FFT-less TFHE: Simpler, Faster and Scale-invariant

Zhen Gu, Wen-jie Lu, and Cheng Hong

Alibaba Group

Abstract. Fully homomorphic encryption (FHE) has been one of the most promising cryptographic tools for secure two-party computation and secure outsourcing computation in recent years. However, the complex bootstrapping procedure in FHE schemes is the main bottleneck of it practical usage, and the `TFHE` scheme is the state-of-the-art for efficient bootstrapping. To further improve the efficiency of bootstrapping in `TFHE`, the number of fast Fourier transforms (FFT) should be reduced since bootstrapping in `TFHE` is mainly composed of vast FFTs. In this paper, we focus on a novel method of decomposing-in-Fourier (DIF) to reduce the number of `FFT`s in bootstrapping of `TFHE`, from $2(\ell+1)n$ to $4n$. As a result, our method would reduce the number of `FFT`s required by each external product in bootstrapping to a constant number rather than varying with decomposing parameters, which leads to a scale-invariant bootstrapping structure.

Keywords: `TFHE`· `FFT`· Bootstrapping

1 Introduction

With the growing interest in data privacy, fully homomorphic encryption (FHE) has been developed rapidly over the past decade. After the first bootstrappable construction by Gentry [7], various schemes and implementations have been designed for simpler and more efficient bootstrapping or more composable primitives. Among these, `BGV`, `B/FV`, `CKKS`, `GSW`, `FHEW` and `TFHE` are the most famous FHE schemes[1][6][2][8][5][3]. In the sense of bootstrapping, `TFHE` is one of the fastest scheme with only milliseconds required to bootstrap a ciphertext. To further improve the performance of bootstrapping, there have been some researches of reducing the amortized complexity for bootstrapping [9][11][4]. However, few significant attempts have been made for reducing the computational complexity of bootstrapping. In this paper, we present `FFT`-less `TFHE`, which is lower in computational complexity than the original `TFHE`.

We briefly describe the roadmap here and details are discussed in the following sections. Step 1, we revisit the `ExtProd` in `TFHE` and analyze the operations from the computational perspective; Step 2, we discuss the relationship between `Decomp` and the number of required `FFT`s in a single `ExtProd`; Step 3, to unleash the relationship between `Decomp` and the number of `FFT`s, we present the constraints of making a valid `Decomp`; Step 4, we present the properties of `FFT` and

show how to perform Decomp in FFT domain; Step 5, we show that the FFT-less TFHE is possible with worst-case and average-case noise analysis. As a closure of the paper, we have some interesting discussions about possible paths to the FFT-free TFHE for the further complexity reduction.

2 Preliminaries

2.1 Learning with Errors and Its Ring Variant

Learning with errors (LWE) is a widely used lattice-based cryptographic scheme that is convinced to resist attacks with quantum computing[12]. LWE and its ring variant (RLWE) have been developed to construct most of nowadays FHE schemes, like BGV, B/FV, CKKS, GSW, FHEW and TFHE [13][1][6][2][8][5][3]. In this paper, $\mathcal{R}$ denotes $\mathbb{Z}[\mathtt{X}]/(\mathtt{X}^N+1)$, and $\mathcal{R}_q$, $\mathcal{R}_2$ denote $\mathcal{R}/q\mathcal{R}$, $\mathcal{R}/2\mathcal{R}$, respectively.

In LWE, a plaintext $\mu \in \mathbb{Z}_q$ is encrypted with a ciphertext $(\vec{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ using a secret key $\vec{\mathfrak{s}} \in \mathbb{Z}_q^n$ with elements in $\{0,1\}$, such that $b = \vec{a} \cdot \vec{\mathfrak{s}} + e$, where $\vec{a}$ is uniformly sampled from $\mathbb{Z}_q^n$ and $e \in \mathbb{Z}_q$ is a noise with discrete gaussian distribution. We can rewrite the LWE ciphertext as $\mathtt{LWE}_{\vec{\mathfrak{s}}}(\mu|\vec{a}, e) = (\vec{a},\ \vec{a} \cdot \vec{\mathfrak{s}} + e)$.

In RLWE, a plaintext $\mu \in \mathcal{R}_q$ is encrypted with a ciphertext $(a, b) \in \mathcal{R}_q \times \mathcal{R}_q$ using a secret key $\mathfrak{s} \in \mathcal{R}_q$ with coefficients in $\{0,1\}$, such that $b = a\mathfrak{s} + e$, where a is uniformly sampled from $\mathcal{R}_q$ and $e \in \mathcal{R}_q$ is a noise with discrete gaussian distribution. We can rewrite the RLWE ciphertext as $\mathtt{RLWE}_{\mathfrak{s}}(\mu|a, e) = (a,\ a\mathfrak{s} + e)$. Moreover, we define the phase function for RLWE ciphertexts as follows:

$$\varphi_{\mathfrak{s}}(\mathtt{RLWE}_{\mathfrak{s}}(\mu|a, e)) = (a\mathfrak{s} + \mu + e) - a\mathfrak{s} = \mu + e \tag{1}$$

In GSW-like schemes such as FHEW and TFHE, RGSW ciphertexts are introduced for homomorphic multiplications. In RGSW, two parameters $\ell, \mathfrak{B}$ are defined for the balance between efficiency and noise propagation. The gadget vector $\vec{g}$ is $\left(Q/\mathfrak{B}^1 \cdots Q/\mathfrak{B}^\ell\right)$, and the gadget matrix is

$$\mathcal{G}^{\ell,\mathfrak{B}} = \begin{bmatrix} \mathbf{Diagonal}(\vec{g}) & 0 \\ 0 & \mathbf{Diagonal}(\vec{g}) \end{bmatrix}. \tag{2}$$

A RGSW ciphertext encrypting a plaintext $\mu \in \mathbb{Z}$ with the secret key $\mathfrak{s} \in \mathcal{R}_q$ is then defined as

$$\mathtt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}(\mu|A, E) = \mu\mathcal{G}^{\ell,\mathfrak{B}} + \mathfrak{Z} = \mu\mathcal{G}^{\ell,\mathfrak{B}} + \begin{bmatrix} A & A\mathfrak{s} + E \end{bmatrix} \tag{3}$$

where $\mathfrak{Z}$ is a matrix of 2ℓ rows and each row contains a RLWE ciphertext encrypting 0 with $\mathfrak{s}$.

2.2 External Product and Fast Fourier Transforms in TFHE

In TFHE, an external product performs the homomorphic multiplication of a RLWE ciphertext and a RGSW ciphertext. For RGSW parameters $\ell, \mathfrak{B}$, the decomposition

Algorithm 1 `ExtProd` in `TFHE`

Require: $(a,b) = \mathtt{RLWE}_{\mathfrak{s}}(\mu_0|a,e), \mathtt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}(\mu_1|A,E)$
Ensure: $\mathbf{ct}_{\mathtt{ExtProd}} = \mathtt{RLWE}_{\mathfrak{s}}(\mu_0\mu_1|a_{\mathtt{ExtProd}}, e_{\mathtt{ExtProd}})$
$\mathbf{ct}_{\mathtt{Decomp}} = \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))$
$\mathbf{ct}_{\mathtt{DecompFFT}}[i] = \mathtt{FFT}(\mathbf{ct}_{\mathtt{Decomp}}[i]) \quad i = 0, \cdots, 2\ell$
$\mathbf{ct}_{\mathtt{ExtProdFFT}} = \mathbf{ct}_{\mathtt{DecompFFT}} \cdot \mathtt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}(\mu_1|A,E)$
$\mathbf{ct}_{\mathtt{ExtProd}}[i] = \mathtt{IFFT}(\mathbf{ct}_{\mathtt{ExtProdFFT}}[i]) \quad i = 0, 1$

of a `RLWE` ciphertext $\mathtt{RLWE}_{\mathfrak{s}}(\mu|a,e) = (a,b)$ is defined as follows

$$\mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b)) = \left[\mathtt{Dig}(a,0) \cdots \mathtt{Dig}(a,\ell-1) \quad \mathtt{Dig}(b,0) \cdots \mathtt{Dig}(b,\ell-1)\right], \tag{4}$$

where $\mathtt{Dig}(\bullet, i) = \left[\left\lfloor \frac{\bullet}{Q/\mathfrak{B}^{i+1}} \right\rceil\right]_{\mathfrak{B}}$. Meanwhile, $\varepsilon^{\ell,\mathfrak{B}}(u)$, the decomposition error of a polynomial u and the maximal decomposition error $\varepsilon^{\ell,\mathfrak{B}}$ are defined as follows, respectively,

$$\varepsilon^{\ell,\mathfrak{B}}(u) = \sum_{i=0}^{\ell-1} \mathtt{Dig}(u,i)\frac{Q}{\mathfrak{B}^{i+1}} - u = \sum_{i=0}^{\ell-1} \mathtt{Dig}(u,i)\vec{g}_i - u \tag{5}$$

$$\varepsilon^{\ell,\mathfrak{B}} = \max_{u \in \mathcal{R}_Q} \left\{ \left\| \varepsilon^{\ell,\mathfrak{B}}(u) \right\|_\infty \right\} = Q/\mathfrak{B}^{\ell} \tag{6}$$

With the decomposition of `RLWE` ciphertexts, the external product of a `RLWE` ciphertext $\mathbf{ct}_0 = \mathtt{RLWE}_{\mathfrak{s}}(\mu_0|a,e)$ and a `RGSW` ciphertext $\mathbf{ct}_1 = \mathtt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}(\mu_1|A,E)$ is defined as

$$\mathtt{ExtProd}(\mathbf{ct}_0, \mathbf{ct}_1) = \mathtt{Decomp}^{\ell,\mathfrak{B}}(\mathbf{ct}_0) \cdot \mathbf{ct}_1. \tag{7}$$

Actually, the external product of $\mathbf{ct}_0$ and $\mathbf{ct}_1$ is a $\mathtt{RLWE}$ ciphertext encrypting $\mu_0\mu_1$ with $\mathfrak{s}$, because

$$
\begin{aligned}
&\mathtt{ExtProd}(\mathbf{ct}_0, \mathbf{ct}_1)\\
=&\mathtt{Decomp}^{\ell,\mathfrak{B}}(\mathbf{ct}_0)\cdot\mathbf{ct}_1\\
=&\mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,\mu_1\mathcal{G}^{\ell,\mathfrak{B}} + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\left[A\ A\mathfrak{s}+E\right]\\
=&\mu_1(\sum_{i=0}^{\ell-1}\mathtt{Dig}(a,i)\vec{g}_i, \sum_{i=0}^{\ell-1}\mathtt{Dig}(b,i)\vec{g}_i)\\
&\quad + \left[\mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A \quad \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A\mathfrak{s} + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,E\right]\\
=&(\mu_1 a + \mu_1\varepsilon^{\ell,\mathfrak{B}}(a) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A, \quad \mu_1 b + \mu_1\varepsilon^{\ell,\mathfrak{B}}(b)\\
&\quad + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A\mathfrak{s} + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,E)\\
&\varphi_{\mathfrak{s}}(\mathtt{ExtProd}(\mathbf{ct}_0, \mathbf{ct}_1))\\
=&\mu_1 b + \mu_1\varepsilon^{\ell,\mathfrak{B}}(b) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A\mathfrak{s} + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,E\\
&\quad - \left(\mu_1 a + \mu_1\varepsilon^{\ell,\mathfrak{B}}(a) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A\right)\mathfrak{s}\\
=&\mu_1\mu_0 + \mu_1 e + \mu_1(\varepsilon^{\ell,\mathfrak{B}}(b) - \varepsilon^{\ell,\mathfrak{B}}(a)\mathfrak{s}) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,E
\end{aligned}
$$

Therefore, the external product of $\mathtt{RLWE}_{\mathfrak{s}}(\mu_0|a,e)$ and $\mathtt{RGSW}^{\ell,\mathfrak{B}}_{\mathfrak{s}}(\mu_1|A,E)$ is $\mathtt{RLWE}_{\mathfrak{s}}(\mu_0\mu_1|a_{\mathtt{ExtProd}}, e_{\mathtt{ExtProd}})$, where

$$
\begin{aligned}
a_{\mathtt{ExtProd}} &= \mu_1 a + \mu_1\varepsilon^{\ell,\mathfrak{B}}(a) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,A\\
e_{\mathtt{ExtProd}} &= \mu_1 e + \mu_1(\varepsilon^{\ell,\mathfrak{B}}(b) - \varepsilon^{\ell,\mathfrak{B}}(a)\mathfrak{s}) + \mathtt{Decomp}^{\ell,\mathfrak{B}}((a,b))\,E.
\end{aligned}
$$

For noise analysis, a straightforward conclusion about the external product is

$$\|e_{\mathtt{ExtProd}}\|_\infty \le \mu_1\|e\|_\infty + \mu_1(1+N)\varepsilon^{\ell,\mathfrak{B}} + 2\ell\mathfrak{B}\|E\|_\infty$$

$\mathtt{ExtProd}$ is a basic building block in the bootstrapping of $\mathtt{TFHE}$. Thus we look into the computational structure of $\mathtt{ExtProd}$ to see how the number of $\mathtt{FFT}$s account to the total computational complexity.

Computational Perspective on $\mathtt{ExtProd}$ We first take a brief glance of the computational structure of $\mathtt{ExtProd}$, and later we count the number of $\mathtt{FFT}$s required in a single $\mathtt{ExtProd}$.

In an $\mathtt{ExtProd}$, 2 polynomials in a $\mathtt{RLWE}$ ciphertext are first decomposed into ℓ polynomials respectively. Then the vector of these 2ℓ polynomials is multiplied by the $\mathtt{RGSW}$ ciphertext, which is matrix of $2\ell\times 2$ polynomials. Consequently, in a single $\mathtt{ExtProd}$, 4ℓ polynomial multiplications are involved. Fast Fourier transforms are applied for the reduction in computational complexity of polynomial multiplications. Usually, the polynomials are converted to its evaluation form via $\mathtt{FFT}$, and converted back to its coefficient form via $\mathtt{IFFT}$. In their evaluation forms, polynomials can be multiplied with point-wise-multiplication ($\mathtt{PWM}$). Therefore, considering the transformations from and to evaluation forms, a single external product can be evaluated via 2ℓ $\mathtt{FFT}$s, 2 $\mathtt{IFFT}$s and 4ℓ $\mathtt{PWM}$s of polynomials:

1. Decompose the two polynomials of the RLWE ciphertext in their coefficient forms.
2. Apply 2ℓ FFTs to the decomposed polynomials and convert them into evaluation forms.
3. Perform 4ℓ PWMs of the 2ℓ decomposed polynomials and the 4ℓ polynomials in the RGSW ciphertext.
4. Sum the products up to the two polynomials in the result RLWE ciphertext.
5. Apply 2 IFFTs to the polynomials in the result RLWE ciphertext and convert them into coefficient form.

In the above procedure, polynomials in the RGSW ciphertext are not transformed into evaluation forms because in the case of bootstrapping, the RGSW ciphertexts are pre-computed bootstrapping key components, and their conversion to evaluation forms is not counted in the computational cost of ExtProds.

Most theoretical papers about homomorphic encryption schemes take fast Fourier transforms and number-theoretic transforms (NTT) as a speeding up technique for polynomial multiplications and would not discuss the details of the transforms. However, in this paper, we take the advantage of the evaluation form to reduce the number of FFTs and we present the details of FFT for further discussions.

Fast Fourier transform is a fast algorithm for computing discrete Fourier transform. We take $\xi = \exp\left(-\frac{2\pi}{2N}\mathtt{I}\right)$, then for a polynomial $f(\mathtt{X}) = \sum_{i=0}^{N-1} a_i \mathtt{X}^i$ in $\mathcal{R}$, the FFT of the coefficient vector $\vec{f}$ of the polynomial f is

$$\begin{aligned}\mathtt{FFT}(\vec{f}) &= \left(f(\xi)\ f(\xi^3)\ \cdots\ f(\xi^{2N-1})\right)^T \\ &= \mathtt{Vand}\left(\xi, \xi^3, \cdots, \xi^{2N-1}\right)\vec{f} \\ &= \begin{bmatrix} 1 & \xi & \cdots & \xi^{N-1} \\ 1 & \xi^3 & \cdots & \xi^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \xi^{2N-1} & \cdots & \xi^{(2N-1)(N-1)} \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix}\end{aligned}$$

In the above formula, $\mathtt{Vand}\left(\xi, \xi^3, \cdots, \xi^{2N-1}\right)$ is the Vandermonde matrix generated by the primitive root ξ. Similarly, the inverse fast Fourier transform, IFFT is

$$\mathtt{IFFT}(\mathtt{FFT}(\vec{f})) = \vec{f} = \frac{1}{N}\mathtt{Vand}\left(\xi^{-1}, \xi^{-3}, \cdots, \xi^{-(2N-1)}\right)\mathtt{FFT}(\vec{f})$$

FFT and IFFT can be evaluated with computational complexity of $\mathcal{O}(N \log_2 N)$.

3 FFT-less External Product

3.1 Magnitude-Preserving Property of FFT

Unlike its finite filed variant NTT, FFT would preserve the magnitude when converting a polynomial from (to) coefficient form to (from) evaluation form. In

brief (but not precisely), polynomials with small coefficients would be converted to evaluation form with small terms; polynomials with large coefficients would be converted to evaluation form with large terms. Such magnitude-preserving property is not possible in NTT, since the wrapping-around in finite field arithmetic is possible to map large coefficients to small terms. In this sense, our proposal can only be applied to those schemes using FFT. In fact, all digit decompositions in any homomorphic encryption scheme would benefit from our proposal.

We first introduce the linearity of FFT and then present a formal description along with proof of the magnitude-preserving property, which would make up all the ingredients required for our FFT-less TFHE.

Property 1 (Linearity of FFT).

$$\texttt{FFT}(s_a a + s_b b) = s_a \texttt{FFT}(a) + s_b \texttt{FFT}(b), \quad \forall s_a, s_b \in \mathbb{C}, \forall a, b \in \mathbb{C}^N \tag{8}$$

Property 2 (Magnitude-preserving Property of FFT).

$$\left\|\texttt{FFT}(a)\right\|_\infty \leq N \left\|a\right\|_\infty, \forall a \in \mathbb{C}^N \tag{9}$$

$$\left\|\texttt{IFFT}(a)\right\|_\infty \leq \left\|a\right\|_\infty, \forall a \in \mathbb{C}^N \tag{10}$$

The above linearity of FFT is quite straightforward. The magnitude-preserving property is a direct corollary of Cauchy-Schwarz Inequality

Proof (Correctness of Property 2).

For arbitrary $a = \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \end{pmatrix} \in \mathbb{C}^N$, we have

$$\begin{aligned}
&\left\|\texttt{FFT}\left(a\right)\right\|_\infty \\
=&\left\|\texttt{Vand}\left(\xi^1, \xi^3, \cdots, \xi^{2N-1}\right) a\right\|_\infty \\
=&\max_{k=0,1,\cdots,N-1} \left\{ \left\| \sum_{i=0}^{N-1} \xi^{(2k-1)i} a_i \right\| \right\} \\
\leq&\max_{k=0,1,\cdots,N-1} \left\{ \sqrt{\left(\sum_{i=0}^{N-1} \left\|\xi^{(2k-1)i}\right\|^2 \right) \left(\sum_{i=0}^{N-1} \left\|a_i\right\|^2 \right)} \right\} \\
=&\max_{k=0,1,\cdots,N-1} \left\{ \sqrt{N \sum_{i=0}^{N-1} \left\|a_i\right\|^2} \right\} \\
\leq&\max_{k=0,1,\cdots,N-1} \left\{ \sqrt{N \times N \left\|a\right\|_\infty^2} \right\} \\
=&N \left\|\vec{a}\right\|_\infty
\end{aligned}$$

The magnitude-preserving property for IFFT can be similarly proven.

3.2 Decomposition in FFT Form

Before presenting our algorithm, we have to investigate why Decomp is used in TFHE and show the main constraints of constructing a valid Decomp. Actually, for ExtProd, its noise estimation is

$$\begin{aligned} e_{\texttt{ExtProd}} &\leq \mu_1 \|e\|_\infty + \mu_1(1+N)\varepsilon^{\ell,\mathfrak{B}} + 2\ell\mathfrak{B}\|E\|_\infty \\ &= \mu_1 \|e\|_\infty + \mu_1(1+N)\frac{Q}{\mathfrak{B}^\ell} + 2\ell\mathfrak{B}\|E\|_\infty . \end{aligned}$$

Therefore, Decomp is used for the reduction of noise growth in ExtProd and there are two constraints for Decomp :

1. The norm of the decomposed ciphertext should not be too large, which in Decomp is upper bounded the base $\mathfrak{B}$.
2. The reconstructed ciphertext, which is the inner product of the decomposed ciphertext and the gadget vector, is close to the original ciphertext with a limited error, which in our case, is $\varepsilon^{\ell,\mathfrak{B}}$.

The necessity of decomposition in coefficient form has not been seriously discussed ever. Actually, in NTT-based implementations, the only choice for digit decomposition is converting to coefficient forms and then decomposing. However, with the magnitude-preserving property and linearity of FFT, decomposition in FFT-based implementations is possible for decomposing in evaluation forms.

With the magnitude-preserving property and linearity of FFT, we present here our proposal of *Decomposition-in-FFT-Form* as follows:

$$\begin{aligned} &\texttt{DecompFFT}^{\ell,\mathfrak{B}}((a,b)) \\ =&\left[\texttt{Dig}_{\texttt{FFT}}(a,0)\cdots\texttt{Dig}_{\texttt{FFT}}(a,\ell-1)\quad \texttt{Dig}_{\texttt{FFT}}(b,0)\cdots\texttt{Dig}_{\texttt{FFT}}(b,\ell-1)\right] \end{aligned}$$

where $\texttt{Dig}_{\texttt{FFT}}(a,i) = \left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}$, and $\vec{g}_{\texttt{FFT}} = N\cdot\vec{g}$, $\mathcal{G}^{\ell,\mathfrak{B}}_{\texttt{FFT}} = N\cdot\mathcal{G}^{\ell,\mathfrak{B}}$.

On one hand, such $\texttt{DecompFFT}^{\ell,\mathfrak{B}}((a,b))$ is a valid decomposition. Actually, since $\|\texttt{FFT}(a)\|_\infty \leq N\|a\|_\infty \leq NQ$, we have

$$\begin{aligned} \varepsilon^{\ell,\mathfrak{B}}_{\texttt{FFT}}(a) =& \texttt{FFT}(a) - \texttt{DecompFFT}^{\ell,\mathfrak{B}}(a)\cdot\vec{g}_{\texttt{FFT}} \\ =& \texttt{FFT}(a) - \sum_{i=0}^{\ell-1}\texttt{Dig}_{\texttt{FFT}}(a,i)\frac{QN}{\mathfrak{B}^{i+1}} \\ =& \texttt{FFT}(a) - \sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\frac{QN}{\mathfrak{B}^{i+1}} \\ =& \texttt{FFT}(a) - \left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{\ell}}\right\rfloor\frac{QN}{\mathfrak{B}^{\ell}} \end{aligned}$$

Therefore, $\varepsilon^{\ell,\mathfrak{B}}_{\texttt{FFT}} = \max_{a\in\mathcal{R}_Q}\left\{\left\|\varepsilon^{\ell,\mathfrak{B}}_{\texttt{FFT}}(a)\right\|_\infty\right\} = \frac{\sqrt{2}QN}{\mathfrak{B}^\ell}$.

On the other hand, the norm of the decomposed ciphertext is upper bounded by $\sqrt{2}\mathfrak{B}$. Therefore, our construction makes a valid decomposition. Our FFT-less external product is then shown in Algorithm.2.

Algorithm 2 $\texttt{ExtProdFFT}$ in $\texttt{TFHE}$

Require: $(a,b)=\texttt{RLWE}_{\mathfrak{s}}\left(\mu_0|a,e\right),\texttt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}\left(\mu_1|A,E\right)$
Ensure: $\mathbf{ct}_{\texttt{ExtProd}}=\texttt{RLWE}_{\mathfrak{s}}\left(\mu_0\mu_1|a_{\texttt{ExtProd}},e_{\texttt{ExtProd}}\right)$
$\mathbf{ct}_{\texttt{DecompFFT}}=\texttt{DecompFFT}^{\ell,\mathfrak{B}}\left((a,b)\right)$
$\mathbf{ct}_{\texttt{ExtProdFFT}}=\mathbf{ct}_{\texttt{DecompFFT}}\cdot\texttt{RGSW}_{\mathfrak{s}}^{\ell,\mathfrak{B}}\left(\mu_1|A,E\right)$
$\mathbf{ct}_{\texttt{ExtProd}}[i]=\left[\lfloor\Re\{\texttt{IFFT}(\mathbf{ct}_{\texttt{ExtProdFFT}}[i])\}\rfloor\right]_Q\quad i=0,1$

3.3 Noise Analysis

To prove the correctness of Algorithm.2, we only to show that the error

$$\epsilon_{\texttt{ExtProd}}=\mathbf{ct}_{\texttt{ExtProd}}[1]-\mathbf{ct}_{\texttt{ExtProd}}[0]\mathfrak{s}-\mu_0\mu_1$$

would not blow up. Actually, assume that

$$\epsilon_i=\texttt{IFFT}(\mathbf{ct}_{\texttt{ExtProdFFT}}[i])-\lfloor\Re\{\texttt{IFFT}(\mathbf{ct}_{\texttt{ExtProdFFT}}[i])\}\rfloor\quad i=0,1$$

then we have

$$\begin{aligned}
&\epsilon_{\texttt{ExtProd}}\\
=&\mathbf{ct}_{\texttt{ExtProd}}[1]-\mathbf{ct}_{\texttt{ExtProd}}[0]\mathfrak{s}-\mu_0\mu_1\\
=&\texttt{IFFT}(\mathbf{ct}_{\texttt{ExtProdFFT}}[1])-\epsilon_1-\texttt{IFFT}(\mathbf{ct}_{\texttt{ExtProdFFT}}[0])\mathfrak{s}+\epsilon_0\mathfrak{s}-\mu_0\mu_1\\
=&\texttt{IFFT}\left(\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(B_i)+\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(B_{i+\ell}+\mu_1\frac{QN}{\mathfrak{B}^{i+1}})\right)-\epsilon_1\\
&-\texttt{IFFT}\left(\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(A_i+\mu_1\frac{QN}{\mathfrak{B}^{i+1}})+\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(A_{i+\ell})\right)\mathfrak{s}+\epsilon_0\mathfrak{s}-\mu_0\mu_1\\
=&\texttt{IFFT}\left(\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(B_i-(A_i+\mu_1\frac{QN}{\mathfrak{B}^{i+1}})\mathfrak{s})\right.\\
&\left.+\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(B_{i+\ell}+\mu_1\frac{QN}{\mathfrak{B}^{i+1}}-A_{i+\ell}\mathfrak{s})\right)-\epsilon_1+\epsilon_0\mathfrak{s}-\mu_0\mu_1\\
=&\texttt{IFFT}\left(\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(-\mu_1\frac{QN}{\mathfrak{B}^{i+1}}\mathfrak{s}+E_i)+\sum_{i=0}^{\ell-1}\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\odot\texttt{FFT}(\mu_1\frac{QN}{\mathfrak{B}^{i+1}}+E_{i+\ell})\right)\\
&-\epsilon_1+\epsilon_0\mathfrak{s}-\mu_0\mu_1\\
=&\mu_1\texttt{IFFT}\left(\sum_{i=0}^{\ell-1}\left(\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\frac{QN}{\mathfrak{B}^{i+1}}-\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\frac{QN}{\mathfrak{B}^{i+1}}\odot\texttt{FFT}(\mathfrak{s})\right)\right)\\
&+\sum_{i=0}^{\ell-1}\texttt{IFFT}\left(\left[\left\lfloor\frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\right)E_i+\sum_{i=0}^{\ell-1}\texttt{IFFT}\left(\left[\left\lfloor\frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}}\right\rfloor\right]_{\mathfrak{B}}\right)E_{i+\ell}-\epsilon_1+\epsilon_0\mathfrak{s}-\mu_0\mu_1
\end{aligned}$$

$$
\begin{aligned}
=&\mu_1 \mathtt{IFFT}\left(\mathtt{FFT}(b) - \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(b) - (\mathtt{FFT}(a) - \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(a)) \odot \mathtt{FFT}(\mathfrak{s})\right) + \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_i \\
&+ \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_{i+\ell} - \epsilon_1 + \epsilon_0\mathfrak{s} - \mu_0\mu_1 \\
=&\mu_1(b - a\mathfrak{s}) - \mu_1 \mathtt{IFFT}(\varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(b) - \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(a) \odot \mathtt{FFT}(\mathfrak{s})) + \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_i \\
&+ \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_{i+\ell} - \epsilon_1 + \epsilon_0\mathfrak{s} - \mu_0\mu_1 \\
=&\mu_1(\mu_0 + e) - \mu_1 \mathtt{IFFT}(\varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(b) - \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(a) \odot \mathtt{FFT}(\mathfrak{s})) + \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_i \\
&+ \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_{i+\ell} - \epsilon_1 + \epsilon_0\mathfrak{s} - \mu_0\mu_1 \\
=&\mu_1 e - \mu_1 \mathtt{IFFT}(\varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(b) - \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(a) \odot \mathtt{FFT}(\mathfrak{s})) + \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_i \\
&+ \sum_{i=0}^{\ell-1} \mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right) E_{i+\ell} - \epsilon_1 + \epsilon_0\mathfrak{s}
\end{aligned}
$$

Therefore, the noise growth of `ExtProdFFT` can be estimated by the following two theorems:

Theorem 1. *The noise growth in worst case of* ***`ExtProdFFT`*** *is*

$$
\|\epsilon_{\texttt{ExtProd}}\|_\infty \le \|\mu_1\|_1 \|e\|_\infty + (1+N)\sqrt{2}\|\mu_1\|_1 \frac{QN}{\mathfrak{B}^l} + 2\sqrt{2}\ell N\mathfrak{B}\|E\|_\infty + (1+N)
$$

Theorem 2. *The noise growth in average case of* ***`ExtProdFFT`*** *is*

$$
\begin{aligned}
\mathtt{Var}(\epsilon_{\texttt{ExtProd}}) = &\|\mu_1\|_1 \mathtt{Var}(e) + \|\mu_1\|_1 \frac{2(Q^2N^2 - \mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}} \left(1 + N\mathtt{Var}(\mathfrak{s}_i) + \mathbb{E}^2(\mathfrak{s}_i)\right) \\
&+ \frac{4\ell\mathfrak{B}(\mathfrak{B}-1)}{3}\mathtt{Var}(E_i) + \frac{1}{12} + \frac{N}{12}\left(4\mathtt{Var}(\mathfrak{s}_i) + \mathbb{E}^2(\mathfrak{s}_i)\right)
\end{aligned}
$$

The Worst Case Noise Growth First of all, as discussed before, $\varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(a), \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}}(b) \le \varepsilon_{\mathtt{FFT}}^{\ell,\mathfrak{B}} \le \sqrt{2}\frac{QN}{\mathfrak{B}^l}$.

Secondly, $\left\|\mathtt{IFFT}\left(\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right)\right\|_\infty \le \left\|\left[\left\lfloor \frac{\mathtt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right\|_\infty \le \sqrt{2}\mathfrak{B}$.

Finally, $\|\epsilon_i\|_\infty \le 1,\ \|\epsilon_0\mathfrak{s}\|_\infty \le 1 * N * 1 = N$.

Therefore, the worst-case noise growth is bounded by:

$$
\begin{aligned}
&\|\epsilon_{\texttt{ExtProd}}\|_\infty \\
=&\left\| \mu_1 e - \mu_1 \texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(b) - \varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a) \odot \texttt{FFT}(\mathfrak{s})) + \sum_{i=0}^{\ell-1} \texttt{IFFT}\left(\left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rceil \right]_{\mathfrak{B}} \right) E_i \right. \\
&\left. + \sum_{i=0}^{\ell-1} \texttt{IFFT}\left(\left[\left\lfloor \frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rceil \right]_{\mathfrak{B}} \right) E_{i+\ell} - \epsilon_1 + \epsilon_0 \mathfrak{s} \right\|_\infty \\
\leq& \|\mu_1\|_1 \|e\|_\infty + (1+N) \|\mu_1\|_1 \varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}} + \ell * \sqrt{2}\mathfrak{B} * N \|E\|_\infty + \ell * \sqrt{2}\mathfrak{B} * N \|E\|_\infty + 1 + N \\
\leq& \|\mu_1\|_1 \|e\|_\infty + (1+N)\sqrt{2} \|\mu_1\|_1 \frac{QN}{\mathfrak{B}^l} + 2\sqrt{2}\ell N \mathfrak{B} \|E\|_\infty + (1+N)
\end{aligned}
$$

which is exactly as stated in Theorem.1.

The Average Case Noise Growth Before going to the analysis of the noise terms, we have a look at the variance of a general case:

Lemma 1. *For a complex vector u with N i.i.d elements, which are uniformly distributed on $\{a + b\mathtt{I} | a, b \in [-T, T] \cap \mathbb{Z}\}$ with $T \in \mathbb{N}^+$, then*

$$
\begin{aligned}
\mathbb{E}(\mathit{IFFT}(u)[i]) &= 0 \\
\texttt{Var}\left(\mathit{IFFT}(u)[i]\right) &= \frac{2T(T+1)}{3N}
\end{aligned}
$$

Actually, we have

$$
\begin{aligned}
\mathbb{E}(\texttt{IFFT}(u)[i]) &= \mathbb{E}(\frac{1}{N} \sum_{j=0}^{N-1} u_j \exp\left(\frac{ij \times 2\pi\mathtt{I}}{2N}\right)) = \frac{1}{N} \sum_{j=0}^{N-1} \mathbb{E}(u_j) \exp\left(\frac{ij \times 2\pi\mathtt{I}}{2N}\right) \\
&= \frac{1}{N} \sum_{j=0}^{N-1} \sum_{s,t=-T}^{T} \frac{s + t\mathtt{I}}{(2T+1)^2} \exp\left(\frac{ij \times 2\pi\mathtt{I}}{2N}\right) = 0 \\
\texttt{Var}\left(\texttt{IFFT}(u)[i]\right) &= \mathbb{E} \left\| \texttt{IFFT}(u)[i] - \mathbb{E}(\texttt{IFFT}(u)[i]) \right\|_2^2 \\
&= \mathbb{E}\left(\frac{1}{N^2} \sum_{j=0,k=0}^{N-1,N-1} u_j \overline{u_k} \exp\left(\frac{i(j-k) \times 2\pi\mathtt{I}}{2N}\right) \right) \\
&= \mathbb{E}\left(\frac{1}{N^2} \sum_{j=0}^{N-1} \|u_j\|_2^2 \right) + \mathbb{E}\left(\frac{1}{N^2} \sum_{0 \leq j \neq k \leq N-1} u_j \overline{u_k} \exp\left(\frac{i(j-k) \times 2\pi\mathtt{I}}{2N}\right) \right) \\
&= \mathbb{E}\left(\frac{1}{N^2} \sum_{j=0}^{N-1} \|u_j\|_2^2 \right) + \frac{1}{N^2} \sum_{0 \leq j \neq k \leq N-1} \mathbb{E}\left(u_j\right) \mathbb{E}\left(\overline{u_k}\right) \exp\left(\frac{i(j-k) \times 2\pi\mathtt{I}}{2N}\right) \\
&= \mathbb{E}\left(\frac{1}{N^2} \sum_{j=0}^{N-1} \|u_j\|_2^2 \right) = \frac{N}{N^2} \mathbb{E}(\|u_j\|_2^2) = \frac{1}{N} \frac{\sum_{s,t=-T}^{T} s^2 + t^2}{(2T+1)^2} = \frac{1}{N} \frac{2(2T+1)\sum_{s=-T}^{T} s^2}{(2T+1)^2}
\end{aligned}
$$

$$= \frac{1}{N}\frac{2}{2T+1}\frac{T(T+1)(2T+1)}{6} \times 2 = \frac{2T(T+1)}{3N}$$

For $\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(b))$, one needs to set $T = \frac{QN}{\mathfrak{B}^\ell} - 1$, and for $\texttt{IFFT}(\left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}})$, one needs to set $T = \mathfrak{B} - 1$, namely

$$\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(b))\right) = \frac{2\frac{QN}{\mathfrak{B}^\ell}(\frac{QN}{\mathfrak{B}^\ell}-1)}{3N} = \frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}}$$
$$\texttt{Var}\left(\texttt{IFFT}(\left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}})\right) = \frac{2\mathfrak{B}(\mathfrak{B}-1)}{3N}$$

Further, we have

$$\begin{aligned}
&\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(b) - \varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a) \odot \texttt{FFT}(\mathfrak{s}))\right)\\
=&\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(b))\right) + \texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a))\mathfrak{s}\right)\\
=&\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}} + N\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a))[i]\mathfrak{s}_i\right)\\
=&\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}} + N\left(\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a))[i]\right)\texttt{Var}\left(\mathfrak{s}_i\right) + \mathbb{E}^2\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a))[i]\texttt{Var}\left(\mathfrak{s}_i\right)\right.\right.\\
&\left.\left.+\texttt{Var}\left(\texttt{IFFT}(\varepsilon_{\texttt{FFT}}^{\ell,\mathfrak{B}}(a))[i]\right)\mathbb{E}^2\left(\mathfrak{s}_i\right)\right)\right)\\
=&\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}} + N(\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}}\texttt{Var}\left(\mathfrak{s}_i\right) + \frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}}\mathbb{E}^2\left(\mathfrak{s}_i\right))\\
=&\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}}\left(1 + N\texttt{Var}\left(\mathfrak{s}_i\right) + \mathbb{E}^2\left(\mathfrak{s}_i\right)\right)
\end{aligned}$$

$$\begin{aligned}
&\texttt{Var}\left(\sum_{i=0}^{\ell-1}\texttt{IFFT}\left(\left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right)E_i + \sum_{i=0}^{\ell-1}\texttt{IFFT}\left(\left[\left\lfloor \frac{\texttt{FFT}(b)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right)E_{i+\ell}\right)\\
=&2N\ell\texttt{Var}\left(\texttt{IFFT}\left(\left[\left\lfloor \frac{\texttt{FFT}(a)}{QN/\mathfrak{B}^{i+1}} \right\rfloor\right]_{\mathfrak{B}}\right)\right)\texttt{Var}\left(E_i\right)\\
=&2N\ell\frac{2\mathfrak{B}(\mathfrak{B}-1)}{3N}\texttt{Var}\left(E_i\right)
\end{aligned}$$

$$\begin{aligned}
&\texttt{Var}\left(e_1 + e_0\mathfrak{s}\right)\\
=&\frac{1}{12} + N(\frac{1}{12}\texttt{Var}\left(\mathfrak{s}_i\right) + \frac{1}{4}\texttt{Var}\left(\mathfrak{s}_i\right) + \frac{1}{12}\mathbb{E}^2\left(\mathfrak{s}_i\right))\\
=&\frac{1}{12} + \frac{N}{12}\left(4\texttt{Var}\left(\mathfrak{s}_i\right) + \mathbb{E}^2\left(\mathfrak{s}_i\right)\right)
\end{aligned}$$

$$\begin{aligned}
&\texttt{Var}\left(\epsilon_{\texttt{ExtProd}}\right)\\
=&\left\|\mu_1\right\|_1\texttt{Var}\left(e\right) + \left\|\mu_1\right\|_1\frac{2(Q^2N^2-\mathfrak{B}^{2\ell})}{3N\mathfrak{B}^{2\ell}}\left(1 + N\texttt{Var}\left(\mathfrak{s}_i\right) + \mathbb{E}^2\left(\mathfrak{s}_i\right)\right) + \frac{4\ell\mathfrak{B}(\mathfrak{B}-1)}{3}\texttt{Var}\left(E_i\right)
\end{aligned}$$

$$+\frac{1}{12}+\frac{N}{12}\left(4\mathtt{Var}\left(\mathfrak{s}_i\right)+\mathbb{E}^2\left(\mathfrak{s}_i\right)\right)$$

Considering the imaginary part is always ignored, the variance can be even smaller, which is

$$\begin{aligned}\mathtt{Var}\left(\epsilon_{\mathtt{ExtProd}}\right)=\left\|\mu_1\right\|_1\mathtt{Var}\left(e\right)+\left\|\mu_1\right\|_1\frac{Q^2N^2-\mathfrak{B}^{2\ell}}{3N\mathfrak{B}^{2\ell}}\left(1+N\mathtt{Var}\left(\mathfrak{s}_i\right)+\mathbb{E}^2\left(\mathfrak{s}_i\right)\right)+\frac{2\ell\mathfrak{B}(\mathfrak{B}-1)}{3}\mathtt{Var}\left(E_i\right)\\+\frac{1}{12}+\frac{N}{12}\left(4\mathtt{Var}\left(\mathfrak{s}_i\right)+\mathbb{E}^2\left(\mathfrak{s}_i\right)\right)\end{aligned}$$

With the above noise analysis, we say Algorithm.2 is correct with similar noise growth as the original `TFHE ExtProd`.

4 Discussions and Future Work

With Algorithm.2, we obtain a `TFHE` external product with fewer `FFT`s. We present a very brief comparison on the number of `FFT`s required by an `ExtProd` in with (w/) or without (w/o) FFT-less method in Figure.1. It is straightforward to see that our method outperform the original `TFHE` in performance as long as $\ell \geq 2$.

Meanwhile, beyond the reduction in computational complexity, Algorithm.2 introduces many other benefits, and we discuss about two most significant benefits here.

Firstly, in Algorithm.2 the number of `FFT`s is no longer dependent on ℓ, which makes Algorithm.2 scale-invariant in the number of `FFT`s. This is quite important in the design of hardware accelerators, since the design would need no changes or redundancy to satisfy the requirement of different choices of ℓ.

Secondly, the order of `FFT` and `Decompose` is switched in Algorithm.2, which makes the very first and the very last operations of Algorithm.2 are a pair of `FFT` and `IFFT`. In `BlindRotate`, `IFFT-FFT` and Matrix-vector multiplication are interleaved. In this way, the computational pattern is much simpler than the original `TFHE`.

With the above two benefits, our algorithm would show advantage over the original `TFHE` in some cases.

4.1 Is `FFT`-free possible?

So far we have presented a `FFT`-less TFHE, a quick question is that, is `FFT`-free TFHE possible? We have not any answer to the question, but some simple observations are presented.

Modular Reduction in `FFT` Domain and The Cloest-Vector Problem A naive thought of transforming `FFT`-less to `FFT`-free is to perform modular reduction in Fourier domain without transforming to and from the normal domain

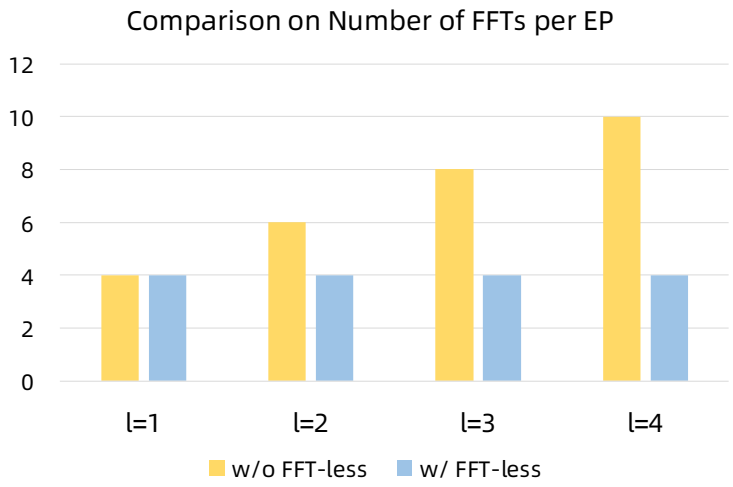

Fig. 1. Comparison on Number of FFTs per `ExtProd`

via `FFT`/`IFFT`s. We found that without the help of `FFT`/`IFFT`s, performing the modular reduction in Fourier domain is equivalent to solving a variant of the Closest-Vector Problem (CVP).

To see this, assuming that we have to obtain $\texttt{FFT}\left(\left[\vec{A}\right]_Q\right)$ from $\texttt{FFT}\left(\vec{A}\right)$, where $A \in \mathbb{Z}^N$, we have to find out a integral linear combination of $\texttt{FFT}\left(Q\mathcal{E}_i\right)$ with $\mathcal{E}_i$ being a vector of zeros expect i-th element being one, such that the linear combination is closest to $\texttt{FFT}\left(\vec{A}\right)$, namely,

$$\texttt{FFT}\left(\left[\vec{A}\right]_Q\right) = \texttt{FFT}\left(\vec{A} - \vec{k}Q\right) = \texttt{FFT}\left(\vec{A}\right) - \texttt{FFT}\left(\vec{k}Q\right) = \texttt{FFT}\left(\vec{A}\right) - \sum_{i=0}^{N-1} k_i \texttt{FFT}\left(Q\mathcal{E}_i\right)$$

where $\left\lceil \frac{\vec{A}}{Q} \right\rfloor$.Therefore, we can see that the modular reduction in Fourier domain without `FFT`/`IFFT`s is equivalent to finding the lattice vector closest to $\texttt{FFT}\left(\vec{A}\right)$ in the lattice generated by the basis $\{\texttt{FFT}\left(Q\mathcal{E}_i\right)\}_{i=0}^{N-1}$.

It is quite interesting to bridge modular reduction in Fourier domain and the CVP. From the computational perspective, the lattice is only linear in dimension and the basis can be easily transformed to a standard basis. Thus, whether or not there exists a efficient way of performing modular reduction in Fourier is itself interesting.

Lazy Reduction for Fewer `FFT`s Though we have not known any way of obtaining `FFT`-free `TFHE`, a method of performing lazy reduction with fewer `FFT`s

than FFT-less TFHE actually exits. In brief, as long as the result of the matrix-vector multiplication in Algorithm.2 still can be reconstructed via $\texttt{DecompFFT}^{\ell,\mathfrak{B}}(\cdot)$, it can be directly bypassed to the next matrix-vector product without performing any IFFT/FFTs. However, the parameters should be carefully selected to satisfy such requirement.

A severe problem of aggressive lazy reduction is that the decomposition redundancy would make the bootstrapping key vulnerable. Consider that the gadget matrix is multiplied by a large scalar for the application of lazy reduction. Then when added to the RGSW ciphertext, it is easy to tell is it encrypting 0 or 1 since the infinity norm of ciphertexts encrypting 1s are significantly larger than the ciphertexts encrypting 0s. Therefore, out of the consideration of security, we suggest the largest term of the gadget vector should be smaller than the average case of the maximal possible term in a RGSW ciphertext.

4.2 Hardware Acceleration

The hardware accelerators like MATCHA, FPT, cuFHE and nuFHE can easily adapt to our FFT-less TFHE [10][14][15]. Actually, FFTs of TFHE occupy most of the resources in these accelerators. Therefore, with the FFT-less technology, they will either obtain higher throughput with similar resources, or remain the throughput with less resource utilization.

References

1. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
2. 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 Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
3. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
4. De Micheli, G., Kim, D., Micciancio, D., Suhl, A.: Faster amortized fhew bootstrapping using ring automorphisms. Cryptology ePrint Archive (2023)
5. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34. pp. 617–640. Springer (2015)
6. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
7. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)

8. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. pp. 75–92. Springer (2013)
9. Guimarães, A., Pereira, H.V., van Leeuwen, B.: Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. Cryptology ePrint Archive (2023)
10. Jiang, L., Lou, Q., Joshi, N.: Matcha: A fast and energy-efficient accelerator for fully homomorphic encryption over the torus. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. pp. 235–240 (2022)
11. Micciancio, D., Sorrell, J.: Ring packing and amortized fhew bootstrapping. Cryptology ePrint Archive (2018)
12. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)
13. Regev, O.: Learning with errors over rings. In: Algorithmic Number Theory: 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings 9. pp. 3–3. Springer (2010)
14. Van Beirendonck, M., D'Anvers, J.P., Verbauwhede, I.: Fpt: a fixed-point accelerator for torus fully homomorphic encryption. arXiv preprint arXiv:2211.13696 (2022)
15. Zhang, J., Cheng, X., Yang, L., Hu, J., Liu, X., Chen, K.: Sok: Fully homomorphic encryption accelerators. arXiv preprint arXiv:2212.01713 (2022)