm{Tr}_{K/\mathbb{Q}}$ of the number field $K = \mathbb{Q}[X]/(X^N + 1)$ zeroizes all the monomials X^i, $0 \neq i \in [N]$ but keeps the constant term. We homomorphically evaluate it to obtain an RLWE ciphertext whose phase is approximately equal to the constant polynomial $N \cdot \mu_0$ whose extra factor N can be easily removed. To minimize the conversion complexity, we describe a recursive algorithm which includes only $\log N$ automorphism evaluations, based on the chain of number fields.

Finally, we present a packing algorithm which takes at most N LWE ciphertexts as the input and returns a single RLWE ciphertext. It can be viewed as a generalization of the LWE-to-RLWE conversion consisting of two phases. Given $n = 2^\ell \leq N$ input ciphertexts of phases $\mu_j \in \mathbb{Z}_q$, the first step is a tree-based algorithm which generates an RLWE ciphertext of phase $\mu \in R_q$ such that $\mu[(N/n) \cdot j] \approx n \cdot \mu_j$ for all $j \in [n]$, i.e., it collects the phases μ_j in an element $\sum_{j \in [j]} \mu_j \cdot Y^j$ of $K_n = \mathbb{Z}[Y]/(Y^n + 1)$. In the following step, we evaluate the field trace Tr_{K/K_n} to annihilate the useless coefficients $\mu[i]$, $(N/n) \nmid i$ and finally return an RLWE ciphertext of phase $\approx N \cdot \sum_{j \in [n]} \mu_j \cdot Y^j$. The whole process requires $(n-1) + \log(N/n)$ homomorphic automorphisms so we achieve an amortized complexity of $< 1 + n^{-1} \cdot \log N$ automorphisms per an LWE ciphertext.

Our conversion algorithms are built on the RLWE KS procedure whose abstract description will be given in the next section. We do not specify the KS method to keep the generality, but we analyze the computational costs of algorithms by counting the number of KS calls or homomorphic automorphisms (see Table 1).

Related Works. In [19, 18], the authors presented a field switching method. Previous works take *ciphertexts* as the input of trace function to reduce the dimension of a base ring dynamically during computation purely for efficiency reasons. Hence there is no need for evaluation key during this step. Meanwhile, we utilize the trace function in our LWE(s)-to-RLWE algorithm in a totally different way for a different purpose. It homomorphically evaluates the field trace on *plaintexts* (phases) to generate a valid RLWE ciphertext over a larger ring R_q from LWE ciphertexts over $\mathbb{Z}_q$.

It has been studied in [29, 12] a method to convert multiple LWE ciphertexts into a single RLWE ciphertext. Given n LWE ciphertexts $\{(b_j, \mathbf{a}_j)\}_{j \in [n]}$, it vertically stacks the i-th entries of all ciphertext in a polynomial for each $i \in [N]$ by $b = \sum_j b_j X^j$ and $a_i = \sum_j \mathbf{a}_j[i] \cdot X^j$, then homomorphically evaluates $b + \sum_i a_i \cdot s_i$ over RLWE system. Different from our packing algorithm, this method has a fixed complexity of N KS operations, independently from the number n of input ciphertexts. This implies that it has to pack $\Omega(N)$ many ciphertexts to achieve the minimal amortized complexity.

Boura et al. [4] combined the previous KS and bootstrapping methods to provide various transformations between HE ciphertexts of different *schemes*. Our work is in an orthogonal direction from [4] since we aim to change the type (algebraic structure) of ciphertexts. In fact, we can improve the performance of [4] by replacing its underlying KS methods by our conversion algorithms.

It is well known that the automorphisms of $\mathrm{Gal}(K/\mathbb{Q})$ can provide advanced functionality to HE system, such as rotation of plaintext slots. We note that our LWE-to-RLWE conversion consists of several iterations each of which evaluates an automorphism and adds the resulting ciphertext to the original input and there have been a few algorithms [22, 7, 8, 9] which are technically similar to our conversion algorithm. However, to the best of our knowledge, this

is the first study which reinterpreted and applied this building block to the KS (conversion) of HE ciphertexts.

2 Background

We denote vectors in bold, e.g. $\mathbf{u}$, and the i-th entry of a vector $\mathbf{u}$ will be denoted by $\mathbf{u}[i]$. For simplicity, we identify $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$ and write the index set $[N] = \{0, 1, \ldots, N-1\}$. For a finite set S, $U(S)$ denotes the uniform distribution on S.

2.1 Cyclotomic Field

Let $\zeta = \exp(\pi i/N)$ for a power-of-two integer N. We denote by $K = \mathbb{Q}(\zeta)$ the $2N$-th cyclotomic field and $R = \mathbb{Z}[\zeta]$ the ring of integers of K. We will identify K (resp. R) with $\mathbb{Q}[X]/(X^N+1)$ (resp. $\mathbb{Z}[X]/(X^N+1)$) with respect to the map $\zeta \mapsto X$. The residue ring of R modulo an integer q will be denoted by $R_q = R/q \cdot R$. For $a, b \in \mathbb{Z}$ (or R, R_q), we informally write $a \approx b \pmod q$ if $a = b + e$ for some small $e \in \mathbb{Z}$ (or R).

An element of K (resp. R, R_q) can be uniquely represented as a polynomial of degree less than N with coefficients in $\mathbb{Q}$ (resp. $\mathbb{Z}$, $\mathbb{Z}_q$). The i-th coefficient of a polynomial $a(X)$ will be denoted by $a[i]$. We use the map $\iota : \mathbf{a} \mapsto \sum_{i \in [N]} \mathbf{a}[i] \cdot X^i$ to identify a polynomial and the vector of its coefficients.

2.2 (Ring) Learning with Errors

Given the dimension N, modulus q and error distribution ψ over $\mathbb{Z}$, the LWE distribution with secret $\mathbf{s} \in \mathbb{Z}^N$ is a distribution over $\mathbb{Z}_q^{N+1}$ which samples $\mathbf{a} \leftarrow U(\mathbb{Z}_q^N)$ and $e \leftarrow \psi$, and returns $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ where $b = \langle \mathbf{a}, \mathbf{s} \rangle + e \pmod q$. The (decisional) LWE assumption of parameter (N, q, χ, ψ) is that it is computationally infeasible to distinguish the LWE distribution of a secret $\mathbf{s} \leftarrow \chi$ from the uniform distribution $U(Z_q^{N+1})$.

The RLWE problem [27] is a variant of LWE based on the ring structure. The key s is chosen from the key distribution χ over R, and an RLWE sample $(b, a) \in R_q^2$ by sampling random a and noise e from $U(R_q)$ and the error distribution ψ over R and computing $b = as + e \pmod q$. The RLWE assumption with parameter (N, q, χ, ψ) is that the RLWE distribution of a secret $s \leftarrow \chi$ and $U(R_q^2)$ are computationally indistinguishable. These assumptions have been widely used to design HE schemes, e.g. [6, 17, 13, 11].

2.3 Gadget Decomposition

Let q be an integer and $\mathbf{g} = (g_0, \ldots, g_{L-1})$ be an integral vector. A *gadget decomposition*, denoted by $\mathbf{g}^{-1} : R_q \to R^L$, is a map which transforms an element $a \in R_q$ into a *short* vector $\mathbf{u} = \mathbf{g}^{-1}(a) \in R^L$ such that $\langle \mathbf{u}, \mathbf{g} \rangle = a \pmod q$. The base (digit) decomposition [6, 5] and prime decomposition [3, 10] are typical examples. This technique has been widely used to control the noise growth during homomorphic operations such as key-switching, which will be described in the next section.

2.4 Basic RLWE Key-Switching

We describe a well known KS method method for RLWE ciphertexts. The goal of KS procedure is to transform a ciphertext into another ciphertext under a different secret key while approximately preserving its phase.

• $\texttt{KSKeyGen}(s \in R, s' \in R)$: Sample $\mathbf{k}_1 \leftarrow U(R_q^L)$ and $\mathbf{e} \leftarrow \chi^L$. Compute $\mathbf{k}_0 = -s' \cdot \mathbf{k}_1 + s \cdot \mathbf{g} + \mathbf{e} \pmod q$ and return the KS key $K = [\mathbf{k}_0 | \mathbf{k}_1] \in R_q^{L \times 2}$.

• $\texttt{KeySwitch}(\mathsf{ct}; K)$: Given an RLWE ciphertext $\mathsf{ct} = (c_0, c_1) \in R_q^2$ and a KS key $K \in R_q^{L\times 2}$, compute and return the ciphertext $\mathsf{ct}' = (c_0, 0) + \mathbf{g}^{-1}(c_1) \cdot K \pmod q$.

A KS key K consists of ℓ RLWE "encryptions" of $s \cdot g_i$ under s', i.e., $K \cdot (1, s') \approx s \cdot \mathbf{g} \pmod q$. For an RLWE ciphertext $\mathsf{ct} \in R_q^2$ and a KS key $K \leftarrow \texttt{KSKeyGen}(s, s')$, the output $\mathsf{ct}' \leftarrow \texttt{KeySwitch}(\mathsf{ct}; K)$ satisfies that

$$\begin{aligned}\langle \mathsf{ct}', (1, s')\rangle &= c_0 + \mathbf{g}^{-1}(c_1) \cdot K \cdot (1, s') \\ &= c_0 + \langle \mathbf{g}^{-1}(c_1), s \cdot \mathbf{g} + \mathbf{e}\rangle = \langle \mathsf{ct}, (1, s)\rangle + e_{ks} \pmod q \end{aligned} \tag{1}$$

for the KS noise $e_{ks} = \langle \mathbf{g}^{-1}(c_1), \mathbf{e}\rangle \in R$.

The complexity of KS is asymptotically quasi-linear with the dimension N, but its concrete performance depends on several factors including the efficiency of gadget decomposition.

2.5 Galois Group and Evaluation of Automorphisms

We recall that $K \geq Q$ is a Galois extension and its Galois group $\mathrm{Gal}(K/\mathbb{Q})$ consists of the automorphisms $\tau_d : \zeta \mapsto \zeta^d$ for $d \in \mathbb{Z}_{2N}^\times$, the invertible residues modulo $2N$. The automorphisms $\tau_d \in \mathrm{Gal}(K/\mathbb{Q})$ gives some distinctive functionalities to HE system. For example, many of RLWE-based schemes such as BGV [6], FV [17] and CKKS [11] utilize the Discrete Fourier Transform (DFT) to encode multiple plaintext values in a single polynomial so that the slots of a ciphertext can be permuted by evaluating an automorphism.

We describe a well-known method to homomorphically evaluate an automorphism τ_d.

• $\texttt{AutoKeyGen}(d \in \mathbb{Z}_{2N}^\times; s \in R)$: Run $A_d \leftarrow \texttt{KSKeyGen}(\tau_d(s), s)$.

• $\texttt{EvalAuto}\left(\mathsf{ct} \in R_q^2, d \in \mathbb{Z}_{2N}^\times; A_d\right)$: Given a ciphertext $\mathsf{ct} = (c_0, c_1) \in R_q^2$, an integer $d \in \mathbb{Z}_{2N}^\times$ and an automorphism key A_d, compute and return the ciphertext $\mathsf{ct}' \leftarrow \texttt{KeySwitch}\left((\tau_d(c_0), \tau_d(c_1)); A_d\right)$.

Security. The homomorphic automorphism algorithm is a simple application of KS, so its security basically relies on the hardness of RLWE for $\texttt{KSKeyGen}$. Moreover, an additional circular security assumption should be made because A_d is a special encryption of $\tau_d(s)$ with secret s.

Correctness. Suppose that $\mathsf{ct} \in R_q^2$ is an RLWE ciphertext such that $\mu = \langle \mathsf{ct}, (1, s)\rangle \pmod q$ and $A_d \leftarrow \texttt{AutoKeyGen}(d; s)$ is an automorphism key. Then the output ciphertext $\mathsf{ct}' \leftarrow \texttt{EvalAuto}(\mathsf{ct}, d; A_d)$ satisfies that

$$\langle \mathsf{ct}', (1, s)\rangle \approx \langle (\tau_d(c_0), \tau_d(c_1)), (1, \tau_d(s))\rangle = \tau_d\left(\langle \mathsf{ct}, (1, s)\rangle\right) = \tau_d(\mu) \pmod q,$$

from the property of $\texttt{KeySwitch}$.

For the rest of this paper, we will simply write $\texttt{EvalAuto}(\mathsf{ct}, d; A_d) = \texttt{EvalAuto}(\mathsf{ct}, d)$ by assuming that an automorphism key $A_d \leftarrow \texttt{AutoKeyGen}(d; s)$ is properly generated and implicitly taken as input of the $\texttt{EvalAuto}$ algorithm. We remark that homomorphic automorphism has almost the same complexity as RLWE KS procedure because computing $\tau_d(c_i)$ is very cheap.

3 Conversion Algorithms

This section presents core ideas and their application to efficient conversion between HE ciphertexts of different secret keys or algebraic structures.

3.1 Functionality of Automorphisms on the Coefficients

We examine how the elements of $\mathrm{Gal}(K/\mathbb{Q})$ act on the coefficients of an input polynomial. Let us define the sets $I_k = \{i \in [N] : 2^k \parallel i\}$[3] for $0 \le k < \log N$ and $I_{\log N} = \{0\}$. Then, the index set $[N]$ can be written as the disjoint union $\bigcup_{0 \le k \le \log N} I_k$. We are interested in how the automorphims $\tau_d(\cdot)$ act on the monomials for $d = 2^\ell + 1$, $1 \le \ell \le \log N$. We note that the map $i \mapsto i \cdot d \pmod N$ is a signed permutation on I_k, i.e., if $i \in I_k$, then $\tau_d(X^i) = \pm X^j$ for some $j \in I_k$. In particular, we see that

$$\begin{aligned} \tau_d(X^i) &= X^i \quad \text{for} \quad i \in \bigcup_{k > \log N - \ell} I_k, \\ \tau_d(X^i) &= -X^i \quad \text{for} \quad i \in I_{\log N - \ell}. \end{aligned} \tag{2}$$

In other words, the map $\mu \mapsto \mu + \tau_d(\mu)$ doubles the coefficients $\mu[i]$ if $2^{\log N - \ell + 1} \mid i$, but zeroizes the coefficients $\mu[i]$ if $2^{\log N - \ell} \parallel i$.

3.2 LWE to LWE

Let $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ be an LWE ciphertext under a secret $\mathbf{s} \in \mathbb{Z}^N$ with phase $\mu_0 = b + \langle \mathbf{a}, \mathbf{s} \rangle \pmod q$. We aim to design an efficient LWE-to-LWE conversion, which replaces the secret of a ciphertext into another secret $\mathbf{s}' \in \mathbb{Z}^N$ while almost preserving the phase μ_0.

Our first idea is to embed $\mathbb{Z}_q^N$ and $\mathbb{Z}_q$ into R_q to utilize the ring structure. We consider the two polynomials

$$\begin{aligned} a := \iota(\mathbf{a}) &= \sum_{i \in [N]} \mathbf{a}[i] \cdot X^i \in R_q, \\ s := \tau_{-1} \circ \iota(\mathbf{s}) &= \sum_{i \in [N]} \mathbf{s}[i] \cdot X^{-i} \in R. \end{aligned}$$

and the polynomial pair $\mathsf{ct} = (b, a) \in R_q^2$. We remark that ct can be viewed as an RLWE ciphertext with secret s satisfying $\langle \mathsf{ct}, (1, s) \rangle[0] = (b + as)[0] = \mu_0$, i.e., its phase $\mu = \langle \mathsf{ct}, (1, s) \rangle \pmod q$ of ct stores $\mu[0] = \mu_0$ in the constant term but all other coefficients, $\mu[i]$ for $0 \ne i \in [N]$, have no valid values.

Though ct is not a canonical RLWE ciphertext, we can still exploit the KS algorithm. If we perform the KS procedure from s to $s' = \tau_{-1} \circ \iota(\mathbf{s}')$, then the output ciphertext also includes a valid value in its constant term from the property of KS. Finally, we can extract an LWE ciphertext with secret $\mathbf{s}'$.

• $\mathtt{LWE\text{-}to\text{-}LWE}((b, \mathbf{a}), K)$: Given an LWE ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ and a KS key $K \in R_q^{L \times 2}$, set the RLWE ciphertext $\mathsf{ct} \leftarrow (b, a) \in R_q^2$ where $a = \iota(\mathbf{a})$. Compute $\mathsf{ct}' = (b', a') \leftarrow \mathtt{KeySwitch}(\mathsf{ct}, K) \in R_q^2$ and let $\mathbf{a}' = \iota^{-1}(a')$. Return the ciphertext $(b'[0], \mathbf{a}') \in \mathbb{Z}_q^{N+1}$.

Correctness. We claim that, if $K \leftarrow \mathtt{KSKeyGen}(\mathbf{s}, \mathbf{s}')$ is a KS key from $\mathbf{s}$ to $\mathbf{s}'$, then $(b'[0], \mathbf{a}')$ is an LWE ciphertext under $\mathbf{s}'$ whose phase is approximately equal to the phase of $(b, \mathbf{a})$ under $\mathbf{s}$. It can be shown by

$$b'[0] + \langle \mathbf{a}', \mathbf{s}' \rangle = (b' + a's')[0] \approx (b + as)[0] = b + \langle \mathbf{a}, \mathbf{s} \rangle \pmod q$$

where the approximate equality is derived from the property of $\mathtt{KeySwitch}$ (1).

[3] $2^k \parallel i$ if and only if $2^k \mid i$ and $2^{k+1} \nmid i$.

Algorithm 1 Homomorphic Evaluation of the Trace Function ($\texttt{EvalTr}_{N/n}$)

Input: ciphertext $\mathsf{ct} = (b, a) \in R_q^2$, a power-of-two integer $n \leq N$.

1: $\mathsf{ct}' \leftarrow \mathsf{ct}$
2: **for** $k = 1$ to $\log(N/n)$ **do**
3: $\quad \mathsf{ct}' \leftarrow \mathsf{ct}' + \texttt{EvalAuto}(\mathsf{ct}'; 2^{\log N - k + 1} + 1)$
4: **return** $\mathsf{ct}' \in R_q^2$

3.3 LWE to RLWE

Our next goal is to design a conversion algorithm from LWE to RLWE. As explained above, if we set an RLWE ciphertext $(b, a = \iota(\mathbf{a})) \in R_q^2$ from an LWE ciphertext $(b, \mathbf{a})$, then its phase has the only valid value in the constant term. Hence, the key question is how to annihilate useless coefficients of μ except the constant term $\mu[0]$ to generate a valid RLWE ciphertext.

We remark that the *field trace* $\mathrm{Tr}_{K/\mathbb{Q}} : K \to \mathbb{Q}$, $a \mapsto \sum_{\tau \in \mathrm{Gal}(K/\mathbb{Q})} \tau(a)$ has the required property, i.e., $\mathrm{Tr}_{K/\mathbb{Q}}(1) = N$ and $\mathrm{Tr}_{K/\mathbb{Q}}(X^i) = 0$ for all $0 \neq i \in [N]$. Therefore, a conversion from LWE into RLWE can be done by evaluating the field trace homomorphically. Moreover, we also propose a recursive algorithm using an algebraic structure of cyclotomic fields to reduce the conversion complexity. To be precise, for the chain of finite fields $K = K_N \geq K_{N/2} \geq \cdots \geq K_1 = \mathbb{Q}$, where K_n denotes the $(2n)$-th cyclotomic field for a power-of-two integer n, the field trace can be expressed as a composition $\mathrm{Tr}_{K/\mathbb{Q}} = \mathrm{Tr}_{K_2/K_1} \circ \cdots \circ \mathrm{Tr}_{K_N/K_{N/2}}$ of $\log N$ field traces and each Galois group $\mathrm{Gal}\left(K_{2^\ell}/K_{2^{\ell-1}}\right)$ has a (unique) nontrivial element $\tau_{2^\ell+1}|_{K_{2^\ell}}$ for $\ell = 1, \ldots, \log N$. Therefore, the evaluation of $\mathrm{Tr}_{K_{2^\ell}/K_{2^{\ell-1}}}$ requires only one homomorphic rotation.

See Alg. 1 for a description of homomorphic trace evaluation Tr_{K_N/K_n} for any power-of-two integer $n \leq N$. We use the parameter $n = 1$ in this section. Finally, we present an LWE-to-RLWE conversion algorithm as follows.

• $\texttt{LWE-to-RLWE}\left((b, \mathbf{a}) \in \mathbb{Z}_q \times \mathbb{Z}_q^N\right)$: Set the RLWE ciphertext $\mathsf{ct} \leftarrow (b, a) \in R_q^2$ where $a = \iota(\mathbf{a})$. Then, run Alg. 1 and return the ciphertext $\mathsf{ct}' \leftarrow \texttt{EvalTr}_{N/1}(\mathsf{ct}) \in R_q^2$.

Correctness. We will prove the correctness of Alg. 1 for an arbitrary $n \leq N$. Let $\mu = \langle \mathsf{ct}, (1, s) \rangle \pmod q$ be the phase of an input ct. We inductively show that the phase $\mu' = \langle \mathsf{ct}', (1, s) \rangle \pmod q$ satisfies

$$\mu' \approx \mathrm{Tr}_{K_N/K_{N/2^k}}(\mu) = 2^k \cdot \sum_{2^k | i \in [N]} \mu[i] \cdot X^i \pmod q \tag{3}$$

at the k iteration. For the base case $k = 0$, the statement is trivially true since $\mu' = \mu$. Now we assume that (3) is true for $k - 1$. In the next k-th iteration, we evaluate the map $\mu' \mapsto \mu' + \tau_d(\mu')$ for $d = 2^{\log N - k + 1} + 1$. We recall from (2) that $\tau_d(X^i) = X^i$ for $2^k \mid i \in [N]$ and $\tau_d(X^i) = -X^i$ for $i \in [N]$ such that $2^{k-1} \parallel i$. From the induction hypothesis,

$$\begin{aligned}
\mu' &\approx 2^{k-1} \cdot \sum_{2^{k-1} | i} \mu[i] \cdot X^i \\
&= 2^{k-1} \cdot \sum_{2^k | i} \mu[i] \cdot X^i + 2^{k-1} \cdot \sum_{2^{k-1} \| i} \mu[i] \cdot X^i \pmod q, \\
\tau_d(\mu') &\approx 2^{k-1} \cdot \sum_{2^k | i} \mu[i] \cdot X^i - 2^{k-1} \cdot \sum_{2^{k-1} \| i} \mu[i] \cdot X^i \pmod q,
\end{aligned}$$

and thereby $\mu' + \tau_d(\mu') \approx 2^k \cdot \sum_{2^k|i} \mu[i] \cdot X^i$. Finally, we obtain

$$\mu' \approx \mathrm{Tr}_{K_N/K_n}(\mu) = (N/n) \cdot \sum_{(N/n)|i \in [N]} \mu[i] \cdot X^i \pmod q$$

after $k = \log(N/n)$ iterations. We remark that the size of noise does not blow up much during the evaluation since $\tau_d(\cdot)$ preserves the size of elements in R.

The correctness of `LWE-to-RLWE` is directly derived from this result with parameter $n = 1$. Given an RLWE encryption $\mathsf{ct} = (b, a)$, we homomorphically compute the field trace $\mathrm{Tr}_{K_N/\mathbb{Q}}$ and the phase $\mu' = \langle \mathsf{ct}', (1, s) \rangle$ of the output ciphertext is approximately equal to $\mathrm{Tr}_{K_N/\mathbb{Q}}(b + as) = N \cdot (b + as)[0] = N \cdot (b + \langle \mathbf{a}, \mathbf{s} \rangle)$, as desired.

3.4 LWEs to RLWE

An LWE ciphertext has a phase in $\mathbb{Z}_q$, which can store only one scalar message, so our LWE-to-RLWE conversion algorithm aims to generate an RLWE ciphertext whose phase μ contains an approximate value of an initial LWE phase in its constant term but zeros in all other coefficients $\mu[i]$, $0 \neq i \in [N]$ for its functionality. However, in general, an RLWE ciphertext can store at most N scalars in the coefficients of its phase. So a natural question is how to efficiently merge multiple LWE ciphertexts into a single RLWE ciphertext?

Suppose that we are given n LWE ciphertexts $\{(b_j, \mathbf{a}_j)\}_{j \in [n]}$ for some $n = 2^\ell \leq N$ and let $\mu_j \in \mathbb{Z}_q$ be the phase of $(b_j, \mathbf{a}_j)$ under the same secret $\mathbf{s} \in \mathbb{Z}^N$. A naive answer for the question above is to run $\mathsf{ct}'_j \leftarrow \texttt{LWE-to-RLWE}((b_j, \mathbf{a}_j)) \in R_q^2$ for all $j \in [n]$ and take their linear combination $\mathsf{ct}' = \sum_{j \in [n]} \mathsf{ct}'_j \cdot Y^j$ for $Y = X^{N/n}$. Then the phase of ct' is approximately equal to $N \cdot \sum_{j \in [n]} \mu_j \cdot Y^j$, which is an element of the ring of integers of K_n. However, this method is not optimal in terms of both complexity and noise growth.

In this section, we present a generalized version of our previous algorithm which takes multiple LWE encryptions as input and returns a single RLWE ciphertext. This conversion consists of two phases: packing and trace evaluation. The first step (Alg. 2) is an FFT-style algorithm which merges $n = 2^\ell$ multiple RLWE ciphertexts into one. The phase μ of an output ciphertext stores the constant terms of input phases in its coefficients $\mu[i]$ for $(N/n) \mid i$. All valid values are now packed into an element of R_n, so in the next step, we use the idea of the previous section to evaluate the field trace Tr_{K_N/K_n} and zeroize useless coefficients.

• $\texttt{LWEs-to-RLWE}\left(\{(b_j, \mathbf{a}_j)\}_{j \in [n]}\right)$: Given $n = 2^\ell$ LWE ciphertexts $(b_j, \mathbf{a}_j) \in \mathbb{Z}_q^{N+1}$, do the following:

1. Set $\mathsf{ct}_j \leftarrow (b_j, a_j) \in R_q^2$ for each $j \in [n]$ where $a_j = \iota(\mathbf{a}_j)$.
2. Run Alg. 2 to get $\mathsf{ct} \leftarrow \texttt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j \in [2^\ell]}\right)$.
3. Compute and return the ciphertext $\mathsf{ct}' \leftarrow \texttt{EvalTr}_{N/n}(\mathsf{ct})$.

The packing algorithm and the subsequent field trace evaluation for $n = 2^\ell$ ciphertexts requires $(n-1)$ and $\log(N/n)$ homomorphic automorphisms, respectively. Hence the total complexity of `LWEs-to-RLWE` is $(n-1) + \log(N/n) < n + \log N$ automorphisms, yielding an amortized complexity less than $(1 + n^{-1} \cdot \log N)$ automorphisms per an input LWE ciphertext. We remark that the asymptotically optimal amortized complexity ($O(1)$ automorphisms) is achieved when $n = \Omega(\log N)$.

Correctness. We first show the correctness of our packing algorithm. For $j \in [2^k]$, let ct_j be input ciphertexts of Alg. 2 such that $\mu_j = \langle \mathsf{ct}_j, (1, s) \rangle[0] \pmod q$. For the output ciphertext

Algorithm 2 Homomorphic Packing of LWE Ciphertexts (`PackLWEs`)

1: **input** ciphertexts $\mathsf{ct}_j = (b_j, a_j) \in R_q^2$ for $j \in [2^\ell]$
2: **if** $\ell = 0$ **then**
3: **return** $\mathsf{ct} \leftarrow \mathsf{ct}_0$
4: **else**
5: $\mathsf{ct}_{even} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_{2j}\}_{j\in[2^{\ell-1}]}\right)$
6: $\mathsf{ct}_{odd} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_{2j+1}\}_{j\in[2^{\ell-1}]}\right)$
7: $\mathsf{ct} \leftarrow \mathsf{ct}_{even} + X^{N/2^\ell} \cdot \mathsf{ct}_{odd} + \mathtt{EvalAuto}\left(\mathsf{ct}_{even} - X^{N/2^\ell} \cdot \mathsf{ct}_{odd}, 2^\ell + 1\right)$
8: **return** ct

$\mathsf{ct} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j\in[2^\ell]}\right)$, we claim that its phase satisfies

$$\mu\left[(N/2^\ell)\cdot j\right] \approx 2^\ell \cdot \mu_j \pmod q \quad \text{for all} \quad j \in [2^\ell], \tag{4}$$

or equivalently, $\mathrm{Tr}_{K_N/K_{2^\ell}}(\mu) \approx N \cdot \sum_{j\in[2^\ell]} \mu_j \cdot Y^j$ for $Y = X^{N/2^\ell}$.

We again use the induction on $0 \le \ell \le \log N$. The base case $\ell = 0$ is trivial since $\mu[0] = \mu_0$. Suppose that our statement is true for some $0 \le \ell - 1 < \log N$. For 2^ℓ input ciphertexts, Alg. 2 first divides them into two groups of size $2^{\ell-1}$ and runs `PackLWEs` twice (in lines 5 and 6). From the induction hypothesis, the output ciphertexts $\mathsf{ct}_{even}, \mathsf{ct}_{odd}$ have phases μ_{even}, μ_{odd} such that

$$\begin{aligned}\mu_{even}\left[(N/2^{\ell-1})\cdot j\right] &\approx 2^{\ell-1}\cdot\mu_{2j} \pmod q,\\ \mu_{odd}\left[(N/2^{\ell-1})\cdot j\right] &\approx 2^{\ell-1}\cdot\mu_{2j+1} \pmod q,\end{aligned}$$

for all $j \in [2^{\ell-1}]$. Then, we compute and return the ciphertext ct whose phase is

$$\begin{aligned}\mu &\approx (\mu_{even} + X^{N/2^\ell}\cdot\mu_{odd}) + \tau_d\left(\mu_{even} - X^{N/2^\ell}\cdot\mu_{odd}, 2^\ell+1\right)\\ &= \mu'_{even} + X^{N/2^\ell}\cdot\mu'_{odd},\end{aligned}$$

for $\mu'_{even} = \mu_{even} + \tau_d(\mu_{even})$ and $\mu'_{odd} = \mu_{odd} + \tau_d(\mu_{odd})$, which satisfy that

$$\begin{aligned}\mu'_{even}\left[(N/2^\ell)\cdot(2j)\right] &\approx 2^\ell\cdot\mu_{2j}, \quad \mu'_{even}\left[(N/2^\ell)\cdot(2j+1)\right] \approx 0 \pmod q,\\ \mu'_{odd}\left[(N/2^\ell)\cdot(2j)\right] &\approx 2^\ell\cdot\mu_{2j+1}, \quad \mu'_{odd}\left[(N/2^\ell)\cdot(2j+1)\right] \approx 0 \pmod q\end{aligned}$$

for all $j \in [2^{\ell-1}]$. Therefore, their linear combination $\mu = \mu'_{even} + X^{N/2^\ell}\cdot\mu'_{odd}$ has coefficients $\mu\left[(N/2^\ell)\cdot j\right] \approx 2^\ell\cdot\mu_j$ for all $j \in [2^\ell]$, as desired.

Now let us discuss about the `LWEs-to-RLWE` algorithm. After running the packing algorithm, the phase μ of $\mathsf{ct} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j\in[n]}\right)$ has $n\cdot\mu_j$ in its coefficients $\mu[i]$ such that $(N/2^\ell) \mid i$. So we homomorphically evaluate the field trace $\mathrm{Tr}_{K_N/K_{2^\ell}}$ on the ciphertext ct to zeroize all other coefficients. It follows from the property of Alg. 1 that the final output $\mathsf{ct}' \leftarrow \mathtt{EvalTr}_{N/2^\ell}(\mathsf{ct})$ satisfies

$$\begin{aligned}\langle \mathsf{ct}', (1,s)\rangle \approx \mathrm{Tr}_{K_N/K_{2^\ell}}(\mu) &= (N/2^\ell)\cdot \sum_{(N/2^\ell)\mid i\in[N]} \mu[i]\cdot X^i\\ &\approx (N/2^\ell)\cdot\sum_{j\in[2^\ell]}(2^\ell\cdot\mu_j)\cdot X^{(N/2^\ell)\cdot j} = N\cdot\sum_{j\in[2^\ell]}\mu_j\cdot Y^j \pmod q\end{aligned}$$

where $Y = X^{N/2^\ell}$, as desired.

Further computation on a packed ciphertext. In a plaintext level, our conversion algorithm computes the function $\mathbb{Z}_q^n \to R_q$, $(\mu_j)_{j\in[n]} \mapsto N \cdot \sum_{j\in[n]} \mu_j \cdot Y^j$, which is not a multiplicative homomorphism. However, it is often required to store multiple values in plaintext slots, instead of coefficients, so that parallel computation is allowed over an encrypted vector of plaintexts.

It has been studied in several researches about HE bootstrapping [19, 23, 9, 8] how to move values from coefficients to slots and vice versa. In the case of BGV, BFV or CKKS, the transformation can be done by evaluating the encoding or decoding functions of the underlying scheme, which are expressed as linear transformations over plaintext vectors. We do not consider it here because this coefficients-to-slots conversion is scheme-dependent. Moreover, its computational cost is cheaper than the main part, so that the total/amortized complexity does not change much even if we add this extra step at the end.

4 Discussion

There are some issues about our conversion algorithms that have not been fully addressed in the previous section. In this section, we present various ideas of modifying or generalizing the algorithms to achieve better efficiency and functionality.

4.1 Removing the Leading Term N

Our conversion algorithms from Section 3.3 and 3.4 introduce the extra term N in the phase of output RLWE ciphertext, so the encrypted plaintext is also scaled by N. We present two available post-processing methods to remove this constant, but we recommend to use the second method if applicable because of its advantages in noise growth (level consumption). Throughout this section, we assume that $\{\mathsf{ct}_j\}_{j\in[n]}$ are n LWE input encryptions of our `LWEs-to-RLWE` and ct' is the output RLWE ciphertext. Their phases are denoted by $\mu_j = \langle \mathsf{ct}_j, (1, \mathbf{s})\rangle \pmod q$ and $\mu = \langle \mathsf{ct}', (1, s)\rangle \pmod q$, respectively.

The first approach is to utilize the functionality of an HE scheme. We describe the idea by providing two specific examples: BFV and CKKS. In the BFV scheme with a plaintext modulus $t > 1$, each phase has the form of $\mu_j = \Delta \cdot m_j + e_j$ for plaintext $m_j \in \mathbb{Z}_t$ and noise $e_j \in \mathbb{Z}$ where $\Delta = \lfloor q/t \rceil$. Hence it satisfies that

$$\mu' = \Delta \cdot (N \cdot \sum_{j\in[n]} m_j \cdot Y^j) + (N \cdot \sum_{j\in[n]} e_j \cdot Y^j + e)$$

for some extra noise $e \in R$. Then we can obtain a valid BFV encryption ct'' of $\sum_{j\in[n]} m_j \cdot Y^j$ by computing $\mathsf{ct}'' \leftarrow \left(N^{-1} \pmod t\right) \cdot \mathsf{ct}'$. However, this method works only when t is co-prime to N, and the final noise $(N^{-1} \pmod t) \cdot (N \cdot \sum_{j\in[n]} e_j \cdot Y^j + e)$ of ct'' is somewhat large. Meanwhile, CKKS is a leveled HE scheme which exploits an approximate encoding method $\mu_j = m_j + e_j$ for plaintext $m_j \in \mathbb{Z}$ and noise $e_j \in \mathbb{Z}$, so that $\mu' = N \cdot \sum_{j\in[n]}(m_j + e_j) + e$ for some noise e. The "rescale" operation of CKKS can remove the term N. To be precise, if the current modulus is $q = q_1 \dots q_\ell$, then we compute $\mathsf{ct}'' \leftarrow \lfloor q_\ell^{-1} \cdot \lfloor q_\ell / N \rceil \cdot \mathsf{ct}' \rceil$, which is a CKKS encryption of $\sum_{j\in[n]} m_i$ with modulus $q_1 \dots q_{\ell-1}$. In both cases, we roughly consume one level of HE system for the post-processing.

We now propose more generic and efficient method by modifying the KS procedure. We recall that a KS key $K \leftarrow \mathtt{KSKeyGen}(s, s') \in R_q^{L\times 2}$ satisfies $K \cdot (1, s') = s \cdot \mathbf{g} + \mathbf{e} \pmod q$ for some noise $\mathbf{e} \in R^L$ whose entries are sampled from the noise distribution χ over R. Our idea is to modify the KS key generation algorithm so that a noise is sampled from the distribution

$N \cdot \chi$ over $N \cdot R$. If N is coprime to q, this new KS key method is secure under the same RLWE assumption since the RLWE distribution with the noise distribution $N \cdot \chi$ is exactly N times of the ordinary RLWE distribution, which is computationally indistinguishable from the uniform distribution over $U(R_q^2)$.

Since the noise during our conversion algorithms is only derived from `EvalAuto`, all extra error terms during the computation (the hidden terms of approximate equalities $\approx$ in our correctness proofs) are elements of $N \cdot R$. As a result, a new variant of our conversion algorithm will return a ciphertext with phase $\mu' = N \cdot \sum_{j \in [n]} \mu_j \cdot Y^j + N \cdot e$ for some noise $e \in R$. Finally, we obtain the ciphertext $\mathsf{ct}'' \leftarrow N^{-1} \cdot \mathsf{ct}' \pmod q$ to remove the scaling factor N. This method is based on the assumption that the ciphertext modulus q is coprime to N, however, it is not a strong requirement because a stronger assumption is usually made in HE schemes to enable an efficient implementation.[4] As a result, this method results in a valid ciphertext that encrypts the desired message but has potentially smaller noise than the first method.

4.2 Special Modulus Technique for KS

Our algorithms are based on the basic KS method described in Section 2.4, but we can replace it by the special modulus [20] variant of KS if necessary. The special modulus technique switches the ciphertext modulus to reduce the noise growth. In this setting, a KS key is generated over a larger modulus $\hat{q} = pq$ for an integer p (called special modulus) with an extra p factor in plaintext part. In this variant of KS operation, the decomposed ciphertext component is multiplied to K over q, and the result is returned back to the original modulus q by multiplying p^{-1} followed by rounding.

We can replace the underlying KS method of our conversion algorithms with this special modulus variant, however, a problem arises if we also apply the technique of previous section to remove the leading term N. Our idea of sampling error from $N \cdot \chi$ is not compatible with the special modulus technique since this KS operation introduces a rounding error which is a multiple of N in general.

We propose another variant of KS, called *lazy modulus-reduction*, to solve the issue. We let a ciphertext stay in the larger modulus $\hat{q}$ during homomorphic evaluation not to have an extra rounding error until the leading factor N is removed. In the case of LWE-to-RLWE conversion, for example, we set an RLWE ciphertext ct from an input LWE ciphertext of phase μ_0 and raise its modulus by $\hat{\mathsf{ct}} = p \cdot \mathsf{ct} \pmod{\hat{q}}$. Then, we perform the trace evaluation in modulo $\hat{q}$ to obtain a ciphertext $\hat{\mathsf{ct}}'$ whose phase is approximately equal to $pN \cdot \mu_0$ modulo $\hat{q}$. Note that the noise is a multiple of N at this point if we applied the technique of scaled noise in the previous section. Therefore, we can remove the factor N and finally recover the original modulus by $\mathsf{ct}' = \lfloor p^{-1} \cdot (N^{-1} \pmod{\hat{q}}) \cdot \hat{\mathsf{ct}}' \rceil \pmod q$.

4.3 General conversion on Modulue LWE

Modulue LWE (MLWE) is a generalized variant of LWE which was proposed to take advantages of both LWE and RLWE [6]. Its hardness has been studied in previous works [26, 1]. We showed three conversion algorithms in Section 3, but our techniques can be generalized for conversion between MLWE ciphertexts.

Let us denote by $K_N = \mathbb{Q}[X]/(X^N + 1)$, $R_N = \mathbb{Z}_q[X]/(X^N + 1)$, $R_{N,q} = R_N/(q \cdot R_N)$ the $(2N)$-th cyclotomic field, the ring of integers, and the residue ring modulo q, respectively. The MLWE problem is parametrized by the dimension n, module rank d, and modulus q. An MLWE sample with secret $\mathbf{s} = (s_i)_{i \in [d]} \in R_{n,q}^d$ is generated by sampling $\mathbf{a}$ uniform at random

[4] The ciphertext modulus q is usually set to be a product of primes 1 modulo $2N$ so that we can utilize an efficient Number Theoretic Transformation (NTT) for polynomial arithmetic in R_q.

from $R_{n,q}^d$ and an error $e \in R_n$ the error distribution over R_n, and returning $(b, \mathbf{a}) \in R_{n,q}^{d+1}$ where $b = -\langle \mathbf{a}, \mathbf{s} \rangle + e \pmod q$. We can define an MLWE-based encryption scheme similar to (R)LWE, and the phase of an MLWE ciphertext $\mathsf{ct} = (b, \mathbf{a})$ is defined as $\mu = \langle \mathsf{ct}, (1, \mathbf{s}) \rangle \in R_{n,q}$. LWE and RLWE are simply MLWE instantiated with $n = 1$ and $d = 1$, respectively.

In this section, we assume that the dimension and rank of MLWE are always powers of two. For $n < N$, we consider R_n as a subring of R_N generated by $Y = X^{N/n}$. We define a bijection $\iota : R_n^{N/n} \to R_N$ and a projection $\pi : R_N \to R_n$ by $\iota : \mathbf{a} \mapsto \sum_{i \in [N/n]} \mathbf{a}[i] \cdot X^i$ and $\pi : a \mapsto \iota^{-1}(a)[0]$, respectively.

Let us present more general variants of our conversion algorithms between MLWE ciphertexts with possibly different parameters (n, d) and (n', d'), from simple to complex cases. We write $N = nd$ and $N' = n'd'$ below.

1. The case $n \leq n'$ and $d \leq d'$ is similar to LWE-to-LWE conversion. First convert the input (n, d)-MLWE ciphertext $(b, \mathbf{a}) \in R_n^{d+1}$ into an RLWE ciphertext $(b, a = \iota(\mathbf{a})) \in R_N^2 \leq R_{N'}^2$, then use the RLWE KS technique over $R_{N'}$ and extract an (n, d')-MLWE ciphertext from resulting ciphertext.
2. If $n \leq n'$ and $d > d'$, split the entries of $\mathbf{a}$ into d/d' groups and merge d' elements of each group into a single element of $R_{N'}$ to obtain an $(N', N/N')$-MLWE ciphertext. Now the secret is a (N/N')-dimensional vector over $R_{N'}$, so run the KS procedure (N/N') times over $R_{N'}$ on individual entries and get an RLWE ciphertext of dimension N'. Finally, extract an (n', d')-MLWE ciphertext.
   As a concrete application, we can accelerate the KS part of TFHE [13], which transforms an LWE ciphertext into another LWE with a smaller dimension.
3. In the above cases $n \leq n'$, more than one ciphertexts can be taken as the input. We can add a pre-processing of packing at most $2^\ell \leq n'/n$ ciphertexts into a single MLWE ciphertext of dimension $2^\ell \cdot n$ using the idea of Alg. 2.
4. Otherwise $n > n'$, a single (n', d')-MLWE cannot store the phase of an (n, d)-MLWE ciphertext. In this case, perform the (n, d)-to-$(n, N'/n)$ transformation first, then extract (n/n')-number of (n', d')-MLWE ciphertexts from it.[5]

5 Implementation

5.1 Experimental Results

We provide a proof-of-concept implementation to show the performance of our conversion algorithms. Our source code is developed in C++ by modifying Microsoft SEAL version 3.4.3 [32]. All experiments are performed on a desktop with an Intel Xeon Silver 4108 CPU @ 1.80 GHz single-threaded with 32 GB memory, compiled with GNU C++ 8.3.0 (-O2).

We set the secret distribution as the uniform distribution over the set of ternary polynomials in R coefficients in $\{0, \pm 1\}$. Each coefficient/entry of (R)LWE error is drawn according to the discrete Gaussian distribution centered at zero with standard deviation $\sigma = 3.2$. See Table 2 for benchmarks of our algorithms.

Our solution supports flexible parameter setting so that it allows the client to take advantages in performance or functionality. We did not specify the type of HE scheme or its plaintext space since the performance of conversion algorithms depend only on the parameters N, $\log q$ and n. The noise of a fresh encryption is of size $O(1)$ and an additional noise from our conversion algorithm is bounded by $\mathrm{poly}(N) = \mathrm{poly}(\log q)$. Hence we can use $\log q - O(\log \log q)$ out of $\log q$ bits of ciphertext modulus for homomorphic functionality.

There are several options to utilize this empty space of $\mathbb{Z}_q$. If we fully use it to store a plaintext, then the ratio between bitsizes of ciphertext modulus and plaintext would be

[5] The last step is similar to extracting N LWE ciphertexts from an RLWE ciphertext of dimension N.

$(N, \log q)$	n	$(2^{12}, 72)$		$(2^{13}, 174)$		$(2^{14}, 389)$	
		Total (Amortized)	Noise	Total (Amortized)	Noise	Total (Amortized)	Noise
LWE to LWE	-	1.85ms	7	7.49ms	8	43.21ms	8
LWE to RLWE	-	18.76ms	18	98.19ms	20	598.49ms	24
LWEs to RLWE	2	17.76ms (8.88ms)	19	104.68ms (52.34ms)	21	584.68ms (292.3ms)	25
	8	25.2ms (3.15ms)	21	127.76ms (15.97ms)	23	798.32ms (99.79ms)	27
	32	66.24ms (2.07ms)	23	315.84ms (9.87ms)	25	1836.48ms (57.4ms)	29

Table 2: Concrete performance of our conversion algorithm measured by total runtime (amortized timing per ciphertext) and noise growth (an upper bound for the bitsize of coefficients of conversion error).

$\log q/(\log q - O(\log\log q)) = 1 + O(\log\log q/\log q)$. It reduces the expansion rate but no more computation is allowed after conversion without bootstrapping. Otherwise, a larger parameter can be chosen to let ciphertexts have more remaining levels after conversion without bootstrapping. In summary, a simple trade-off between the expansion rate and computational capability can be made by the encryptor accordingly.

5.2 Lightweight Communication with Homomorphic Functionality

HE is an attractive solution for secure outsourced computation on the cloud, however, there still remain some problems of performance. Since RLWE encryption schemes are functional but comparably expensive, a client must have a device (encryptor) with enough memory and computational power. Moreover, the ciphertext expansion rate can be reasonably small only when we pack a large number of values in a single ciphertext, i.e., the total communication cost blows up if the client must send a small amount of information frequently. To mitigate this issue, Naehrig et al. [30] gave a blueprint that the client sends data, encrypted by a light-weight symmetric encryption scheme, as well as a homomorphically encrypted secret key of the cryptosystem. Then, the cloud homomorphically evaluates its decryption circuit to get homomorphically encrypted data. In this scenario, the main challenge is to construct a symmetric encryption with low communication cost (expansion rate) and conversion complexity.

The first attempt was made by Gentry et al. [20], which evaluated the AES-128 circuit using the BGV scheme. The main implementation takes about 4 minutes to evaluate an entire AES decryption operation on 120 blocks. Since then, other HE-friendly symmetric encryption schemes such as LowMC [2], FLIP [28] and Rasta [16] have been designed to reduce multiplicative depth and minimize the cost of homomorphic decryption. These block/stream ciphers have advantages in communication cost and encryption timing, but the transformation of ciphertexts brings considerable computation overhead to the cloud side. Prior works have several minutes' latency for the transformation and have to collect a number of ciphertexts to achieve the minimal amortized complexity.

We suggest to use an LWE-based symmetric encryption on the edge device and let the cloud perform LWE-to-RLWE transformation for homomorphic arithmetic (multiplication). Coron et al. [14] compressed the public key of an LWE-based encryption scheme by storing its random part as a seed of a pseudo-random number generator (PRNG). We adapt the

same idea to reduce the size of ciphertexts. To be precise, a symmetric key LWE encryption of secret $\mathbf{s}$ is of the form $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ for a random vector $\mathbf{a} \leftarrow U(\mathbb{Z}_q^N)$ and $b = -\langle \mathbf{a}, \mathbf{s} \rangle + \mu \pmod{q}$ where μ is the phase from the input which is a randomized encoding of plaintext (by Gaussian sampling). Since the second component $\mathbf{a}$ is purely random over $\mathbb{Z}_q^N$, we can modify the encryption algorithm such that it samples a seed se and takes it as the input of a PRNG $f : \{0,1\}^* \to \mathbb{Z}_q^N$ to generate $\mathbf{a} = f(\mathsf{se})$. As a result, a ciphertext can be represented as a pair (b, se), and this variant remains semantically secure in the random oracle model. Moreover, when a client sends multiple LWE ciphertexts to the cloud, the same seed can be reused by computing the random part of the i-th ciphertext by $\mathbf{a}_i = f(\mathsf{se}; i)$. Hence, the communication cost per ciphertext is only $\log q$ bits.

The use of an LWE-based scheme has several advantages compared to prior works based on either block or stream ciphers: (1) Our solution has better conversion latency and amortized timings, and a smooth trade-off between them based on our packing algorithm. As discussed in Section 3.4, it requires to collect only $\Omega(\log N)$ ciphertexts to obtain an optimal amortized complexity. Meanwhile, previous solutions have fixed conversion latency timings (e.g. 4.1, 14.2, 14.5, and 7.7 minutes of AES-128, LowMC, FLIP, and Rasta, respectively) and require to pack hundreds ciphers to achieve the minimal amortized complexity of a few seconds per input ciphertext, compared to several milliseconds of our approach. (2) Our algorithms are more generic in the sense that they preserve the phases of input ciphertexts approximately. Therefore, it is allowed to use any type of HE scheme including BGV, BFV with a non-binary plaintext space and CKKS for approximate arithmetic. On the other hand, the decryptions of block or stream ciphers are usually Boolean circuits, so it is required to use an HE scheme with a binary plaintext space. Therefore, this imposes a limitation that the resulting ciphertext supports only binary operations after conversion to an HE ciphertext. (3) It is more flexible in terms of parameter setting. (4) LWE-based schemes are additively homomorphic, so linear operations can be done over input LWE ciphertexts before converting them into RLWE one.

If the client wants to send only a few bits of information at a time, we can take a smaller ciphertext modulus q and hence speed up the computation. As discussed above, if the client wishes to minimize the communication cost, then the expansion rate $\log q/(\log q - O(\log\log q)) = 1 + O(\log\log q/\log q)$ can be arbitrary close to 1 by fully utilizing the modulus space to store a plaintext. Also in practice, this value is considerably small. As shown in Table 2, the expansion rate can be reduced down to $174/(174-20) \approx 1.13$ or $389/(389-24) \approx 1.07$ when $(N, \log q) = (2^{13}, 174)$ or $(2^{14}, 389)$, respectively.

References

1. Albrecht, M.R., Deo, A.: Large modulus Ring-LWE ≥ Module-LWE. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 267–296. Springer (2017)
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 430–454. Springer (2015)
3. Bajard, J.C., Eynard, J., Hasan, M.A., Zucca, V.: A full RNS variant of FV like somewhat homomorphic encryption schemes. In: International Conference on Selected Areas in Cryptography. pp. 423–442. Springer (2016)
4. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. Cryptology ePrint Archive, Report 2018/758 (2018), `https://eprint.iacr.org/2018/758`
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer (2012)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proc. of ITCS. pp. 309–325. ACM (2012)
7. Carpov, S., Sirdey, R.: Another compression method for homomorphic ciphertexts. In: Proceedings of the 4th ACM International Workshop on Security in Cloud Computing. pp. 44–50. ACM (2016)

8. Chen, H., Han, K.: Homomorphic lower digits removal and improved fhe bootstrapping. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 315–337. Springer (2018)
9. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 360–384. Springer (2018)
10. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full RNS variant of approximate homomorphic encryption. In: Selected Areas in Cryptography – SAC 2018. pp. 347–368. Springer (2019)
11. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437. Springer (2017)
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: Fast fully homomorphic encryption over the torus. Journal of Cryptology pp. 1–58
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security. pp. 3–33. Springer (2016)
14. Coron, J.S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Proc. of EUROCRYPT, LNCS, vol. 7237, pp. 446–464. Springer (2012)
15. Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T.: CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 142–156. ACM (2019)
16. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: a cipher with low ANDdepth and few ANDs per bit. In: Annual International Cryptology Conference. pp. 662–692. Springer (2018)
17. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), `https://eprint.iacr.org/2012/144`
18. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Field switching in BGV-style homomorphic encryption. Journal of Computer Security **21**(5), 663–684 (2013)
19. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Public Key Cryptography–PKC 2012, pp. 1–16. Springer (2012)
20. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Advances in Cryptology–CRYPTO 2012, pp. 850–867. Springer (2012)
21. Gentry, C., et al.: Fully homomorphic encryption using ideal lattices. In: STOC. vol. 9, pp. 169–178 (2009)
22. Halevi, S., Shoup, V.: Algorithms in HElib. In: Advances in Cryptology–CRYPTO 2014. pp. 554–571. Springer (2014)
23. Halevi, S., Shoup, V.: Bootstrapping for HElib. In: Advances in Cryptology–EUROCRYPT 2015, pp. 641–670. Springer (2015)
24. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1209–1222. ACM (2018)
25. Kim, M., Song, Y., Li, B., Micciancio, D.: Semi-parallel logistic regression for GWAS on encrypted data. Cryptology ePrint Archive, Report 2019/294 (2019), `https://eprint.iacr.org/2019/294`
26. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
27. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010. pp. 1–23 (2010)
28. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 311–343. Springer (2016)
29. Miccianco, D., Sorrell, J.: Ring packing and amortized FHEW bootstrapping. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
30. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 113–124. ACM (2011)
31. Riazi, M.S., Laine, K., Pelton, B., Dai, W.: Heax: High-performance architecture for computation on homomorphically encrypted data in the cloud. arXiv preprint arXiv:1909.09731 (2019)
32. Microsoft SEAL (release 3.4). `https://github.com/Microsoft/SEAL` (Oct 2019), microsoft Research, Redmond, WA.
33. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Designs, codes and cryptography **71**(1), 57–81 (2014)