FHEW-like Leveled Homomorphic Evaluation: Refined Workflow and Polished Building Blocks

Ruida Wang[1,2⋆], Jincheol Ha[3⋆], Xuan Shen[1,2], Xianhui Lu[1,2(✉)⋆⋆], Chunling Chen[1,2], Kunpeng Wang[1,2], and Jooyoung Lee[3(✉)⋆⋆⋆]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{wangruida,shenxuan,luxianhui,chenchunling,wangkunpeng}@iie.ac.cn

[2] School of Cybersecurity, University of Chinese Academy of Sciences, Beijing, China

[3] Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
{smilecjf,hicalf}@kaist.ac.kr

Abstract. In FHEW-like cryptosystems, the leveled homomorphic evaluation (LHE) mode performs bootstrapping after circuit evaluation rather than after each gate. The core procedure and the performance bottleneck are known as circuit bootstrapping (CBS). This paper revisits the LHE mode by refining the workflow and proposing polished building blocks:

1. Algorithmic Enhancements
   - We introduce an NTT-based CBS algorithm, patched from WWL+ [Eurocrypt24], achieving up to a 2.9× efficiency improvement.
   - We present an FFT-based CBS that is 3.5× faster than the most efficient FFT-CBS implementation, with a key size reduction of 37.5×.
2. Refined Leveled Homomorphic Evaluation and Applications
   - We propose the HalfCBS algorithm, which is 2.4× faster than the traditional CBS algorithm, enabling a flexible leveled evaluation.
   - By applying these improvements to evaluate general look-up tables (LUTs), we can evaluate a 16-8 LUT in 311 ms, which is $2^{13.61}\times$ faster than the trivial FHE mode implementation.
   - When applied to AES transciphering, our enhancements yield a 2.9× improvement in efficiency and a 31.6× reduction in key size compared to the state of the art, Thunderbird [TCHES24].
3. High-Precision LHE
   - To handle multi-bit inputs, we propose a high-precision LHE framework. Compared to the WoP-PBS [JoC23], the compute efficiency (resp. key size) is improved by factors from 9.7 to 16.2 (resp. 3.4 to 4.4) according to the parameters.

Keywords: Homomorphic Encryption, FHEW/TFHE, Leveled Homomorphic Evaluation, Circuit Bootstrapping, High-Precision

⋆ Both authors contributed equally to this work.

⋆⋆ Xianhui Lu was supported by Chinese Academy of Sciences Project for Young Scientists in Basic Research [NO.YSBR-035].

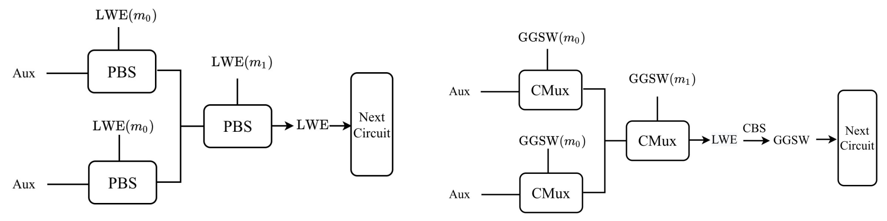
⋆⋆⋆ Jooyoung Lee was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.2022-0-01047, Development of statistical analysis algorithm and module using homomorphic encryption based on real number operation].

1 Introduction

Fully Homomorphic Encryption (FHE) has long been a cornerstone in the field of privacy-enhancing technologies, enabling computations on encrypted data without decryption. TFHE [10, 11, 12], proposed as an extension of FHEW [18], is a homomorphic encryption scheme that supports fast bootstrapping to refresh noise. Recently, Micciancio et al. unified these schemes into a single framework of FHEW-like cryptosystems [23].

FHEW-like cryptosystems are friendly to Boolean evaluation, where each gate is integrated with a fast bootstrapping operation. This concept later be extended to the evaluation of negacyclic univariate functions, known as functional bootstrapping [6] or programmable bootstrapping [14] (PBS[4]). The evaluation strategy using PBS, also referred to as FHE mode, combines function evaluation (Eval.) and noise refreshing (Refr.) steps, making it user-friendly by eliminating the need to manage noise growth. However, it faces a computational burden when dealing with large-scale circuits due to the huge number of bootstrapping.

To deal with this issue, Chillotti et al. [14] proposed the Leveled Homomorphic Evaluation (LHE) mode for FHEW-like cryptosystem that decouples gate evaluation from bootstrapping. The LHE mode uses external product to evaluate controlled selector gates (CMux) to build the circuits, reducing the complexity of the evaluation step from $O(nN\log N)$[5] to $O(N\log N)$ per gate, achieving hundreds times improvement. However, in LHE mode, the input and output of the circuit evaluations are two different ciphertext types, which destroys its composability. Therefore, unlike FHE mode, LHE mode requires both a refreshing step and a conversion (Conv.) step to transform the ciphertext form. The current evaluation methods [11, 14, 27], as shown in Figure 1, combine them into a circuit bootstrapping (CBS) algorithm.



(a) FHE: $O(nN\log N)$ per gate (b) LHE: $O(N\log N)$ per gate

Fig. 1: Toy examples to evaluate a 2-digit LUT in the FHE mode and LHE mode.

Despite the advancement in the evaluation step, the high cost of circuit bootstrapping in the LHE mode diminishes the efficiency. To provide a clearer demonstration, we divide the fully homomorphic evaluation into three steps in Table

[4] In this paper, we use PBS as the abbreviation for this technique.

[5] n denotes the Learning with Error (LWE) dimension and N denotes the ring dimension in General LWE (GLWE).

Mode	Refr.			Conv.			Eval. (per Gate)		
	Method	NTT/FFTs	Storage	Method	NTT/FFTs	Storage	Method	NTT/FFTs	Storage
FHE		-			-		PBS	$O(nN \log N)$	$O(nN)$
LHE[11]	ℓ PBS	$O(nN \log N)$	$O(nN)$	Private KS	$O(N^2)$	$O(N^2)$	CMux	$O(N \log N)$	-
LHE[14]	PBSmanyLUT	$O(nN \log N)$	$O(nN)$	Private KS	$O(N^2)$	$O(N^2)$	CMux	$O(N \log N)$	-
LHE[27]	PBSmanyLUT	$O(nN \log N)$	$O(nN)$	HomTrace, SS	$O(N \log^2 N)$	$O(N \log N)$	CMux	$O(N \log N)$	-

Table 1: Dividing Bit-Wise FHEW-like solutions in a novel perspective. We divids the circuit bootstrapping in the LHE mode into Refr. and Conv. steps.

1, with more comprehensive content in Table 2. As shown in Table 1, while the LHE mode reduces the computational complexity of the Eval. step, the storage and computational costs of the Refr. and Conv. steps are significantly higher.

There are two works representing milestone breakthroughs for the Refr. and Conv. steps: Chillotti et al. [14] introduced the PBS evaluating multiple look-up tables (PBSmanyLUT), reducing the computational cost of the Refr. step from ℓ PBS to that of a single PBS, where ℓ denotes the gadget length of the GSW ciphertext[6]. Building on this, Wang et al. [27] proposed a faster and smaller circuit bootstrapping framework using homomorphic trace evaluation (HomTrace) [8] and scheme switching (SchemeSwitch) [16] to replace private key switching during the Conv. step (called WWL+ method in this paper). This method reduces the computational complexity (rsp. storage) of the Conv. step from $O(N^2)$ to $O(N \log^2 N)$ (rsp. $O(N \log N)$), rendering it negligible compared to the Refr. step in terms of both computation time and evaluation key size.

However, the LHE mode still faces several challenges:

A **The Phase Amplification Issue in WWL+**[7]: The circuit bootstrapping algorithm proposed by Wang et al. encounters phase amplification issues during the Conv. Step. Specifically, HomTrace amplifies the noise generated in the Refr. Step by a factor of N. This necessitates their work increasing the gadget length ℓ_{ep} in the Refr. step to manage the noise and achieve the desired decryption failure probability. This adjustment will significantly impact the efficiency of the circuit bootstrapping (see Appendix A).

B **The Discrepancy Between Asymptotic Complexity and Concrete Cost**: WWL+ leveraged the inverse of N to improve the Conv. step, restricting this method to NTT-based systems. However, there is a gap between their theoretical optimizations and the implementation performance. This is due to the concrete cost of NTT is higher than FFT for the large modulus sizes in LHE mode (54-bit in NTT setting, 64-bit in FFT setting), although they have a same asymptotic complexity[8]. Consequently, the CBS implementation in WWL+ only achieves a 1.31× speedup compared to the FFT-based

[6] There is another method called multiple-value programmable bootstrapping (MV-PBS) [6] that enables multiple look-up tables, but it is not suitable in CBS that evaluates several functions of different scaling factors.

[7] The authors of [27] confirmed this issue and presented adjusted parameter sets and performance at Eurocrypt 2024 conference.

[8] For small-sized modulus, NTT may have an advantage due to its better parallelism.

TFHEpp CBS (see Table 9 in [27]). Considering issue A, this improvement further reduces to 1.1×, failing to deliver their theoretical performance.[9]

C **The Inflexibility and Computational Intensity Problem**: In the current LHE framework [11, 14, 27], the Refr. and Conv. steps are bonded executed, known as circuit bootstrapping. It leads to a inflexibility and unnecessary overhead for evaluating small-scale circuits. Specifically, even if the Eval. step computes only one gate, it still requires consecutive execution of the heavy Refr. step with the Conv. step.

D **User-Unfriendly Concerns**: The LHE mode requires more computational steps compared to the FHE mode. This often necessitates expert intervention to configure the various parameters, balancing noise management and efficiency. This complexity also makes it challenging to provide a clear and comparative evaluation of the efficiency benefits of LHE mode over FHE mode. As a result, few developers opt to use the LHE mode when building applications based on the FHEW-like systems.

E **A Lack of Efficient High-Precision LHE Solution:** In LHE mode, circuits are constructed by CMux gates, which limits the inputs to bit-wise ciphertext. However, real-world applications often encode data into integer (or multi-bit chunks) rather than single bits. To process high-precision inputs in the LHE mode, Bergerat et al. proposed the new WoP-PBS (programmable bootstrapping without padding) algorithm in JoC23 [3]. It involves to extract (Extr.) and convert multi-bit data into single bit GGSWs for CMux evaluation, and support higher precision to handle multi-bit plaintext spaces. Despite its capabilities, the WoP-PBS algorithm itself is too heavy, so that it is better than previous works [21, 14] only for large-size input LUT (e.g. larger than 8-bit) [3]. For instance, processing 8-bit inputs with WoP-PBS takes more than 4,400 ms and requires a 375 MB key size.

1.1 Our Contributions

In this paper, we address the aforementioned issues and comprehensively enhance the efficiency of the LHE mode in the FHEW-like cryptosystems. For *algorithms*, we propose two improved CBS algorithms by introducing Contributions I and II. These two algorithms are complementary, achieving optimal performance in the NTT and FFT domains, respectively. For LHE evaluation and *applications*, we design a flexible LHE framework (Contribution III). This framework, combined with our improved CBS algorithm, is applied to general homomorphic look-up tables and AES transciphering, achieving the best practical performance to date, as detailed in Contribution IV. Finally, to address the newly identified challenge of handling *high-precision* inputs in LHE mode, we propose the HP-LHE framework. It incorporates our enhanced CBS algorithm and has been comprehensively optimized for high-precision inputs (Contribution V). The details of each contribution are as follows:

[9] We do not intend to undermine Wang et al.'s work; their circuit bootstrapping algorithm remains the fastest to date and compresses key sizes by 13 times compared to TFHEpp.

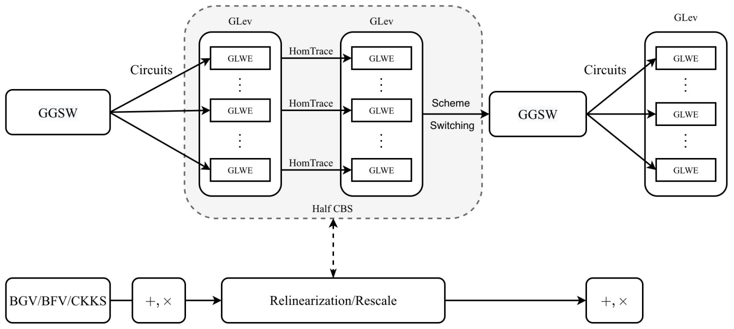


Fig. 2: A Flexible Leveled Homomorphic Evaluation (LHE) Framework.

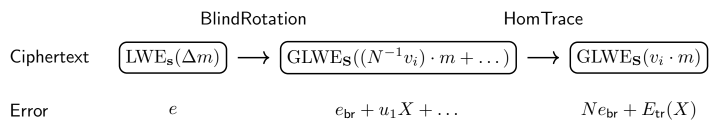
I. Patched NTT-Based Circuit Bootstrapping (CBS): To address the phase amplification issue identified in WWL+ (Challenge A), we propose an NTT-based CBS leveraging a newly designed pre-processing method. It reduces the noise growth from $(N^2 V_{\mathsf{pbs}} + ...)$ to $(V_{\mathsf{pbs}} + ...)$ compared to WWL+, where V_{pbs} denotes the variance of the error generated in the Refr. step. Under the same decryption failure probability, we can then reduce the gadget length in the Refr. step from 2/4/7 to 1/2/2, respectively, decreasing the latency (rsp. compressing the key size) of CBS by factors up to **2.9 (rsp. 3.4)**.

II. FFT-Based CBS: In response to the performance discrepancy between theory and implementation (Challenge B), we introduce an FFT-based circuit bootstrapping algorithm. It is important to note that this is not merely an implementation difference, as we detailed in the technique overview subsection. Our implementation on TFHE-rs achieves a speedup up to 15.4× compared to WWL+ OpenFHE-based implementation, and is **3.1** times faster than their AVX-512 NTT-based implementation[10]. Additionally, our method outperforms the state-of-the-art FFT-based CBS implementation in the TFHEpp library by **3.5**× (rsp. **37.5**×) in terms of the time efficiency (rsp. key size compression).

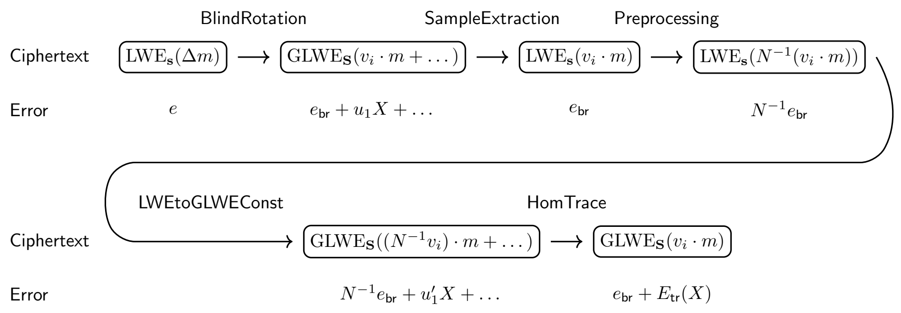
III. A Flexible Leveled Homomorphic Evaluation (LHE) Framework: To tackle the inflexibility and computational intensity of the LHE mode (Challenge C), we propose a flexible LHE framework using a newly designed HalfCBS algorithm to achieve the circuit composability as illustrated in Figure 2, and thereby decouples the Refr. and Conv. steps. This algorithm is particularly suited for small-scale circuit evaluations since the HalfCBS is **2.4** times faster than CBS.

IV. Crypto Tools & Applications: To address the user-unfriendly concerns (Challenge D), we first provide a suite of parameter evaluation tools to lower the barrier to using the LHE mode. We then compare the efficiency of LHE mode and FHE mode in computing general look-up tables under reasonable parameter settings. Our result shows that we can evaluate an 8-8 LUT in 136 ms, which is **111**× faster than the FHE mode using gate bootstrapping. Finally, we highlight

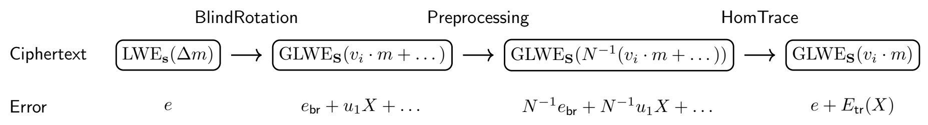
[10] Since WWL+ implementation based on AVX-512 NTT is not open-source, the latter result is derived from their paper.


BlindRotation
HomTrace
Ciphertext
$\mathrm{LWE}_{\mathbf{s}}(\Delta m)$
$\mathrm{GLWE}_{\mathbf{S}}((N^{-1}v_i)\cdot m+\dots)$
$\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m)$
Error
e
$e_{\mathsf{br}}+u_1X+\dots$
$Ne_{\mathsf{br}}+E_{\mathsf{tr}}(X)$

(a) WWL+ method


BlindRotation
SampleExtraction
Preprocessing
Ciphertext
$\mathrm{LWE}_{\mathbf{s}}(\Delta m)$
$\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m+\dots)$
$\mathrm{LWE}_{\mathbf{s}}(v_i\cdot m)$
$\mathrm{LWE}_{\mathbf{s}}(N^{-1}(v_i\cdot m))$
Error
e
$e_{\mathsf{br}}+u_1X+\dots$
e_{br}
$N^{-1}e_{\mathsf{br}}$
LWEtoGLWEConst
HomTrace
Ciphertext
$\mathrm{GLWE}_{\mathbf{S}}((N^{-1}v_i)\cdot m+\dots)$
$\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m)$
Error
$N^{-1}e_{\mathsf{br}}+u_1'X+\dots$
$e_{\mathsf{br}}+E_{\mathsf{tr}}(X)$

(b) A patched method using Chen et al.'s [8] preprocessing


BlindRotation
Preprocessing
HomTrace
Ciphertext
$\mathrm{LWE}_{\mathbf{s}}(\Delta m)$
$\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m+\dots)$
$\mathrm{GLWE}_{\mathbf{S}}(N^{-1}(v_i\cdot m+\dots))$
$\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m)$
Error
e
$e_{\mathsf{br}}+u_1X+\dots$
$N^{-1}e_{\mathsf{br}}+N^{-1}u_1X+\dots$
$e+E_{\mathsf{tr}}(X)$

(c) Our improved method

Fig. 3: The Circuit Bootstrapping Workflow

transciphering as a scenario to demonstrate the power of the LHE mode. For instance, we implement AES transciphering within **15.6** seconds, which is more than 6.6× faster than the FHE mode solutions [26, 4], and outperforms the state of the art [28] by **2.9**×. Additionally, our approach achieves the lowest decryption failure probability among the aforementioned implementations.

V. High Precision LHE (HP-LHE) Framework: To deal with high-precision inputs (Challenge E), we propose an HP-LHE framework optimized from WoP-PBS proposed by Bergerat et al.. Our method can process a message of 4/6/8-bit precision in 64/113/310 milliseconds. Compared to the state-of-the-art method [3] implemented in `tfhe-rs` library, the evaluation time (resp. key size) is improved by factors from **9.7** to **16.2** (resp. **3.4** to **4.4**).

1.2 Technique Overview

Patched NTT-Based Circuit Bootstrapping (CBS): WWL+ sets the scaling factor of the $\mathsf{PBSmanyLUT}$ output to $N^{-1}\Delta \bmod q$ to guarantee the correctness of the Conv. step. However, it causes error amplification from trace evaluation as Figure 3a. A possible solution to address this issue is to execute the efficient conversion algorithm from LWE to GLWE ciphertext proposed by Chen et al. [8] after $\mathsf{PBSmanyLUT}$. The key point is to multiply N^{-1} to both

$$\begin{tikzcd}
\text{Ciphertext} & \boxed{\mathrm{LWE}_{\mathbf{s}}(\Delta m)} \arrow[r, "\mathsf{BlindRotation}"] & \boxed{\mathrm{GLWE}_{\mathbf{S}}((v_i/N)\cdot m+\dots)} \arrow[r, "\mathsf{HomTrace}"] & \boxed{\mathrm{GLWE}_{\mathbf{S}}(v_i\cdot m)} \\
\text{Error} & e & e_{\mathsf{br}}+u_1X+\dots & Ne_{\mathsf{br}}+E_{\mathsf{tr}}(X)
\end{tikzcd}$$

Fig. 4: Adjusting the input scaling factor by v_i/N in FFT domain.

the message and error term, such that only homomorphic trace evaluation error is added after the conversion. We further decompose this workflow as Figure 3b, and then simplify and improve this method as Figure 3c.

FFT-Based CBS: There are two technical subtleties that prevent the above improved WWL+ method from being applied in the FFT domain: (a) there does not exist an inverse of $N \bmod q$ in FFT domains for the aforementioned pre-processing, and (b) the inherent errors in FFT will be amplified by $\mathsf{HomTrace}$ and compromise the correctness of circuit bootstrapping.

For the first issue, one possible solution is to set the scaling factor of the input LWE ciphertext to $\varDelta/N$. However, the initial error is still amplified by N (see Figure 4). Inspired by this limited approach, we present a new pre-processing method that divides both the scaling factor and the initial error by N using modulus switching and modulus raising, see Figure 5 for the pictorial description. Then 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.

To overcome the second barrio, we propose a *split FFT* method to reduce the FFT error by splitting the automorphism key of 64-bit precision into two parts, using them in separate multiplications, and then combining the results. To give a theoretical issuance, we bound the failure probability of this method following the FFT error analysis proposed by Bergerat et al. [3].

A Flexible Leveled Homomorphic Evaluation (LHE) Framework: Our approach is based on two key observations: (a) the cost of the Conv. step is less than the Refr. step; (b) using CBS for small-scale circuits results in wasted depth. Based on these insights, we developed the $\mathsf{HalfCBS}$ algorithm, which only converts the ciphertext form but does not refresh the noise (as illustrated in Figure 2). This algorithm embeds the conversion from $\mathrm{LWE}_{\mathbf{s}}(v_i \cdot m)$ to $\mathrm{GLWE}_{\mathbf{S}}(v_i \cdot m + \dots)$ within the circuit evaluation, rather than using $\mathsf{PBSmanyLUT}$, thereby creating a decoupled LHE computation framework. The Refr. step is only invoked when the noise is near overflow. Intuitively, this approach aligns more closely with the construction of LHE schemes such as BGV/BFV/CKKS.

Crypto Tools & Applications: Our parameter evaluation tool is distinguished by its accuracy and automation. Accuracy comes from precise and compact noise analysis. Automation is achieved through our discovery that the noise in gadget decomposition exhibits a V-shaped relationship with the decomposition length and base. This insight allows for the automated adjustment of gadget parameters. Under reasonable parameter selection, we optimize AES transciphering by

FHE Solution	Mode	Extr.	Refr.	Conv.	Eval.
BGV/BFV/CKKS	LHE	-	Bootstrapping	Relinearization	+, Tensor Product
Bit-Wise FHEW/TFHE	FHE	-	PBS per Gate		
	LHE [11]	-	PBS	Private KS	CMux Gate
	LHE [14]	-	PBSmanyLUT	Private KS	CMux Gate
	LHE [27]	-	PBSmanyLUT	HomTrace, SS	CMux Gate
High-Precision FHEW/TFHE	FHE	-	Tree-PBS		
	LHE [3]	δ PBS	$\delta\ell$ PBS	Private KS	CMux-Tree
	LHE Ours	$\lceil \frac{\delta}{\tau} \rceil$ PBSmanyLUT		GLWE HomTrace, SS	CMux-Tree

Table 2: Dividing FHE solutions in our new perspective.

using our proposed FFT-based CBS algorithm and our flexible LHE framework. Additionally, we employ following techniques to further improve the performance of the AES transciphering: (a) Using dimension-2 GLWE ciphertexts to manage noise growth from CMux operations following Bon et al. [4]. (b) Designing a keyed-Sbox to minimize noise growth during the AddRoundKey. (c) Introducing the 8-24 LUT proposed by Wei et al. [28] to replace 8-8 LUTs to reduce noise growth from the MixColumns.

High Precision LHE (HP-LHE) Framework: We show some details of the WoP-PBS framework proposed by Bergerat et al. [3] in Table 2 and Figure 7a. We further enhance this approach with our HP-LHE framework using the following techniques with a pictorial description in Figure 7b:

(a) Observing that both the extraction and refreshing steps deal with the ciphertext scaling factors, we integrate them together and redesign a PBSmanyLUT to achieve both the Extr. and Refr. operations. This integration reduces the total number of PBS operations from $\delta(\ell + 1)$ to δ, where δ is the message bit-size in a single LWE ciphertext and ℓ is the gadget length for PBS.

(b) We further propose a multi-bit extraction algorithm that can extract τ bits from ciphertext $\text{LWE}(m)$ at the cost of a single PBS, reducing the number of PBS to $\lceil \frac{\delta}{\tau} \rceil$. Specifically, we divide a δ-bit message into some τ-bit chunks of the form $\sum_{j=0}^{\tau-1} m_j 2^j$, each of which can be extracted into $(m_{\tau-1}, m_{\tau-1} \oplus m_{\tau-2}, \ldots, m_{\tau-1} \oplus m_0)$ at the cost of single PBS. We then use a modified LUT accordingly that outputs $\text{L}[m]$ based on the extracted bits.

(c) Managing the HomTrace noise by adjusting the gadget length has a lower bound, which prevents HomTrace from being used in high-precision evaluation. We present a high-precision HomTrace algorithm based on the GLWE dimension switching, which breaks through the noise lower bound, enabling it to be used in our HP-LHE framework. We can then reduce the computational complexity of the conversion step from $O(N^2)$ to $O(N \log^2 N)$.

1.3 Paper Organization

In Section 2, we briefly review the FHEW-like cryptosystems. In Section 3, we propose our patched NTT-based circuit bootstrapping and FFT-based circuit bootstrapping, then show the benchmark from our implementation. Section 4 presents the HalfCBS algorithms and a flexible LHE framework, and shows the LHE application performance in look-up table evaluations and AES transciphering. Section 5 introduces the high-precision LHE framework, and compares it with WoP-PBS and the FHE mode solution. Section 6 concludes this paper.

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 $[a, b[= \{x \in \mathbb{R} : a \le x < b\}$. 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 $\mathbb{Z}_q$. 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.

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

2.2 FHEW-like Cryptosystem

In this section, we briefly review the core concepts of the FHEW-like scheme. 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 $\mathbb{B}_N[X] = \mathbb{B}[X]/(X^N + 1)$.

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 $\Delta \le q/p$ 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 FHEW-like 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 $\varDelta$. The decryption works correctly if the error contained in the ciphertext is small enough to be eliminated during the rounding by $\varDelta$.

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 length 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.

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

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

GGSW Ciphertexts. In the case of nonlinear operations such as multiplication, FHEW-like cyptosystems use another type of ciphertext called generalized GSW (GGSW) [20]. 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 length 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.

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

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 length ℓ decomposes an input $a \in \mathbb{Z}_q$ into a vector $(a_1, \ldots, a_\ell) \in \mathbb{Z}_q^\ell$ such that

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

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

The controlled mux gate, dubbed CMux, is the key operation used in FHEW-like cryptosystem. 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 FHEW/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, FHEW-like schemes require 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.

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 [12]. In this paper, we describe it in two steps: LWE to Lev, 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 $j = 1, \dots, \ell$ using PBS. Since it computes ℓ PBS operations on the same LWE input, PBSmanyLUT proposed by Chillotti et al. [14] 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 $\mathbf{S} = (S_1, \dots, S_k)$, 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 $i = 1, \dots, k+1$ and $j = 1, \dots, \ell$ 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.[11] We refer to Appendix B.3 for a brief overview of the functional keyswitching.

2.4 Automorphism and Trace

The automorphism and trace can be defined on the polynomial ring $\mathcal{R}_N = \mathbb{Z}[X]/(X^N + 1)$ 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

$$\mathrm{Tr}(\mu(X)) := \sum_{d \in \mathbb{Z}_{2N}^{\times}} \tau_d(\mu(X)) = N\mu_0. \tag{1}$$

We then refer Algorithm 1 to describe the algorithm evaluating homomorphic automorphism, and Algorithm 2 to describe the algorithm evaluating homomorphic trace. We present more detailed descriptions in Appendix B.4.

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

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 length $\ell$
   Output: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}(X)}(M(X^d))$
1  $\mathbf{C} = (A_1, \dots, A_k, B)$
2  $\mathbf{C}' \leftarrow (A'_1, \dots, A'_k, B') = (A_1(X^d), \dots, A_k(X^d), B(X^d))$
3  $\mathbf{C}' \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}', \mathrm{AutoKey}_d)$
4  return $\mathbf{C}'$
```

Algorithm 2: Evaluating Trace $\mathsf{HomTrace}(\mathbf{C})$

```
   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: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}(X)}(N \cdot m_0)$
1  $\mathbf{C}' \leftarrow \mathbf{C}$
2  for $d = 1$ to $\log(N)$ do
3      $\mathbf{C}' \leftarrow \mathbf{C}' + \mathsf{EvalAuto}(\mathbf{C}', 2^{\log N - d + 1} + 1)$
4  return $\mathbf{C}'$
```

3 Our Improved Circuit Bootstrapping Algorithms

3.1 Patched NTT-Based CBS

We first briefly introduce the phase amplification issue in WWL+, and propose our patched NTT-based circuit bootstrapping algorithm.

Phase Amplification. WWL+ employed $\mathsf{HomTrace}$ and $\mathsf{SchemeSwitch}$ to construct their faster and smaller circuit bootstrapping [27]. It is crucial to point that the trace evaluation will amplify the phase of the ciphertext containing both message and error, see Equation 1. However, WWL+ just set the scaling factor to $N^{-1}\Delta \bmod q$ to deal with the message amplification, causing an error amplified by a factor of N. Without removing this error amplification, one has to use a larger gadget decomposition length to decrease the PBS error, degrading the overall performance, see Appendix A. We then propose our patched NTT-based CBS algorithm with a novel pre-processing method.

Patched NTT-Based CBS. Our patched NTT-based circuit bootstrapping method can be described as the following two steps.

Step 1 Refr. $\mathrm{LWE}_{\mathbf{s}}(\Delta m)$ to refreshed $\mathrm{GLev}_{\mathbf{S}}(m + \cdots)$ by:
- $\mathsf{PBSmanyLUT}$ [14] without sample extraction, or
- automorphism-based multi-value blind rotation [27].

Step 2 Conv. $\mathrm{GLev}_{\mathbf{S}}(m + \cdots)$ to $\mathrm{GGSW}_{\mathbf{S}}(m)$ conversion by:
- $\mathsf{Preprocess}$ $\times N^{-1}$: $\mathrm{GLev}_{\mathbf{S}}(m + \cdots) \to \mathrm{GLev}_{\mathbf{S}}(N^{-1}m + \cdots)$
- $\mathsf{HomTrace}$: $\mathrm{GLev}_{\mathbf{S}}(N^{-1}m + \cdots) \to \mathrm{GLev}_{\mathbf{S}}(m)$

– SchemeSwitch [16]: $\mathrm{GLev}_{\mathbf{S}}(m) \to \mathrm{GGSW}_{\mathbf{S}}(m)$.

We present Theorem 1 to analyze the noise growth in our patched algorithm. Additionally, we provide a detailed re-analysis of the CBS algorithm proposed by Wang et al. [27] in Appendix A. Our method can significantly reduce the noise growth from $(N^2 V_{\mathsf{pbs}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}})$ to $(V_{\mathsf{pbs}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}})$, removing an N^2 multiplicative factor in the error variance.

Theorem 1. *Let* $\mathbf{c}$ *be an LWE ciphertext of phase* $\mu = \Delta m + e_{\mathsf{in}}$ *under a secret key* $\mathbf{s} = (s_1, \ldots, s_{kN})$ *where the ciphertext modulus* q *is a prime. Then, our patched NTT-based CBS algorithm returns a GGSW ciphertext* $\mathbf{C}$ *of phase* $m + E_{\mathsf{cbs}}(X)$ *under the GLWE secret key* $\mathbf{S} = (S_1, \ldots, S_k)$ *corresponding to* $\mathbf{s}$ *where the variance* V_{cbs} *of* $E_{\mathsf{cbs}}(X)$ *is given as follows.*

$$V_{\mathsf{cbs}} \le V_{\mathsf{pbs}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}.$$

where V_{pbs} *denotes the* PBSmanyLUT[12] *error variance,* V_{tr} *denotes the* HomTrace *error variance, and* V_{ss} *denotes the scheme switching error variance.*

Proof Sketch. After pre-processing with N^{-1}, the phase of the ciphertext is

$$N^{-1} \left\lceil \frac{q}{B^j} \right\rfloor m + N^{-1} y_1 X + \ldots + N^{-1} y_{N-1} X^{N-1} + N^{-1} E_{\mathsf{pbs}}(X).$$

Then the trace evaluation change the phase to

$$\left\lceil \frac{q}{B^j} \right\rfloor m + e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X),$$

where e_{pbs} is the constant term of E_{pbs} without amplify it. We give the full proof in Appendix G for self-completeness. □

3.2 FFT-Based CBS

Under the large modulus used by circuit bootstrapping, FFT outperforms NTT in terms of the concrete cost and implemented performance [30, 1]. We then propose an FFT-based CBS method following our patched method. Compared to the algorithm described in Section 3.1, our FFT-based CBS algorithm uses the following techniques: (a) New pre-processing method designed for the FFT domains, and (b) Split FFT-based evaluation to handle the FFT error.

New Pre-processing. In the patched NTT-based CBS, we use $N^{-1} \bmod q$ to mitigate the phase amplification. However, the existence of N^{-1} is guaranteed when (N, q) are co-prime, while it is not the case in the FFT domain where both N and q are powers-of-two. To overcome this limitation, we propose a new pre-processing method using modulus switching (see Section 2.2 for details) and modulus raising, as shown in Figure 5.

[12] In this paper, we focus on PBSmanyLUT, the conclusions deduced from the automorphism-based multi-value blind rotation are similar.

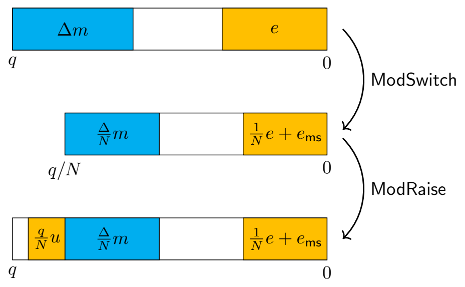


Fig. 5: New pre-processing method on an FFT domain.

Let $\mathbf{C} = \mathrm{GLWE}_{q,\mathbf{S}}(\Delta m)$ be an LWE ciphertext of phase $\mu = \Delta m + e$ modulo q under a GLWE secret key $\mathbf{S} \in \mathbb{B}_N[X]^k$, where q and N are both powers of two. 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 a GLWE ciphertext $\mathbf{C}' = \mathrm{GLWE}_{\frac{q}{N},\mathbf{S}}(\frac{\Delta}{N}m)$ of phase $\mu' = \frac{1}{N}\mu + E_{\mathsf{ms}} = \frac{\Delta}{N}m + \frac{1}{N}e + E_{\mathsf{ms}}$ modulo $\frac{q}{N}$.

As the input message and error are both divided by N, one can cancel out the phase amplification from trace evaluation. Additionally, only the constant e_{ms} of E_{ms} survives after $\mathsf{HomTrace}$. 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 $\mathbf{C}' = (A_1, \ldots, A_k, B) \in \mathcal{R}_{q/N}^{k+1}$. Then,

$$B - \langle (A_1, \ldots, A_n), \mathbf{S} \rangle = \mu' + \frac{q}{N} \cdot U \tag{2}$$

for some $U \in \mathbb{Z}[X]/(X^N + 1)$ since the phase of $\mathbf{C}'$ is μ' modulo $\frac{q}{N}$ under the secret key $\mathbf{S}$. The modulus raising interprets each coefficient of $\mathbf{C}'$ in $\mathbb{Z}_{q/N}$ as an element of $\mathbb{Z}_q$ of the same value, obtaining a GLWE ciphertext $\mathbf{C}'' = (A_1, \ldots, A_k, B) \in \mathcal{R}_q^{k+1}$ of phase $\mu' + \frac{q}{N} \cdot U$ modulo q by (2).

After the above modulus switching and modulus raising, the input GLWE ciphertext of phase μ is changed to the GLWE ciphertext of phase $\mu'' = \frac{1}{N}\mu + e_{\mathsf{ms}} + \frac{q}{N} \cdot U$ under the same GLWE 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. In terms of error, the modulus switching error e_{ms} is amplified by N after $\mathsf{HomTrace}$, while it is still small enough compared to other terms. We refer to Appendix B.1 for the details.

Finally, one can replace the pre-processing step in the patched WWL+ algorithm with this new method. Then, the homomorphic trace evaluation after blind rotation outputs a GLWE ciphertext of a phase

$$N\mu'' + E_{\mathsf{tr}} = \mu + N e_{\mathsf{ms}} + E_{\mathsf{tr}} \tag{3}$$

under the corresponding GLWE secret key $\mathbf{S}$ where E_{tr} is the error of the homomorphic trace evaluation.

Split FFT. Most error analyses in TFHE on FFT domains, especially for PBS, do not deal with the errors generated from the FFT-based polynomial multiplication. However, one cannot ignore the impact of the FFT error in the CBS automorphism-based conversion step, since it will be amplified by N during $\mathsf{HomTrace}$, possibly making it the largest term among output errors.

To reduce the FFT errors, we split a polynomial of 64-bit precision into two parts. For instance, let $F \in \mathcal{R}_{2^{64},N}$ such that $\|F\|_\infty \leq B/2$ 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:

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

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*.

A key point to use split FFT is finding 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 do this, we should first estimate the FFT error. 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. Bergerat et al. [3] proposed an estimation for the variance of FFT error in PBS as

$$2^{-2\chi - 2.6} \cdot n\ell q^2 B^2 N^2 (k+1)^{13},$$

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 the FFT error variance of the external product, we then derive the formula for the FFT error of a single polynomial multiplication $F \cdot G$ as $2^{-2\chi-2.6} B_1^2 B_2^2 N^2$. When it comes to the split FFT-based GLWE keyswitching, the FFT error variance of the lower part (resp. the upper part) is given by

$$2^{2(b-\chi)-2.6} \ell B^2 N^2 k \quad (\text{resp. } 2^{2(64-b-\chi)-2.6} \ell B^2 N^2 k). \tag{4}$$

Using (4), one can find a proper b to guarantee the exact computation on the upper part with a negligible failure probability. For instance, 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 PBS. On the other hand, there might be an FFT error for the lower part, which should be added to the final error. The split FFT reduces the FFT error significantly, making it much smaller

[13] There is also an FFT error estimate proposed by Klemsa et al., we analysis the difference in Appendix C.

than the additive error of the GLWE keyswitching for the parameters used in this paper. We refer to Appendix C for the detailed analysis.

We then present Theorem 2 to analyze the noise growth in our FFT-based circuit bootstrapping algorithm.

Theorem 2. *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, our FFT-based CBS algorithm returns a GLWE ciphertext* $\mathbf{C}$ *of phase* $\mu + E_{\mathsf{cbs}}(X)$ *under the GLWE secret key* $\mathbf{S} = (S_1, \ldots, S_k)$ *corresponding to* $\mathbf{s}$ *where the variance* V_{cbs} *of* $E_{\mathsf{cbs}}(X)$ *is given as follows.*

$$V_{\mathsf{cbs}} \leq V_{\mathsf{pbs}} + N^2 V_{\mathsf{ms}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}$$

where V_{pbs} *denotes the* PBSmanyLUT *output error variance,* V_{ms} *denotes the modulus switching error,* V_{tr} *denotes the* HomTrace *output error variance, and* V_{ss} *denotes the scheme switching output error variance.* V_{pbs}*,* V_{tr} *and* V_{ss} *should contain their FFT error variance.*

Proof Sketch. The proof is analogous to that of Theorem 1 except that there are additional pre-processing error and FFT error. We give the full proof in Appendix G for self-completeness. □

3.3 Performance

3.3.1 Patched NTT-Based CBS

Parameters. We employ the same security parameters as outlined in Table 3 of [27]. By using our novel pre-processing method to mitigate phase amplification, we enhance the effectiveness of noise management. This allows us to recalibrate the noise control parameters while maintaining the similar max depth (our parameter-gen tool is provided[14] to lower the barrier to using CBS and the LHE mode). The newly recommended parameters are presented in Table 3.

The parameters n, N and k represent the dimension of the LWE, the dimension of the ring polynomial, and the dimension of the GLWE, respectively. $\ell, \ell_{\mathsf{ep}}, \ell_{\mathsf{tr}}, \ell_{\mathsf{ss}}$ and $B, B_{\mathsf{ep}}, B_{\mathsf{tr}}, B_{\mathsf{ss}}$ denote the gadget decomposition length and base in the circuit evaluation, external product, HomTrace and scheme switching, respectively.

Performance. We implement our patched NTT-based circuit bootstrapping algorithm as shown in Table 4 in OpenFHE library[1], which supports the the FHEW scheme in the NTT settings. The implementation of patched CBS are provided[15]. The evaluation environment is a PC with 11th Gen Intel (R) Core(TM) i5-11500 @ 2.70GHz and 32GB of RAM, running Ubuntu 22.04.2 LTS. The execution time is the average on 1000 trails of CBS.

[14] https://github.com/LightFHE/CircuitBootstrap/blob/main/parameters_gen.py

[15] https://github.com/LightFHE/CircuitBootstrap/tree/main

Sets	n	N	k	ℓ_{ep}	B_{ep}	ℓ_{tr}	B_{tr}	ℓ_{ss}	B_{ss}	ℓ	B	Max Depth
CMux_1	571	2048	1	1	2^{26}	3	2^{13}	1	2^{28}	4	2^3	38
CMux_2	571	2048	1	2	2^{17}	3	2^{13}	2	2^{19}	4	2^4	533
CMux_3	571	2048	1	2	2^{17}	6	2^8	2	2^{19}	4	2^5	73,951

Table 3: The recommended parameter sets for patched NTT-based CBS noise management. For each parameter set, we have listed the max supported depth.

Implementations	Sets	Key Size (MB)	Run Time (ms)
TFHE-NTT	TFHE_{pp}	202.04	877
WWL+	CMux_O_1	30.6	102.5
Our-NTT	CMux_1	**15.5**	**77.5**
TFHE_{pre}-NTT	TFHE_{pp}	1147.50[16]	484.08
WWL+	CMux_O_2	45.9	128.2
Our-NTT	CMux_2	**30.57**	**102.8**
WWL+	CMux_O_3	107	298.3
Our-NTT	CMux_3	**31.01(3.5×)**	**103.9(2.9×)**

Table 4: Our patched circuit bootstrapping performance.

Comparison. We compare our patched circuit bootstrapping with the state-of-the-art NTT-based CBS implementations, as shown in Table 4. According to the analysis in Table 7 of [27], the original TFHE method supports a maximum circuit depth of 9, while the pre-computed version, TFHE_{pre}, supports 294. To ensure fairness, our comparison uses parameter sets supporting similar depths.

Compared to the TFHE method, our result achieves a 11.3× performance improvement and 13.0× key size compression. Compared to the faster TFHE_{pre} method, the key size is 37.5× smaller, while it remaining 4.7× faster. Finally, compared to WWL+ under the similar decryption failure prob., our NTT-based CBS decreases the latency (rsp. compresses the key size) by factors up to 2.9 (rsp. 3.5). The key size used in this paper is the compressed size (see Appendix B.7).

3.3.2 FFT-Based CBS

Parameters. For a fair comparison with our patched NTT-based CBS, we employ the same level of security parameters by migrating it into the FFT setting in this section. We have set the LWE standard deviation σ_{LWE} (resp. GLWE standard deviation σ_{GLWE}) by σ/Q_{ks} (resp. σ/Q) where σ is the standard deviation for the error, Q_{ks} (resp. Q) is the maximum modulus size under the LWE dimension of n (resp. kN) in the NTT setting. For the LWE keyswitching, we use a gadget length of $\ell_{\mathsf{ks}} = 5$ with a base $B_{\mathsf{ks}} = 2^2$. The ciphertext modulus of $q = 2^{64}$ is used in our FFT-based CBS.

The recommended parameter sets for our FFT-based circuit bootstrapping are summarized in Table 5. The additional column of b_{tr} denotes the split FFT base to reduce the FFT error of $\mathsf{HomTrace}$. One can find that our FFT-based

[16] We update the key size by revising a parameter flaw in WWL+.

CBS uses a smaller gadget base compared to the NTT-based CBS. The reason is that we have to consider the FFT error, making the optimal parameters slightly different from the NTT-based CBS and reducing the max supported depth.

Sets	n	N	k	ℓ_{ep}	B_{ep}	ℓ_{tr}	B_{tr}	b_{tr}	ℓ_{ss}	B_{ss}	ℓ	B	Max Depth
CMux_1	571	2048	1	1	2^{23}	3	2^{13}	2^{42}	1	2^{26}	4	2^3	8
CMux_2	571	2048	1	2	2^{15}	3	2^{13}	2^{42}	2	2^{17}	4	2^4	1194
CMux_3	571	2048	1	2	2^{15}	6	2^{8}	2^{37}	2	2^{17}	4	2^5	13410

Table 5: The recommended parameter sets for our FFT-based CBS noise management. For each parameter set, we have listed the max supported depth.

We also provide a parameter analysis tool for the FFT-based LHE mode that considers the FFT error.[17]

Performance. We implement our FFT-based circuit bootstrapping algorithm using the TFHE-rs library [30] of version 0.5.3, which supports the TFHE scheme in the FFT settings. The implementation of our FFT-based CBS are provided[18]. For the benchmark in the FFT setting, we used Intel i5-13600K @ 5.30 GHz with 128 GB RAM. The time is measured by `criterion` benchmarking module of `Rust` with 1,000 samples.

Comparison. We compare our FFT-based circuit bootstrapping with the other FFT-based libraries: TFHEpp and MOSFHET. The detailed results are shown in Table 6. Compared to the faster $\mathrm{TFHE}_{\mathrm{pre}}$ method, our FFT-based CBS enjoys $3.45\times$ faster running time and $37.5\times$ smaller key size.

4 Improved Leveled Homomorphic Evaluation

4.1 A Flexible LHE Framework

The LHE framework is inflexible since the Conv. step is always bonded executed with the heavy Refr. step, denoted as circuit bootstrapping (CBS). However, given that the supported circuit evaluation depth increases exponentially with the circuit bootstrapping parameters, using CBS for small-scale circuits to convert ciphertext types results in wasted depth.

We then propose a $\mathsf{HalfCBS}$ algorithm inputs $\mathbf{C} = \mathrm{GGSW}_{\mathbf{S}}(m)$ and outputs $\mathbf{C} = \mathrm{GGSW}_{\mathbf{S}}(L[m])$ to achieve the circuit composability without refreshing noise. Compared to our proposed CBS algorithms described in Section 3.1 and Section 3.2, the $\mathsf{HalfCBS}$ algorithm (Algorithm 3) does not have a Refr. step. Specifically, it use ℓ look up table circuits $\mathsf{Circuit}_{m\to v_i\cdot L[m]}$ instead of $\mathsf{PBSmanyLUT}$ to generate $\mathrm{GLev}_{\mathbf{S}}(m+\cdots)$, where ℓ denotes the GGSW gadget length. Then its computational complexity can be reduced from $O(nN\log N)$ to $O(N\log^2 N)$ compared to the whole CBS algorithm, significantly improving the efficiency. We then propose Theorem 3 to analyze the noise growth in $\mathsf{HalfCBS}$.

[17] https://github.com/KAIST-CryptLab/FFT-based-CircuitBootstrap/tree/main/fft_error_analysis

[18] https://github.com/KAIST-CryptLab/FFT-based-CircuitBootstrap/tree/main

Methods	Sets	Key Size (MB)	Run Time (ms)
TFHE$_{\text{pre}}$-FFT	TFHEpp	1360	63.20
TFHE$_{\text{pre}}$-FFT	MOSFHET	5201.88	152.83
Our-FFT	CMux_2	36.31	**18.34**
Our-FFT	CMux_1	18.44	**13.28**
Our-FFT	CMux_3	36.83	**19.38**

Table 6: Our proposed FFT-based circuit bootstrapping performance compared to TFHEpp and MOSFHET.

Algorithm 3: HalfCBS

```
   Input: $\mathbf{C} = \mathrm{GGSW}_{\mathbf{S}}(m)$
   Input: $\ell$ look up table circuits $\mathsf{Circuit}_{m \to v_i \cdot L[m]}$, $i \in \{1, \ldots, \ell\}$
   Input: Automorphism keys under $\mathbf{S}$
   Input: Scheme switching key under $\mathbf{S}$
   Output: $\mathbf{C} = \mathrm{GGSW}_{\mathbf{S}}(L[m])$
1  for $i = 1$ to $\ell$ do
2      $\mathbf{C}'_i \leftarrow \mathsf{Circuit}_{m \to v_i \cdot L[m]}(\mathbf{C})$
3  $\mathbf{C}' \leftarrow \{\mathbf{C}'_i\}_{i \in \{1,\ldots,\ell\}}$
4  $\mathbf{C}' \leftarrow \mathsf{Preprocess}(\mathbf{C}')$
5  $\mathbf{C}' \leftarrow \mathsf{HomTrace}(\mathbf{C}')$
6  $\mathbf{C} \leftarrow \mathsf{SchemeSwitch}(\mathbf{C}')$
7  return $\mathbf{C}$
```

Theorem 3. *Let $\mathbf{C}$ be a GGSW ciphertext of phase $\mu = m + e_{\mathsf{in}}$ under a secret key $\mathbf{S} = (S_1, \ldots, S_k)$. Then our HalfCBS algorithm returns a GGSW ciphertext $\mathbf{C}$ of phase $m + E_{\mathsf{half\text{-}cbs}}(X)$ where the variance $V_{\mathsf{half\text{-}cbs}}$ of $E_{\mathsf{half\text{-}cbs}}(X)$ is given as follows.*

$$V_{\mathsf{half\text{-}cbs}}(X) \leq V_{\mathsf{circuit}} + V_{\mathsf{pre}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}.$$

where $V_{\mathsf{circuit}} \leq (1 + kN) \left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12} \right) + (k+1)\ell N \left(\frac{B^2+2}{12} \right) \sigma_{\mathsf{in}}^2$ denotes the circuit evaluation error variance, where d is the circuit depth. V_{pre} denotes the pre-processing error variance, V_{tr} denotes the trace evaluation error variance, and V_{ss} denotes the scheme switching error variance, and $\sigma_{\mathsf{in}}^2 = \mathrm{Var}(e_{\mathsf{in}})$. V_{circuit}, V_{tr} and V_{ss} should contain their FFT error variance in the FFT setting.

Proof. One can directly derive the equation $V_{\mathsf{circuit}} \leq (1 + kN) \left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12} \right) + (k+1)\ell N \left(\frac{B^2+2}{12} \right) \sigma_{\mathsf{in}}^2$ from the CMux construction and Lemma 4. $V_{\mathsf{pre}} = N^2 V_{\mathsf{ms}}$ comes from Theorem 2 in the FFT setting, otherwise $V_{\mathsf{pre}} = 0$ in the NTT setting following our new proposed pre-processing method. □

Theorem 3 demonstrates that the HalfCBS algorithm amplifies the input noise in each round rather than refreshing it. Therefore, it is more suitable for tasks with a small number of circuits and low circuit depth. However, our flexible LHE framework can incorporate the following optimization methods to handle complex tasks. For simplicity, these methods are not included in Algorithm 3:

Hybrid use of HalfCBS and CBS algorithms: Our LHE framework can make a flexible and mixed use of both HalfCBS and CBS algorithms. Specifically, based on a compact noise assessment, the HalfCBS algorithm can be used when the noise level is low, and the CBS algorithm can be employed when the noise is about to overflow. This technique is conducted in the AES transciphering implementation in Section 4.3, where a HalfCBS round is shown to be 2.4 times faster than a CBS round.

4.2 General Look-Up Table Evaluation

To demonstrate the power of LHE mode, we introduce the general Look-Up Table(LUT) evaluation. Specifically, LUT is commonly used in homomorphic applications since it can represent general functions [13]. In the FHE mode, LUT is evaluated using PBS-tree [6, 21]. On the other hand, in the LHE mode, LUT is evaluated via circuit bootstrapping and CMux-tree [11, 12, 3].

For an LUT $L : \mathbb{B}^d \to \mathcal{R}^s$, it can be expressed with s sub-LUT $L_i : \mathbb{B}^d \to \mathcal{R}$, each of which contains a list of 2^d inputs and 1 output. Then in order to evaluate L_i in the LHE mode, we need to convert the d inputs and build a binary decision tree composed of $2^d - 1$ CMux gates. Therefore the total complexity of evaluation L requires d times circuit bootstrappings and $s \cdot (2^d - 1)$ CMux gates. We denote the latency using $d \cdot T_{\mathsf{cbs}} + (s(2^d - 1)) \cdot T_{\mathsf{cmux}}$, where T_{cbs} and T_{cbs} denote the execution time of circuit bootstrapping and CMux gates, respectively. On the other hand, in FHE mode, the LUT evaluation time is $(d + s(2^d - 1)) \cdot T_{\mathsf{pbs}}$ accordingly [12], where T_{pbs} denotes the PBS time.

We then implement the general LUT in both the NTT domain with OpenFHE and FFT domain with TFHE-rs. Furthermore, we adopted the packing strategy proposed in [11] to further optimize performance as detailed in Appendix D.

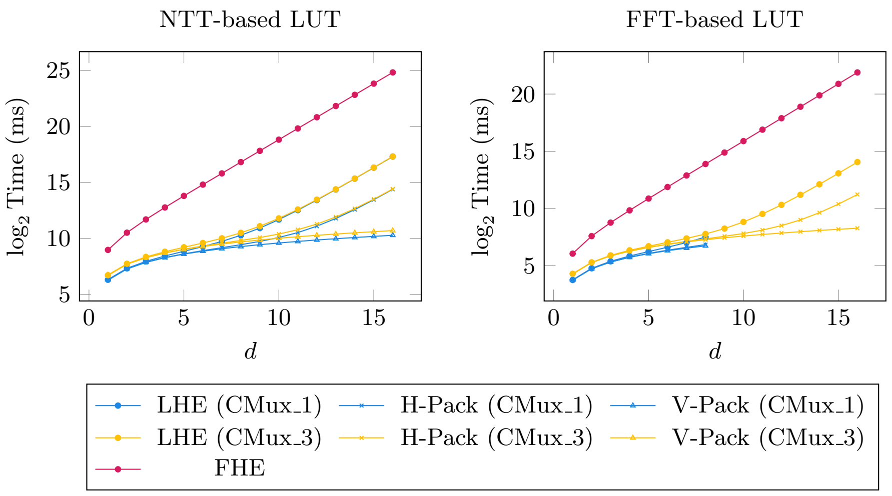


Fig. 6: Comparison of LUT evaluation using GBS and CBS.

Our experimental results demonstrates that, based on our FFT-based circuit bootstrapping algorithm, we can evaluate 8/16-bit (to 8-bit) look-up tables in 136.2/310.8 milliseconds in the LHE mode, respectively. Compared to the FHE mode, the performance improvement factors are $2^{6.8}/2^{13.61}$, respectively. This factor tends to 2^{22} for lager input size ($2^{21.84}$ for 32-bit input), see Figure 8 in Appendix D for pictures until $d = 32$ to check the details.

4.3 AES Transciphering

Homomorphic AES evaluation is one of the relevant applications of the transciphering framework, which combines a symmetric cipher with a homomorphic encryption scheme, see details in Appendix E. This hybrid approach aims to reduce computation and communication costs of the client-side at the cost of homomorphic decryption of the symmetric cipher on the server-side [24]. Several works evaluating AES have been proposed [19, 9, 15, 17, 4, 26, 29, 27, 28]. Among these, the fastest method to date (in a single thread) is based on the LHE mode in the FHEW-like cryptosystem [27, 28].

In the LHE mode, single-bit message encoding makes bit shifting operations nearly cost-free. Additionally, homomorphic XOR can be efficiently implemented using homomorphic addition, which incurs minimal computational overhead. As a result, the LHE mode allows for almost free evaluation of AddRoundKey, ShiftRows, and MixColumns in AES transciphering in terms of computation time. Consequently, the primary cost arises from computing the 8-bit AES S-box (SubBytes) using 8-8 lookup tables and performing circuit bootstrapping. For a detailed step-by-step evaluation, please refer to Appendix E.1.

n	N	k	ℓ_{ks}	B_{ks}	HalfCBS						CBS ($\vartheta = 3$)								Failure Prob.
					ℓ_{tr}	B_{tr}	ℓ_{ss}	B_{ss}	ℓ	B	ℓ_{pbs}	B_{pbs}	ℓ_{tr}	B_{tr}	ℓ_{ss}	B_{ss}	ℓ	B	
768	1024	2	3	2^4	8	2^6	2	2^{19}	6	2^4	1	2^{23}	3	2^{13}	2	2^{19}	7	2^2	$2^{-36.8}$

Table 7: Parameters for the AES evaluation.

We then implement the AES transciphering based on our FFT-based CBS. For our implementation, we use the parameter set as listed in Table 7. We used the LWE standard deviation of $2^{-17.12}$ that achieves 128-bit security according to the lattice-estimator [2]. For the GLWE standard deviation, we use the same value as in the recommended parameters of TFHE-rs. Additionally, we propose the following techniques to further improve the performance:

1. **Flexible LHE:** We use our proposed flexible LHE framework. For instance, the latency of the HalfCBS round is 808.78 ms, which is **2.4** times faster than the CBS round (1920.1 ms). However, the whole AES transciphering is too large to use HalfCBS for all of the procedures. We currently use it 1 time without affecting the decryption failure prob., and one can adjust it flexibly.
2. **Modified AES evaluation:** We embed the AES round key into the Sbox to generate a keyed Sbox encrypted by FHE. Then we can eliminate the

Scheme	Evaluation Mode	Hardware	F.P.	Performance
BGV	LHE [19]	i5-3320M	2^{-40}	1,080s[19]
FHEW-like	FHE [26]	i7-12700H	2^{-23}	270s
	FHE [4]	/	2^{-23}	103s
	LHE [29]	i5-12500	/	86s
	LHE [27]	i5-11500	2^{-32}	73.8s[20]
	LHE [28]	i5-11500	/	46s (769 MB)
	LHE (Our)	i5-13600K	2^{-37}	16.5s (19.10 MB)
	Flex. LHE (Our)	i5-13600K	2^{-37}	15.6s (24.32 MB)

Table 8: Our AES evaluation performance compared with the state of the art.

homomorphic addition (AddRoundKey) before look-up tables in AES transciphering. Furthermore, we mix SubBytes, MixColumns, and ShiftRows together into four 8-24 LUTs following Wei et al.'s method [28], reducing the number of homomorphic additions and the noise growth.

Table 8 shows the benchmark result and comparison with previous works. Although the benchmark environments are varying, one can find that our results outperform all the previous results.

5 High-Precision Input LHE

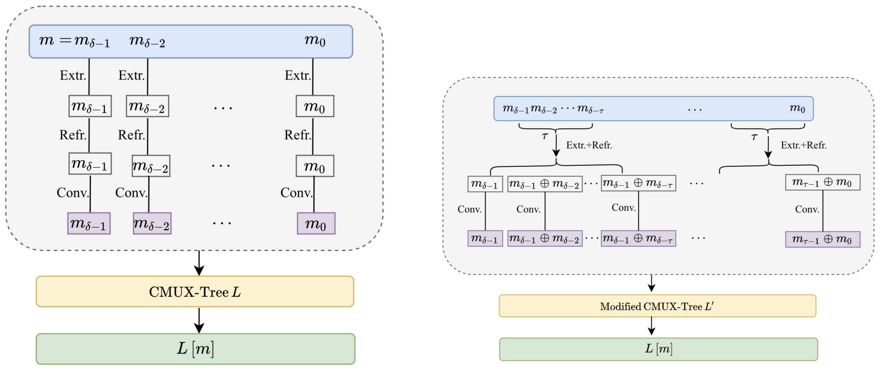
In Section 3 and Section 4, we show the advantage of LHE mode dealing with bit-wise input. However, there is also a common approach for FHEW-like cryptosystems to handle large-size messages, which involves decomposing the input into multi-bit chunks[21], and thereby supporting some linear operations without modulus and tree-based LUT [6, 13, 14, 21].

These evaluation methods lead to a challenge for the LHE mode to deal with high-precision inputs instead of bits. To respond to this issue, Bergerat et al.[3] proposed the WoP-PBS algorithm, as shown in Figure 7a. This algorithm extracts each bit from the ciphertext chunks, and converts them into GGSWs to perform CMux evaluation. In this section, we propose a new HP-LHE framework by improving the WoP-PBS algorithm, as shown in Figure 7b.

[19] Gentry et al. also proposed an AES evaluation without bootstrapping, with a latency of 240s. However, the output from this method does not support further evaluation, so that it is not suitable for transciphering.

[20] Wang et al. evaluate AES transciphering with their WWL+ circuit bootstrapping method [27], achieved a latency of 26 seconds. However, this method is affected by phase amplification, resulting in a higher decryption failure probability. Therefore, we reanalyzed their method in Appendix A, provided new parameter sets, and used the performance under these parameters as a benchmark.

[21] There is a general parameter setting method in `tfhe-rs` library for `shortint` type, called `PARAM_MESSAGE_a_CARRY_b`. It encodes a message bits and b carry bits, where the carry bits are reserved for linear operations.



$m = m_{\delta-1}$ $m_{\delta-2}$ m_0
Extr.
Refr.
Conv.
CMUX-Tree L
$L[m]$
$m_{\delta-1}m_{\delta-2}\cdots m_{\delta-\tau}$
τ
Extr.+Refr.
$m_{\delta-1} \oplus m_{\delta-2}$
$m_{\delta-1} \oplus m_{\delta-\tau}$
$m_{\tau-1} \oplus m_0$
Modified CMUX-Tree L'

(a) WoP-PBS Method [3]   (b) Our HP-LHE Method

Fig. 7: The High-Precision Solutions in the LHE Mode.

5.1 New HP-LHE Framework

We present our insightful observation and idea for the construction in technique overview, and further detailed full algorithm in Appendix F. For short, we improve WoP-PBS and present our HP-LHE by using the following techniques:

– We integrate the extraction and refreshing step by modified $\mathsf{PBSmanyLUT}$.
– We propose a high-precision $\mathsf{HomTrace}$ to improve the conversion step.
– We present a multi-bit extraction method to enhance the extraction step.

Integrate Extr. with Refr. Suppose that an LWE ciphertext of a δ-bit message m scaled by $\varDelta = \lceil q/2^{\delta} \rceil$ is given where $m = \sum_{j=0}^{\delta-1} m_j 2^j$ and $m_0, \ldots, m_{\delta-1} \in \{0,1\}$. In the original WoP-PBS, each bit of the message requires $(\ell+1)$ PBS operations where ℓ is the gadget length of the CBS output.[22]

The bit extraction step and the CBS step are separated in the algorithm of the original WoP-PBS, while we can describe it together as follows. For each iteration, the LWE ciphertext $\mathrm{LWE}(\lceil q/2 \rceil \cdot m_0)$ of the LSB m_0 is obtained by multiplying $\mathrm{LWE}(\varDelta m)$ by $2^{\delta-1}$. Given $\mathrm{LWE}(\lceil q/2 \rceil \cdot m_0)$ as an input,

– the bit extraction step outputs $\mathrm{LWE}(\varDelta m)$ by a single PBS operation, and
– the CBS step outputs $\mathrm{Lev}(m)$ by ℓ PBS operations and converts it into $\mathrm{GGSW}(m)$ by the followed Conv. step.

The output $\mathrm{LWE}(\varDelta m)$ of the bit extraction step is subtracted from the input ciphertext $\mathrm{LWE}(\varDelta m)$, obtaining an input ciphertext $\mathrm{LWE}(\varDelta' m')$ to the next iteration where $\varDelta' = \lceil q/2^{\delta-1} \rceil$ and $m' = \sum_{j=1}^{\delta-1} m_j 2^j$.

From the above description, one can find that the PBS operations take the same input ciphertext $\mathrm{LWE}(\lceil q/2 \rceil \cdot m_0)$. Furthermore, the output $\mathrm{LWE}(\varDelta m_0)$ of the bit extraction step can be obtained from the output $\mathrm{GGSW}(m)$ of the CBS step. So that we can generate $\mathrm{GGSW}(m)$ by a single $\mathsf{PBSmanyLUT}$, reducing the number of blind rotations from $\delta(\ell+1)$ to δ.

[22] To be precise, the MSB does not require a PBS operation for the bit extraction.

Algorithm 4: High Precision Conversion

```
Input: $\mathbf{C}_i = \mathrm{GLWE}_{\mathbf{S}}(v_i \cdot m + u_{1,i}X + \cdots + u_{N-1,i}X^{N-1})$ for $i = 1, \ldots, \ell$
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 $\mathbf{S}'$
Input: Scheme switching key under $\mathbf{S}$
Output: $\overline{\overline{\mathbf{C}}} = \mathrm{GGSW}_{\mathbf{S}}(m)$
1 for $i = 1$ to $\ell$ do
2     $\mathbf{C}'_i \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}_i, \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'})$
3     $\mathbf{C}'_i \leftarrow \mathsf{Preprocess}(\mathbf{C}'_i)$
4     $\mathbf{C}'_i \leftarrow \mathsf{HomTrace}(\mathbf{C}')$
5     $\mathbf{C}'_i \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}', \mathrm{KS}_{\mathbf{S}'\to\mathbf{S}})$
6 $\overline{\mathbf{C}'} \leftarrow \{\mathbf{C}'_i\}_{i=1}^{\ell}$
7 $\overline{\overline{\mathbf{C}}} \leftarrow \mathsf{SchemeSwitch}(\overline{\mathbf{C}}')$
8 return $\overline{\overline{\mathbf{C}}}$
```

Improved Conv. with HP-HomTrace The Conv. step takes the blind rotation output and converts it into a GGSW ciphertext. The WWL+ method replaces the heavy private functional key switching with HomTrace and scheme switching, improving both computation cost and key size. However, in terms of error growth, this method cannot support higher precision because the error variance of the evaluation key is amplified by $O(N^3)$ during the conversion step. Although the error induced by HomTrace was small enough to be used in the bit-input LHE mode, we need high-precision HomTrace evaluation method to support the multi-bit input LHE mode.

We propose a high-precision HomTrace approach by combining GLWE dimension switching as follows. Let $\mathbf{S}$ be a GLWE secret key of a dimension k, we first 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 key switching (see Algorithm 5 in Appendix B.2). Then, after pre-processing on the switched GLWE ciphertext to handle phase amplification, we evaluate the trace on the large dimension. Finally, we switch it back into the origin.

We propose Algorithm 4 for high precision conversion step, and present Theorem 4 to bound the conversion noise. We note that HomTrace is evaluated in the larger GLWE dimension k' in Algorithm 4, so one can make V_{tr} in Theorem 4 much smaller than that in Theorem 1 and 2 for the same N.

Theorem 4. *Let $\mathbf{c}$ be an LWE ciphertext of a message m under a secret key $\mathbf{s} = (s_1, \ldots, s_{kN})$ and $\mathbf{S} = (S_1, \ldots, S_k)$ be the GLWE ciphertext corresponding to $\mathbf{s}$. Let $\mathbf{S}' = (S'_1, \ldots, S'_k)$ be a GLWE secret key of a dimension k' such that $k' > k$. Provided that the FFT error is negligible when the FFT domain is used, our CBS algorithms along with high-precision conversion (Algorithm 4) returns a GGSW ciphertext $\overline{\overline{\mathbf{C}}}$ of m under $\mathbf{S}$ whose error variance $V_{\mathsf{hp\text{-}cbs}}$ is given as* *follows.*

$$V_{\mathsf{hp\text{-}cbs}} = V_{\mathsf{pbs}} + V_{\mathsf{pre}} + V_{\mathbf{S}\to\mathbf{S}'} + \frac{N}{2}(V_{\mathsf{tr}} + V_{\mathbf{S}'\to\mathbf{S}}) + V_{\mathsf{ss}}$$

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_{pre} is the pre-processing noise variance, and V_{tr} is the trace evaluation variance under $\mathbf{S}'$.

Proof. Compared to our previous CBS algorithms, additional GLWE keyswitching errors are induced during Conv. step. For the first keyswitching error (from $\mathbf{S}$ to $\mathbf{S}'$), the subsequent pre-processing prevents the error amplification by $\mathsf{HomTrace}$ and only its constant term is remained after $\mathsf{HomTrace}$, so the multiplication factor $N/2$ of the scheme switching is not multiplied. For the second keyswitching error (from $\mathbf{S}'$ to $\mathbf{S}$), the multiplication factor of the scheme switching is multiplied together with the $\mathsf{HomTrace}$ error. $\square$

Multi-Bit Extraction Algorithm. By the above two techniques, one can extract and convert each bit of the message using a single blind rotation. To reduce it further, we try to process several message bits (τ bits) in a single blind rotation. We then propose multi-bit extraction algorithm as follows.

Suppose that a scaled input ciphertext $\mathrm{LWE}(\lceil q/2^\tau \rfloor) \cdot (m)$ of a τ-bit chunk m is given where $m = \sum_{j=0}^{\tau-1} m_j \cdot 2^j$. After obtaining $\mathrm{GLev}(\lceil q/2 \rfloor \cdot m_{\tau-1} + \dots)$ by $\mathsf{PBSmanyLUT}$, one can also compute $\mathrm{GLev}(\lceil q/2 \rfloor \cdot (m_{\tau-1} \oplus m_j) + \dots)$ by MV-PBS [21]. We refer to Appendix F for the details of the MV-PBS operation for the multi-bit extraction. Building on this, the conversion step outputs $\left(\mathrm{GGSW}(m'_j)\right)_{j=0}^{\tau-1}$, where $m'_{\tau-1} = m_{\tau-1}$ and $m'_j = m_{\tau-1} \oplus m_j$ for $j = 0, \dots, \tau-2$. Then, by easily building a circuit $\mathsf{Circuit}_{(m'_{\tau-1},\dots,m'_0)\mapsto \Delta\cdot L[m]}$, one can compute $\mathrm{LWE}(\Delta L[m])$ by evaluating the circuit using $\left(\mathrm{GGSW}(m'_j)\right)_{j=0}^{\tau-1}$.

5.2 Performance

Environments and Parameters. We implement our new HP-LHE framework using the TFHE-rs library [30] of version 0.5.3. The overall benchmark environment is the same as that of Section 3.3.2.

We used the same security parameters for our new HP-LHE framework with that of the recommended parameter sets for WoP-PBS in the TFHE-rs library, named `WOPBS_PARAM_MESSAGE_a_CARRY_a_KS_PBS` for $a \in \{2, 3, 4\}$, all of which uses $N = 2048$ and $k = 1$. The name of these parameter sets indicates that its plaintext encoding has no padding bit and a-bit carry and a-bit message. We denote such plaintext encoding as `WOPBS_a_a` in short. The recommended parameter sets for each of the encodings are summarized in Table 9. We note that our parameter set for `WOPBS_3_3` uses a smaller PBS gadget length than that of the TFHE-rs library.

Comparison with WoP-PBS. We compare our new HP-LHE framework with the WoP-PBS implemented in the TFHE-rs library. The detailed results are

Encoding	n	ℓ_{ks}	B_{ks}	ℓ_{pbs}	B_{pbs}	ℓ_{tr}	B_{tr}	b_{tr}	ℓ_{ss}	B_{ss}	$\ell_{k\to k'}$	$B_{k\to k'}$	$b_{k\to k'}$	$\ell_{k'\to k}$	$B_{k'\to k}$	$b_{k'\to k}$	ℓ	B	ϑ
`WOPBS_2_2`	769	3	2^4	2	2^{15}	7	2^7	2^{36}	2	2^{16}	-	-	-	-	-	-	4	2^4	2
`WOPBS_3_3`	873	2	2^7	3	2^{11}	4	2^{12}	2^{41}	4	2^{10}	3	2^{15}	2^{44}	3	2^{13}	2^{42}	4	2^5	2
`WOPBS_4_4`	953	2	2^7	4	2^9	6	2^9	2^{39}	4	2^{10}	3	2^{15}	2^{44}	4	2^{10}	2^{40}	8	2^3	3

Table 9: Recommended parameter sets for our new HP-LHE framework.

Encoding	Method	δ	τ	Time (ms)				Max Depth			Key Size (MB)
				Extr.	Refr.	Conv.	Total	F.P.-32	F.P.-80	F.P.-128	
`WOPBS_2_2`	TFHE-rs	4	1	69.73	253.40	231.91	555.04	171	62	37	170.01
	Ours	4	1	82.19		18.12	100.31	558	203	119	49.33
		4	2	39.55		17.93	57.48	516	161	77	49.33
`WOPBS_3_3`	TFHE-rs	6	1	217.79	761.12	696.09	1675.00	1017	315	150	365.13
	Ours	6	1	196.81		40.99	237.80	1393	507	299	82.80
		6	2	91.46		41.09	132.55	1289	404	196	82.80
		6	3	62.15		41.32	103.47	292	-	-	82.80
`WOPBS_4_4`	TFHE-rs	8	1	337.86	2237.5	1897.5	4472.86	6669	2428	1429	375.13
	Ours	8	1	365.54		150.23	515.77	17073	5335	2572	120.45
		8	2	178.64		149.75	328.39	11523	-	-	120.45

Table 10: Our new HP-LHE framework performance.

shown in Table 10. The maximum CMux depth after circuit bootstrapping is computed according to various failure probabilities. Our method has a larger max-depth (only except for the case of $(\delta, \tau) = (6, 3)$), so our performance improvement does not come from degrading success probability.

According to the plaintext encoding, our HP-LHE framework improves the running time of the WoP-PBS method by factors from 9.7 to 16.2 for the parameters supporting failure probability of 2^{-128}. For the relaxed parameters that only support failure probability of 2^{-32}, our method is even more than 16.19× faster. In terms of the key size, our method reduces it by factors from 3.4 to 4.4.

6 Conclusion

The current design and application of FHEW-like schemes are mainly focused on gate bootstrapping, which hides the sophisticated parameter configuration and provides a user-friendly interface to the application developers. However, for most of the applications, leveled homomorphic evaluation presents more competitive solutions. In this paper, we refined the workflow of leveled homomorphic evaluation based on FHEW-like schemes, making it clearer, more flexible and easy to use. By decoupling the most expensive circuit bootstrapping into three fine-grained operations, we significantly reduce the need for time consuming operations. In addition to workflow improvements, main building blocks such as $\mathsf{HomTrace}$, FFT multiplication, parameter evaluation, multi-bits LUT are carefully polished. Based on the improvement above, the homomorphic of AES can be speed up by 2.9× and the evaluation of 8-32 LUT can be speed up by $2^{21.84}\times$.

References

[1] Al Badawi, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: Openfhe: Open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 53–63. WAHC'22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560827.3563379, `https://doi.org/10.1145/3560827.3563379`

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

[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

[6] Carpov, S., Izabachène, M., Mollimard, V.: New Techniques for Multi-value Input Homomorphic Evaluation and Applications. In: Matsui, M. (ed.) CT-RSA 2019. pp. 106–126. Springer (2019). https://doi.org/10.1007/978-3-030-12612-4_6

[7] 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

[8] 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)

[9] Cheon, J.H., Coron, J.S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32. pp. 315–335. Springer (2013)

[10] 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)

[11] 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)

[12] 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

[13] 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

[14] 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

[15] Coron, J.S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17. pp. 311–328. Springer (2014)

[16] 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)

[17] Doröz, Y., Hu, Y., Sunar, B.: Homomorphic aes evaluation using the modified ltv scheme. Designs, Codes and Cryptography **80**, 333–358 (2016)

[18] 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)

[19] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the aes circuit. In: Annual Cryptology Conference. pp. 850–867. Springer (2012)

[20] 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

[21] Guimarães, A., Borin, E., Aranha, D.F.: Revisiting the functional bootstrap in tfhe. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 229–253 (2021)

[22] 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)

[23] Micciancio, D., Polyakov, Y.: Bootstrapping in fhew-like cryptosystems. In: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography. pp. 17–28 (2021)

[24] 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

[25] Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 113–124 (2011)

[26] 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

[27] 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)

[28] Wei, B., Lu, X., Wang, R., Liu, K., Li, Z., Wang, K.: Thunderbird: Efficient homomorphic evaluation of symmetric ciphers in 3gpp by combining two modes of tfhe. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 530–573 (2024)

[29] 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)

[30] 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

Supplementary Materials

A Re-analysis of WWL+

This section reanalyzes the circuit bootstrapping proposed in [27] and fixes previous errors.

The original NTT-based circuit bootstrapping method can be described as the following two steps.

Step 1 Refr. $\mathrm{LWE}_s(\Delta m)$ to refreshed $\mathrm{GLev}_{\mathbf{S}}(N^{-1}m + \cdots)$ by:
- PBSmanyLUT [14] without sample extraction, or
- automorphism-based multi-value blind rotation [27].

Step 2 Conv. $\mathrm{GLev}_{\mathbf{S}}(N^{-1}m + \cdots)$ to $\mathrm{GGSW}_{\mathbf{S}}(m)$ conversion by:
- HomTrace: $\mathrm{GLev}_{\mathbf{S}}(N^{-1}m + \cdots) \rightarrow \mathrm{GLev}_{\mathbf{S}}(m)$
- SchemeSwitch: $\mathrm{GLev}_{\mathbf{S}}(m) \rightarrow \mathrm{GGSW}_{\mathbf{S}}(m)$.

We propose Theorem 5 to provide a detailed re-analysis of the noise growth in the CBS algorithm proposed by Wang et al. [27].

Theorem 5. *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 prime. Then, our patched NTT-based CBS algorithm returns a GGSW ciphertext* $\mathbf{C}$ *of phase* $\mu + E_{\mathsf{cbs}}(X)$ *under the GLWE secret key* $\mathbf{S} = (S_1, \ldots, S_k)$ *corresponding to* $\mathbf{s}$ *where the variance* V_{cbs} *of* $E_{\mathsf{cbs}}(X)$ *is given as follows.*

$$V_{\mathsf{cbs}} \leq N^2 V_{\mathsf{pbs}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}.$$

where V_{pbs} *denotes the* PBSmanyLUT [23] *output error variance,* V_{tr} *denotes the* HomTrace *output error variance, and* V_{ss} *denotes the scheme switching output error variance.*

Proof. The Refr. step outputs a GLev ciphertext, where the phase of the j-th GLWE ciphertext is

$$N^{-1} \left\lceil \frac{q}{B^j} \right\rfloor m + y_1 X + \ldots + y_{N-1} X^{N-1} + E_{\mathsf{pbs}}(X),$$

where $y_i X^i$ are some redundant terms and $E_{\mathsf{pbs}}(X)$ is the PBSmanyLUT error. Subsequent trace evaluation can eliminate the power terms of X and multiply the constant term by a factor of N. Therefore the phase of HomTrace is

$$\left\lceil \frac{q}{B^j} \right\rfloor m + N e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X),$$

[23] In this paper, we focus on PBSmanyLUT, the conclusions deduced from the automorphism-based multi-value blind rotation are similar.

where e_{pbs} is the constant term of E_{pbs} and $E_{\mathsf{tr}}(X)$ is the error induced by $\mathsf{HomTrace}$. Lastly, the phase after scheme switching with S_i is

$$\begin{aligned}&\left(\left\lceil \frac{q}{B^j} \right\rceil m + N e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X)\right) \cdot S_i + E_{\mathsf{ss}}(X)\\ =&\left\lceil \frac{q}{B^j} \right\rceil m S_i + N e_{\mathsf{pbs}} S_i + E_{\mathsf{tr}}(X) S_i + E_{\mathsf{ss}}(X),\end{aligned}$$

where $E_{\mathsf{ss}}(X)$ is the error induced by scheme switching.

Since all of the additive errors $e_{\mathsf{pbs}} S_i$, $E_{\mathsf{tr}}(X) S_i$ and $E_{\mathsf{ss}}(X)$ are independent, the noise variance

$$V_{\mathsf{cbs}} = N^2 V(e_{\mathsf{pbs}} S_i) + V(E_{\mathsf{tr}}(X) S_i) + V_{\mathsf{ss}}.$$

Since the secret key S_i follows uniform binary distribution, we have $V(e_{\mathsf{pbs}} S_i) = V_{\mathsf{pbs}}$ thanks to e_{pbs} is only a constant term. Furthermore, $V(E_{\mathsf{tr}}(X) S_i) \leq \frac{N}{2} V_{\mathsf{tr}}$, where $\frac{N}{2}$ is the ring expansion factor. Substituting these estimates into the above formula, we obtain

$$V_{\mathsf{cbs}} \leq N^2 V_{\mathsf{pbs}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}.$$

□

Based on the noise analysis from Theorem 5, we have adjusted the noise control parameters, as listed in Table 11 and implemented them in `OpenFHE`. We note that the key size in this paper assumes that the evaluation keys are compressed (see Appendix B.7).

Sets	ℓ_{ep}	B_{ep}	ℓ_{tr}	B_{tr}	ℓ_{ss}	B_{ss}	ℓ	B	Max Depth	Key Size	# of NTTs	Time (ms)
CMux_O_1	2	2^{17}	3	2^{13}	1	2^{28}	4	2^3	8	30.6 MB	3610	102.5
CMux_O_2	3	2^{13}	5	2^{9}	1	2^{28}	4	2^4	520	45.9 MB	4840	128.2
CMux_O_3	7	2^{7}	7	2^{7}	2	2^{19}	4	2^5	271,500+	107 MB	9500	298.3

Table 11: The recommended parameter sets for refined NTT based CBS noise management. For each parameter set, we have listed the corresponding max supported circuit depth, circuit bootstrapping key size, and the number of needed NTT/FFTs.

B FHEW-like Cryptosystem Operations

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 length ℓ, we assume the decomposition error is uniformly sampled from $\llbracket -\frac{q}{2B^\ell}, \frac{q}{2B^\ell} \llbracket$ as analogous to [14]. As mentioned in Section 2.2, we only deal with the binary secret key in this section.[24] The proofs given in this section comes from [8, 14, 16, 7] with a slight modification generalizing GLWE dimension k.

B.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.*

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

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

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

□

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'.

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

B.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.

Algorithm 5: GLWE keyswitching $\mathsf{GLWE\_KS}$

```
    Input: $\mathbf{C} = \mathrm{GLWE}_{\mathbf{S}}(M)$ under $\mathbf{S} = (S_1, \ldots, S_k)$
    Input: $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i] = \mathrm{GLev}_{\mathbf{S}'}^{(B,\ell)}(S_i)$ for $i = 1, \ldots, k$ with decomposition base $B$
            and length $\ell$ under $\mathbf{S}' = (S'_1, \ldots, S'_{k'})$
    Output: $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}'}(M)$
1   $\mathbf{C} = (A_1, \ldots, A_k, A_{k+1})$
2   $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j] = \mathrm{GLWE}_{\mathbf{S}'}\left(\frac{q}{B^j} \cdot S_i\right)$ for $i \in [k]$ and $j \in [\ell]$
3   $\mathbf{C}' \leftarrow (0, \cdots, 0, A_{k+1}) = \mathrm{GLWE}^0_{\mathbf{S}'}(A_{k+1}) \in \mathcal{R}_{q,N}^{k'+1}$
4   for $i = 1$ to $k$ do
5       Decompose $A_i$ as $\sum_{j=1}^{\ell} A'_{i,j} \cdot \frac{q}{B^j} + E'_i$ with $\|A'_{i,j}\|_\infty \le \frac{B}{2}$ and $\|E'_i\|_\infty \le \frac{q}{2B^\ell}$
6       $\mathbf{C}' \leftarrow \mathbf{C}' - \sum_{j=1}^{\ell} A'_{i,j} \cdot \mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}[i][j]$
7   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.*

$$V_{\mathsf{ks}} \le kN\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right) + k\ell N\left(\frac{B^2+2}{12}\right)\sigma^2_{\mathbf{S}\to\mathbf{S}'}.$$

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 $E'_i \leftarrow [\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$, $A_{i,j} \leftarrow [\![-B/2, B/2[\![$ 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.

$$\mathrm{Var}(E_{\mathsf{ks}}) \le kN\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right) + k\ell N\left(\frac{B^2+2}{12}\right)\sigma^2_{\mathbf{S}\to\mathbf{S}'}.$$

□

B.3 Functional Keyswitching

In this subsection, we summarize the LWE to GLWE public/private functional keyswitching. For the detailed analysis of the keyswitching, we refer to [12, 14].

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 length, 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,

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

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. length). 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.4 Homomorphic 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.

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

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.

Computing the trace by its definition requires one to compute the automorphism N times. For efficient homomorphic trace evaluation, Chen et al. [8] 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.

$$\begin{aligned} \mathrm{Tr}_{N/n}(\mu(X)) &= \mathrm{Tr}_{N/(N/2)} \circ \cdots \circ \mathrm{Tr}_{2n/n}(\mu(X)) \\ &= \frac{N}{n}(\mu_0 + \mu_{\frac{N}{n}} X^{\frac{N}{n}} + \cdots + \mu_{N-\frac{N}{n}} X^{N-\frac{N}{n}}) \end{aligned} \tag{5}$$

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 B.2 for the details of GLWE keyswitching.

Finally, we give Lemma 3 to measure the HomTrace output noise:

Lemma 3 (HomTrace 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* $\mathrm{Tr}(\mu) + E_{\mathsf{tr}}$ *under* $\mathbf{S}$ *where the variance* V_{tr} *of* E_{tr} *is given as follows.*

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

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

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

□

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

B.5 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.

Algorithm 6: SchemeSwitch

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

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.*

$$V_{\mathsf{ext}} \leq (1 + kN)\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right)\ell_2(M)^2 + (k+1)\ell N \left(\frac{B^2+2}{12}\right)\sigma_{\mathsf{ext}}^2.$$

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 $\mathrm{Var}(E_{i,j}) = \sigma_{\mathsf{ext}}^2$ 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 $\|A'_{i,j}\|_\infty \leq \frac{B}{2}$ and $\|E'_{i,j}\|_\infty \leq \frac{q}{2B^\ell}$ 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 $E'_i \leftarrow [\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$, $A'_{i,j} \leftarrow [\![-B/2, B/2[\![$ and $\mathbf{S}$ is a binary secret key, the variance of E_{ks} is given as follows.

$$\mathrm{Var}(E_{\mathsf{ks}}) \leq (1 + kN)\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right)\ell_2(M)^2 + (k+1)\ell N \left(\frac{B^2+2}{12}\right)\sigma_{\mathsf{ext}}^2.$$

□

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_{in}^2. Let σ_{ssk}^2 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* $V_{\mathsf{out}} \le \frac{N}{2} \cdot \sigma_{\mathsf{in}}^2 + V_{\mathsf{ss}}$ *where*

$$V_{\mathsf{ss}} \le \frac{(1+kN)N}{2}\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell_{\mathsf{ss}}}} + \frac{1}{12}\right) + (k+1)\ell N\left(\frac{B_{\mathsf{ss}}^2 + 2}{12}\right)\sigma_{\mathsf{ssk}}^2.$$

B.6 Programmable Bootstrapping and Its Failure Probability

All the homomorphic operations increase the internal noise, while only programmable bootstrap refreshes it. Hence, the error variance of the output ciphertext of PBS is independent of that of the input ciphertext, provided that the PBS operation succeeds. Chillotti et al. [14] proposed a theorem about the error variance of PBSmanyLUT and its failure probability. It describes PBSmanyLUT on FFT domains where only one ciphertext modulus q is used and it is switched into $2N$ temporarily just before PBSmanyLUT, while it is analogous on NTT domains where various ciphertext moduli are used.

Suppose that an LWE ciphertext of input noise variance σ_{in}^2 and a scaling factor Δ_{in} is given to PBSmanyLUT evaluating 2^{ϑ} LUTs. Then the failure probability of PBSmanyLUT is at most $P = 1 - \mathrm{erf}\left(\frac{\Gamma}{\sqrt{2}}\right)$, 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 $\omega = 2N \cdot 2^{-\vartheta}$. Provided that PBSmanyLUT does not fail, the error variance V_{pbs} of the output satisfies the following.

$$V_{\mathsf{pbs}} \le n\ell(k+1)N\frac{B^2+2}{12}\sigma_{\mathsf{bsk}}^2 + n\frac{q^2 - B^{2\ell}}{24B^{2\ell}}\left(1 + \frac{kN}{2}\right) + \frac{nkN}{32} + \frac{n}{16}\left(1 - \frac{kN}{2}\right)^2$$

where B and ℓ are gadget decomposition base and gadget length, respectively, and σ_{bsk}^2 is the error variance of the bootstrapping key. We refer to [14] for the details.

B.7 Evaluation Key Size

In this subsection, we describe the size of various evaluation keys used in FHEW/TFHE according to the parameters, summarizing the result in Table 12. As evaluation keys are encryptions of secret information, we begin with the description of ciphertext sizes.

Ciphertext Size An LWE ciphertext $(a_1, \dots, a_n, b) \in \mathbb{Z}_q^{n+1}$ 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 $\log q$.[25] 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 12a summarizes the size of each type of FHEW/TFHE ciphertext.

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 $(B_{\mathsf{tr}}, \ell_{\mathsf{tr}})$.

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 12b 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

[25] 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.

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 $S_{k+1} = -1$.[26] Table 12c summarizes the evaluation keys for the bootstrapping operations in FHEW/TFHE.

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

(a) Size of FHEW/TFHE ciphertexts in bits.

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

(b) Size of various FHEW/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 FHEW/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 12: Size of FHEW/TFHE ciphertexts and evaluation keys in bits. The size of seeds or auxiliary information is ignored.

C Error Analysis of the Split FFT

Klemsa [22] proposed an upper bound for the FFT error of (negacyclic) polynomial multiplication as follows[27]:

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

However, the experimental result shows that the above theoretical bound is a loose upper bound to be used in practice. It seems to come from the gap

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

[27] The second-order terms are neglected.

between the worst-case and average-case analysis, since most FHEW/TFHE parameters are chosen based on the average-case analysis using the independence heuristic [12, 14]. So that we use Bergerat et al.'s method [3] to estimate the FFT error in this paper.

We then analyze the error variance of the split FFT, described in Section 3.2. 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 13 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 (4). The Failure Prob. column denotes the failure probability of the exact polynomial multiplication of the upper part in the split FFT, computed by the Std. Dev. column. We compare the FFT error induced by the lower part to the gadget decomposition error $q/B^{2\ell}$, and conclude that the split FFT allows us to neglect the FFT error during HomTrace.

	N	k	B	ℓ	b	Upper Part		Lower Part	
						Std. Dev.	Failure Prob.	Std. Dev.	$q/B^{2\ell}$
WOPBS_2_2	2048	1	2^{7}	7	36	$2^{-6.90}$	2^{-2565}	$2^{1.10}$	2^{14}
WOPBS_3_3	2048	2	2^{12}	4	41	$2^{-6.80}$	2^{-2245}	$2^{11.2}$	2^{15}
	2048	1	2^{15}	3	44	$2^{-7.51}$	2^{-5978}	$2^{16.49}$	2^{18}
	2048	2	2^{13}	3	42	$2^{-7.01}$	2^{-2992}	$2^{12.99}$	2^{24}
WOPBS_4_4	2048	2	2^{9}	6	39	$2^{-7.51}$	2^{-5978}	$2^{6.49}$	2^{9}
	2048	1	2^{9}	6	39	$2^{-7.51}$	2^{-5978}	$2^{16.49}$	2^{18}
	2048	2	2^{9}	6	39	$2^{-6.80}$	2^{-2245}	$2^{7.20}$	2^{23}
CMux_1	2048	1	2^{13}	3	42	$2^{-7.51}$	2^{-5978}	$2^{12.49}$	2^{24}
CMux_2	2048	1	2^{13}	3	42	$2^{-7.51}$	2^{-5978}	$2^{12.49}$	2^{24}
CMux_3	2048	1	2^{8}	6	37	$2^{-7.01}$	2^{-2992}	$2^{2.99}$	2^{15}
AES (LHE)	1024	2	2^{13}	3	41	$2^{-7.01}$	2^{-2992}	$2^{10.99}$	2^{15}
AES (flex. LHE)	1024	2	2^{6}	8	35	$2^{-7.30}$	2^{-4485}	$2^{-1.30}$	2^{15}

Table 13: Standard deviations of the split FFT for the GLWE keyswitching under the parameters used in this paper.

D Packing Method in The LHE Mode

1. **Horizontal packing:** The core idea of horizontal packing is to compute all sub-LUTs L_i simultaneously, rather than separately. Specifically, the s sub-LUT results $L_i(x)$ corresponding to the same input x are embedded into a single GLWE ciphertext using coefficient encoding and evaluate the binary decision tree as described before. Finally using the cost-free sample

extraction to extract the s results simultaneously, thereby reducing the total number of CMux gates from $s \cdot (2^d - 1)$ to $2^d - 1$. It is noteworthy that under coefficient encoding, a RLWE sample can accommodate up to N messages, and in practice s is always less than N.

2. **Vertical packing:** In vertical packing, the N inputs of the sub-LUT f_i are encoded into a single GLWE sample. Consequently, for each sub-LUT, the inputs are divided into $2^d/N$ blocks. Now we only need the $d - \log N$ most significant bits to compute a CMux tree with depth $d - \log N$, rather than the full CMux tree. After that, we obtain the block that contains $L_i(x)$. And the $\log N$ least significant bits are utilized in blind rotation to bring the coefficient $L_j(x)$ in constant term and followed with sample extraction. Using this method, the total number of CMux gates is reduced to $2^d/N - 1 + \log N$.

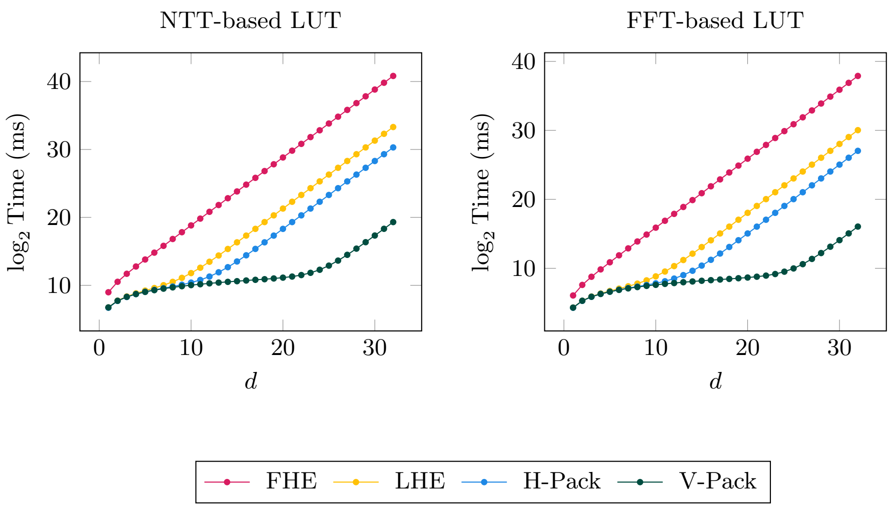


Fig. 8: Comparison of LUT evaluation using GBS and CBS.

E Transciphering and AES Evaluation

Transciphering is an innovative approach that combines the strengths of symmetric encryption with FHE to address the challenges of ciphertext expansion.

Specifically, FHE ciphertexts are typically much larger than the original plaintext, posing significant challenges for devices with limited resources. Transciphering, first proposed by Naehrig et al. [25], offers a solution to this problem. The core idea is to use symmetric encryption for data transmission, which is then converted into homomorphic ciphertext by the server for further processing. Here's how it works:

1. **Symmetric encryption for transmission:** The client encrypts the data using a symmetric encryption scheme, resulting in a ciphertext $\mathcal{E}_k(m)$. Then it generates a homomorphic encryption of the symmetric key, $\mathsf{Enc}(k)$, sends both of them to the server.
2. **Homomorphic decryption and evaluation on the server:** The server homomorphically evaluates the decryption circuit of the symmetric encryption using $\mathsf{Enc}(k)$ and $\mathcal{E}_k(m)$ to obtain the homomorphic ciphertext of the data, then perform homomorphic evaluation.

$$\mathsf{Eval}_{\mathcal{E}^{-1}}(\mathsf{Enc}(k), \mathcal{E}_k(m)) = \mathsf{Enc}(\mathcal{E}^{-1}(k, \mathcal{E}_k(m))) = \mathsf{Enc}(m)$$

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. [8] 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 $m + v_1 X + \cdots + v_{N-1} X^{N-1}$ under $\mathbf{S}_{\mathsf{src}}$ by $\mathsf{LWEtoGLWEConst}$ 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 $m + v_1 X + \cdots + v_{N-1} X^{N-1}$ 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. [8] 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 FFT-based CBS. In our LHE mode, 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 to keep the plaintext encoding.

When our flexible LHE mode is used, the S-box output is redundantly obtained in a gadget decomposed form. Since the scaling factors are smaller than $\lceil q/2 \rceil$, homomorphic addition no longer corresponds to XOR, so the subsequent linear operations work as integer addition, increasing the magnitude of the internal message. To reduce the error growth by non-binary message space in the flexible LHE mode, it is important to reduce the number of additions.

For that purpose, we adopt two modified AES evaluation techniques. The first one is evaluating 8-24 LUT to pre-compute the field multiplication in the MixColumns layer. Since the cost of evaluating LUT is much smaller than that of circuit bootstrapping, we can almost freely pre-compute the field multiplication for the MixColumns layer, reducing the number of additions in the subsequent linear layer. The other one is using keyed S-box to skip the AddRoundKey layer. By transferring the keyed S-box instead of the ciphertexts of the round key, one can integrate AddRoundKey and SubBytes.

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. In this perspective, 8-24 LUT and keyed S-box evaluation allow us to choose a compact parameter in the (flexible) LHE mode.

F Detailed Description for Our HP-LHE Mode

Idea. First, we integrate the Extr. step with the Refr. step by $\mathsf{PBSmanyLUT}$, reducing the number of overall PBS operations. The Extr. step extracts each bit of the message from the ciphertext, obtaining ciphertexts containing message bits scaled by $\lceil q/2 \rceil$. To extract the message bit by PBS without increasing the polynomial size, the Extr. step moves the LSB to the MSB by constant multiplication, changes its scaling factor to subtracts it from original the ciphertext, and repeats this process until all the bits are extracted. The followed Refr. step changes the scaling factor of the extracted ciphertext to the gadget components by PBS. We found that both Extr. and Refr. perform PBS operations to change the scaling factor of a single-bit ciphertext, so we integrate them into a single $\mathsf{PBSmanyLUT}$ operation per each bit.

After the Refr. step, the resulting GLev ciphertext is converted to the GGSW ciphertext in the Conv. step. The WWL+ method has improved the Conv. step significantly in terms of both computation time and key sizes in the bit-wise input LHE setting, while it's higher error growth compared to private key switching makes it hard to be used in the high-precision input LHE mode. We resolved this issue by proposing a high-precision HomTrace method based on GLWE dimension switching. By performing HomTrace under a larger GLWE dimension, we obtained high enough precision at the cost of increased computation cost.

Lastly, we propose a multi-bit extraction method to reduce the number of PBSmanyLUT operations further. The number of PBSmanyLUT operations is the same with the number of the extraction operations, so we extracts message bits in a small chunk of τ-bit, where $\tau \in \{2, 3\}$, using a single PBSmanyLUT operation combined with the MV-PBS method [6]. The extracted bit is masked by the MSB of the extracted chunk, while we can evaluate LUT correctly by modifying the function to evaluate using the masked input bits.

Algorithm 7: CBS of HP-LHE Framework

```
Input: $\mathbf{c} = \mathrm{LWE}_{\mathbf{s}}(\lceil q/2^{\delta} \rceil \cdot m)$ where $\tau \mid \delta$
Input: $\ell$ test polynomials $F_i$ to compute $\frac{v_i}{2} \cdot (-1)^{x_{\tau-1}+1}$ from a $\tau$-bit input
       $x = \sum_{k=0}^{\tau-1} x_k \cdot 2^k$ for $i = 1, \ldots, \ell$
Input: $\tau - 2$ polynomials $G_j$ such that $G_j \cdot F_i$ becomes a test polynomial to
       compute $\frac{v_i}{2} \cdot (-1)^{(x_{\tau-1} \oplus x_j)+1}$ from a $\tau$-bit input $x = \sum_{k=0}^{\tau-1} x_k \cdot 2^k$ for
       $j = 0, \ldots, \tau - 2$
Input: Bootstrapping keys under $\mathbf{S}$
Input: Automorphism keys under $\mathbf{S}$
Input: Scheme switching key under $\mathbf{S}$
Output: $\overline{\overline{\mathbf{C}}}_j = \mathrm{GGSW}(m'_j)$ for $j = 0, \ldots, \delta - 1$ where $m'_j = m_j$ if $\tau \mid (j+1)$
        and $m'_j = m_{\lceil (j+1)/\tau \rceil \cdot \tau} \oplus m_j$ otherwise.
1  for $k = 0$ to $\delta/\tau - 1$ do
2      $\mathbf{c}' \leftarrow 2^{\delta-(k+1)*\tau} \cdot \mathbf{c}$    /* $\mathbf{c}' = \mathrm{LWE}(\frac{q}{2^{\tau}} \cdot \sum_{j=0}^{\tau-1} m_{k\cdot\tau+j} \cdot 2^j)$ */
3      $\overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)} \leftarrow \mathsf{PBSmanyLUT}\left(\mathbf{c}'; (F_i)_{i=1}^{\ell}\right)$ (except sample extraction)
        /* $\overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)}[i] = \mathrm{GLWE}(\frac{v_i}{2} \cdot (-1)^{m_{k\cdot\tau+(\tau-1)}+1} + \ldots)$ */
4      for $i = 1$ to $\ell$ do
5          for $j = 0$ to $\tau - 2$ do
6              $\overline{\mathbf{C}}_{k\cdot\tau+j}[i] \leftarrow G_j \cdot \overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)}$
                /* $\overline{\mathbf{C}}_{k\cdot\tau+j}[i] = \mathrm{GLWE}(\frac{v_i}{2} \cdot (-1)^{(m_{k\cdot\tau+(\tau-1)} \oplus m_{k\cdot\tau+j})+1} + \ldots)$ */
7              $\overline{\mathbf{C}}_{k\cdot\tau+j}[i] \leftarrow \overline{\mathbf{C}}_{k\cdot\tau+j}[i] + \mathrm{GLWE}^0(v_i/2)$
                /* $\overline{\mathbf{C}}_{k\cdot\tau+j}[i] = \mathrm{GLWE}(v_i \cdot (m_{k\cdot\tau+(\tau-1)} \oplus m_{k\cdot\tau+j}) + \ldots)$ */
8          $\overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)}[i] \leftarrow \overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)} + \mathrm{GLWE}^0(v_i/2)$
            /* $\overline{\mathbf{C}}_{k\cdot\tau+(\tau-1)}[i] = \mathrm{GLWE}(v_i \cdot m_{k\cdot\tau+(\tau-1)} + \ldots)$ */
9      for $j = 0$ to $\tau - 1$ do
10         $\overline{\overline{\mathbf{C}}}_{k\cdot\tau+j} \leftarrow \mathsf{Conv}(\overline{\mathbf{C}}_{k\cdot\tau+j})$    /* $\overline{\overline{\mathbf{C}}}_{k\cdot\tau+j} = \mathrm{GGSW}(m'_{k\cdot\tau+j})$ */
11     $\mathbf{C}'' \leftarrow \mathsf{Circuit}_{(m'_{k\cdot\tau+(\tau-1)}, \ldots, m'_{k\cdot\tau}) \to \frac{q}{2^{\delta-k\cdot\tau}} \cdot L[\sum_{j=0}^{\tau-1} m_{k\cdot\tau+j} \cdot 2^j]}\left((\overline{\overline{\mathbf{C}}}_{k\cdot\tau+j})_{j=0}^{\tau-1}\right)$
12     $\mathbf{c}'' \leftarrow \mathsf{SampleExtract}(\mathbf{C}'')$    /* $\mathbf{c}'' = \mathrm{LWE}(\frac{q}{2^{\delta-k\cdot\tau}} \sum_{j=0}^{\tau-1} m_{k\cdot\tau+j} \cdot 2^j)$ */
13     $\mathbf{c} \leftarrow \mathbf{c} - \mathbf{c}''$    /* $\mathbf{c} = \mathrm{LWE}(\frac{q}{2^{\delta-(k+1)\tau}} \sum_{j=0}^{\delta-(k+1)\tau} m_{j+(k+1)\tau} \cdot 2^j)$ */
14 return $(\overline{\overline{\mathbf{C}}}_j)_{j=0}^{\delta-1}$
```

Multi-Bit Extraction. For simplicity, we only describe the case of $\tau = 2$. Suppose that an input ciphertext $\mathrm{LWE}(\lceil q/2^2 \rceil \cdot (2m_1 + m_0))$ of a 2-bit message chunk is given. Then one can output both $\mathrm{LWE}(v \cdot m_1)$ and $\mathrm{LWE}(v \cdot (m_1 \oplus m_0))$ by 1-depth PBS where v is a component of the gadget vector. Although WoP-PBS originally outputs $\mathrm{GGSW}(m_0)$ and $\mathrm{GGSW}(m_1)$ to use them evaluate LUT by CMux, it is also possible to evaluate the same function using CMux by $\mathrm{GGSW}(m_0 \oplus m_1)$ and $\mathrm{GGSW}(m_1)$ by slightly modifying the table.

To compute both $\{\mathrm{LWE}(v_j \cdot m_1)\}_{j=1}^{\ell}$ and $\{\mathrm{LWE}(v_j \cdot (m_1 \oplus m_0))\}_{j=1}^{\ell}$ in a single $\mathsf{PBSmanyLUT}$ operation, ϑ should be increased by 1 to evaluate twice many functions. However, using a larger ϑ increases the failure probability of the $\mathsf{PBSmanyLUT}$ operation. Combined with the small scaling factor $\lceil q/2^2 \rceil$ of the input ciphertext in the multi-bit extraction, increasing ϑ might lead too large failure probability. To keep the value of ϑ, we opted the multi-value PBS proposed by Carpov et al. [6].

The test polynomial to compute $\mathrm{LWE}(v \cdot m_1)$ from $\mathrm{LWE}(\lceil q/2^2 \rceil (2m_1 + m_0))$ using PBS is given as follows.

$$f_{m_1}(X) = -\frac{v}{2}\left(1 + X + \cdots + X^{N-1}\right).$$

Blind rotation on this test polynomial outputs $\mathrm{GLWE}((-1)^{m_1+1}\frac{v}{2} + \dots)$, and one can obtain $\mathrm{GLWE}(v \cdot m_1 + \dots)$ by adding a constant $v/2$. The test polynomial f_{m_1} is, in fact, the same with the common test polynomial used in the MV-PBS, so one can evaluate additional functions (of the same scaling factor) by multiplying some polynomial to the blind rotation output on f_{m_1}. For example, the test polynomial $f_{m_1 \oplus m_0}$ to compute $\mathrm{LWE}(v \cdot (m_1 \oplus m_0))$ from the same input is given as follows.

$$f_{m_1 \oplus m_0}(X) = -\frac{v}{2}\left(1 + X + \cdots + X^{N/2-1} - X^{N/2} - \cdots - X^{N-1}\right).$$

From $f_{m_1 \oplus m_0}(X) = -X^{N/2} \cdot f_{m_1}(X)$, the MV-PBS method evaluates $f_{m_1 \oplus m_0}$ on the same input by multiplying $-X^{N/2}$ to the blind rotation output on f_{m_1}.

G Proofs

G.1 Theorem.1

Proof. The Refr. step outputs a GLev ciphertex, where the phase of the j-th GLWE ciphertext is

$$\left\lceil \frac{q}{B^j} \right\rceil m + y_1 X + \ldots + y_{N-1} X^{N-1} + E_{\mathsf{pbs}}(X),$$

where $y_i X^i$ are some redundant terms and $E_{\mathsf{pbs}}(X)$ is the $\mathsf{PBSmanyLUT}$ error. After pre-processing by multiplying with N^{-1}, the phase is

$$N^{-1}\left\lceil \frac{q}{B^j} \right\rceil m + N^{-1} y_1 X + \ldots + N^{-1} y_{N-1} X^{N-1} + N^{-1} E_{\mathsf{pbs}}(X).$$

Subsequent trace evaluation can eliminate the the power terms of X and multiply the constant term by a factor of N. Therefore the phase of $\mathsf{HomTrace}$ is

$$\left\lceil \frac{q}{B^j} \right\rceil m + e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X),$$

where e_{pbs} is the constant term of E_{pbs} and $E_{\mathsf{tr}}(X)$ is the error induced by $\mathsf{HomTrace}$. Lastly, the phase after i-th scheme switching is

$$\begin{aligned}&\left(\left\lceil \frac{q}{B^j} \right\rceil m + e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X)\right) \cdot S_i + E_{\mathsf{ss}}(X)\\ =& \left\lceil \frac{q}{B^j} \right\rceil mS_i + e_{\mathsf{pbs}}S_i + E_{\mathsf{tr}}(X)S_i + E_{\mathsf{ss}}(X),\end{aligned}$$

where $E_{\mathsf{ss}}(X)$ is the error induced by scheme switching. Since all of the additive errors $e_{\mathsf{pbs}}S_i$, $E_{\mathsf{tr}}(X)S_i$ and $E_{\mathsf{ss}}(X)$ are independent, the noise variance is

$$V_{\mathsf{cbs}} = V(e_{\mathsf{pbs}}S_i) + V(E_{\mathsf{tr}}(X)S_i) + V_{\mathsf{ss}}.$$

Given that the secret key S_i follows uniform binary distribution, we have $V(e_{\mathsf{pbs}}S_i) = V_{\mathsf{pbs}}$ thanks to e_{pbs} is only a constant term. Furthermore, $V(E_{\mathsf{tr}}(X)S_i) \leq \frac{N}{2}V_{\mathsf{tr}}$, where N is the ring expansion factor. Substituting these estimate into the above formula, we obtain

$$V_{\mathsf{cbs}} \leq V_{\mathsf{pbs}} + \frac{N}{2}V_{\mathsf{tr}} + V_{\mathsf{ss}}.$$

□

G.2 Theorem.2

Proof. The proof is analogous to that of Theorem 1 except that there is an additional pre-processing error. The Refr. step outputs a GLev ciphertext whose j-th GLWE ciphertext is

$$\frac{q}{B^j}m + y_1X + \cdots + y_{N-1}X^{N-1} + E_{\mathsf{pbs}}(X),$$

where y_iX^i are some redundant terms and $E_{\mathsf{pbs}}(X)$ is the $\mathsf{PBSmanyLUT}$ error. After pre-processing, one obtains a GLev ciphertext whose j-th GLWE ciphertext has a phase of

$$\frac{1}{N}\left(\frac{q}{B^j}m + y_1X + \cdots + y_{N-1}X^{N-1} + E_{\mathsf{pbs}}(X)\right) + E_{\mathsf{ms}}(X) + \frac{q}{N}U(X)$$

where $E_{\mathsf{ms}}(X)$ is the modulus switching error from q to q/N and $U(X)$ is a redundant terms caused by modulus raising from q/N to q. Subsequent trace evaluation eliminates all the coefficients except the constant term, which is multiplied by N. Hence, the phase after $\mathsf{HomTrace}$ is

$$\frac{q}{B_j}m + e_{\mathsf{pbs}} + Ne_{\mathsf{ms}} + E_{\mathsf{tr}}(X)$$

where e_{pbs} (resp. e_{ms}) is the constant term of $E_{\mathsf{pbs}}(X)$ (resp. $E_{\mathsf{ms}}(X)$) and $E_{\mathsf{tr}}(X)$ is the error induced by $\mathsf{HomTrace}$. The final scheme switching operation converts $\mathrm{GLev}(m)$ obtained from $\mathsf{HomTrace}$ into $\mathrm{GGSW}(m)$, whose error variance V_{cbs} is given as follows.

$$V_{\mathsf{cbs}} \leq V_{\mathsf{pbs}} + N^2 V_{\mathsf{ms}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}}$$

where V_{ss} is the scheme switching error. $\square$