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 progress on homomorphic encryption (HE) has 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 that switch between different (R)LWE-based HE schemes to take advantage 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) [26] 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 [5, 14, 39]. Recent studies demonstrated that this technology shows reasonable performance in real-world tasks such as biomedical analysis and machine learning [31, 32, 19].

Currently, all the best-performing HE schemes, such as BGV [9], BFV [7, 21], TFHE [17] and CKKS [15], 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 the efficient use of the ciphertext packing technique [41]. Each HE scheme has its own pros and cons, but it has been relatively less studied how to take advantage of various HE schemes by converting ciphertexts of different types [6].

Our Contribution. In this paper, we provide a toolkit to transform (R)LWE-based ciphertexts and generate another ciphertext under a new key or of a different structure. Specifically, we present three conversion methods: (1) to perform a new key-switching (KS) operation between LWE ciphertexts; (2) to transform an LWE ciphertext into an RLWE-based ciphertext; and (3) to merge multiple LWE ciphertexts 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 of input ciphertext(s). Moreover, there is no reduction of ciphertext level (modulus) because all building blocks (e.g. homomorphic automorphism) are depth-free. The proposed methods have wide applications in the literature: for example, our LWEs-to-RLWE packing method can improve the performance of [6, 10] which present a hybrid framework between different HE schemes. In addition, our basic algorithms are easily generalizable to design the key-switching between (R)LWE ciphertexts with different dimensions, or more generally, Module LWE [9, 33] based schemes with different parameters.

Finally, we present experimental results to demonstrate that our techniques achieve better asymptotic and concrete performance than previous works [37, 16]. Moreover, we provide a secure outsourcing solution of storage and computation to a cloud with low communication cost. A client encrypts data via an LWE-based symmetric encryption on a lightweight device. On receiving LWE ciphertexts, the public server transforms or packs them into RLWE encryptions to provide better functionality for homomorphic arithmetic. Compared to prior works based on block or stream ciphers [25, 3, 35, 20], our approach has advantages in terms of flexibility, functionality and efficiency.

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 a small error. Similarly, in the case of RLWE over $R = \mathbb{Z}[X]/(X^N + 1)$ and its residue ring $R_q = R/qR$, 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 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 the initial cryptosystem using an HE system if a homomorphically encrypted secret key is given. Furthermore, 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 performing the full decryption which usually includes expensive (non-arithmetic) operations such as bit extraction or rounding [24, 12].

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 an RLWE-based HE system. In fact, the evaluation key consists of N key-switching keys from individual $\mathbf{s}[i]$ to the RLWE secret and the conversion requires N RLWE 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 the 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 $\mathbf{s}'$, we first perform a RLWE KS procedure from s to $s' = \sum_{i \in [N]} \mathbf{s}'[i] \cdot X^{-i}$. Then the phase of the output ciphertext is approximately equal to μ in R, so it is enough to extract an LWE ciphertext from the ciphertext.

Our second algorithm is an efficient conversion from LWE to RLWE. In the example above, the RLWE ciphertext ct cannot be directly used for further homomorphic computation 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 for $0 \neq i \in [N]$ but keeps the constant term (scaled by a factor of N). We homomorphically evaluate the trace function to obtain an RLWE ciphertext whose phase is approximately equal to the constant polynomial $N \cdot \mu_0$ (the extra factor N can be

Type	Previous works [37, 16]		This work	
	Complexity	Storage	Complexity	Storage
LWE-to-LWE	$O(dN^2)$	dN^2	$O(dN \log N)$	$2dN$
LWE-to-RLWE	$O(dN^2)$	$2dN^2$	$O(dN \log^2 N)$	$2dN \log N$
nLWEs-to-RLWE	$O(dN^2 \log N)$	$2dN^2$	$O(dN \log N(n + \log(N/n)))$	$2dN \log N$

Table 1: Computational costs (number of scalar operations) and storage (number of $\mathbb{Z}_q$ elements to store a switching key) of conversion algorithms. N denotes the dimension of (R)LWE, n denotes the number of input LWE ciphertexts to be packed in an RLWE ciphertext, and d denotes the gadget decomposition degree.

easily removed). To minimize the conversion complexity, we present a recursive algorithm that includes only $\log N$ automorphism evaluations, based on the tower of number fields. Furthermore, our algorithm reduces the number of the key-switching keys to $\log N$ compared to N of the previous method.

Finally, we present a packing algorithm that takes at most N LWE ciphertexts as the input and returns a single RLWE ciphertext. Suppose that we are given $n \leq N$ input ciphertexts of phases $\mu_j \in \mathbb{Z}_q$. A naive solution is to perform our LWE-to-RLWE conversion on each LWE ciphertext and adds up the output RLWE ciphertexts into a single ciphertex, which requires $n \log N$ homomorphic automorphisms. We can improve the complexity by performing the FFT-style ciphertext packing algorithm. 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's in an element $\sum_{j \in [n]} \mu_j \cdot Y^j$ 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 $\mu[i]$ for $(N/n) \nmid i$ 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.

Related Works. In [24, 23], the authors presented a method to switch the underlying field of HE ciphertexts. In these works, *ciphertexts* were taken as the input of the trace function to reduce the dimension of the base ring dynamically during computation purely for efficiency reasons. Meanwhile, in our LWE(s)-to-RLWE algorithm, we utilize the trace function in a totally different way for a different purpose. We homomorphically evaluate 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 [16, 37] 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 ciphertexts in a polynomial by $b = \sum_{j \in [n]} b_j \cdot X^j$ and $a_i = \sum_{j \in [n]} \mathbf{a}_j[i] \cdot X^j$ for $i \in [N]$. Then it homomorphically evaluates $b + \sum_i a_i \cdot s_i$ over an RLWE-based HE scheme. Different from our packing algorithm, this method has a fixed complexity of N RLWE KS operations, independently from the number n of input ciphertexts. This implies that it needs to pack $\Omega(N)$ many ciphertexts to achieve the minimal amortized complexity.

Boura et al. [6] presented various transformations between ciphertexts of different RLWE-based HE schemes. Our work is in an orthogonal direction to [6] since we aim to switch the secret key or change the type of ciphertexts (e.g. LWE, RLWE) while preserving their phases (encoded plaintexts). In fact, we can improve the performance of [6] by replacing the underlying KS methods by our conversion algorithms.

In Table 1, we provide the performance of previous works and analyze the computational costs of our algorithms. Our LWE-to-RLWE conversion consists of several iterations in which we evaluate an automorphism and add the resulting ciphertext to the original input. There have been proposed a few algorithms [29, 11, 12, 13] which are technically similar to our conversion algorithm. However, to the best of our knowledge, this is the first study to reinterpret and apply this building block to the KS (conversion) of HE ciphertexts.

Recently, Gentry and Halevi [22] and Brakerski et al. [8] presented a new framework that compresses multiple HE ciphertexts into a single ciphertext with nearly optimal rate $1-o(1)$. Our approach solves an associated but fundamentally different problem. In our application, we could build a lightweight and low-latency communication from the client to the cloud because fresh ciphertexts are high-rate and extremely small. However, they should be packed or converted into an RLWE ciphertext before computation. Meanwhile, previous works [22, 8] aim to compress HE ciphertexts after computation and thereby minimize the communication cost from the cloud to the client.

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 set of representatives 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 is denoted by $R_q = R/qR$. 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 $U(\mathbb{Z}_q^{N+1})$.

The RLWE problem [34] is a variant of LWE which has been widely used to design HE schemes, e.g. [9, 21, 17, 15]. 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.

2.3 Gadget Decomposition

Let q be an integer and $\mathbf{g} = (g_0, \ldots, g_{d-1})$ be an integral vector. A *gadget decomposition* [36], denoted by $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^d$, is a map satisfying $\langle \mathbf{g}^{-1}(a), \mathbf{g}\rangle = a \pmod q$ for all $a \in \mathbb{Z}_q$. We can naturally extend its domain and define $\mathbf{g}^{-1} : R_q \to R^d$ by $a = \sum_{i\in[N]} a_i \cdot X^i \mapsto \sum_{i\in[N]} \mathbf{g}^{-1}(a_i) \cdot X^i$.

The base (digit) decomposition [9, 7] and prime decomposition [5, 14] are typical examples. This technique has been widely used to control the noise growth during homomorphic computation such as key-switching, which will be described in the next section.

2.4 Key Switching

We describe a well known KS 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 $\mathbf{k}_1 \leftarrow U(R_q^d)$ and $\mathbf{e} \leftarrow \chi^d$. Compute $\mathbf{k}_0 = -s' \cdot \mathbf{k}_1 + s \cdot \mathbf{g} + \mathbf{e} \pmod q$ and return the KS key $\mathbf{K} = [\mathbf{k}_0 | \mathbf{k}_1] \in R_q^{d \times 2}$.

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

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

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

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

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 [9], BFV [7, 21] and CKKS [15] 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 $\tau_d : a(X) \to a(X^d)$.

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

• $\texttt{EvalAuto}\left(\mathsf{ct} \in R_q^2, d \in \mathbb{Z}_{2N}^\times; \mathbf{A}_d\right)$: 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 $\mathbf{A}_d$, compute and return the ciphertext $\mathsf{ct}' \leftarrow \texttt{KeySwitch}\left((\tau_d(c_0), \tau_d(c_1)); \mathbf{A}_d\right)$.

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 $\mathbf{A}_d$ 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 $\mathbf{A}_d \leftarrow \texttt{AutoKeyGen}(d; s)$ is an automorphism key. Then the output ciphertext $\mathsf{ct}' \leftarrow \texttt{EvalAuto}(\mathsf{ct}, d; \mathbf{A}_d)$ 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 $\texttt{KeySwitch}$.

In the rest of this paper, we simply write $\texttt{EvalAuto}(\mathsf{ct}, d; \mathbf{A}_d) = \texttt{EvalAuto}(\mathsf{ct}, d)$ by assuming that an automorphism key $\mathbf{A}_d \leftarrow \texttt{AutoKeyGen}(d; s)$ is properly generated and implicitly taken as input of the $\texttt{EvalAuto}$ algorithm. We remark that homomorphic automorphism has almost the same complexity as the KS procedure because the computation of $\tau_d(c_i)$ 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 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 automorphism $\tau_d(\cdot)$ acts 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 the 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 we define 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 valid RLWE ciphertext, we can still apply 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 $\mathbf{s}'$.

• $\texttt{LWE-to-LWE}((b, \mathbf{a}), \mathbf{K})$: Given an LWE ciphertext $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ and a KS key $\mathbf{K} \in R_q^{L\times 2}$, set the RLWE ciphertext $\mathsf{ct} \leftarrow (b, a) \in R_q^2$ where $a = \iota(\mathbf{a})$. Compute $\mathsf{ct}' = (b', a') \leftarrow \texttt{KeySwitch}(\mathsf{ct}, \mathbf{K}) \in R_q^2$ 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 $\mathbf{K} \leftarrow \texttt{KSKeyGen}(s, s')$ is a KS key from s to 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 $\texttt{KeySwitch}$ (see Equation (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 ($\mathtt{EvalTr}_{N/n}$)

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

$\mathsf{ct}' \leftarrow \mathsf{ct}$

for $k = 1$ to $\log(N/n)$ **do**

  $\mathsf{ct}' \leftarrow \mathsf{ct}' + \mathtt{EvalAuto}(\mathsf{ct}'; 2^{\log N - k + 1} + 1)$

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 $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$, then its phase has the valid value only 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. A naive solution is to evaluate each automorphism $\tau(\cdot)$ and add up all the resulting ciphertexts, and therefore it requires N KS operations. We now describe a recursive algorithm which uses an algebraic structure of cyclotomic fields for reducing the conversion complexity. To be precise, for the tower 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.

- $\mathtt{LWE\text{-}to\text{-}RLWE}\left((b, \mathbf{a}) \in \mathbb{Z}_q \times \mathbb{Z}_q^N\right)$: 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 $\mathsf{ct}' \leftarrow \mathtt{EvalTr}_{N/1}(\mathsf{ct}) \in R_q^2$.

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 iteration k. 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. 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 $\mathsf{ct}'_j \leftarrow \mathtt{LWE\text{-}to\text{-}RLWE}((b_j, \mathbf{a}_j)) \in R_q^2$ 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.

- $\mathtt{LWEs\text{-}to\text{-}RLWE}\left(\{(b_j, \mathbf{a}_j)\}_{j \in [n]}\right)$: 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 $\mathsf{ct} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j \in [n]}\right)$.
3. Compute and return the ciphertext $\mathsf{ct}' \leftarrow \mathtt{EvalTr}_{N/n}(\mathsf{ct})$.

The packing algorithm and the subsequent field trace evaluation for $n = 2^\ell$ ciphertexts require $(n-1)$ 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 this conversion algorithm achieves the asymptotically optimal amortized complexity ($O(1)$ automorphisms) when $n = \Omega(\log N)$.

Correctness. We first show the correctness of our packing algorithm. For $j \in [2^\ell]$, 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 $\mathsf{ct} \leftarrow \mathtt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j \in [2^\ell]}\right)$, we claim that its phase satisfies

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

Algorithm 2 Homomorphic Packing of LWE Ciphertexts ($\texttt{PackLWEs}$)

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:     $\mathsf{ct}_{even} \leftarrow \texttt{PackLWEs}\left(\{\mathsf{ct}_{2j}\}_{j\in[2^{\ell-1}]}\right)$
6:     $\mathsf{ct}_{odd} \leftarrow \texttt{PackLWEs}\left(\{\mathsf{ct}_{2j+1}\}_{j\in[2^{\ell-1}]}\right)$
7:     $\mathsf{ct} \leftarrow \left(\mathsf{ct}_{even} + X^{N/2^\ell}\cdot \mathsf{ct}_{odd}\right) + \texttt{EvalAuto}\left(\mathsf{ct}_{even} - X^{N/2^\ell}\cdot \mathsf{ct}_{odd}, 2^\ell + 1\right)$
8:     **return** ct

We again use the induction on $\ell \geq 0$. The base case $\ell = 0$ is trivial since $\mu[0] = \mu_0$. Suppose that our statement is true for some $0 \leq \ell - 1 < \log N$. For 2^ℓ input ciphertexts, Alg. 2 first divides them into two groups of size $2^{\ell-1}$ and runs $\texttt{PackLWEs}$ 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

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

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

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

for $\mu'_{even} = \mu_{even} + \tau_d(\mu_{even})$ and $\mu'_{odd} = \mu_{odd} + \tau_d(\mu_{odd})$, which satisfies 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 $\texttt{LWEs-to-RLWE}$ algorithm. After running the packing algorithm, the phase μ of $\mathsf{ct} \leftarrow \texttt{PackLWEs}\left(\{\mathsf{ct}_j\}_{j\in[n]}\right)$ has $n\cdot\mu_j$ in its coefficients $\mu[i]$ such that $(N/n) \mid i$. So we homomorphically evaluate the field trace Tr_{K_N/K_n} on the ciphertext ct to zeroize all other coefficients. It follows from the property of Alg. 1 that the final output $\mathsf{ct}' \leftarrow \texttt{EvalTr}_{N/n}(\mathsf{ct})$ satisfies

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

where $Y = X^{N/n}$, as desired.

Removing the Leading Term. Our conversion algorithms from LWE(s) to RLWE introduce the auxiliary term N to the phase of output RLWE ciphertext. We present two available methods to remove this constant, but recommend to use the second method if applicable. 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 a post-processing procedure that utilizes 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

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

for some extra noise $e \in R$. As in [24, 4], 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 noise $(N^{-1} \pmod t)) \cdot (N \cdot \sum_{j\in[n]} e_j \cdot Y^j + e)$ of ct'' becomes 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 the HE system.

The other solution is a general and more efficient method which pre-processes the inputs. At the beginning of computation, we multiply the input LWE ciphertexts by the constant $N^{-1} \pmod q$, so that their phases μ_j are also multiplied by the same factor. If we run the same algorithm on the ciphertexts of phase $N^{-1} \cdot \mu_j \pmod q$, then the leading term N is cancelled out and the output would be an RLWE ciphertext whose phase is $\approx N \cdot \sum_{j\in[n]}(N^{-1} \cdot \mu_j) \cdot Y^j = \sum_{j\in[n]} \mu_j \cdot Y^j$ as desired. This method is based on the assumption that the ciphertext modulus q is co-prime to N, but it is not a strong requirement because a stronger assumption is usually made in HE schemes in order to enable an efficient implementation.[4] We recommend to use the second method if applicable because it is depth-free and has better noise growth compared to the first approach.

Further Computation on a Packed Ciphertext. In a plaintext level, our conversion algorithm computes the function $\mathbb{Z}_q^n \to R_q$, $(\mu_j)_{j\in[n]} \mapsto \sum_{j\in[n]} \mu_j \cdot Y^j$, which is not a multiplicative homomorphism. However, it is often required to pack multiple values in plaintext slots, instead of coefficients, so that parallel computation (e.g. element-wise addition or multiplication) is allowed over an encrypted vector of plaintexts.

It has been studied in several researches about HE bootstrapping [24, 30, 13, 12] how to represent 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] 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.

4 Implementation

4.1 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 3.5.1 [40]. All experiments are performed on a desktop with an Intel Core i7-4770K CPU running a single thread at 3.50 GHz, compiled with Clang 9.0.0 (-O3).

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 $\sigma = 3.2$. Table 2 presents timing results and noise growth for our conversion algorithms with various LWE parameters. The selected parameter sets provide at least 128-bit of security level according to the LWE estimator [2] and HE security standard white paper [1]. We adapt an RNS-friendly decomposition method [5] and exploit an efficient NTT in order to optimize the basic polynomial arithmetic. As discussed in Section 3.4, the LWEs-to-RLWE conversion algorithm achieves a better amortized running time as the number of input LWE ciphertexts n increases.

$(N, \log q)$	n	$(2^{12}, 72)$		$(2^{13}, 174)$		$(2^{14}, 389)$	
		Total (Amortized)	Noise	Total (Amortized)	Noise	Total (Amortized)	Noise
LWE to LWE	-	1.03 ms	7	4.81 ms	8	27.1 ms	10
LWE to RLWE	-	11.2 ms	18	57.7 ms	21	361 ms	23
LWEs to RLWE	2	11.4 ms (5.70 ms)	18	58.7 ms (29.4 ms)	21	364 ms (182 ms)	23
	8	16.8 ms (2.10 ms)	20	83.2 ms (10.4 ms)	22	492 ms (61.5 ms)	24
	32	45.0 ms (1.41 ms)	20	209 ms (6.53 ms)	22	1168 ms (36.5 ms)	24

Table 2: Concrete performance of our conversion algorithms measured by total running time (amortized timing per ciphertext) and noise growth (an upper bound on the bit size of coefficients of conversion errors).

Our solution supports flexible parameter setting so that it allows the client to take advantage in performance or functionality. We did not specify the type of HE scheme or its plaintext space since the performance of our conversion algorithms depend only on the parameters N, $\log q$ and n. The noise of a fresh encryption is of size $O(1)$ and the 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 a ciphertext modulus for further homomorphic computation. We refer the reader to Appendix A which provides noise analysis of our conversion algorithms.

There are several options to utilize the space $\mathbb{Z}_q$ for storing the phase of an LWE ciphertext. If we use the full space to store a plaintext, then the expansion ratio between bitsizes of ciphertext modulus and plaintext would be $\log q/(\log q - O(\log \log q)) = 1 + O(\log \log q/\log q)$. It achieves nearly optimal rate but no more homomorphic 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.

4.2 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. So, the total communication cost blows up if the client must send a small amount of information frequently. To mitigate this issue, Naehrig et al. [38] 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. [25], 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 [3], FLIP [35], and Rasta [20] 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 the LWE-to-RLWE conversion for homomorphic computation (multiplication). Coron et al. [18] 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 $f : \{0,1\}^* \rightarrow \mathbb{Z}_q^N$ 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 of messages 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 decryption of block or stream cipher is usually Boolean circuits, so it is required to use an HE scheme with a binary plaintext space to homomorphically evaluate its decryption circuit. Therefore, this imposes a limitation that the resulting HE ciphertext supports only binary operations after conversion. (3) Parameter setting is more flexible and the unit ciphertext size is much smaller compared to the block/stream ciphers. (4) LWE-based schemes are additively homomorphic, so linear operations can be done over input LWE ciphertexts before converting them into RLWE ciphertexts.

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 $1 + o(1)$ can be almost optimal by utilizing the whole space to store a plaintext. For example, as shown in Table 2, the expansion rate can be reduced down to $174/(174 - 21) \approx 1.14$ or $389/(389 - 23) \approx 1.06$ when $(N, \log q) = (2^{13}, 174)$ or $(2^{14}, 389)$, respectively.

References

1. Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption security standard. Tech. rep., HomomorphicEncryption.org, Toronto, Canada (November 2018)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
3. 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)
4. Alperin-Sheriff, J., Peikert, C.: Practical bootstrapping in quasilinear time. In: Annual Cryptology Conference. pp. 1–20. Springer (2013)
5. 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)
6. 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`
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer (2012)
8. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: Theory of Cryptography Conference. pp. 407–437. Springer (2019)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proc. of ITCS. pp. 309–325. ACM (2012)
10. Carpov, S., Gama, N., Georgieva, M., Troncoso-Pastoriza, J.R.: Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption. (2019), `https://eprint.iacr.org/2019/101`
11. 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)
12. 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)
13. 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)
14. 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)
15. 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)
16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology pp. 1–58

17. 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)
18. 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)
19. 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)
20. 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)
21. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), `https://eprint.iacr.org/2012/144`
22. Gentry, C., Halevi, S.: Compressible fhe with applications to pir. In: Theory of Cryptography Conference. pp. 438–464. Springer (2019)
23. 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)
24. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Public Key Cryptography–PKC 2012, pp. 1–16. Springer (2012)
25. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Advances in Cryptology–CRYPTO 2012, pp. 850–867. Springer (2012)
26. Gentry, C., et al.: Fully homomorphic encryption using ideal lattices. In: STOC. vol. 9, pp. 169–178 (2009)
27. Halevi, S., Polyakov, Y., Shoup, V.: An improved RNS variant of the BFV homomorphic encryption scheme. In: Cryptographers' Track at the RSA Conference. pp. 83–105. Springer (2019)
28. Halevi, S., Shoup, V.: Design and implementation of a homomorphic-encryption library. IBM Research (Manuscript) (2013)
29. Halevi, S., Shoup, V.: Algorithms in HElib. In: Advances in Cryptology–CRYPTO 2014. pp. 554–571. Springer (2014)
30. Halevi, S., Shoup, V.: Bootstrapping for HElib. In: Advances in Cryptology–EUROCRYPT 2015, pp. 641–670. Springer (2015)
31. 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)
32. 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`
33. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
34. 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)
35. 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)
36. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 700–718. Springer (2012)
37. 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)
38. 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)
39. 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)
40. Microsoft SEAL (release 3.5). `https://github.com/Microsoft/SEAL` (Apr 2020), microsoft Research, Redmond, WA.
41. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Designs, codes and cryptography **71**(1), 57–81 (2014)

A Noise analysis

The key switching procedure in Section 2.4 is the only source of an extra noise during our conversion algorithms. Recall that the key-switching procedure $\texttt{KeySwitch}(\mathsf{ct} = (c_0, c_1); \mathbf{K})$ introduces the noise $e_{ks} = \langle \mathbf{g}^{-1}(c_1), \mathbf{e} \rangle$ where $\mathbf{e}$ is the noise of the KS key $\mathbf{K}$. We make a heuristic assumption (which has been widely used in HE researches, e.g. [25, 28, 16]) such that a KS noise behaves as if its coefficients are sampled independently from a Gaussian distribution with a fixed variance, which will be denoted by V_{ks}. For a random variable $a = \sum_{i \in [N]} a_i \cdot X^i$ over R, we denote by $\mathsf{Var}(a)$ the maximum among the variances of its coefficients $\{\mathsf{Var}(a_i) : 0 \le i < N\}$.

In practice, we need to specify the gadget decomposition method to compute V_{ks}. For example, suppose that the ciphertext modulus $q = \prod_{0 \le i < d} q_i$ is a product of relatively co-prime integers and the gadget decomposition is defined as $R_q \to \prod_{i \in [d]} R_{q_i}, a \mapsto \mathbf{g}^{-1}(a) = (a \pmod{q_i})_{0 \le i < d}$.[5] Then, the coefficients of $e_{ks} = \langle \mathbf{g}^{-1}(c_1), \mathbf{e} \rangle$ have the common variance $V_{ks} \le \frac{1}{12} N \sigma^2 \cdot \sum_{i \in [d]} q_i^2$ where σ^2 is the variance of RLWE error distribution.

A.1 LWE to LWE

Technically, our LWE-to-LWE conversion includes only one KS procedure between RLWE ciphertexts and then we extract an LWE ciphertext from the output ciphertext. As shown in the correctness proof in Section 3.2, the additional noise in the final LWE ciphertext is equal to the constant term of the KS noise, whose variance is V_{ks}.

A.2 LWE to RLWE

We will analyze the noise of homomorphic trace evaluation ($\texttt{EvalTr}_{N/n}$ in Alg. 1) since the LWE-to-RLWE conversion is a special case where $n = 1$.

We showed that if $\mu = b + as \pmod{q}$ is the phase of the input ciphertext ct, then the phase of ct' is $\mathrm{Tr}_{K_N/K_{N/2^k}}(\mu) + e_k$ for some error e_k after k iterations. We will estimate the variance of e_k using the induction on k.

If $k = 0$, we have $e_0 = 0$. For $1 \le k \le \log(N/n)$, we denote by $e'_k \in R$ the additional noise from the homomorphic automorphism at the k-th iteration. Then, we get $e_k = e_{k-1} + \tau_d(e_{k-1}) + e'_k$ for $d = 2^{\log N - k + 1} + 1$ and its variance is bounded by $\mathsf{Var}(e_k) \le 4 \cdot \mathsf{Var}(e_{k-1}) + V_{ks}$. Therefore, the noise of the output ciphertext from Alg. 1 is bounded by $\mathsf{Var}(e_k) \le (1 + 4 + \cdots + 4^{k-1}) \cdot V_{ks} \le \frac{1}{3}\left((N/n)^2 - 1\right) \cdot V_{ks}$.

Our LWE-to-RLWE algorithm is the case of $n = 1$ (or equivalently $k = \log N$) which returns a ciphertext whose phase is $\mathrm{Tr}_{K/\mathbb{Q}}(\mu) + e_{\log N}$ for some $e_{\log N}$ such that $\mathsf{Var}(e_{\log N}) \le \frac{1}{3}(N^2 - 1) \cdot V_{ks}$.

A.3 LWEs to RLWE

We first analyze the noise growth of Alg. 2. We showed that if $\{\mathsf{ct}_j = (b_j, a_j)\}_{j \in [2^\ell]}$ are the input RLWE ciphertexts such that $\mu_j = (b_j + a_j \cdot s)[0]$, then the phase μ of output ciphertext satisfies that $\mu[(N/2^\ell) \cdot j] = 2^\ell \cdot \mu_j + e_{\ell,j} \pmod{q}$ for all $j \in [2^\ell]$ and for some $e_{\ell,j} \in \mathbb{Z}$. If $\ell = 0$, then there is no extra noise from the packing algorithm. In the case of $\ell > 0$, we divide the input ciphertexts into two groups and run the packing algorithm on each subgroup separately. Suppose that the phases of ct_{even} and ct_{odd} satisfy

$$\mu_{even}[(N/2^{\ell-1}) \cdot j] = 2^{\ell-1} \cdot \mu_{2j} + e_{\ell-1,2j} \pmod{q},$$
$$\mu_{odd}[(N/2^{\ell-1}) \cdot j] = 2^{\ell-1} \cdot \mu_{2j+1} + e_{\ell-1,2j+1} \pmod{q}$$

[5] This method is called the prime decomposition which is widely used in the construction of RNS-friendly HE schemes such as [5, 27, 32, 40].

for some errors $e_{\ell-1,2j}, e_{\ell-1,2j+1} \in \mathbb{Z}$. Let $e'_\ell(X)$ be the additional noise from the evaluation of automorphism $\texttt{EvalAuto}(\mathsf{ct}_{even} - X^{N/2^\ell} \cdot \mathsf{ct}_{odd}, 2^\ell + 1)$ and $e'_{\ell,j}$ the $(N/2^\ell)) \cdot j$-th coefficient of $e'_\ell(X)$ for $j \in [2^\ell]$. Then, we get a relation $e_{\ell,j} = 2e_{\ell-1,j} + e'_{\ell,j}$ between errors from the equation $\mu = \mu'_{even} + X^{N/2^\ell} \cdot \mu'_{odd} + e'_\ell(X)$ for all $j \in [2^\ell]$. Since $e'_{\ell,j}$ has a fixed variance V_{ks} for all ℓ and j, we have $\mathsf{Var}(e_{\ell,j}) = 4 \cdot \mathsf{Var}(e_{\ell-1,j}) + V_{ks}$. Finally, we use the induction on ℓ and show that $\mathsf{Var}(e_{\ell,j}) = (1 + 4 + \cdots + 4^{\ell-1}) \cdot V_{ks} = \frac{1}{3}(n^2 - 1) \cdot V_{ks}$ when $n = 2^\ell$.

In our LWEs-to-RLWE conversion, the packing algorithm is followed by the trace evaluation $\texttt{EvalTr}_{N/n}$ whose noise growth is analyzed above. Hence, the phase of the output ciphertext from the LWEs-to-RLWE conversion satisfies that $\mu = (N/n) \cdot \left(\sum_{j \in [n]} (n\mu_j + e_{\ell,j}) \cdot X^{(N/n) \cdot j} \right) + e_k(X) \pmod q$ where e_k denotes the noise from trace evaluation and $k = \log(N/n)$. Therefore, the variance of total noise $(N/n) \cdot \left(\sum_{j \in [n]} e_{\ell,j} \cdot X^{(N/n) \cdot j} \right) + e_k(X)$ is bounded by $(N/n)^2 \cdot \mathsf{Var}(e_{\ell,j}) + \mathsf{Var}(e_k) \leq \frac{1}{3}(N^2 - 1) \cdot V_{ks}$.