Patching and Extending the WWL+ Circuit Bootstrapping Method to FFT Domains

Jincheol Ha and Jooyoung Lee

KAIST, Daejeon, Korea,
{smilecjf,hicalf}@kaist.ac.kr

Abstract. TFHE is a homomorphic encryption scheme supporting fast bootstrapping. There are two kinds of bootstrapping in TFHE: programmable bootstrapping (also known as gate bootstrapping) and circuit bootstrapping. Circuit bootstrapping offers more functionality than programmable bootstrapping, but requires heavier computational cost and larger evaluation key size. A recent work by Wang et al. improving circuit bootstrapping using homomorphic trace evaluation seems to mitigate its heavy cost, while we observe some flaws in their error analysis.
In this paper, we patch the circuit bootstrapping method proposed by Wang et al. with correct error analysis and extend the ciphertext modulus from a prime modulus to a power-of-two modulus, enabling FFT-based implementation of our patched method. In addition, we propose a high precision WWL+ method by adopting GLWE keyswitching, improving the circuit bootstrapping time (resp. key size) of WoP-PBS proposed by Bergerat et al. by factors from 3.26 to 7.22 (resp. 2.39 to 2.63). We also patch the parameter selection used in the AES evaluation by the WWL+ method, obtaining 26.301s for a single AES evaluation in a single thread.

Keywords: homomorphic encryption, TFHE, fast Fourier transform, ciphertext conversion, circuit bootstrapping, WoP-PBS, AES evaluation

1 Introduction

TFHE [8, 9, 10], proposed as a variant of FHEW [16], is a homomorphic encryption scheme based on GSW [17] that supports fast bootstrapping. Both FHEW and TFHE have been proposed as homomorphic encryption schemes supporting homomorphic evaluation of Boolean circuits: FHEW bootstrapping evaluates the homomorphic NAND gate of two bits [16] and TFHE allows one to build various Boolean operations on top of it [8], while every binary Boolean gate is homomorphically evaluated using a single fast bootstrapping of FHEW/TFHE.

The idea of evaluating NAND gates during bootstrapping can be extended to evaluation of any (negacyclic) univariate function. In the context of TFHE, it is called programmable bootstrapping (PBS) [11, 12]. By the PBS operation, TFHE is extended to a leveled homomorphic encryption scheme that supports almost free linear operations (in terms of computation time) and non-linear

operations by the programmable bootstrapping on the plaintext space of a small precision [11, 13, 3].

In order to evaluate a large circuit, an alternative method of bootstrapping method, called circuit bootstrapping, has been proposed [9]. It takes an LWE ciphertext of a single-bit message as input, producing the corresponding generalized GSW ciphertext, which enables evaluating the controlled selector gate called CMux. Circuit bootstrapping-based function evaluation is efficient in terms of evaluation time when the input size of the function is large [9, 3], while circuit bootstrapping itself is much heavier than programmable bootstrapping in terms of both evaluation time and key size.

Both types of bootstrapping methods require a significant amount of polynomial multiplications over $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N+1)$ where q is the ciphertext modulus and N is a power-of-two. To boost the speed of polynomial multiplication, FHEW/TFHE-like schemes employ either fast Fourier transform (FFT) on a power-of-two ciphertext modulus or number theoretic transform (NTT) on a prime ciphertext modulus. Both methods reduce polynomial multiplication cost to $O(N \log N)$. So their computational costs are asymptotically of the same order, while there are significant differences in using FFT and NTT in practice.

Using FFT allows one to use any power-of-two ciphertext modulus ($q \in \{2^{32}, 2^{64}\}$ is mostly used), while it might produce computational error according to q, N and the precision of the floating point number. For example, double-precision floating-point representation using 64 bits only provides precision of 53 bits, so polynomial multiplication modulo 2^{64} cannot be computed exactly using double-precision FFT.[1]

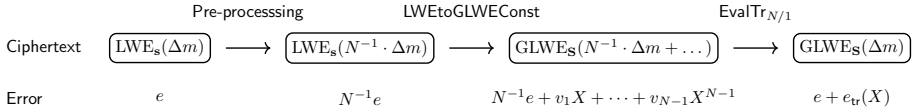
On the other hand, NTT allows one to compute polynomial multiplication exactly while it requires an NTT-friendly ciphertext modulus q. Over the power-of-two cyclotomic polynomial ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N+1)$ where N is a power-of-two, NTT does not support a power-of-two ciphertext modulus, increasing the overall cost of modulo operations.[2]

1.1 Motivation

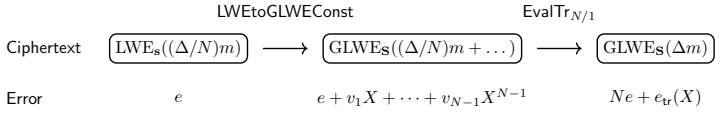
Recently, Wang et al. [24] proposed a new circuit bootstrapping method (called WWL+ method in this paper) that reduces both computation time and evaluation key size. Circuit bootstrapping operation can be divided into two steps: the first step of programmable bootstrapping to refresh the input ciphertext and change its scaling factor, and the second step of private keyswitching to reform the ciphertext from LWE to (generalized) GSW. One of the main ideas of the WWL+ method is to replace private keyswitching by homomorphic trace

[1] There is a data type for floating point number of arbitrary precision, but it degrades the FFT performance significantly.

[2] To overcome this limitation, there is an approach to change the quotient polynomials to support NTT on a power-of-two modulus [18]. However, the set of functions that can be evaluated by PBS (called PBS-friendly in the paper) depends on the quotient polynomials.



(a) Pre-processing of multiplying N^{-1} mod q, which does not amplify the initial error but is only possible on an NTT domain.



(b) Adjusting the input scaling factor by Δ/N, which is possible on both NTT and FFT domains but amplifies the initial error by N.

Fig. 1: Two possible methods of mitigating the phase amplification in the automorphism-based LWE to GLWE conversion.

evaluation [7] and scheme switching [15], rendering the second step negligible compared to the first step in terms of both computation time and evaluation key size.

However, we observe that the authors made two mistakes in the error analysis: 1) the error amplification by the trace evaluation is not considered, and 2) a multiplication factor in the error growth by the scheme switching is missing.[3] This implies that the PBS error should decrease, degrading the overall performance of their method.

The first issue occurs on the homomorphic trace evaluation, proposed by Chen et al. [7] to efficiently convert an LWE ciphertext into the corresponding GLWE ciphertext. Compared to the previous conversion method called packing keyswitching [13], this method reduces the computational cost (resp. key size) of LWE to GLWE conversion from $O(N^2)$ to $O(N\log^2 N)$ (resp. $O(N\log N)$) where N is the polynomial size. There is a straightforward way of converting an LWE ciphertext into a GLWE ciphertext with additional unknown coefficients in non-constant terms (when the LWE secret key can be converted into the GLWE secret key). Since trace evaluation multiplies the constant term by N and removes all the non-constant coefficients, it is possible to convert an LWE ciphertext into the corresponding GLWE ciphertext if the multiplication of the constant term by N can be canceled out. In this paper, this multiplication by N is called the *phase amplification* of the trace evaluation.

Chen et al. [7] handled this phase amplification by assuming the existence of N^{-1} mod q, which is reasonable on the NTT domain, and proposing a pre-processing step of multiplying N^{-1} mod q to the input before the trace evaluation. Figure 1a describes the automorphism-based conversion with pre-processing on an NTT domain proposed in [7]. With this pre-processing, only homomorphic

[3] The authors of [24] confirmed the issues and presented adjusted parameters and performance at Eurocrypt 24.

trace evaluation error is added through the conversion. Unfortunately, this approach does not apply to an FFT domain since there is no $N^{-1} \bmod q$ in an FFT domain since N and q, both being power of two, are not relatively prime. One possible way of using the automorphism-based conversion on an FFT domain is to set the scaling factor of an input LWE ciphertext to Δ/N ($N^{-1}\Delta \mod q$ in an NTT domain) to cancel out the amplification on the scaling factor. However, the initial error is still amplified by N in this method (see Figure 1b). Thus, this approach is limited only to cases where the input ciphertext has a small enough error considering the error amplification.

Wang et al. [24] implemented their WWL+ method on NTT domains, while they employed the second pre-processing method of adjusting the input scaling factor. In order to avoid the error amplification by trace evaluation, the WWL+ method should be patched by using the first pre-processing method of multiplying $N^{-1} \bmod q$. This forces the WWL+ method to use NTT domains, limiting its functionality on various domains.

The second issue is raised from the scheme switching, proposed by De Micheli et al. [15] to efficiently convert a GLWE ciphertext into the corresponding GGSW ciphertext. Wang et al. assumed that the scheme switching error is linearly added [24], while we observe that the scheme switching process basically multiplies the variance of the input error roughly by N as mentioned in [15]. In order to avoid performance degradation of the PBS step due to the multiplication factor in the error growth, we provide correct and tight error analysis of the scheme switching step in the WWL+ method.

1.2 Our Contributions

In this paper, we patch the WWL+ circuit bootstrapping method by revising the error analysis and extend it by proposing a new pre-processing method to handle phase amplification of trace evaluation. In this way, our method works on FFT domains. We also propose a new method of high precision conversion to evaluate a function of multi-bit outputs using our patched WWL+ circuit bootstrapping, in contrast to the original WWL+ method which is targeted only for a function of single bit outputs.

Handling Phase Amplification on FFT Domains The main idea of our new pre-processing working on FFT domains is as follows. First, the input is divided by N using modulus switching. It takes negligible time compared to homomorphic trace evaluation, while it produces an additional error and reduces the ciphertext modulus from q to q/N. Then, the ciphertext modulus is recovered from q/N to q by modulus raising. Although the modulus raising introduces an unknown term $(q/N) \cdot u$, it vanishes by trace evaluation. See Figure 2 for the pictorial description of our pre-processing method on an FFT domain. Since our pre-processing divides both the scaling factor and the initial error by N, the phase amplification by N is canceled out only except for the modulus switching error, which is much smaller than the homomorphic trace evaluation error for practical parameters.

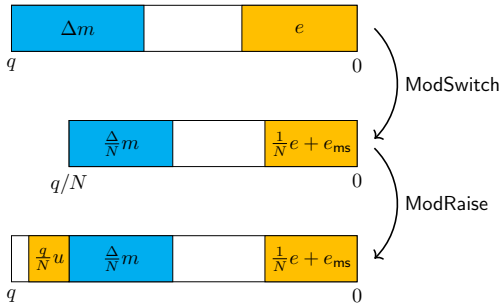


Fig. 2: New pre-processing method on an FFT domain.

Another practical issue on the automorphism-based conversion on an FFT domain is the amplification of the FFT error when the ciphertext modulus q is 2^{64}. Although our pre-processing mitigates the amplification of the input itself, the FFT error is still amplified by the homomorphic automorphism. According to the parameters, this amplified FFT error might be larger than the homomorphic trace evaluation error. We reduce the FFT error by splitting the automorphism key of 64-bit precision into two (or four) parts, using them in separate multiplications, and then combining the results. The above method is dubbed the *split FFT* method in this paper. We bound the failure probability of this method using the analysis of the FFT error variance proposed by Bergerat et al. [3].

Combining our new pre-processing and the split FFT, we implement the LWE to GLWE conversion by trace evaluation using the `tfhe-rs` library [26]. We implemented our result to compare it to the currently used conversion method based on packing keyswitching [13]. Our benchmark shows that computational cost (resp. transmission key size) for LWE to GLWE conversion is improved by factors of 13.25 (resp. 61.54), 20.29 (resp. 157.54) and 37.09 (resp. 728.5) for $N \in \{2048, 8192, 32768\}$.

Patching the WWL+ Method and Extending to FFT Domains As mentioned above, there are two flaws in the error analysis of the WWL+ method. A flaw from the phase amplification can be patched easily on NTT domains by empoying a proper pre-processing method proposed by Chen et al [7]. When it comes to FFT domains, our new pre-processing method mitigates the phase amplification, allowing the WWL+ method to work on both FFT and NTT domains. The other flaw is that multiplicative error growth has been ignored in scheme switching. We prove that scheme switching in the WWL+ method amplifies only the additive error of the LWE of GLWE conversion, not the PBS output error, obtaining correct and tight error analysis.

We implement the patched (CMux-based) WWL+ method on an FFT domain using `tfhe-rs` library [26] and apply it to improve WoP-PBS operation proposed by Bergerat et al. [3]. For certain sets of parameters, WoP-PBS requires high precision circuit bootstrapping to evaluate a function of a multi-bit output.

Hence, we proposed a high precision automorphism-based LWE to GLWE conversion method by combining the GLWE keyswitching, leading to high precision WWL+ circuit bootstrapping. To obtain benchmarks for the improvement, we use the recommended WoP-PBS parameters given in the `tfhe-rs` library. Then it turns out that the circuit bootstrapping time (resp. key size) is improved by factors from 3.26 to 7.22 (resp. 2.39 to 2.63) according to the parameters.

We also apply the patched WWL+ circuit bootstrapping to evaluate the AES cipher. The AES evaluation has been considered one of the practical applications of the original WWL+ method, while we observe a flaw in their selection of parameters in [24]. We implemented the AES evaluation with the patched WWL+ method under proper parameters, obtaining the benchmark result of 26.301 s in a single thread.

1.3 Paper Organization

In Section 2, we briefly review the TFHE scheme and automorphism-based operations. In Section 3, we describes our new automorphism-based LWE to GLWE conversion method without phase amplification and shows the benchmark from our implementation. Section 4 provides the patch of the WWL+ circuit bootstrapping method on an FFT domain and its applications.

2 Preliminaries

2.1 Notations

Throughout the paper, bold letters denote vectors (or matrices). The nearest integer to r is denoted $\lfloor r \rceil$. For real numbers a and b such that $a < b$, we write $[a, b[= \{x \in \mathbb{R} : a \leq x < b\}$. For two integers a and b, $\mathbb{Z} \cap [a, b[$ is denoted $[\![a, b[\![$. For an integer q, we identify $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ with $[\![-q/2, q/2[\![$, and $[\cdot]_q$ denotes the mod$q$ reduction into $\mathbb{Z}_q$.

For a polynomial $P(X) = p_0 + p_1 X + \cdots + p_{N-1} X^{N-1} \in \mathbb{Z}_q[X]$, its ℓ_1, ℓ_2 and ℓ_∞ norms are defined as follows.

$$\begin{aligned} \ell_1(P) &= |p_0| + |p_1| + \cdots + |p_{N-1}|, \\ \ell_2(P) &= \sqrt{|p_0|^2 + |p_1|^2 + \cdots + |p_{N-1}|^2}, \\ \ell_\infty(P) &= \max_{0 \leq i \leq N-1} |p_i|. \end{aligned}$$

The set $\mathbb{B}$ and $[n]$ denote $\{0, 1\}$ and $\{1, 2, \ldots, n\}$, respectively, for a positive integer n. The set $\mathbb{N}$ denotes the set of all positive integers. The set $\mathbb{Z}_q^\times$ denotes the multiplicative subgroup of $\mathbb{Z}_q$.

For a set S, we will write $a \leftarrow S$ to denote that a is chosen from S uniformly at random. For a probability distribution $\mathcal{D}$, $a \leftarrow \mathcal{D}$ denotes that a is sampled according to the distribution $\mathcal{D}$. Unless stated otherwise, all logarithms are to the base 2.

In the context of TFHE, we use p and q to denote the moduli of messages and ciphertexts, respectively. For a power-of-two N, the cyclotomic ring $\mathbb{Z}[X]/(X^N+1)$ is denoted $\mathbb{Z}_N[X]$. We also write $\mathcal{R}_{q,N} = \mathbb{Z}_q[X]/(X^N+1)$ and $\mathbb{B}_N[X] = \mathbb{B}[X]/(X^N+1)$.

The noise variance of an error polynomial $E(X) \in \mathcal{R}_{q,N}$ is denoted by its ℓ_∞-norm; $\mathrm{Var}(E(X)) \leq \sigma^2$ if and only if all coefficients of $E(X)$ have variances at most σ^2. For simplicity, we say that an error $E(X)$ has a variance σ^2 when it has a variance at most σ^2.

2.2 TFHE

In this section, we briefly review the core concepts of the TFHE scheme. Although TFHE itself is mathematically defined over the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ [10], it is common to use the discretized torus $\frac{1}{q}\mathbb{Z}/\mathbb{Z}$ for $q \in \{2^{32}, 2^{64}\}$ considering its implementation. Then we can identify the discretized torus $\frac{1}{q}\mathbb{Z}/\mathbb{Z}$ with $\mathbb{Z}_q$, which is commonly used in the recent descriptions of TFHE [12, 13, 3]. In this paper, most of the notations for TFHE follow those in [13].

LWE, RLWE, and GLWE Ciphertexts Under a secret key $\mathbf{S} \in \mathcal{R}_{q,N}^k$, a message $M \in \mathcal{R}_{p,N}$ is encrypted into a generalized LWE (GLWE) ciphertext $\mathbf{C} \in \mathcal{R}_{q,N}^{k+1}$ with a scaling factor Δ such that $\Delta \leq q/p$ as follows [5].

$$\mathbf{C} = \mathrm{GLWE}_{q,\mathbf{S}}(\Delta \cdot M) = (A_1, \ldots, A_k, B = \sum_{i=1}^{k} A_i \cdot S_i + [M \cdot \Delta]_q + E)$$

where $\mathbf{S} = (S_1, \ldots, S_k)$, $A_i \leftarrow \mathcal{R}_{q,N}$ for $i = 1, 2, \ldots, k$, and $E \leftarrow \chi_\sigma$ for some Gaussian distribution χ_σ as the error distribution. $(A_1, \ldots, A_k)$ and B are called the mask and the body of the GLWE ciphertext $\mathbf{C}$, respectively, and k is called the GLWE dimension. It is common to use the binary secret key in the TFHE scheme, so we only deal with the binary secret key in this paper. Some of the subscripts $q, \mathbf{S}$ might be omitted when they are clear from the context.

A GLWE ciphertext with $N = 1$ is called an LWE ciphertext. In this case, it is common to use n to denote the LWE dimension instead of k, so that an LWE ciphertext is usually denoted $(a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$. When $k = 1$, a GLWE ciphertext is called a ring LWE (RLWE) ciphertext. In this paper, we distinguish LWE ciphertexts from GLWE ciphertexts of $N > 1$.

The decryption of a GLWE ciphertext is to compute its *phase*, which is defined as $B - \langle (A_1, \ldots, A_k), \mathbf{S} \rangle$, followed by rounding the phase by the scaling factor Δ. The decryption works correctly if the error contained in the ciphertext is small enough to be eliminated during the rounding by Δ.

From the definition of the GLWE ciphertext, the sum of two GLWE ciphertexts under the same secret key results in the sum of their internal plaintexts in $\mathcal{R}_{p,N}$. Multiplying the ciphertext by a scalar is possible by iterating addition several times. Both the addition and the scalar multiplication increase the error of the resulting ciphertext linearly.

Lev and GLev Ciphertexts Let $B \in \mathbb{N}$ be a power-of-two and $\ell \in \mathbb{N}$. A GLev ciphertext of $\overline{\mathbf{C}} \in \mathcal{R}_{q,N}^{(k+1)\ell}$ of $M \in \mathcal{R}_{q,N}$ with a decomposition base B and a decomposition level ℓ under a GLWE secret key $\mathbf{S} \in \mathbb{B}_N[X]^k$ is defined as a vector of ℓ GLWE ciphertexts of $M \in \mathcal{R}_{q,N}$ with scaling factors q/B^j for $j = 1, \ldots, \ell$ as follows.

$$\overline{\mathbf{C}} = \mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(M) = \left(\mathrm{GLWE}_{\mathbf{S}}\left(\frac{q}{B^j} \cdot M\right)\right)_{j \in [\ell]}.$$

When $N = 1$, it is called a Lev ciphertext.

GGSW Ciphertexts In the case of nonlinear operations such as multiplication, TFHE uses another type of ciphertext called generalized GSW (GGSW) [17]. Let $B \in \mathbb{N}$ be a power-of-two and $\ell \in \mathbb{N}$. A GGSW ciphertext $\overline{\overline{\mathbf{C}}} \in \mathcal{R}_{q,N}^{(k+1)\ell \times (k+1)}$ of a message $M \in \mathcal{R}_{q,N}$ with a decomposition base B and a decomposition level ℓ under a secret key $\mathbf{S} \in \mathbb{B}_N[X]^k$ is an $(k+1)\ell \times (k+1)$ matrix over $\mathcal{R}_{q,N}$ defined as follows.

$$\overline{\overline{\mathbf{C}}} = \mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M) = \left(\mathrm{GLWE}_{\mathbf{S}}\left(\frac{q}{B^j}(-S_i \cdot M)\right)\right)_{(i,j) \in [k+1] \times [\ell]}$$

where $\mathbf{S} = (S_1, \ldots, S_k)$, $S_{k+1} = -1$, and each GLWE ciphertext is considered a row having $k+1$ columns of polynomials in $\mathcal{R}_{q,N}$. One can also represent $\overline{\overline{\mathbf{C}}}$ as a vector of $k+1$ GLev ciphertexts $(\mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(-S_i \cdot M))_{i=1}^{k+1}$.

Gadget Decomposition Let $B \in \mathbb{N}$ be a power-of-two and $\ell \in \mathbb{N}$. The gadget decomposition $\mathrm{GadgetDecomp}^{(B,\ell)}$ with a base B and a level ℓ decomposes an input $a \in \mathbb{Z}_q$ into a vector $(a_1, \ldots, a_\ell) \in \mathbb{Z}_q^\ell$ such that

$$a = \sum_{j=1}^{\ell} a_j \cdot \frac{q}{B^j} + e$$

where $a_j \in [\![-B/2, B/2[\![$ for all $j = 1, \ldots, \ell$ and the decomposition error e satisfies $|e| \leq \frac{q}{2B^\ell}$.

The gadget decomposition can be extended to a polynomial in $\mathcal{R}_q$ (or $\mathcal{R}_{q,N}$) by applying the decomposition to its coefficients. When it is applied to a vector of polynomials, it outputs a vector of decomposition vectors of the input polynomials.

External Product and CMux Gate The external product $\boxdot$ between a GGSW ciphertext $\overline{\overline{\mathbf{C}}}_1$ and a GLWE ciphertext $\mathbf{C}_2$ is defined as

$$\overline{\overline{\mathbf{C}}}_1 \boxdot \mathbf{C}_2 = \mathrm{GadgetDecomp}^{(B,\ell)}(\mathbf{C}_2) \cdot \overline{\overline{\mathbf{C}}}_1$$

where (B, ℓ) is the decomposition parameter of $\overline{\overline{\mathbf{C}}}_1$ and $\mathrm{GadgetDecomp}^{(B,\ell)}$ is the gadget decomposition with a base B and a level ℓ.

The external product between GGSW and GLWE ciphertexts defines homomorphic module scalar multiplication on the discretized torus $\frac{1}{q}\mathbb{Z}/\mathbb{Z}$. Roughly speaking, the external product increases the error by the magnitude of the plaintext in the GGSW ciphertext. Thus it is common to use GGSW ciphertexts encrypting a single bit of message in the external product.

The controlled mux gate, dubbed CMux, is the key operation used in TFHE. Suppose that two GLWE ciphertexts $\mathbf{C}_0$ and $\mathbf{C}_1$ are given along with a secret boolean value b encrypted to a GGSW ciphertext $\overline{\overline{\mathbf{C}}}$, where all three ciphertexts are encrypted with the same key $\mathbf{S}$. Then one may select $\mathbf{C}_b$ without knowing b by

$$\mathrm{CMux}(\overline{\overline{\mathbf{C}}}, \mathbf{C}_0, \mathbf{C}_1) = (\mathbf{C}_1 - \mathbf{C}_0) \boxdot \overline{\overline{\mathbf{C}}} + \mathbf{C}_0.$$

Programmable Bootstrapping The programmable bootstrapping (PBS) of TFHE supports an extra functionality that evaluates a function for free during the bootstrapping. Suppose that an LWE ciphertext $\mathbf{c} = (a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$ of a phase $\mu = \Delta m + e$ under a secret key $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{B}^n$ is given. The PBS operation outputs a refreshed LWE ciphertext $\mathbf{c}' \in \mathbb{Z}_q^{kN}$ of the message $f(m)$ under a secret key $\mathbf{s}' \in \mathbb{B}^{kN}$ by the following steps.

1. Encode the function f on a new GLWE ciphertext under a different secret key $\mathbf{S}' \in \mathbb{B}_N[X]^k$. The half of the function values of f are redundantly encoded in the coefficients of the plaintext of the (trivial) GLWE ciphertext.
2. (Modulus switching) Compute $\tilde{\mathbf{c}} = (\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in \mathbb{Z}_{2N}^{n+1}$ where
$$\tilde{a}_i = \lfloor a_i \cdot (2N)/q \rceil \text{ and } \tilde{b} = \lfloor b \cdot (2N)/q \rceil,$$
obtaining an LWE ciphertext of a phase $\tilde{\mu} \approx \lfloor \mu \cdot (2N)/q \rceil$.
3. (Blind rotation) Multiply $X^{-\tilde{b}+\sum_{i=1}^n \tilde{a}_i s_i} = X^{-\tilde{\mu}}$ to the GLWE ciphertext encoding the function using a bootstrapping key $\{\mathrm{GGSW}_{\mathbf{S}'}(s_i)\}_{i=1}^n$; multiply either 1 or $X^{-\tilde{a}_i}$ according to $s_i \in \{0, 1\}$ by the CMux gate.
4. (Sample extraction) Extract the constant term of the GLWE ciphertext, obtaining an LWE ciphertext of $f(m)$ under the secret key $\mathbf{s}' \in \mathbb{B}^{kN}$ which is a reordering of the coefficients of $\mathbf{S}'$.

Since $X^N = -1$ in the ring $\mathcal{R}_{q,N}$, it is only possible to evaluate a negacyclic function $f : \mathbb{Z}_p \to \mathbb{Z}_q$ such that $f(x + p/2) = -f(x)$ by encoding only half of the function values. To evaluate an arbitrary function, TFHE requires one padding bit of zero in the MSB of μ to guarantee $\tilde{\mu} < N$.

LWE Keyswitching The input and output LWE dimensions might be different for the PBS operation. To improve the performance of PBS, it is common to use a smaller input LWE dimension than the output LWE dimension. Hence, one needs to switch the LWE dimension before the PBS operation, and this step is called the *keyswitching*. In practice, the keyswitching operation is performed only once just before the PBS operation to match the LWE dimension rather than after every PBS operation.

Plaintext Encoding in TFHE To keep the padding bit zero, Bergerat et al. [3] proposed a new encoding method for TFHE splitting the traditional plaintext space into three parts: one (or more) bit of padding at the MSB, the carry subspace after the padding bits, and the message subspace in the remaining bits. By tracking the maximum possible value in the ciphertext, it clears the carry space before the padding bit is filled. For example, one of the recommended parameters of `tfhe-rs` library for `shortint` type, called `PARAM_MESSAGE_2_CARRY_2_KS_PBS`, uses the encoding that consists of two message bits, two carry bits, and one padding bit.

2.3 Circuit Bootstrapping

The circuit bootstrapping is a bootstrapping process that converts an LWE ciphertext of a single bit into the corresponding GGSW ciphertext [10]. In this paper, we describe circuit bootstrapping in two steps: LWE to Lev conversion and Lev to GGSW conversion.

The first step is LWE to Lev conversion by PBS. Given an LWE ciphertext $\mathrm{LWE}_{\mathbf{s}}(\Delta m)$ of a single bit message m with some scaling factor Δ, one can compute $\mathrm{Lev}_{\mathbf{s}}^{(B,\ell)}(m)$ by gathering its internal LWE ciphertexts $\mathrm{LWE}_{\mathbf{s}}(q/B^j \cdot m)$ for $j = 1, \ldots, \ell$ using PBS. Since it computes ℓ PBS operations on the same LWE input, PBSmanyLUT proposed by Chillotti et al. [13] can improve this step without increasing the PBS error.

The next step is Lev to GGSW conversion by private functional keyswitching. For a GLWE secret key $\mathbf{S} = (S_1, \ldots, S_k)$, the private functional keyswitching operation converts $\mathrm{LWE}_{\mathbf{s}}(q/B^j \cdot m)$ contained in $\mathrm{Lev}_{\mathbf{s}}^{(B,\ell)}(m)$ into $\mathrm{GLWE}_{\mathbf{S}}(q/B^j(-S_i \cdot m))$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$ where $S_{k+1} = -1$ for convenience, obtaining GLev ciphertexts $\{\mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(-S_i \cdot m)\}_{i=1}^{k+1}$. These are the internal GLev ciphertexts of the GGSW ciphertext $\mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(m)$ of m, so the Lev ciphertext can be converted into the GGSW ciphertext using $k+1$ private functional keyswitchings.[4] We refer to Appendix A for a brief overview of the functional keyswitching.

2.4 Automorphism and Trace

Let $K = \mathbb{Q}[X]/(X^N + 1)$ be the number field where N is a power-of-two. Since K is a Galois extension of $\mathbb{Q}$, its Galois group $\mathrm{Gal}(K/\mathbb{Q})$ consists of the automorphisms $\tau_d : \mu(X) \mapsto \mu(X^d)$ for $d \in \mathbb{Z}_{2N}^{\times}$. Then the field trace $\mathrm{Tr}_{K/\mathbb{Q}} : K \to \mathbb{Q}$, defined by

$$\mathrm{Tr}_{K/\mathbb{Q}}(\mu(X)) = \sum_{\sigma \in \mathrm{Gal}(K/\mathbb{Q})} \sigma(\mu(X))$$

satisfies the following equation.

$$\mathrm{Tr}_{K/\mathbb{Q}}(\mu(X)) = N\mu_0$$

[4] To be precise, it requires k private functional keyswitchings and a single public functional keyswitching since $S_{k+1} = -1$.

where $\mu(X) = \mu_0 + \mu_1 X + \cdots + \mu_{N-1} X^{N-1}$.

The automorphism and trace can be defined analogously on the ring of integer $\mathcal{R}_N = \mathbb{Z}[X]/(X^N + 1)$ and its residue ring $\mathcal{R}_{q,N} = \mathcal{R}_N / qR_N$ modulo q. For $d \in \mathbb{Z}_{2N}^\times$, the automorphism τ_d on $\mathcal{R}_N$ (or $\mathcal{R}_{q,N}$) is defined by $\tau_d : \mu(X) \mapsto \mu(X^d)$, and the trace function Tr on $\mathcal{R}_N$ (or $\mathcal{R}_{q,N}$) is defined by

$$\mathrm{Tr}(\mu(X)) := \sum_{d \in \mathbb{Z}_{2N}^\times} \tau_d(\mu(X)) = N\mu_0.$$

Computing the trace by its definition requires one to compute the automorphism N times. For efficient homomorphic trace evaluation, Chen et al. [7] proposed a recursive algorithm as follows: let $K_n = \mathbb{Q}[X]/(X^n + 1)$ be the $2n$-th cyclotomic field for a power-of-two n. Then the field extension $K \geq \mathbb{Q}$ can be described as a tower of fields $K = K_N \geq K_{N/2} \geq \cdots \geq K_1 = \mathbb{Q}$. For $1 \leq i < j \leq \log N$, the trace $\mathrm{Tr}_{K_{2^j}/K_{2^i}}$ can be expressed as a composition

$$\mathrm{Tr}_{K_{2^j}/K_{2^i}} = \mathrm{Tr}_{K_{2^j}/K_{2^{j-1}}} \circ \cdots \circ \mathrm{Tr}_{K_{2^{i+1}}/K_{2^i}}.$$

Since $\mathrm{Gal}(K_{2^k}/K_{2^{k-1}}) = \{\tau_1, \tau_{2^k+1}\}$ for all $k = 1, \ldots, \log N$, computing $\mathrm{Tr}_{K_{2^j}/K_{2^i}}$ using the above composition requires only $j - i$ automorphisms, where K_n is identified with

$$\{a_0 + a_1 X^{\frac{N}{n}} + \cdots + a_{n-1} X^{N - \frac{N}{n}} : a_0, \ldots, a_{n-1} \in \mathbb{Q}\} \subseteq K_N.$$

As an analogue, let $\mathrm{Tr}_{N/n}$ be the trace on $\mathcal{R}_{q,N}/\mathcal{R}_{q,n}$ where n and N are power-of-two such that $n \mid N$. Then, $\mathrm{Tr}_{N/n} : R_{q,N} \to R_{q,n}$ satisfies the following equation.

$$\begin{aligned} \mathrm{Tr}_{N/n}(\mu(X)) &= \mathrm{Tr}_{N/(N/2)} \circ \cdots \circ \mathrm{Tr}_{2n/n}(\mu(X)) \\ &= \frac{N}{n}(\mu_0 + \mu_{\frac{N}{n}} X^{\frac{N}{n}} + \cdots + \mu_{N-\frac{N}{n}} X^{N-\frac{N}{n}}) \end{aligned} \tag{1}$$

where $\mathcal{R}_{q,n}$ is identified with

$$\{a_0 + a_1 X^{\frac{N}{n}} + \cdots + a_{n-1} X^{N - \frac{N}{n}} : a_0, \ldots, a_{n-1} \in \mathbb{Z}_q\} \subseteq \mathcal{R}_{q,N}.$$

Using the above relation, one can compute $\mathrm{Tr} = \mathrm{Tr}_{N/1}$ on $\mathcal{R}_N$ (or $\mathcal{R}_{q,N}$) by only $\log N$ automorphisms.

The number of automorphisms for the trace evaluation is important since the trace function is evaluated by a series of homomorphic automorphisms based on GLWE keyswitching. For $d \in \mathbb{Z}_{2N}^\times$, the automorphism τ_d maps $M(X)$ into $M(X^d)$. Given a GLWE secret key $\mathbf{S}(X) \in \mathcal{R}_{q,N}^k$, a GLWE ciphertext $\mathrm{GLWE}_{\mathbf{S}(X)}(M(X))$ of $M(X)$ under $\mathbf{S}(X)$ can be regarded as one $\mathrm{GLWE}_{\mathbf{S}(X^d)}(M(X^d))$ of $M(X^d)$ under $\mathbf{S}(X^d)$. By switching the key of $\mathrm{GLWE}_{\mathbf{S}(X^d)}(M(X^d))$ from $\mathbf{S}(X^d)$ to $\mathbf{S}(X)$, one can obtain the GLWE ciphertext of $M(X^d)$ under the original secret key $\mathbf{S}(X)$. We refer to Appendix C.2 for the details of GLWE keyswitching.

Algorithm 1 describes the algorithm evaluating homomorphic automorphism, and Algorithm 2 describes the algorithm evaluating homomorphic trace using (1).

Algorithm 1: Evaluating Automorphism $\mathsf{EvalAuto}(\mathbf{C}, d)$

Input: $\mathbf{C} = \mathrm{GLWE}_{\mathbf{S}(X)}(M(X))$, $d \in \mathbb{Z}_{2N}^{\times}$
Input: $\mathrm{AutoKey}_d = \mathrm{KS}_{\mathbf{S}(X^d)\to\mathbf{S}(X)}$ with decomposition base B and level ℓ
Output: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}(X)}(M(X^d))$

$\mathbf{C} = (A_1, \ldots, A_k, B)$
$\mathbf{C}' \leftarrow (A_1', \ldots, A_k', B') = (A_1(X^d), \ldots, A_k(X^d), B(X^d))$
$\mathbf{C}' \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}', \mathrm{AutoKey}_d)$
return $\mathbf{C}'$

Algorithm 2: Evaluating Trace $\mathsf{EvalTr}_{N/n}(\mathbf{C})$

Input: $\mathbf{C} = \mathrm{GLWE}_{\mathbf{S}(X)}(M(X))$ where
$M(X) = m_0 + m_1 X + \cdots + m_{N-1} X^{N-1}$
Input: $\mathrm{AutoKey}_d = \mathrm{KS}_{\mathbf{S}(X^d)\to\mathbf{S}(X)}$ for all $d \in \mathbb{Z}_{2N}^{\times}$
Output: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}(X)}\left(\frac{N}{n}\sum_{j=0}^{n-1} m_{j\cdot\frac{N}{n}} X^{j\cdot\frac{N}{n}}\right)$

$\mathbf{C}' \leftarrow \mathbf{C}$
for $d = 1$ *to* $\log(N/n)$ **do**
  $\mathbf{C}' \leftarrow \mathbf{C}' + \mathsf{EvalAuto}(\mathbf{C}', 2^{\log N - d + 1} + 1)$
return $\mathbf{C}'$

3 Automorphism-based Conversion without Phase Amplification on FFT Domains

3.1 Conversion on NTT Domains

Chen et al. [7] proposed efficient LWE to GLWE conversion using homomorphic automorphism on GLWE ciphertexts.[5] The key idea of their methods is to clear unnecessary coefficients of the GLWE ciphertext by homomorphic trace evaluation. Employing automorphism-based conversion, the time complexity of the LWE to GLWE (resp. n LWEs to GLWE) conversion is reduced by a factor of $N/\log^2 N$ (resp. $N/(n + \log(N/n))$), and the key size is reduced by a factor of $N/\log N$ compared to the packing keyswitching method based on public functional keyswitching [13].

Suppose that an LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^{kN+1}$ of phase μ under a secret key $\mathbf{s} = (s_1, \ldots, s_{kN})$ is given where k is a positive integer and N is a power-of-two. Let $\mathbf{S} = (S_1, \ldots, S_k)$ be the GLWE secret key corresponding to $\mathbf{s}$ such that

$$S_i = \sum_{j=0}^{N-1} s_{(i-1)N+j} X^j \tag{2}$$

for $i \in 1, \ldots, k$. Then one can find a GLWE ciphertext $(\mathbf{A}, B) \in R_{q,N}^{k+1}$ whose phase under the secret key $\mathbf{S}$ has the constant term μ by reordering $\mathbf{a}$ into the

[5] They only considered the RLWE case of $k = 1$, but it can be generalized to any GLWE dimension.

coefficients of $\mathbf{A}$ properly, i.e.,

$$B - \langle \mathbf{A}, \mathbf{S} \rangle = (\mu + v_1 X + v_2 X^2 + \cdots + v_{N-1} X^{N-1})$$

for some $v_1, \ldots, v_{N-1} \in \mathbb{Z}_q$. The above LWE to GLWE conversion that is only valid on the constant term is denoted

$$(\mathbf{A}, B) \leftarrow \mathsf{LWEtoGLWEConst}((\mathbf{a}, b))$$

in this paper. To annihilate the unintended coefficients $v_1, \ldots, v_{N-1}$, one can homomorphically evaluate its trace, obtaining a GLWE ciphertext of phase $N\mu + E_{\mathsf{tr}}(X)$ under the secret key $\mathbf{S}$ where $E_{\mathsf{tr}}(X)$ is the error induced by homomorphic trace evaluation.

The homomorphic trace evaluation not only removes the unnecessary coefficients but also multiplies the constant term of its input phase by N. To mitigate this phase amplification, multiplying the input LWE ciphertext by $N^{-1} \bmod q$ is needed as pre-processing. Then, using Algorithm 2 with $n = 1$, one can convert an LWE ciphertext into a GLWE ciphertext at the cost of $\log N$ homomorphic automorphism evaluations. We note that this pre-processing is based on the assumption that the ciphertext modulus q is relatively prime to N, which holds on an NTT domain.

3.2 LWE to GLWE Conversion on FFT Domains with a New Pre-processing Method

The automorphism-based conversion methods are advantageous in terms of both time complexity and key size. The issue of phase amplification by N can be handled by multiplying $N^{-1} \bmod q$ when N and q are relatively prime, which holds on an NTT domain. However, on an FFT domain with a power-of-two ciphertext modulus q, one cannot simply cancel out the phase amplification in the same way since q is a multiple of N.

One possible solution is to set the scaling factor of the input ciphertext to Δ/N in order to obtain the desired scaling factor Δ after the trace evaluation. For example, [6] and [14] use the automorphism-based GLWE expansion that extracts $\mathrm{GLWE}(Nm_i)$ from $\mathrm{GLWE}(m_0 + m_1 X + \cdots + m_{N-1} X^{N-1})$ for $i = 0, 1, \ldots, N-1$. Using the scaling factor of Δ/N for the input GLWE ciphertext, the output can be regarded as a GLWE ciphertext with the desired scaling factor of Δ.

The problem with the above approach is that the error inside the input phase is still amplified by N (see Figure 1b). This error amplification is not significant when the input error is small enough, in particular, when the input is a fresh ciphertext [6, 14]. However, if one wants to apply this conversion to the output of PBS, the error amplification requires the PBS error to be at least N times smaller, making the PBS operation slower. This is an undesirable trade-off since PBS operation is heavier than the LWE to GLWE conversion in most cases.

New Pre-processing Method on FFT Domains To overcome this limitation on an FFT domain, we propose a new pre-processing method using modulus switching and modulus raising (see Figure 2). Let $\mathbf{c} = \mathrm{LWE}_{q,\mathbf{s}}(\Delta m)$ be an LWE ciphertext of phase $\mu = \Delta m + e$ modulo q under an LWE secret key $\mathbf{s} \in \mathbb{B}^{kN}$ where q and N are powers of two. Let $\mathbf{S} \in \mathbb{B}_N[X]^k$ be the GLWE secret key corresponding to $\mathbf{s}$. The modulus switching of $\mathbf{c}$ from q to q/N divides its phase by N at the cost of additional modulus switching error e_{ms}, obtaining an LWE ciphertext $\mathbf{c}' = \mathrm{LWE}_{\frac{q}{N},\mathbf{s}}(\frac{\Delta}{N}m)$ of phase $\mu' = \frac{1}{N}\mu + e_{\mathsf{ms}}$ modulo $\frac{q}{N}$. As the input phase is divided by N, one can cancel out the phase amplification by trace evaluation.

However, the modulus switching consumes the ciphertext modulus, reducing it from q to q/N. To recover the ciphertext modulus, we use the modulus raising from q/N to q. Let $\mathbf{c}' = (a_1, \ldots, a_n, b) \in \mathbb{Z}_{q/N}^{n+1}$. Then,

$$b - \langle (a_1, \ldots, a_n), \mathbf{s} \rangle = \mu' + \frac{q}{N} \cdot u \tag{3}$$

for some $u \in \mathbb{Z}$ since the phase of $\mathbf{c}'$ is μ' modulo $\frac{q}{N}$ under the secret key $\mathbf{s}$. The modulus raising interprets each component of $\mathbf{c}'$ in $\mathbb{Z}_{q/N}$ as an element of $\mathbb{Z}_q$ of the same value, obtaining an LWE ciphertext $\mathbf{c}'' = (a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$ of phase $\mu' + \frac{q}{N} \cdot u$ modulo q by (3).

After the above modulus switching and modulus raising, the input LWE ciphertext of phase μ is changed to the LWE ciphertext of phase $\mu'' = \frac{1}{N}\mu + e_{\mathsf{ms}} + \frac{q}{N} \cdot u$ under the same LWE secret key and modulus q. Although the value of u is unknown, the term $\frac{q}{N} \cdot u$ will vanish by trace evaluation that multiplies it by N modulo q.

This pre-processing takes negligible time compared to the homomorphic trace evaluation, so the overall computational overhead is also negligible. In terms of error, an additional error appears by the modulus switching and is amplified by N, while it is still small enough compared to the homomorphic trace evaluation error for practical parameters. We refer to Appendix C.1 for the details.

LWE to GLWE Conversion on the FFT Domain The pre-processing based on modulus switching and modulus raising returns the LWE ciphertext $\mathbf{c}''$ of phase $\mu'' = \frac{1}{N}\mu + e_{\mathsf{ms}} + \frac{q}{N} \cdot u$. Then, the homomorphic trace evaluation after $\mathsf{LWEtoGLWEConst}$ outputs a GLWE ciphertext of a phase

$$N\mu'' + E_{\mathsf{tr}}(X) = \mu + Ne_{\mathsf{ms}} + E_{\mathsf{tr}}(X) \tag{4}$$

under the corresponding GLWE secret key $\mathbf{S}$ where $E_{\mathsf{tr}}(X)$ is the error of the homomorphic trace evaluation. The exact algorithm is described in Algorithm 3.

Theorem 1. *Let* $\mathbf{c}$ *be an LWE ciphertext of phase* μ *under a secret key* $\mathbf{s} = (s_1, \ldots, s_{kN})$ *where the ciphertext modulus* q *is a power-of-two. Then, Algorithm 3 returns a GLWE ciphertext* $\mathbf{C}$ *of phase* $\mu + E_{\mathsf{conv}}(X)$ *under the GLWE secret key* $\mathbf{S} = (S_1, \ldots, S_k)$ *corresponding to* $\mathbf{s}$ *where the variance* V_{conv} *of* $E_{\mathsf{conv}}(X)$

Algorithm 3: Automorphism-based LWE to GLWE Conversion on an FFT Domain

Input: $\mathbf{c} = \mathrm{LWE}_{q,\mathbf{s}}(m)$ where q is a power-of-two and $\mathbf{s} = (s_1, \ldots, s_{kN})$
Input: Automorphism keys
Output: $\mathbf{C} = \mathrm{GLWE}_{q,\mathbf{S}}(m)$ where $\mathbf{S} = (S_1, \ldots, S_k)$ is the GLWE secret key corresponding to $\mathbf{s}$

$\mathbf{c} \leftarrow \mathsf{ModSwitch}_{q \to q/N}(\mathbf{c})$
$\mathbf{c} \leftarrow \mathsf{ModRaise}_{q/N \to q}(\mathbf{c})$
$\mathbf{C} \leftarrow \mathsf{LWEtoGLWEConst}(\mathbf{c})$
$\mathbf{C} \leftarrow \mathsf{EvalTr}_{N/1}(\mathbf{C})$
return $\mathbf{C}$

is given as follows.

$$\begin{aligned} V_{\mathsf{conv}} &= N^2 V_{\mathsf{ms}} + V_{\mathsf{tr}} \\ &\le N^2 V_{\mathsf{ms}} + \frac{N^2-1}{3} V_{\mathsf{auto}} \\ &\le \frac{(kN+1)N^2}{12} + \frac{N^2-1}{3}\left(\frac{kN}{12}\left(\frac{q^2}{B_{\mathsf{auto}}^{2\ell_{\mathsf{auto}}}} - 1\right) + k\ell_{\mathsf{auto}} N \left(\frac{B_{\mathsf{auto}}}{2}\right)^2 \sigma_{\mathsf{ak}}^2\right) \end{aligned}$$

where σ_{ak}^2 is the noise variance of the automorphism key with the gadget decomposition parameter $(B_{\mathsf{auto}}, \ell_{\mathsf{auto}})$.

Proof. One can directly derive the equation $V_{\mathsf{conv}} = N^2 V_{\mathsf{ms}} + V_{\mathsf{tr}}$ from (4) with the independence of e_{ms} and $E_{\mathsf{tr}}(X)$. The inequalities on V_{ms}, V_{auto} and V_{tr} come from Lemma 1, 2, and 3 respectively. □

3.3 Handling FFT Error for Polynomial Multiplication

Theorem 1 does not consider the error caused by the FFT-based polynomial multiplication. Most error analyses in TFHE on FFT domains, especially for PBS, do not deal with such FFT errors since the FFT error is much smaller than the PBS output error for practical cases. On the other hand, one cannot ignore the impact of the FFT error in the automorphism-based conversion since it is amplified by N during trace evaluation, resulting in a larger error than the conversion error.

To reduce the FFT errors and perform the conversion with high precision, we split a polynomial of 64-bit precision into two parts. Let $F \in \mathcal{R}_{2^{64},N}$ such that $\|F\|_\infty \le B/2$ and $G \in \mathcal{R}_{2^{64},N}$. Then one can represent G as

$$G = G_0 + G_1 \cdot 2^b$$

where the coefficients of G_0 (resp. G_1) are all contained in $[\![0, 2^b[\![$ (resp. $[\![0, 2^{64-b}[\![$) and $b \in [\![0, 64[\![$. Splitting the multiplier G decomposes the polynomial multiplication $F \cdot G$ into two polynomial multiplications with smaller multipliers as

follows.

$$F \cdot G = (F \cdot G_0) + 2^b \cdot (F \cdot G_1)\,.$$

If the multiplication $F \cdot G_1$ whose result is scaled by 2^b can be computed exactly by FFT, then one can compute $F \cdot G$ with a smaller FFT error at the cost of two FFT multiplications. We call this strategy to compute polynomial multiplication by FFT with a smaller error as *split FFT*.

Using a larger b results in a larger FFT error for $F \cdot G_0$ and a smaller FFT error for $F \cdot G_1$, so one need to choose a proper b such that $F \cdot G_1$ can be computed exactly with a negligible failure probability and $F \cdot G_0$ has as small FFT error as possible. To find such b, we might use some previous works on the variance of the FFT error.

Klemsa [20] proposed an upper bound for the FFT error of (negacyclic) polynomial multiplication as follows. Let $\|F\|_\infty \leq B_1$, $\|G\|_\infty \leq B_2$ and χ be the bit-precision of the floating point representation, which is 53 for double-precision. Let $E_{\mathsf{fft}}(X) \in \mathbb{R}[X]/(X^N+1)$ be the error of the output of $F \cdot G$ computed by FFT before rounding. Then Proposition 1 in [20] gives the following upper bounds.[6]

$$\begin{aligned}\log \|E_{\mathsf{fft}}\|_\infty &\leq (2\log N - 4) \cdot \log(\sqrt{2}+1) + \log B_1 + \log B_2 - \chi + 9/2 + \log 3,\\ \log \mathrm{Var}(E_{\mathsf{fft}}) &\leq 4\log N + 2\log B_1 + 2\log B_2 - 2\chi - 3.\end{aligned}$$

However, the experimental result shows that the above theoretical bound is a loose upper bound to be used in practice (see Section 5.3 in [20]). The gap between the bound proposed by Klemsa and the practical result seems to come from the gap between the worst-case and average-case analyses; most TFHE parameters are chosen based on the average-case analysis using the independence heuristic [10, 13].

Later, Bergerat et al. [3] proposed a tighter estimation for the FFT error in the PBS operation by deducing the formula using data from various parameter settings. They provided the formula for the variance of FFT error in PBS as $2^{-2\chi-2.6} \cdot n\ell q^2 B^2 N^2 (k+1)$ where (B, ℓ) is the gadget decomposition parameters for PBS and k is the GLWE dimension. Since the FFT error variance of PBS is n times of the FFT error variance of external product, we obtain the formula for the FFT error of a single polynomial multiplication $F \cdot G$ for $\|F\|_\infty \leq B_1$ and $\|G\|_\infty \leq B_2$ as $2^{-2\chi-2.6} B_1^2 B_2^2 N^2$. When it comes to the split FFT, the FFT error variance of $F \cdot G_0$ (resp. $F \cdot G_1$) is given by

$$2^{2(b-\chi)-2.6} \ell B^2 N^2 (k+1) \quad (\text{resp. } 2^{2(64-b-\chi)-2.6} \ell B^2 N^2 (k+1)). \tag{5}$$

Using (5), one can find a proper b to guarantee the exact computation of $F \cdot G_1$ with a negligible failure probability.

Defining the failure probability of the split FFT as that of the exact computation of $F \cdot G_1$, the value of b for each parameter set used in this paper is chosen to obtain the failure probability of the split FFT smaller than about 2^{-2000}, enabling one to ignore the failure probability of the split FFT compared to that of

[6] The second-order terms are neglected.

PBS. We perform an experiment to verify that (5) can tightly bound the FFT error variance. On the other hand, there might be an FFT error for $F \cdot G_0$. For the parameters used in this paper (such that $N \leq 8192$), this FFT error is much smaller than additive error of the GLWE keyswitching, so we neglect it in the error analysis. We refer to Appendix B for the details.

To reduce the FFT error further, one can split the multiplier G into more than two parts. For example, G can be split into 4 parts as follows.

$$G = G_0 + G_1 \cdot 2^{16} + G_2 \cdot 2^{32} + G_3 \cdot 2^{48}$$

where the coefficients of G_0, G_1, G_2 and G_3 are all contained in $[\![0, 2^{16}[\![$. The above split FFT method is dubbed split16 in this paper, and used for the parameters such that $N = 32768$. When $F \cdot G_i$ can be computed exactly for all $i = 0, 1, 2, 3$ by FFT, split16 can exactly compute $F \cdot G$ at the cost of 4 FFT-based multiplication operations.

We note that a similar method has been proposed by Kim et al. [19] to speed up keyswitching on an NTT domain of a large ciphertext modulus. Subsequently, Belorgey et al. [2] proposed to decompose a ciphertext modulus into natural bases of the form 2^K, enabling FFT-based multiplication. On the other hand, the goal of the split FFT is to reduce the FFT error, rather than improving computational cost.

The split FFT decreases its multiplication error at the cost of larger computational cost. That said, this performance degradation is negligible in the entire circuit bootstrapping since homomorphic trace evaluation takes much smaller time than programmable bootstrapping.

3.4 High Precision Conversion by GLWE Dimension Switching

For some applications, one might need LWE to GLWE conversion of high precision. Using a larger gadget decomposition level increases precision of TFHE operations, but there might be a lower bound on the output error obtained by changing only gadget decomposition parameters. To achieve a smaller error variance, one possible approach is to increase other TFHE parameters, obtaining evaluation keys of smaller error variances.

We propose a high precision conversion method by combining GLWE keyswitching as follows. First, we convert an input LWE ciphertext into the corresponding GLWE ciphertext using LWEtoGLWEConst before the pre-processing. Let $\mathbf{S}$ be the corresponding GLWE secret key of a dimension k. We switch the GLWE ciphertext into the corresponding GLWE ciphertext under a new GLWE secret key $\mathbf{S}'$ of a larger dimension k' than k by GLWE keyswitching (see Algorithm 5 in Appendix C.2). Then, after applying our pre-processing by modulus switching and modulus raising on the switched GLWE ciphertext,[7] we evaluate the trace function on the larger GLWE dimension. Finally, we switch back the output of

[7] Modulus switching and modulus raising can be defined analogously on GLWE ciphertext.

the trace evaluation into the original GLWE secret key by GLWE keyswitching. In this paper, the GLWE keyswitching that changes GLWE dimension is called *GLWE dimension switching*.

Algorithm 4: High Precision Automorphism-based LWE to GLWE Conversion by GLWE dimension switching

Input: $\mathbf{c} = \mathrm{LWE}_{q,\mathbf{s}}(m)$ where q is a power-of-two and $\mathbf{s} = (s_1, \ldots, s_{kN})$
Input: GLWE keyswitching keys $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}$ and $\mathrm{KS}_{\mathbf{S}'\to\mathbf{S}}$ where $\mathbf{S} = (S_1, \ldots, S_k)$ is the corresponding GLWE secret key of $\mathbf{s}$ and $\mathbf{S}' = (S'_1, \ldots, S'_{k'})$ wheres $k' > k$
Input: Automorphism keys under $\mathbf{S}'$
Output: $\mathbf{C} = \mathrm{GLWE}_{q,\mathbf{S}}(m)$

$\mathbf{C} \leftarrow \mathsf{LWEtoGLWEConst}(\mathbf{c})$
$\mathbf{C}' \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}, \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'})$
$\mathbf{C}' \leftarrow \mathsf{ModSwitch}_{q\to q/N}(\mathbf{C}')$
$\mathbf{C}' \leftarrow \mathsf{ModRaise}_{q/N\to q}(\mathbf{C}')$
$\mathbf{C}' \leftarrow \mathsf{EvalTr}_{N/1}(\mathbf{C}')$
$\mathbf{C} \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}', \mathrm{KS}_{\mathbf{S}'\to\mathbf{S}})$
return $\mathbf{C}$

Theorem 2. *Let* $\mathbf{c}$ *be an LWE ciphertext of phase* μ *under a secret key* $\mathbf{s} = (s_1, \ldots, s_{kN})$ *and* $\mathbf{S} = (S_1, \ldots, S_k)$ *be the corresponding GLWE secret key. Let* $\mathbf{S}' = (S'_1, \ldots, S'_{k'})$ *be a GLWE secret key of a dimension* k' *such that* $k' > k$. *Then, Algorithm 4 returns a GLWE ciphertext* $\mathbf{C}$ *of phase* $\mu + E_{\mathsf{conv}}(X)$ *under* $\mathbf{S}$ *where the variance* V_{conv} *of* $E_{\mathsf{conv}}(X)$ *is given as follows.*

$$V_{\mathsf{conv}} = V_{\mathbf{S}\to\mathbf{S}'} + N^2 V_{\mathsf{ms}} + V_{\mathsf{tr}} + V_{\mathbf{S}'\to\mathbf{S}}$$

where $V_{\mathbf{S}\to\mathbf{S}'}$ *(resp.* $V_{\mathbf{S}'\to\mathbf{S}}$*) is the GLWE keyswitching noise variance from* $\mathbf{S}$ *to* $\mathbf{S}'$ *(resp.* $\mathbf{S}'$ *to* $\mathbf{S}$*),* V_{ms} *is the modulus switching noise variance, and* V_{tr} *is the trace evaluation variance under* $\mathbf{S}'$.

Proof. Let μ be the phase of the input LWE ciphertext $\mathbf{c}$ under $\mathbf{s}$. The GLWE ciphertext $\mathbf{C}$ obtained by $\mathsf{LWEtoGLWEConst}$ on $\mathbf{c}$ has a phase of

$$\mu + v_1 X + \cdots + v_{N-1} X^{N-1}$$

under $\mathbf{S}$ corresponding to $\mathbf{s}$ where $v_1, \ldots, v_{N-1}$ are unknown coefficients. The following GLWE keyswitching from $\mathbf{S}$ to $\mathbf{S}'$ switches GLWE dimension from k to k' with an additive error $E_{\mathbf{S}\to\mathbf{S}'}(X)$, obtaining a GLWE ciphertext of phase

$$\mu + v_1 X + \cdots + v_{N-1} X^{N-1} + E_{\mathbf{S}\to\mathbf{S}'}(X)$$

under $\mathbf{S}'$. Now, our pre-processing method is applied to the GLWE ciphertext. After the pre-processing, one obtains a GLWE ciphertext of phase

$$\frac{1}{N}(\mu + v_1 X + \cdots + v_{N-1} X^{N-1} + E_{\mathbf{S}\to\mathbf{S}'}(X)) + E_{\mathsf{ms}}(X) + \frac{q}{N} \cdot U(X)$$

under $\mathbf{S}'$ where $E_{\mathsf{ms}}(X)$ is the modulus switching error and $U(X)$ is a polynomial induced by modulus raising from q/N to q. Subsequent trace evaluation multiplies the constant term by N and removes all the other coefficients, resulting in a GLWE ciphertext of phase

$$\mu + e_{\mathbf{S}\to\mathbf{S}'} + Ne_{\mathsf{ms}} + E_{\mathsf{tr}}(X)$$

under $\mathbf{S}'$ where $e_{\mathbf{S}\to\mathbf{S}'}$ and e_{ms} are constant terms of $E_{\mathbf{S}\to\mathbf{S}'}$ and E_{ms}, respectively, and $E_{\mathsf{tr}}(X)$ is the error induced by trace evaluation. Lastly, the GLWE ciphertext under $\mathbf{S}'$ is switched back into a corresponding GLWE ciphertext under $\mathbf{S}$ by GLWE keyswitching. The output GLWE ciphertext has a phase of

$$\mu + e_{\mathbf{S}\to\mathbf{S}'} + Ne_{\mathsf{ms}} + E_{\mathsf{tr}}(X) + E_{\mathbf{S}'\to\mathbf{S}}(X)$$

under $\mathbf{S}$ where $E_{\mathbf{S}'\to\mathbf{S}}(X)$ is an additive error of GLWE keyswitching from $\mathbf{S}'$ to $\mathbf{S}$. Since all of the additive errors $e_{\mathbf{S}\to\mathbf{S}'}$, e_{ms}, E_{tr}, and $E_{\mathbf{S}'\to\mathbf{S}}$ are independent, the noise variance V_{conv} is given as follows.

$$V_{\mathsf{conv}} = V_{\mathbf{S}\to\mathbf{S}'} + N^2 V_{\mathsf{ms}} + V_{\mathsf{tr}} + V_{\mathbf{S}'\to\mathbf{S}}.$$

□

3.5 Performance

In this section, we implement our automorphism-based conversion methods using the `tfhe-rs` library [26] of version 0.5.3, which supports the TFHE scheme on FFT domains, comparing the results to the packing keyswitching.[8] Our experiments are executed in Intel i5-13600K @ 5.30 GHz.[9] Time is measured by `criterion` benchmarking module of `Rust` with 1,000 samples and error is the average on 1,000 measurements. The benchmark result of the high precision conversion described in Section 3.4 is not given here, while its application can be found in Section 4.2. All the implementations in this paper are publicly available.[10]

For the packing keyswitching, we consider two possible cases: one from an LWE dimension to the corresponding GLWE dimension directly (denoted 'PackingKS w/o LWE KS'), and the other from a smaller LWE dimension by using LWE keyswitching before packing keyswitching (denoted 'PackingKW w/ LWE KS') as in the PBS operation. The first method only induces an error from the packing keyswitching, but the computational cost and key size are proportional to N^2. On the other hand, the other one has a smaller computational cost and key size proportional to nN at the cost of an additional error that comes from the LWE keyswitching, where n is the output LWE dimension of the LWE

[8] For packing keyswitching, `keyswtich_lwe_ciphertext_into_glwe_ciphertext` supported by the `tfhe-rs` library is used for the benchmark.

[9] It has 6 P-cores @ 5.30 GHz and 8 E-cores @ 3.90 GHz, and we used a single P-core for the benchmark.

[10] https://github.com/KAIST-CryptLab/PatchingWWLp

keyswitching. We note that the packing keyswitching with the LWE keyswitching cannot reduce the output error by increasing the gadget decomposition level of the packing keyswitching since the LWE keyswitching error dominates the packing keyswitching error.

The parameters used in the benchmark of the LWE to GLWE conversion methods are summarized in Table 1. They come from the recommended parameter sets of `shortint` type in the `tfhe-rs` library (see Appendix G for the details), which is also used for the building block of a large precision integer.

Parameter Sets	Name in the `tfhe-rs` Library	n	N	k
`PARAM_2_2`	`PARAM_MESSAGE_2_CARRY_2_KS_PBS`	742	2048	1
`PARAM_3_3`	`PARAM_MESSAGE_3_CARRY_3_KS_PBS`	864	8192	1
`PARAM_4_4`	`PARAM_MESSAGE_4_CARRY_4_KS_PBS`	996	32768	1

Table 1: Parameter Sets used in the benchmark of the LWE to GLWE conversion. Given parameter sets come from the recommended sets of parameters for `shortint` type in the `tfhe-rs` library.

Table 2 shows the benchmark results. The 'Gadget Decomp.' column denotes the gadget decomposition parameters used for the packing keyswitching key or the automorphism key according to the conversion methods. 'Key Size' denotes the size of the packing keyswitching key or the trace evaluation key according to the conversion method. The details of the key size are given in Appendix F.

Parameter Sets	Method	Gadget Decomp. (B, ℓ)	FFT Type	Time (ms)	Error $(\|\cdot\|_\infty)$	Key Size (MB)
`PARAM_2_2`	PackingKS w/o LWE KS	$(2^{24}, 1)$	-	6.252	$2^{42.97}$	32
	PackingKS w/ LWE KS	$(2^{24}, 1)$	-	5.883	$2^{52.66}$	11.59
	Ours	$(2^{12}, 3)$	vanilla	0.472	$2^{42.26}$	0.52
		$(2^{13}, 3)$	$b = 42$	0.788	$2^{39.69}$	0.52
`PARAM_3_3`	PackingKS w/o LWE KS	$(2^{30}, 1)$	-	92.30	$2^{38.79}$	512
	PackingKS w/ LWE KS	$(2^{30}, 1)$	-	32.39	$2^{52.39}$	54
	Ours	$(2^{12}, 4)$	$b = 43$	4.550	$2^{32.36}$	3.25
		$(2^{10}, 5)$	$b = 41$	4.791	$2^{30.38}$	4.063
`PARAM_4_4`	PackingKS w/o LWE KS	$(2^{32}, 1)$	-	1402	$2^{41.81}$	8192
	PackingKS w/ LWE KS	$(2^{32}, 1)$	-	153.2	$2^{50.01}$	249
	Ours	$(2^{15}, 3)$	split16	37.80	$2^{38.29}$	11.25
		$(2^{13}, 4)$	split16	40.91	$2^{35.29}$	15

Table 2: Performance of the LWE to GLWE conversion.

In the case of converting a single LWE ciphertext into a GLWE ciphertext, our method improves the computation time (resp. key size) by factors of 13.25 (resp. 61.54), 20.29 (resp. 157.54) and 37.09 (resp. 728.5) for $N = 2048$, 8192 and 32768, respectively, to obtain a similar output error. Even compared to the packing keyswitching with the LWE keyswitching, our method achieves 12.46 (resp. 22.29), 7.12 (resp. 16.62) and 4.05 (resp. 22.13) times smaller computation time (resp. key size).

4 Patching WWL+ Circuit Bootstrapping

4.1 Revising Pre-processing and Error Analysis

The WWL+ circuit bootstrapping method can be described as the following two steps.

Step 1. Improving LWE to Lev conversion by

- employing PBSmanyLUT [13], and
- improving the automorphism-based blind rotation proposed by Lee et al. [21].

Step 2. Improving Lev to GGSW conversion by employing

- the automorphism-based LWE to GLWE conversion, and
- the GLWE to GGSW conversion by scheme switching proposed by De Micheli et al. [15].

In this section, we summarize two flaws of their error analysis in Step 2 and give a patch for the algorithm and the (corrected) error analysis.

Phase Amplification As the automorphism-based LWE to GLWE conversion is employed,[11] it is crucial to cancel out the phase amplification to avoid amplification of the PBS error. However, they have just set the scaling factor of the PBS output to $N^{-1}\Delta \bmod q$, causing error amplification by trace evaluation. Without removing this error amplification, one has to use a larger PBS decomposition level to decrease the PBS output error, degrading the overall performance. As they have used NTT domains for their implementation, it can be patched easily by applying the proper pre-processing of multiplying $N^{-1} \bmod q$. When it comes to FFT domains, our technique enables one to cancel out the phase amplification by trace evaluation, patching the WWL+ circuit bootstrapping.

[11] To be precise, they have evaluated the trace function on the blind rotation output without sample extraction. That said, there is no difference in our description since all the unnecessary coefficients vanish by trace evaluation in either way.

Scheme Switching The other flaw is on the error growth by the scheme switching, which converts a GLev ciphertext into the corresponding GGSW ciphertext by external product. Given a GLWE ciphertext $\mathrm{GLWE}(q/B^j \cdot m)$ under a GLWE secret key $\mathbf{S} = (S_1, \ldots, S_k)$, the scheme switching outputs $\mathrm{GLWE}(q/B^j \cdot (-S_i \cdot m))$, a component of $\mathrm{GGSW}(m)$, using external product by the scheme switching key $\mathrm{GGSW}(-S_i)$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$ where $S_{k+1} = -1$ and ℓ is the gadget decomposition level of the output GGSW ciphertext. Unlike the case of PBS, the scheme switching uses GGSW ciphertexts of polynomials. Since external product by a GGSW ciphertext not only adds a linear error but also multiplies the input error by the plaintext of the GGSW ciphertext, the scheme switching multiplies the input error variance by at most $\ell_1(S_i)$ [15].

The input to the scheme switching contains both the PBS error and the trace evaluation error, so the multiplication factor of the error growth by the scheme switching requires high precision PBS. However, we observe that the PBS error only remains in the constant term after the trace evaluation since it clears all the coefficients except the constant term. Since TFHE uses binary secret key, the PBS error does not increase by multiplication by S_i in terms of ℓ_∞-norm.[12] Hence, we conclude that using LWE to GLWE conversion with high enough precision mitigates error growth by the scheme switching.

Patch on the WWL+ Circuit Bootstrapping Method We summarize our patch on the WWL+ circuit bootstrapping and its error analysis as follows.

1. Given an LWE ciphertext $\mathrm{LWE}(q/2 \cdot m)$ of a single bit m, one computes $\mathrm{Lev}(m)$ by $\mathsf{PBSmanyLUT}$.
2. $\mathrm{GLev}(m)$ is obtained by applying the automorphism-based LWE to GLWE conversion to each LWE ciphertext inside $\mathrm{Lev}(m)$.
3. $\mathrm{GGSW}(m)$ is computed from $\mathrm{GLev}(m)$ by the scheme switching.

Let V_{pbs}, V_{conv} and V_{ss} be the variances of the PBS output error, the additive error of the automorphism-based LWE to GLWE conversion, and the additive error of the external product in the scheme switching used in the patched WWL+ circuit bootstrapping. Provided that $\mathsf{PBSmanyLUT}$ works without failure, the output GGSW ciphertext has an error variance V_{out} satisfying the following.

$$V_{\mathsf{out}} \leq V_{\mathsf{pbs}} + N\, V_{\mathsf{conv}} + V_{\mathsf{ss}}.$$

The conversion noise variance V_{conv} can be obtained by Theorem 1 or Theorem 2 according to the conversion method. The PBS[13] output error variance V_{pbs} and

[12] One might insist that the PBS error is increased in ℓ_2 norm, but such error growth also occurs in the private functional keyswitching.

[13] In this paper, we use the PBS workflow in [13], which is slightly different from that in [24].

the additive scheme switching error variance V_{ss} are given as follows.

$$V_{\mathsf{pbs}} \leq n\ell_{\mathsf{pbs}}(k+1)N\frac{B_{\mathsf{pbs}}^2+2}{12}\sigma_{\mathsf{bsk}}^2 + n\frac{q^2 - B_{\mathsf{pbs}}^{2\ell_{\mathsf{pbs}}}}{24B_{\mathsf{pbs}}^{2\ell_{\mathsf{pbs}}}}\left(1+\frac{kN}{2}\right) + \frac{nkN}{32} + \frac{n}{16}\left(1-\frac{kN}{2}\right)^2, \tag{6}$$

$$V_{\mathsf{ss}} \leq \frac{(1+kN)N}{12}\left(\frac{q^2}{B^{2\ell_{\mathsf{ss}}}}-1\right) + (k+1)\ell N\left(\frac{B_{\mathsf{ss}}}{2}\right)^2\sigma_{\mathsf{ssk}}^2 \tag{7}$$

where σ_{bsk}^2 (resp. σ_{ssk}^2) is the noise variance of the PBS key (resp. scheme switching key) and $(B_{\mathsf{pbs}}, \ell_{\mathsf{pbs}})$ (resp. $(B_{\mathsf{ss}}, \ell_{\mathsf{ss}})$) is the corresponding decomposition parameter. We refer to Appendix D and C for the details, respectively.

Remark 1. We note that the multiplication factor N of the conversion variance V_{conv} is an upper bound that comes from multiplication by S_i, so the amount of multiplicative error growth might vary according to the characteristic of the conversion error in practice. For example, the result in Table 4 shows that the automorphism-based conversion error grows by 2–3 bits while the high precision conversion error grows approximately by 5 bits.

Applications We implement the patched WWL+ circuit bootstrapping on FFT domains (except the improved automorphism-based blind rotation) using the `tfhe-rs` library and apply it to the following two applications; the first is the without-padding programmable bootstrapping (WoP-PBS) based on circuit bootstrapping proposed by Bergerat et al. [3]. There are some recommended parameters for the WoP-PBS in the `tfhe-rs` library, so we apply the patched faster and smaller circuit bootstrapping on these parameters. The other is homomorphic AES circuit evaluation, one of the applications of the WWL+ circuit bootstrapping.

4.2 Application to WoP-PBS

Bergerat et al. [3] proposed a WoP-PBS method based on the circuit bootstrapping to evaluate large precision LUT efficiently. By extracting bits sequentially from the least significant bit, their WoP-PBS performs PBS operations only for a single bit of message, allowing one to use a small polynomial size for more than 4 bits of plaintext precision. Hence, all of the recommended WoP-PBS parameters in `tfhe-rs` use GLWE parameters k and N such that $kN \leq 2048$, even for 8-bit plaintext precision.

Instead, it requires high precision for its PBS and GGSW conversion to encode a plaintext of a larger precision. For example, one of the recommended sets of parameters named `WOPBS_PARAM_MESSAGE_4_CARRY_4_KS_PBS` encodes an 8-bit plaintext with a scaling factor $\Delta = 2^{56}$, requiring its circuit bootstrapping to have a small enough error so that 8-bit homomorphic LUT using the circuit bootstrapping outputs supports such plaintext encoding.

In this section, we improve the circuit bootstrapping used in the WoP-PBS method by employing the patched WWL+ circuit bootstrapping combined with the high precision LWE to GLWE conversion described in Section 3.4. We implement our method and compare the result to the previous method on the parameters in Table 3, which are recommended parameters in the `tfhe-rs` library (see Appendix G for the details).

Paramter Sets	Name in the `tfhe-rs` Library	n	Gadget Decomp. (B, ℓ)		
			PBS	PrivKS	CBS
WOPBS_2_2	WOPBS_PARAM_MESSAGE_2_CARRY_2_KS_PBS	769	$(2^{15}, 2)$	$(2^{15}, 2)$	$(2^5, 3)$
WOPBS_3_3	WOPBS_PARAM_MESSAGE_3_CARRY_3_KS_PBS	873	$(2^9, 4)$	$(2^9, 4)$	$(2^6, 3)$
WOPBS_4_4	WOPBS_PARAM_MESSAGE_4_CARRY_4_KS_PBS	953	$(2^9, 4)$	$(2^9, 4)$	$(2^4, 6)$

Table 3: The recommended sets of parameters in the `tfhe-rs` library that we used in our implementation of the WoP-PBS. All the parameters use $N = 2048$ and $k = 1$. 'PBS' (resp. 'PrivKS') denotes parameters for PBS (resp. private functional keyswitching), and 'CBS' denotes those for the GGSW output.

PBSmanyLUT by Refreshing The first optimization technique of the WWL+ circuit bootstrapping is employing PBSmanyLUT in the LWE to Lev conversion. PBSmanyLUT allows one to evaluate several LUTs on the same input by a single PBS operation. However, evaluating a larger number of LUTs by PBSmanyLUT requires the input ciphertext to have a smaller error.

To use PBSmanyLUT regardless of its error-sensitive characteristic, we propose to give an additional PBS key to *refresh* the input LWE ciphertext of a single bit message to guarantee the correctness of PBSmanyLUT. This refreshing key need not to achieve as high precision as the original PBS key for the LWE to Lev conversion. Hence, we give a refreshing key with the same parameters as the original PBS key only except the gadget decomposition parameters: $(B_{\mathsf{refresh}}, \ell_{\mathsf{refresh}}) = (2^{23}, 1)$.

Since all the other parameters for the refreshing key are the same as the original PBS key, it is guaranteed that the PBS operation with the refreshing key works correctly on a given input if and only if the original PBS key works correctly on the same input. By analyzing the variance of the refreshing error, we see that PBSmanyLUT allows one to evaluate LWE to Lev conversion in a single PBS operation with a failure probability less than 2^{-241} in our parameters. We refer to Appendix D for more details.

Lev to GGSW Conversion A series of private packing keyswitchings to convert a GLev ciphertext into the corresponding GGSW ciphertext are replaced by the automorphism-based LWE to GLWE conversion and the scheme swithcing

Method	Gadget Decomp. Parameters					Error	
	$(B_{\mathsf{priv}}, \ell_{\mathsf{priv}})$	$(B_{\mathsf{auto}}, \ell_{\mathsf{auto}})$	$(B_{\mathsf{ss}}, \ell_{\mathsf{ss}})$	$(B_{k\to k'}, \ell_{k\to k'})$	$(B_{k'\to k}, \ell_{k'\to k})$	GLev	GGSW
PrivKS	$(2^{15}, 2)$	-	-	-	-	-	$2^{36.96}$
Ours	-	$(2^7, 7)$	$(2^8, 6)$	-	-	$2^{34.24}$	$2^{36.50}$
PrivKS	$(2^9, 4)$	-	-	-	-	-	$2^{31.06}$
Ours	-	$(2^6, 10)$	$(2^6, 9)$	$(2^{15}, 3)$	$(2^5, 10)$	$2^{25.14}$	$2^{30.37}$

Table 4: The experimental result of the error of Lev to GGSW conversion by the private functional keyswitching and our (high precision) automorphism-based conversion with the scheme switching. $N = 2048$ and $k = 1$ are used, and $k' = 2$ is used for the high precision automorphism-based conversion. The first two rows correspond to `WOPBS_2_2` and the last two rows correspond to `WOPBS_3_3` and `WOPBS_4_4`.

in the WWL+ method. As mentioned in Section 4.1, our patched error analysis shows that parameters should be selected considering the multiplicative factor of the trace evaluation error by the scheme switching. We experimentally found proper parameters for our automorphism-based conversion (with GLWE dimension switching) and the scheme switching to achieve smaller error growth for the Lev to GGSW conversion compared to the current method based on private functional keyswitching. The result is shown in Table 4.

Parameter Sets	Method	Time (ms)				Error		Key Size (MB)			
		Refresh	Step 1	Step 2	Total	GGSW	Ext. Prod	Refresh	Step 1	Step 2	Total
`WOPBS_2_2`	`tfhe-rs`	-	66.90	64.34	131.24	$2^{42.06}$	$2^{53.38}$	-	48.06	128	176.06
	Ours	14.27	22.41	3.56	40.24	$2^{41.99}$	$2^{53.35}$	24.03	48.06	1.39	73.48
`WOPBS_3_3`	`tfhe-rs`	-	147.01	144.31	291.32	$2^{36.30}$	$2^{50.13}$	-	109.13	256	365.13
	Ours	15.59	43.28	11.04	69.91	$2^{36.25}$	$2^{50.12}$	27.28	109.13	2.36	138.77
`WOPBS_4_4`	`tfhe-rs`	-	310.62	314.38	625.00	$2^{36.66}$	$2^{46.29}$	-	119.13	256	375.13
	Ours	16.99	47.46	22.16	86.61	$2^{36.66}$	$2^{46.29}$	29.78	119.13	2.36	151.27

Table 5: Performance of the circuit bootstrapping on the recommended sets of WoP-PBS parameters in the `tfhe-rs` library.

Benchmark Our improved method based on the patched WWL+ circuit bootstrapping is applied to the WoP-PBS, and its implementation results are summarized in Table 5, where they are obtained in the same environment as in Section 3.5. 'GGSW Error' denotes the output GGSW ciphertext error and 'Ext. Prod. Error' denotes the resulting error of external product by the output GGSW

ciphertext to a fresh GLWE ciphertext. We observe that our method improves circuit bootstrapping time (resp. key size) by factors from 3.26 to 7.22 (resp. from 2.39 to 2.63) according to the parameters while giving similar output errors.

4.3 Revising AES Evaluation

Homomorphic AES evaluation is one of the relevant applications of the transciphering framework - a hybrid framework combining a symmetric cipher with a homomorphic encryption scheme to reduce computation and communication costs of the client-side at the cost of homomorphic decryption of the symmetric cipher on the server-side [22]. Several works evaluating AES on TFHE have been proposed [4, 23, 25, 24], and the fastest method (in a single thread) until now is based on circuit bootstrapping.

The first method of AES evaluation based on circuit bootstrapping, dubbed Fregata, has been proposed by Wei et al. [25]. Fregata uses an LWE ciphertext encrypting a single bit in the MSB. Such plaintext encoding is called 2-encoding in [4]. The advantage of this plaintext encoding in Boolean circuit evaluation is that homomorphic XOR evaluation becomes almost free as it corresponds to homomorphic addition, enjoying free evaluation of AddRoundKey, ShiftRows, and MixColumns in terms of computation time. Hence, Fregata spends most of its computation time to evaluate the 8-bit AES S-box using circuit bootstrapping.

Wang et al. [24] used a similar approach with their WWL+ circuit bootstrapping method to evaluate AES, while we observe a flaw on their selection of parameters. They choose the parameter set `CMUX1` of maximum depth 8, which is the maximum circuit depth it can support before the next circuit bootstrapping [24], considering the evaluation of the 8-bit AES S-box. However, it requires the maximum depth of at least 56 to evaluate the subsequent MixColumns operation without bootstrapping since MixColumns computes the sum of at most 7 output bits of the AES S-box (see Appendix E.2 for the details). The gadget decomposition level for PBS should be increased from 1 to 2 to achieve such depth, so their benchmark result should be revised.

	N	k	Time (s)				Key Size (MB)	Failure Prob.
			LWE KS	SubBytes	Linear	Total		
Set-I	2048	1	0.030	34.308	0.010	34.348	49.50	$2^{-497.32}$
Set-II	1024	2	0.029	26.263	0.009	26.301	36.92	$2^{-242.63}$
Set-III	512	4	0.029	27.536	0.008	27.573	30.80	$2^{-75.60}$

Table 6: Benchmark result of our AES evaluation. 'Failure Prob.' denotes the failure probability of a single AES evaluation.

We implement the AES evaluation using our patched version of WWL+ method. As additional optimization, we improve the LWE keyswitching before

PBS by extending the method based on GLWE keyswitching proposed by Chen et al. [7] to GLWE dimension switching. For our implementation, we used the following parameters.

- $n = 768$ and $(k, N) \in \{(1, 2048), (2, 1024), (4, 512)\}$.[14]
- $(B_{\mathsf{ds}} = 2^4, \ell_{\mathsf{ds}} = 3)$ for the GLWE dimension switching.
- The other decomposition parameters are the same as those of `WOPBS_2_2` used for our patched version of WWL+ method in Section 4.2 without refreshing.

Table 6 shows the benchmark result. The benchmark environment is the same as in Section 3.5. One can find that Set-II of $N = 1024$ and $k = 2$ provides the highest performance of 26.301 s in terms of computation time. We refer to Appendix E for the details of our AES evaluation method and its error analysis.

Acknowledgement

We thank the authors of [24] for their constructive comments regarding trace evaluation, parameter selection, and AES transciphering discussed in this work. Jooyoung Lee was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.2022-0-01047, Development of statistical analysis algorithm and module using homomorphic encryption based on real number operation].

References

[1] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of Learning with Errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015). https://doi.org/doi:10.1515/jmc-2015-0016

[2] Belorgey, M.G., Carpov, S., Gama, N., Guasch, S., Jetchev, D.: Revisiting Key Decomposition Techniques for FHE: Simpler, Faster and More Generic. Cryptology ePrint Archive, Paper 2023/771 (2023), `https://eprint.iacr.org/2023/771`

[3] Bergerat, L., Boudi, A., Bourgerie, Q., Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Parameter Optimization and Larger Precision for (T)FHE. Journal of Cryptology **36**, 28 (2023). https://doi.org/10.1007/s00145-023-09463-5

[4] Bon, N., Pointcheval, D., Rivain, M.: Optimized Homomorphic Evaluation of Boolean Functions. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 302–341 (Jul 2024). https://doi.org/10.46586/tches.v2024.i3.302-341

[5] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption without Bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 309–325. ACM (2012). https://doi.org/10.1145/2633600

[14] We use the LWE standard deviation of $2^{-17.12}$ that achieves 128-bit security according to the lattice-estimator [1]. For the GLWE standard deviation, we use the same value as in the recommended parameters of `tfhe-rs`.

[6] Chen, H., Chillotti, I., Ren, L.: Onion Ring ORAM: Efficient Constant Bandwidth Oblivious RAM from (Leveled) TFHE. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 345–360. CCS '19, ACM (2019). https://doi.org/10.1145/3319535.3354226
[7] Chen, H., Dai, W., Kim, M., Song, Y.: Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts. In: Sako, K., Tippenhauer, N.O. (eds.) Applied Cryptography and Network Security. pp. 460–479. Springer (2021)
[8] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. vol. 10031, pp. 3–33. Springer (2016)
[9] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. Springer International Publishing, Cham (2017)
[10] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast Fully Homomorphic Encryption Over the Torus. Journal of Cryptology **33**, 34–91 (2020). https://doi.org/10.1007/s00145-019-09319-x
[11] Chillotti, I., Joye, M., Ligier, D., Orfila, J.B., Tap, S.: CONCRETE: Concrete operates on ciphertexts rapidly by extending TfhE. In: WAHC 2020-8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (2020)
[12] Chillotti, I., Joye, M., Paillier, P.: Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks. In: Dolev, S., Margalit, O., Pinkas, B., Schwarzmann, A. (eds.) Cyber Security Cryptography and Machine Learning. pp. 1–19. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-78086-9_1
[13] Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. pp. 670–699. Springer (2021). https://doi.org/10.1007/978-3-030-92078-4_23
[14] Cong, K., Das, D., Park, J., Pereira, H.V.: SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 563–577. CCS '22, ACM (2022). https://doi.org/10.1145/3548606.3560702
[15] De Micheli, G., Kim, D., Micciancio, D., Suhl, A.: Faster Amortized FHEW Bootstrapping Using Ring Automorphisms. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. pp. 322–353. Springer Nature Switzerland, Cham (2024)
[16] Ducas, L., Micciancio, D.: FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. vol. 9056, pp. 617–640. Springer (2015)
[17] Gentry, C., Sahai, A., Waters, B.: Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. vol. 8042, pp. 75–92. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_5
[18] Joye, M., Walter, M.: Liberating TFHE: Programmable Bootstrapping with General Quotient Polynomials. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. p. 1–11. WAHC'22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560827.3563376

[19] Kim, M., Lee, D., Seo, J., Song, Y.: Accelerating HE Operations from Key Decomposition Technique. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 70–92. Springer Nature Switzerland, Cham (2023)
[20] Klemsa, J.: Fast and Error-Free Negacyclic Integer Convolution Using Extended Fourier Transform. In: Dolev, S., Margalit, O., Pinkas, B., Schwarzmann, A. (eds.) Cyber Security Cryptography and Machine Learning. pp. 282–300. Springer International Publishing, Cham (2021)
[21] Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW Bootstrapping with Small Evaluation Keys, and Applications to Threshold Homomorphic Encryption. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 227–256. Springer Nature Switzerland, Cham (2023)
[22] Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can Homomorphic Encryption be Practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop. p. 113–124. ACM (2011). https://doi.org/10.1145/2046660.2046682
[23] Trama, D., Clet, P.E., Boudguiga, A., Sirdey, R.: A Homomorphic AES Evaluation in Less than 30 Seconds by Means of TFHE. In: Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. p. 79–90. WAHC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3605759.3625260
[24] Wang, R., Wen, Y., Li, Z., Lu, X., Wei, B., Liu, K., Wang, K.: Circuit Bootstrapping: Faster and Smaller. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 342–372. Springer Nature Switzerland, Cham (2024)
[25] Wei, B., Wang, R., Li, Z., Liu, Q., Lu, X.: Fregata: Faster Homomorphic Evaluation of AES via TFHE. In: Athanasopoulos, E., Mennink, B. (eds.) Information Security. pp. 392–412. Springer Nature Switzerland, Cham (2023)
[26] Zama: TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data (2022), https://github.com/zama-ai/tfhe-rs

A Functional Keyswitching

In this section, we summarize the LWE to GLWE public/private functional keyswitching. For the detailed analysis of the keyswitching, we refer to [10, 13].

Let $\mathbf{s} = (s_1, \ldots, s_n)$ be a LWE secret key and $\mathbf{S} = (S_1, \ldots, S_k)$ be a GLWE secret key. The keyswitching key is given by $\mathrm{KS}_i = \mathrm{GLev}^{(B,\ell)}_{\mathbf{S}}(s_i)$ for $i = 1, \ldots, n$ where B and ℓ are decomposition base and level, respectively, for the LWE to GLWE keyswitching.

Given an LWE ciphertext $\mathbf{c} = (a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$ of m with respect to $\mathbf{s}$, let $(a_{i,1}, \ldots, a_{i,\ell})$ be the gadget decomposition of a_i for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell$. Let $\mathrm{KS}_{i,j} = \mathrm{GLWE}_{\mathbf{S}}(q/B^j \cdot s_i)$ be the GLWE ciphertext of s_i with a scaling factor q/B^j contained in KS_i. The LWE to GLWE keyswitching outputs a GLWE ciphertext $\mathbf{C}$ of m given as follows.

$$\mathbf{C} = \mathrm{GLWE}^0(b) - \sum_{i=1}^{n} \sum_{j=1}^{\ell} a_{i,j} \cdot \mathrm{KS}_{i,j}$$

where $\mathrm{GLWE}^0(b)$ denotes the trivial GLWE encryption of b, namely,

$$(0, \ldots, 0, b) \in \mathcal{R}_{q,N}^{k+1}.$$

By the linear property of the inner product and gadget decomposition, one can check that $\mathbf{C}$ is a GLWE encryption of m (with the same scaling factor as the input $\mathbf{c}$) with respect to $\mathbf{S}$.

Public Functional Keyswitching The LWE to GLWE keyswitching can be generalized to evaluate a public Lipschitz function while converting LWE ciphertexts into the GLWE ciphertext. Let $f : \mathbb{Z}_q^t \to \mathbb{Z}_q$ be a public Lipschitz function to evaluate on t LWE ciphertexts $\mathbf{c}^{(z)} = (a_1^{(z)}, \ldots, a_n^{(z)}, b^{(z)})$ of m_z for $z = 1, \ldots, t$. Then, the following $\mathbf{C}$ is a GLWE ciphertext of $f(m_1, \ldots, m_t)$.

$$\mathbf{C} = \mathrm{GLWE}^0(f(b^{(1)}, \ldots, b^{(t)})) - \sum_{i=1}^{n} \sum_{j=1}^{\ell} \tilde{a}_{i,j} \, \mathrm{KS}_{i,j}$$

where $(\tilde{a}_{i,1}, \ldots, \tilde{a}_{i,\ell})$ is the gadget decomposition of the value $f(a_i^{(1)}, \ldots, a_i^{(t)})$ for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell$. The above keyswitching that evaluates a public function f is called the LWE to GLWE public functional keyswitching.

Private Functional Keyswitching When the Lipschitz function $f : \mathbb{Z}_q^t \to \mathbb{Z}_q$ to evaluation during the keyswitching is private, it requires an private functional keyswitching key $\{\mathrm{KS}_{z,i}^{(f)}\}_{(z,i)\in[t]\times[n+1]}$ defined as follows ($s_{n+1} = -1$ for convenience).

$$\mathrm{KS}_{z,i}^{(f)} = \mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(f(0, \ldots, 0, s_i, 0, \ldots, 0))$$

where s_i is at position z and B (resp. ℓ) is the decomposition base (resp. level). Let $\mathrm{KS}_{z,i,j}^{(f)} = \mathrm{GLWE}_{\mathbf{S}}(q/B^j \cdot f(0, \ldots, 0, s_i, 0, \ldots, 0))$ be the GLWE ciphertext of $f(0, \ldots, 0, s_i, 0, \ldots, 0)$ with the scaling factor of q/B^j contained in $\mathrm{KS}_{z,i}^{(f)}$.

Let $\mathbf{c}^{(z)} = (a_1^{(z)}, \ldots, a_{n+1}^{(z)})$ be an LWE ciphertext of m_z for $z = 1, \ldots, t$. Then, the following $\mathbf{C}$ is a GLWE ciphertext of $f(m_1, \ldots, m_t)$.

$$\mathbf{C} = -\sum_{z=1}^{t} \sum_{i=1}^{n+1} \sum_{j=1}^{\ell} \tilde{a}_{i,j}^{(z)} \, \mathrm{KS}_{z,i,j}^{(f)}$$

where $(\tilde{a}_{i,1}^{(z)}, \ldots, \tilde{a}_{i,\ell}^{(z)})$ is the gadget decomposition of $a_i^{(z)}$ for $z = 1, \ldots, t$ and $i = 1, \ldots, n+1$. The above keyswitching that evaluates the private function f is called the LWE to GLWE private functional keyswitching.

B Error Analysis of the Split FFT

In this section, we analyze the error variance of the split FFT, described in Section 3.3. The split FFT is based on the observation that the FFT error

increases as the bound B of the input polynomial increases. Considering its application to GLWE keyswitching, we measure the FFT error of the summation of products of gadget decomposed polynomials and random polynomials. Table 7 shows standard deviation of the split FFT error on the parameters used in this paper. The Std. Dev. column denotes the standard deviation computed by square root of (5). The Failure Prob. column denotes failure probability of the exact polynomial multiplication of the upper part in the split FFT, computed by the Std. Dev. column. The Gadget Decomp. column denotes upper bound of standard deviation of the gadget decomposition error, which is estimated by $\sqrt{\frac{kN}{12}\left(\frac{q^2}{B^{2\ell}}-1\right)}$ appearing in Lemma 2. One can find that the FFT error induced by the lower part is smaller than the gadget decomposition error.

	N	k	B	ℓ	b	Upper Part		Lower Part	
						Std. Dev.	Failure Prob.	Std. Dev.	Gadget Decomp.
Sec. 3.5	2048	1	13	3	42	$2^{-7.01}$	2^{-2992}	$2^{12.99}$	$2^{27.71}$
	8192	1	12	4	43	$2^{-6.80}$	2^{-2245}	$2^{15.2}$	$2^{19.71}$
	8192	1	10	5	41	$2^{-6.64}$	2^{-1797}	$2^{11.36}$	$2^{17.71}$
	32768	1	15	3	split16	$2^{-7.01}$	2^{-2292}	-	-
	32768	1	13	4	split16	$2^{-8.8}$	2^{-35835}	-	-
Sec. 4.2	2048	1	7	7	37	$2^{-7.40}$	2^{-5125}	$2^{2.60}$	$2^{17.71}$
	2048	2	6	10	36	$2^{-6.85}$	2^{-2395}	$2^{1.15}$	$2^{7.21}$
	2048	1	15	3	44	$2^{-7.01}$	2^{-2992}	$2^{16.99}$	$2^{21.71}$
	2048	2	5	10	35	$2^{-6.85}$	2^{-2395}	$2^{-0.85}$	$2^{17.21}$
Sec. 4.3	512	4	7	7	35	$2^{-6.74}$	2^{-2053}	$2^{-0.74}$	$2^{17.71}$
	1024	2	7	7	36	$2^{-7.10}$	2^{-3419}	$2^{0.90}$	$2^{17.71}$
	2048	1	7	7	37	$2^{-7.40}$	2^{-5125}	$2^{2.60}$	$2^{17.71}$

Table 7: Standard deviations of the split FFT for the GLWE keyswitching under the parameters used in this paper.

C TFHE Operations and Error Analysis

As in most TFHE/FHEW-like cryptosystems, we analyze the noise growth based on the heuristic assumption such that the noises of coefficient in ciphertexts follow independent Gaussian distribution (or sub-Gaussian) centered at 0 of some standard deviation σ. We denote the noise variance of a key in terms of ℓ_∞-norm, giving an upper bound of the variance of all coefficients of the key components. For the gadget decomposition with a base 2^B and a level ℓ, we assume the decomposition error is uniformly sampled from $[\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$ as analogous to [13]. As mentioned in Section 2.2, we only deal with the binary secret key in this

section.[15] The proofs given in this section comes from [7, 13, 15, 6] with a slight modification generalizing GLWE dimension k.

C.1 Modulus Switching

Let q and q' be ciphertext moduli such that $q' < q$. Given a GLWE ciphertext $\mathbf{C} = (A_1, \ldots, A_{k+1}) \in \mathcal{R}_{q,N}^{k+1}$ of M under $\mathbf{S} = (S_1, \ldots, S_k)$, the modulus switching from q to q' outputs a GLWE ciphertext $\mathbf{C}' = (A'_1, \ldots, A'_{k+1}) \in \mathcal{R}_{q',N}^{k+1}$ of $\frac{q'}{q}M$ under $\mathbf{S}$ where $A'_i = \left\lfloor \frac{q'}{q} A_i \right\rceil$ for $i = 1, \ldots, k+1$.

Lemma 1 (Modulus Switching). *Let $\mathbf{C} \in \mathcal{R}_{q,N}^{k+1}$ be a GLWE ciphertext of a phase μ under $\mathbf{S}$. Then, modulus switching outputs a GLWE ciphertext $\mathbf{C}' \in \mathcal{R}_{q',N}^{k+1}$ of a phase $\frac{q'}{q}\mu + E_{\mathsf{ms}}$ under $\mathbf{S}$ where the variance V_{ms} of E_{ms} is given as follows.*

$$V_{\mathsf{ms}} \leq \frac{kN+1}{12}.$$

Proof. Let $\mathbf{C} = (A_1, \ldots, A_{k+1})$ and $\mathbf{C}' = (A'_1, \ldots, A'_{k+1})$ where $A'_i = \lfloor \frac{q'}{q} A_i \rceil$ for $i = 1, \ldots, k+1$. Then, one can represent A'_i as

$$A'_i = \frac{q'}{q} A_i + E'_i$$

where coefficients of E'_i are uniformly and independently sampled from $[-\frac{1}{2}, \frac{1}{2})$. The phase of $\mathbf{C}'$ under $\mathbf{S}$ is given as follow.

$$\langle \mathbf{C}', (-\mathbf{S}, 1) \rangle = \frac{q'}{q} \left(A_{k+1} - \sum_{i=1}^{k} A_i S_i \right) + \left(E'_{k+1} - \sum_{i=1}^{k} E'_i S_i \right).$$

Let $E_{\mathsf{ms}} = E'_{k+1} - \sum_{i=1}^{k} E'_i S_i$. From $E'_i \leftarrow [-\frac{1}{2}, \frac{1}{2})$ and $\mathbf{S}$ is a binary secret key, one obtain

$$\mathrm{Var}(E_{\mathsf{ms}}) \leq \frac{kN+1}{12}.$$

□

For an LWE ciphertext, the modulus switching error increment e_{ms} has variance bounded above by $\frac{n+1}{12}$. We note that e_{ms} (and E_{ms}) does not depend on q and q'.

C.2 GLWE Keyswitching

Let $\mathbf{S}$ and $\mathbf{S}'$ be two GLWE secret keys of dimensions k and k', respectively, and of the same polynomial size N. The GLWE keyswitching from $\mathbf{S}$ to $\mathbf{S}'$ changes a GLWE ciphertext of M under $\mathbf{S}$ to another GLWE ciphertext of M under $\mathbf{S}'$ using the GLWE keyswitching key $\{\mathrm{GLev}_{\mathbf{S}'}(S_i)\}_{i=1}^{k}$, a set of k GLev ciphertexts of S_i, $i = 1, \ldots, k$. The precise description of the algorithm is given in Algorithm 5.

[15] The result is the same for the ternary secret key, while is not for the Gaussian secret key.

Algorithm 5: GLWE keyswitching GLWE_KS

Input: $\mathbf{C} = \mathrm{GLWE}_{\mathbf{S}}(M)$ under $\mathbf{S} = (S_1, \ldots, S_k)$
Input: $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i] = \mathrm{GLev}^{(B,\ell)}_{\mathbf{S}'}(S_i)$ for $i = 1, \ldots, k$ with decomposition base B and level ℓ under $\mathbf{S}' = (S'_1, \ldots, S'_{k'})$
Output: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}'}(M)$

$\mathbf{C} = (A_1, \ldots, A_k, A_{k+1})$
$\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j] = \mathrm{GLWE}_{\mathbf{S}'}\left(\frac{q}{B^j} \cdot S_i\right)$ for $i \in [k]$ and $j \in [\ell]$
$\mathbf{C}' \leftarrow (0, \cdots, 0, A_{k+1}) = \mathrm{GLWE}^0_{\mathbf{S}'}(A_{k+1}) \in \mathcal{R}^{k'+1}_{q,N}$
for $i = 1$ *to* k **do**
    Decompose A_i as $\sum_{j=1}^{\ell} A'_{i,j} \cdot \frac{q}{B^j} + E'_i$ with $\|A'_{i,j}\|_\infty \leq \frac{B}{2}$ and $\|E'_i\|_\infty \leq \frac{q}{2B^\ell}$
    $\mathbf{C}' \leftarrow \mathbf{C}' - \sum_{j=1}^{\ell} A'_{i,j} \cdot \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j]$
return $\mathbf{C}'$

Lemma 2 (GLWE Keyswitching). *Let $\mathbf{C}$ be a GLWE ciphertext of a phase μ under $\mathbf{S}$. Let $\sigma^2_{\mathbf{S}\to\mathbf{S}'}$ be the noise variance of the GLWE keyswitching key from $\mathbf{S}$ to $\mathbf{S}'$. Then, Algorithm 5 returns a GLWE ciphertext $\mathbf{C}'$ of a phase $\mu + E_{\mathsf{ks}}(X)$ under $\mathbf{S}'$ where the variance V_{ks} of $E_{\mathsf{ks}}(X)$ is given as follows.*

$$V_{\mathsf{ks}} \leq \frac{kN}{12}\left(\frac{q^2}{B^{2\ell}} - 1\right) + k\ell N\left(\frac{B}{2}\right)^2 \sigma^2_{\mathbf{S}\to\mathbf{S}'}.$$

Proof. The output $\mathbf{C}'$ can be represented as follows.

$$\mathbf{C}' = \mathrm{GLWE}^0_{\mathbf{S}'}(A_{k+1}) - \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j} \cdot \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j].$$

From $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j] = \mathrm{GLWE}_{\mathbf{S}'}(\frac{q}{B^j} \cdot S_i)$, let $\langle \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j], (-\mathbf{S}', 1)\rangle = \frac{q}{B^j} \cdot S_i + E_{i,j}$ where $\mathrm{Var}(E_{i,j}) = \sigma^2_{\mathbf{S}\to\mathbf{S}'}$ for $i \in [k]$ and $j \in [\ell]$. Then, one obtain

$$\begin{aligned}
\langle \mathbf{C}', (-\mathbf{S}', 1)\rangle &= A_{k+1} - \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j}\left(\frac{q}{B^j} \cdot S_i + E_{i,j}\right) \\
&= A_{k+1} - \sum_{i=1}^{k}\left((A_i + E'_i) \cdot S_i + \sum_{j=1}^{\ell} A'_{i,j} \cdot E_{i,j}\right) \\
&= \mu - \sum_{i=1}^{k} E'_i \cdot S_i + \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j} \cdot E_{i,j}.
\end{aligned}$$

Since $E'_i \leftarrow [\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$, $\|A'_{i,j}\|_\infty \leq B/2$ and $\mathbf{S}$ is a binary secret key, the variance of $E_{\mathsf{ks}} = -\sum_{i=1}^{k} E'_i \cdot S_i + \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j} \cdot E_{i,j}$ is given as follows.

$$\mathrm{Var}(E_{\mathsf{ks}}) \leq \frac{kN}{12}\left(\frac{q^2}{B^{2\ell}} - 1\right) + k\ell N\left(\frac{B}{2}\right)^2 \sigma^2_{\mathbf{S}\to\mathbf{S}'}.$$

□

C.3 EvalTrace

Given a GLWE ciphertext of $\mathbf{C}$ of M under $\mathbf{S}$, Algorithm 2 outputs a GLWE ciphertext of $\mathrm{Tr}_{N/n}(\mu)$ with an additive noise increment. The correctness of the algorithm is described in Section 2.4, and we bound the variance of the additive noise increment here.

Lemma 3 (Trace Evaluation). *Let $\mathbf{C}$ be a GLWE ciphertext of a phase μ under $\mathbf{S}$. Let V_{auto} be the variance of the noise increment by the homomorphic automorphism evaluation. Then, Algorithm 2 returns a GLWE ciphertext $\mathbf{C}'$ of a phase $\mathrm{Tr}_{N/n}(\mu)+E_{\mathsf{tr}}$ under $\mathbf{S}$ where the variance V_{tr} of E_{tr} is given as follows.*

$$V_{\mathsf{tr}} \leq \frac{(N/n)^2-1}{3} V_{\mathsf{auto}}.$$

where V_{auto} is the variance of the noise increment by homomorphic automorphism evaluation EvalAuto *in Line 3.*

Proof. Let E_d be the increased error polynomial after the d-th iteration of Line 3. Then, E_d satisfies the following relation.

$$E_d = E_{d-1} + \tau_{2^{\log N - d+1}}(E_{d-1}) + E_{\mathsf{auto},d}$$

where $E_{\mathsf{auto},d}$ is the error increment by EvalAuto in the d-th iteration. Then, one obtain

$$\mathrm{Var}(E_d) \leq 2^2 \,\mathrm{Var}(E_{d-1}) + V_{\mathsf{auto}}.$$

From $E_0 = 0$, $V_{\mathsf{tr}} = \mathrm{Var}(\log(N/n))$ satisfies the following.

$$V_{\mathsf{tr}} \leq \sum_{d=0}^{\log(N/n)-1} 4^d V_{\mathsf{auto}} \leq \frac{(N/n)^2-1}{3} V_{\mathsf{auto}}.$$

□

We note that V_{auto} can be upper bounded by Lemma 2 since EvalAuto evaluates a single GLWE keyswitching operation.

C.4 Scheme Switching

Let $\mathbf{S} = (S_1, \ldots, S_k)$ be a GLWE secret key. The scheme switching changes a GLev ciphertext $\mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(M)$ of M to a GGSW ciphertext $\mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M)$ of M using the scheme switching key $\{\mathrm{GGSW}_{\mathbf{S}}^{(B_{\mathsf{ss}},\ell_{\mathsf{ss}})}(S_i)\}_{i=1}^{k+1}$, a set of $k+1$ GGSW ciphertexts of S_i for $i = 1, \ldots, k+1$ under $\mathbf{S}$ where $S_{k+1} = -1$. The precise algorithm is given in Algorithm 6.

Lemma 4 (External Product). *Let $\mathbf{C} = \mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M)$ be a GGSW ciphertext of M having variance σ_{ext}^2 under $\mathbf{S}$ and $\mathbf{c}$ be a GLWE ciphertext of a phase μ under $\mathbf{S}$. Then, external product $\mathbf{C} \boxdot \mathbf{c}$ outputs a GLWE ciphertext of a phase $\mu \cdot M + E_{\mathsf{ext}}$ under $\mathbf{S}$ where the variance V_{ext} of E_{ext} is given as follows.*

$$V_{\mathsf{ext}} \leq \frac{(k+1)N}{12}\left(\frac{q^2}{B^{2\ell}} - 1\right)\ell_2(M)^2 + (k+1)\ell N\left(\frac{B}{2}\right)^2 \sigma_{\mathsf{ext}}^2.$$

Algorithm 6: Scheme Switching

Input: $\mathbf{C} = \mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(M)$ under $\mathbf{S} = (S_1, \ldots, S_k)$
Input: $\mathrm{SS}[i] = \mathrm{GGSW}_{\mathbf{S}}^{(B_{\mathsf{ss}},\ell_{\mathsf{ss}})}(-S_i)$ for $i = 1, \ldots, k+1$ where $S_{k+1} = -1$
Output: $\mathbf{C}' = \mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M)$ such that $\mathbf{C}'_{i,j} = \mathrm{GLWE}_{\mathbf{S}}(-\frac{q}{B^j} \cdot MS_i)$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$

$\mathbf{C}_j = \mathrm{GLWE}_{\mathbf{S}}(\frac{q}{B^j} \cdot M)$ for $j = 1, \ldots, \ell$
for $i = 1$ *to* $k+1$ **do**
  for $j = 1$ *to* ℓ **do**
    $\mathbf{C}'_{i,j} \leftarrow \mathrm{SS}[i] \boxdot \mathbf{C}_j$
return $\mathbf{C}'$

Proof. Let $\mathbf{S} = (S_1, \ldots, S_k)$ and $\mathbf{C} = (\mathbf{C}_{i,j})_{(i,j)\in[k+1]\times[\ell]}$ such that

$$\mathbf{C}_{i,j} = \mathrm{GLWE}_{\mathbf{S}}\left(\frac{q}{B^j}(-S_i \cdot M)\right)$$

where $\langle \mathbf{C}_{i,j}, (-\mathbf{S}, 1)\rangle = \frac{q}{B^j}(-S_i \cdot M) + E_{i,j}$ and $\mathrm{Var}(E_{i,j}) = \sigma^2_{\mathsf{ext}}$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$. Let $\mathbf{c} = (A_1, \ldots, A_{k+1})$ and

$$A_i = \sum_{j=1}^{\ell} A'_{i,j} \cdot \frac{q}{B^j} + E'_i$$

be the gadget decomposition of A_i such that $\|A'_{i,j}\|_\infty \le \frac{B}{2}$ and $\|E'_{i,j}\|_\infty \le \frac{q}{2B^\ell}$ for $i = 1, \ldots, k+1$. Then the output of external product $\mathbf{C} \boxdot \mathbf{c}$ can be represented as $\sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j} \cdot \mathbf{C}_{i,j}$. Then, the phase of the output is given as follows.

$$\begin{aligned}
\langle \mathbf{C} \boxdot \mathbf{c}, (-\mathbf{S}, 1)\rangle &= \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j}\left(\frac{q}{B^j}(-S_i \cdot M) + E_{i,j}\right) \\
&= \sum_{j=1}^{\ell} A'_{k+1,j} \cdot \left(\frac{q}{B^j}M + E_{k+1,j}\right) - \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j}\left(\frac{q}{B^j}M - E_{i,j}\right) \\
&= \mu \cdot M + \left(E'_{k+1} - \sum_{i=1}^{k} E'_i S_i\right) M + \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j}E_{i,j}.
\end{aligned}$$

Let $E_{\mathsf{ks}} = (E'_{k+1} - \sum_{i=1}^{k} E'_i S_i)M + \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j}E_{i,j}$. Since $E'_i \leftarrow [\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$, $\|A'_{i,j}\|_\infty \le \frac{B}{2}$ and $\mathbf{S}$ is a binary secret key, the variance of E_{ks} is given as follows.

$$\mathrm{Var}(E_{\mathsf{ks}}) \le \frac{(k+1)N}{12}\left(\frac{q^2}{B^{2\ell}} - 1\right)\ell_2(M)^2 + (k+1)\ell N\left(\frac{B}{2}\right)^2 \sigma^2_{\mathsf{ext}}.$$

□

Since scheme switching computes the output GGSW ciphertext using external output by the scheme switching key, the noise increment of scheme switching can be analyzed by Lemma 4.

Lemma 5 (Scheme Switching). *Let* $\mathbf{C}$ *be a GLev ciphertext of* M *having variance* σ^2_{in}. *Let* σ^2_{ssk} *be the noise variance of the scheme switching key. Then, Algorithm 6 returns a GGSW ciphertext* $\mathbf{C}'$ *of* M *having variance* V_{out} *such that* $V_{\mathsf{out}} \leq N\sigma^2_{\mathsf{in}} + V_{\mathsf{ss}}$ *where*

$$V_{\mathsf{ss}} \leq \frac{(1+kN)N}{12}\left(\frac{q^2}{B^{2\ell_{\mathsf{ss}}}} - 1\right) + (k+1)\ell N\left(\frac{B_{\mathsf{ss}}}{2}\right)^2 \sigma^2_{\mathsf{ssk}}.$$

D Failure Probability of PBSmanyLUT for WoP-PBS after Refreshing

By employing the refreshing operation before PBSmanyLUT, we can fix the input noise variance to PBSmanyLUT by $V_{\mathsf{refresh}} + V_{\mathsf{ks}}$ where V_{refresh} denotes the variance of the output of the refreshing operation and V_{ks} denotes the variance of the noise increment of the LWE keyswitching operations. Since the refreshing operation is a PBS operation with its own gadget decomposition parameters, the output variance can be computed by (6) with replacing the gadget decomposition parameter by $(B_{\mathsf{refresh}}, \ell_{\mathsf{refresh}})$.

When it comes to the LWE keyswitching, the variance V_{ks} of its noise increment can be upper bounded as follow by applying Theorem 2 in [13].[16]

$$V_{\mathsf{ks}} \leq kN\ell_{\mathsf{ks}}\sigma^2_{\mathbf{s}'\to\mathbf{s}}\left(\frac{B^2_{\mathsf{ks}}}{12} + \frac{1}{6}\right) + \frac{kN}{24}\left(\frac{q^2}{B^{2\ell_{\mathsf{ks}}}_{\mathsf{ks}}} + \frac{1}{2}\right)$$

where $\sigma^2_{\mathbf{s}'\to\mathbf{s}}$ is the noise variance of the LWE keyswitching key from $\mathbf{s}' \in \mathbb{B}^{kN}$ to $\mathbf{s} \in \mathbb{B}^n$ with the decomposition parameters $(B_{\mathsf{ks}}, \ell_{\mathsf{ks}})$.

	$\sigma^2_{\mathsf{refresh}}$	σ^2_{ks}	σ^2_{in}	ϑ	Γ	Failure Prob.
WOPBS_2_2	$2^{97.01}$	$2^{113.03}$	$2^{113.03}$	2	31.815	$2^{-735.48}$
WOPBS_3_3	$2^{97.19}$	$2^{115.35}$	$2^{115.35}$	2	18.132	$2^{-241.66}$
WOPBS_4_4	$2^{97.32}$	$2^{113.25}$	$2^{113.25}$	3	18.235	$2^{-244.38}$

Table 8: The noise variance and the failure probability of PBSmanyLUT after refreshing to evaluate LWE to Lev conversion in a single PBS operation in our improved WoP-PBS method. The polynomial size N is 2048 and the ciphertext modulus q is 2^{64}.

[16] More precisely, Theorem 2 in [13] gives an upper bound for the noise variance of LWE to GLWE packing keyswitching, which can be applied to LWE to LWE keyswitching.

Now, one can estimate the failure probability of PBSmanyLUT as follows. Suppose that an LWE ciphertext of input noise variance σ_{in}^2 and scaling factor Δ_{in} is given to PBSmanyLUT evaluating 2^{ϑ} LUTs. The failure probability of PBSmanyLUT is at most $P = 1 - \mathsf{erf}\left(\frac{\Gamma}{\sqrt{2}}\right)$, where $\Gamma = \frac{\omega\Delta_{\mathsf{in}}}{2q\sigma}$ and

$$\sigma^2 = \frac{\omega^2}{q^2}\left(\sigma_{\mathsf{in}}^2 - \frac{1}{12}\right) + \frac{n}{48}\left(\frac{\omega^2}{q^2} + 2\right) + \frac{1}{12}$$

for $\omega = 2N \cdot 2^{-\vartheta}$. Using $\sigma_{\mathsf{in}}^2 = V_{\mathsf{refresh}} + V_{\mathsf{ks}}$ for the PBSmanyLUT operation after the refreshing, its failure probability can be obtained by above. Table 8 summarizes the result.

E Details on AES Evaluation

In this section, we describe the details of our AES implementation by TFHE and its experimental results of the error growth. We also give an analysis for the parameters, called `CMUX1`, used for the AES evaluation in [24].

E.1 AES Round Function Evaluation

Since the AES circuit is basically a repetition of its round function, it is enough to describe how to evaluate the AES round function. The AES round function consists of SubBytes, ShiftRows, MixColumns and AddRoundKey.

LWE Keyswitching Prior to SubBytes, one has to perform LWE keyswitching on the LWE ciphertexts to use PBS for SubBytes evaluation. Although the LWE keyswitching operation takes a smaller computation time compared to the PBS operation, one cannot simply neglect it. Instead of using the previous LWE keyswitching method, we employ the following optimization based on GLWE dimension switching, which is an extension of the method proposed by Chen et al. [7] based on GLWE keyswitching.

Consider LWE keyswitching from an LWE secret key $\mathbf{s}_{\mathsf{src}} \in \mathbb{Z}_q^{n_{\mathsf{src}}}$ to another one $\mathbf{s}_{\mathsf{dst}} \in \mathbb{Z}_q^{n_{\mathsf{dst}}}$ where there is a power-of-two N such that N divides both n_{src} and n_{dst}. Then, there are corresponding GLWE secret keys $\mathbf{S}_{\mathsf{src}}$ and $\mathbf{S}_{\mathsf{dst}}$ to $\mathbf{s}_{\mathsf{src}}$ and $\mathbf{s}_{\mathsf{dst}}$ where their GLWE dimensions are $k_{\mathsf{src}} = n_{\mathsf{src}}/N$ and $k_{\mathsf{dst}} = n_{\mathsf{dst}}/N$, respectively. Using the GLWE keyswitching from $\mathbf{S}_{\mathsf{src}}$ to $\mathbf{S}_{\mathsf{dst}}$, one can perform LWE keyswitching as follows.

1. Given an input LWE ciphertext $\mathbf{c}$ of m under $\mathbf{s}_{\mathsf{src}}$, one computes a GLWE ciphertext $\mathbf{C}$ of $m + v_1X + \cdots + v_{N-1}X^{N-1}$ under $\mathbf{S}_{\mathsf{src}}$ by LWEtoGLWEConst where $v_1, \ldots, v_{N-1}$ are unknown coefficients.
2. The GLWE ciphertext $\mathbf{C}$ under $\mathbf{S}_{\mathsf{src}}$ is switched to a GLWE ciphertext $\mathbf{C}'$ of the same plaintext $m + v_1X + \cdots + v_{N-1}X^{N-1}$ under the different key $\mathbf{S}_{\mathsf{dst}}$ by GLWE keyswitching.

3. Then, an LWE ciphertext $\mathbf{c}'$ of m under $\mathbf{s}_{\mathsf{dst}}$ can be extracted from the GLWE ciphertext $\mathbf{C}'$ under $\mathbf{S}_{\mathsf{dst}}$.

Chen et al. [7] have proposed the above method as an efficient LWE keyswitching only for the case where $n_{\mathsf{src}} = n_{\mathsf{dst}} = N$, while our extended algorithm using GLWE dimension switching enables to change LWE dimension if there is a common power-of-two divisor N of the input and output LWE dimensions. To employ this optimization, we choose $n = 768 = 3 \cdot 256$ for the input LWE dimension of PBS.

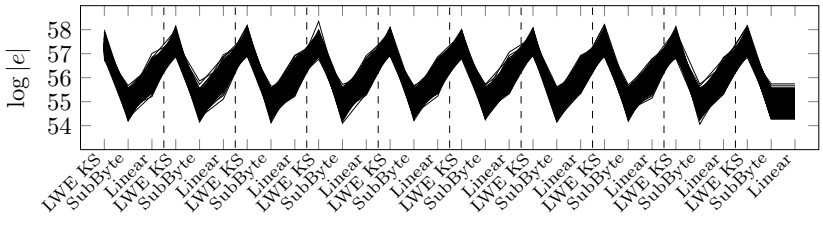
SubBytes The AES S-box is evaluated using the GGSW ciphertext of the input bits obtained by our patched version of the WWL+ method. To keep the plaintext encoding, the AES S-box from 8-bit input to 8-bit output is decomposed into the corresponding 8 tables of 8-bit input to 1-bit output. Then, 8 GGSW ciphertexts of 8 input bits to the AES S-box can evaluate the decomposed table by external product, obtaining 8 LWE ciphertexts of the corresponding 8 output bits to the AES S-box. The most computational overhead in SubBytes comes from the circuit bootstrapping of the 128 input LWE ciphertexts.

Linear Operations Since XOR operation is free under the plaintext encoding that places a single bit plaintext in the MSB of the ciphertext, the other operations such as ShiftRows, MixColumns and AddRoundKey that only require XOR operations can be evaluated freely. That said, we note that homomorphic XOR operation is free *only in terms of computation time*, so the error growth by the linear operations should be considered in the selection of parameters.

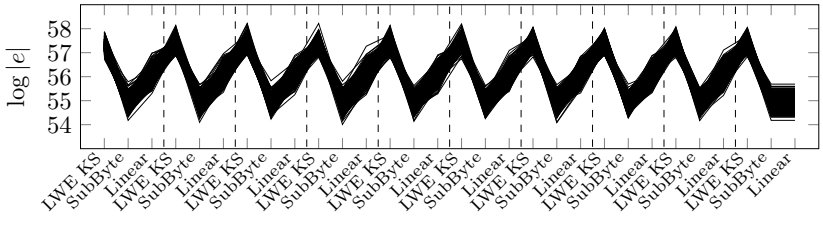
Experimental Result of the Error Growth The experimental result of the error growth in our AES evaluation is given in Figure 3. The graphs show the maximum error of the 128 LWE ciphertexts obtained by each step of AES evaluation for the parameter sets used in our implementation (see Table 6). Each line denotes a single experimental result among 10,000 measurements. One can find that there is non-negligible error growth in the linear operations[17] and the LWE keyswitching error dominates the others.

Failure Probability Based on the observation from the experimental result, we compute the failure probability of our AES implementation as follows. Since the error growth is dominated by the LWE keyswitching, we can upper-bound the noise variance of the input LWE ciphertext to PBS by $\sigma_{\mathsf{in}}^2 \leq 2V_{\mathsf{ks}}$ where V_{ks} denotes the variance of the noise increment of the LWE keyswitching operation. As we implement the LWE keyswitching based on GLWE keyswitching, V_{ks} can be computed by Lemma 2. The failure probability of our AES evaluation in Table 6 is derived from multiplying that of the $\mathsf{PBSmanyLUT}$ operation by the

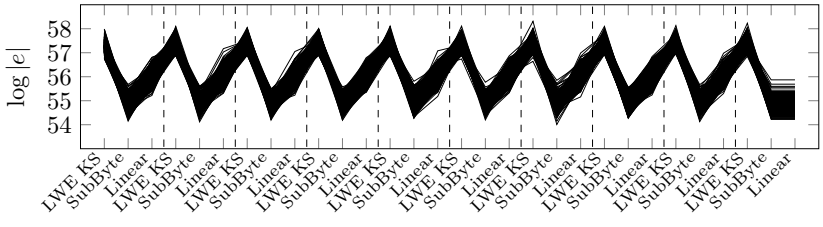
[17] There is almost no error growth in the final round since it does not contain MixColumns.



(a) Set-I



(b) Set-II



(c) Set-III

Fig. 3: The experimental result of the error growth for our AES implementation.

number of PBSmanyLUT operations in the AES evaluation, which is 1280. Table 9 shows the variances V_{ks} and σ^2_{in}, and the corresponding failure probabilities for a single PBSmanyLUT operation and a single AES evaluation.

	N	ϑ	Noise Variance		Failure Prob.	
			V_{ks}	σ^2_{in}	PBSmanyLUT	AES
Set-I	2048	2	$2^{112.95}$	$2^{113.95}$	$2^{-507.64}$	$2^{-497.32}$
Set-II	1024	2	$2^{112.95}$	$2^{113.95}$	$2^{-252.95}$	$2^{-242.63}$
Set-III	512	2	$2^{112.95}$	$2^{113.95}$	$2^{-85.93}$	$2^{-75.60}$

Table 9: The noise variance and failure probability for the recommended sets of parameters in our AES implementation. The ciphertext modulus q is 2^{64}.

E.2 Error Analysis for the Parameters used in WWL+

In this section, we give the detailed error analysis for the AES evaluation using `CMUX1` parameters *by following the estimation method used in* [24]. Before going on, we note that the WWL+ method in [24] is implemented in a FHEW-like setting that uses two kinds of ciphertext moduli: a small modulus $\underline{q}$ for the input LWE ciphertext to PBS and a large modulus $\bar{Q}$ for the output ciphertext to PBS. For the `CMUX1` parameter set, $\underline{q} = 2^{10}$ and $\bar{Q} \approx 2^{54}$ are used (the exact value of $\bar{Q}$ is not given).

Let σ^2_{ggsw} be the output variance of the WWL+ circuit bootstrapping method. Although there is a flaw in the error analysis related to the scheme switching, we simply estimate $\sigma^2_{\mathsf{ggsw}} = \sigma^2_{\mathsf{pbs}}$ since the PBS error dominates the others.[18] According to their estimation, σ^2_{pbs} is given as follows.

$$\sigma^2_{\mathsf{pbs}} = 2n\left(\frac{1}{6}N\ell_{\mathsf{pbs}}B^2_{\mathsf{pbs}} \cdot \sigma^2_{\mathsf{bsk}} + \frac{1}{3}(N+1)\varepsilon^2_{\mathsf{pbs}}\right)$$

where σ^2_{bsk} is the noise variance of the PBS key and $\varepsilon_{\mathsf{pbs}}$ is the gadget decomposition error for PBS, which is defined by

$$\varepsilon = \frac{q}{2B^\ell}$$

for the corresponding decomposition parameters (B, ℓ).

They consider the additive error growth of external product by a GGSW ciphertext of a noise variance σ^2_{ggsw} as follows.

$$\sigma^2_{\mathsf{add}} = \frac{1}{12}N\ell_{\mathsf{cbs}}B^2_{\mathsf{cbs}} \cdot \sigma^2_{\mathsf{ggsw}} + \frac{1}{6}(N+1)\varepsilon^2_{\mathsf{cbs}}$$

[18] To be precise, [24] uses blind rotation without sample extraction. We simply denote it by PBS as described in Section 4.1.

where $(B_{\mathsf{cbs}}, \ell_{\mathsf{cbs}})$ is the decomposition parameter of the GGSW ciphertext and $\varepsilon_{\mathsf{cbs}}$ is the corresponding decomposition error. Then, evaluating t-bit LUT induces an additive error of variance $t\sigma^2_{\mathsf{add}}$.

To perform circuit bootstrapping again to the output of t-bit LUT, there are several pre-switching required such as LWE keyswitching, (non-sparse) modulus switching from $\bar{Q}$ to $\underline{q}$, and spares modulus switching from $\underline{q}$ to $2N$. Only considering the sparse modulus switching error,[19] the failure probability of $\mathsf{PBSmanyLUT}$ is estimated by $1 - \mathrm{erf}(\frac{2N}{4\sqrt{2}\sigma})$ where

$$\sigma^2 = \left(\frac{2N}{\bar{Q}}\right)^2 \cdot (t\sigma^2_{\mathsf{add}}) + \left(\frac{n}{2} + 1\right) \cdot \left(\frac{2^{2\vartheta}}{12} - \frac{N^2}{3\underline{q}^2}\right) + \frac{nN^2}{4\underline{q}^2}.$$

Due to the summation of at most 7 output bits of the AES S-box by MixColumns, each of which noise variance is $8\sigma^2_{\mathsf{add}}$, it is reasonable to consider the noise variance of the input ciphertext to the next circuit bootstrapping is $56\sigma^2_{\mathsf{add}}$. For the `CMUX1` parameters with $\vartheta = 2$ and $t = 56$, the failure probability is given by $2^{-8.988}$. Considering a single AES evaluation requires $1280 \approx 2^{10.322}$ $\mathsf{PBSmanyLUT}$ operations, we can see that the `CMUX1` parameters cannot guarantee a correct AES evaluation result in practice.

F Evaluation Key Size

In this section, we describe the size of various evaluation keys used in TFHE according to the parameters, summarizing the result in Table 10. As evaluation keys are encryptions of secret information, we begin with the description of ciphertext sizes.

Ciphertext Size An LWE ciphertext $(a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$ consists of $n+1$ elements in $\mathbb{Z}_q$, so its size is given by $(n+1)\log q$ bits. If the LWE ciphertext is a fresh one such that no homomorphic operation is performed on it yet, then one can compress the random mask a into a seed for generating it. Such LWE ciphertexts are called seeded LWE ciphertexts. Ignoring the seed size by assuming that one seed generates all the random masks for multiple seeded ciphertexts, the size of the seeded LWE ciphertext is only $\log q$.[20] In the case of a GLWE ciphertext $(A_1, \ldots, A_k, B) \in \mathcal{R}_{q,N}^{k+1}$, it is size of $(k+1)N\log q$ bits. When it is compressed similarly, the seeded GLWE ciphertext is of $N\log q$ bits. For GLev and GGSW ciphertexts, they can be considered as a vector of ℓ and $\ell(k+1)$ GLWE ciphertexts, respectively. Table 10a summarizes the size of each type of TFHE ciphertext.

[19] The non-sparse modulus switching error is small enough. In the case of the LWE keyswitching, its gadget decomposition parameters are not given in [24].

[20] In the `tfhe-rs` library, auxiliary information such as the LWE dimension or ciphertext modulus type is saved together. We ignore such additional data size assuming that it is fixed in the transciphering framework.

GLWE Keyswitching Key A GLWE keyswitching key from a key $\mathbf{S}_{\mathsf{src}} \in \mathcal{R}_{q,N}^{k_{\mathsf{src}}}$ of dimension k_{src} to another key $\mathbf{S}_{\mathsf{dst}} \in \mathcal{R}_{q,N}^{k_{\mathsf{dst}}}$ of dimension k_{dst} with the same polynomial size N is a set of k_{src} GLev ciphertexts $\{\mathrm{GLev}_{\mathbf{S}_{\mathsf{dst}}}^{(B_{\mathsf{ks}},\ell_{\mathsf{ks}})}(S_i)\}_{i=1}^{k_{\mathsf{src}}}$ where $\mathbf{S}_{\mathsf{src}} = (S_1, \ldots, S_{k_{\mathsf{src}}})$.

Trace Evaluation Key A trace evaluation key on a GLWE secret key $\mathbf{S} \in \mathcal{R}_{q,N}^{k}$ of dimension k is a set of $\log N$ automorphism keys, each of which is a GLWE keyswitching key on the same GLWE dimension k and a gadget decomposition parameters of $(B_{\mathsf{auto}}, \ell_{\mathsf{auto}})$.

Scheme Switching Key A scheme switching key on a GLWE secret key $\mathbf{S} \in \mathcal{R}_{q,N}^{k}$ of dimension k is a set of k GGSW ciphertexts $\{\mathrm{GGSW}_{\mathbf{S}}^{(B_{\mathsf{ss}},\ell_{\mathsf{ss}})}(S_i)\}_{i=1}^{k}$ where $\mathbf{S} = (S_1, \ldots, S_k)$.

Packing Keyswitching Key A packing keyswitching key from a LWE secret key $\mathbf{s} \in \mathbb{Z}_q^n$ of dimension n to a GLWE secret key $\mathbf{S} \in \mathcal{R}_{q,N}^{k}$ of dimension k is a set of GLev ciphertext $\{\mathrm{GLev}_{\mathbf{S}}(s_i)\}_{i=1}^{n}$. Table 10b summarizes the evaluation key size.

PBS Key Let $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{B}^n$ be an LWE secret key and $\mathbf{S}' = (S'_1, \ldots, S'_k) \in \mathbb{B}_N[X]^k$ be a GLWE secret key with its corresponding LWE secret key $\mathbf{s}' \in \mathbb{B}^{kN}$. A PBS key is from $\mathbf{s}$ to $\mathbf{s}'$ a set of n GGSW ciphertexts $\{\mathrm{GGSW}_{\mathbf{S}'}^{(B_{\mathsf{pbs}},\ell_{\mathsf{pbs}})}(s_i)\}_{i=1}^{n}$. Since the PBS operation takes an input LWE ciphertext under a different LWE secret key, one needs a corresponding LWE keyswitching key for the PBS operation, which is a set of kN Lev ciphertexts $\{\mathrm{Lev}_{\mathbf{s}}^{(B_{\mathsf{ks}},\ell_{\mathsf{ks}})}(s'_i)\}_{i=1}^{kN}$.

Circuit Bootstrapping Key Let $\mathbf{s} \in \mathbb{B}^n$, $\mathbf{S} \in \mathbb{B}_N[X]^k$ and $\mathbf{s}' \in \mathbb{B}^{kN}$ be defined the same as above. The (previous) circuit bootstrapping takes an input LWE ciphertext under $\mathbf{s}$ and outputs a corresponding GGSW ciphertext under $\mathbf{S}$ using a sequence of PBS operations and private functional keyswitching operations. The private functional keyswitching operation for the circuit bootstrapping, which switches an LWE ciphertext $\mathrm{LWE}_{\mathbf{s}}(m)$ into a GLWE ciphertext $\mathrm{GLWE}_{\mathbf{S}}(-S_i \cdot m)$ for $i = 1, \ldots, k+1$, requires a set of $k+1$ GLev ciphertexts $\{\mathrm{GLev}_{\mathbf{S}}^{(B_{\mathsf{priv}},\ell_{\mathsf{priv}})}(-S_i)\}_{i=1}^{k+1}$ where $\mathbf{S} = (S_1, \ldots, S_k)$ and $S_{k+1} = -1$.[21] Table 10c summarizes the evaluation keys for the bootstrapping operations in TFHE.

G Detailed Parameter Sets

In this section, we give detailed information on the parameters used in this paper. All the parameters come from the recommended parameters of the `tfhe-rs`

[21] To be precise, the private keyswitching from LWE(m) to GLWE($-S_{k+1} \cdot m$) is a packing keyswitching since $-S_{k+1} \cdot m = m$.

	LWE	Lev	GLWE	GLev	GGSW
Normal	$(n+1)\log q$	$\ell(n+1)\log q$	$(k+1)N\log q$	$\ell(k+1)N\log q$	$\ell(k+1)^2N\log q$
Seeded	$\log q$	$\ell\log q$	$N\log q$	$\ell N\log q$	$\ell(k+1)N\log q$

(a) Size of TFHE ciphertexts in bits.

	GLWE KS Key	Trace Evaluation Key	Scheme Switching Key	Packing KS Key
Normal	$\ell_{\mathsf{ks}}k_{\mathsf{src}}(k_{\mathsf{dst}}+1)N\log q$	$\ell_{\mathsf{auto}}k(k+1)N\log N\log q$	$\ell_{\mathsf{ss}}k(k+1)^2N\log q$	$\ell_{\mathsf{pack}}n(k+1)N\log q$
Seeded	$\ell_{\mathsf{ks}}k_{\mathsf{src}}N\log q$	$\ell_{\mathsf{auto}}kN\log N\log q$	$\ell_{\mathsf{ss}}k(k+1)N\log q$	$\ell_{\mathsf{pack}}nN\log q$

(b) Size of various TFHE evaluation keys in bits.

	LWE KS Key	PBS Key	Private Functional KS Key
Normal	$\ell_{\mathsf{ks}}(n+1)kN\log q$	$\ell_{\mathsf{pbs}}(k+1)^2nN\log q$	$\ell_{\mathsf{priv}}k(k+1)^2N^2\log q$
Seeded	$\ell_{\mathsf{ks}}kN\log q$	$\ell_{\mathsf{pbs}}(k+1)nN\log q$	$\ell_{\mathsf{priv}}k(k+1)N^2\log q$

(c) Size of evaluation keys for the TFHE bootstrapping operations in bits. The PBS operation requires the LWE keyswitching key and the PBS key, and the circuit bootstrapping operation requires all kinds of keys in the table.

Table 10: Size of TFHE ciphertexts and evaluation keys in bits. The size of seeds or auxiliary information is ignored.

library only except the parameters used for AES evaluation in Section 4.3. Table 11 shows the detailed information on the recommended parameter sets of the `tfhe-rs` library used in this paper.

Parameter Sets	LWE		GLWE			Gadget Decomp. Parameters			
	n	σ_{LWE}	N	k	σ_{GLWE}	$(B_{\mathsf{ks}}, \ell_{\mathsf{ks}})$	$(B_{\mathsf{pbs}}, \ell_{\mathsf{pbs}})$	$(B_{\mathsf{priv}}, \ell_{\mathsf{priv}})$	$(B_{\mathsf{cbs}}, \ell_{\mathsf{cbs}})$
PARAM_MESSAGE_2_CARRY_2_KS_PBS	742	0.00000769849454709433	2048	1	0.00000000000000029403601535432533	$(2^3, 5)$	$(2^{23}, 1)$	-	-
PARAM_MESSAGE_3_CARRY_3_KS_PBS	864	0.00000757998020150446	8192	1	0.00000000000000000002168404344971009	$(2^3, 6)$	$(2^{15}, 2)$	-	-
PARAM_MESSAGE_4_CARRY_4_KS_PBS	996	0.00000006767666038309478	32768	1	0.00000000000000000002168404344971009	$(2^3, 7)$	$(2^{15}, 2)$	-	-
WOPBS_PARAM_MESSAGE_2_CARRY_2_KS_PBS	769	0.000004313154647504185	2048	1	0.00000000000000029403601535432533	$(2^6, 2)$	$(2^{15}, 2)$	$(2^{15}, 2)$	$(2^5, 3)$
WOPBS_PARAM_MESSAGE_3_CARRY_3_KS_PBS	873	0.000000642879711284378 9	2048	1	0.00000000000000029403601535432533	$(2^{10}, 1)$	$(2^9, 4)$	$(2^9, 4)$	$(2^6, 3)$
WOPBS_PARAM_MESSAGE_4_CARRY_4_KS_PBS	953	0.000000148673396941 1098	2048	1	0.00000000000000029403601535432533	$(2^{11}, 1)$	$(2^9, 4)$	$(2^9, 4)$	$(2^4, 6)$

Table 11: The detailed information of the recommended sets of parameters of the `tfhe-rs` library used in this paper. The ciphertext modulus q is 2^{64} for all the parameters.