# Refined TFHE Leveled Homomorphic Evaluation and Its Application


Ruida Wang[*†]
wangruida@iie.ac.cn
Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
School of Cybersecurity, UCAS
Beijing, China

Jincheol Ha[*‡]
jincheolha@cryptolab.co.rk
CryptoLab Inc.
Seoul, Korea

Xuan Shen
shenxuan@iie.ac.cn
Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
School of Cybersecurity, UCAS
Beijing, China

Xianhui Lu[§¶]
luxianhui@iie.ac.cn
Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
School of Cybersecurity, UCAS
Beijing, China

Chunling Chen
chenchunling@iie.ac.cn
Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
School of Cybersecurity, UCAS
Beijing, China

Kunpeng Wang
wangkunpeng@iie.ac.cn
Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS
School of Cybersecurity, UCAS
Beijing, China

Jooyoung Lee[‖¶]
hicalf@kaist.ac.kr
KAIST
Daejeon, Korea



## Abstract

TFHE is a fully homomorphic encryption scheme over the torus that supports fast bootstrapping. Its primary evaluation mechanism is based on gate bootstrapping and programmable bootstrapping (PBS), which computes functions while simultaneously refreshing noise. PBS-based evaluation is user-friendly and efficient for small circuits; however, the number of bootstrapping operations increases exponentially with the circuit depth. To address the challenge of efficiently evaluating large-scale circuits, Chillotti et al. introduced a leveled homomorphic evaluation (LHE) mode at Asiacrypt 2017. This mode decouples circuit evaluation from bootstrapping, resulting in a speedup of hundreds of times over PBS-based methods. However, the remaining circuit bootstrapping (CBS) becomes a performance bottleneck, even though its frequency is linear with the circuit depth.

In this paper, we refine the LHE mode by mitigating the high cost of CBS. First, we patch the NTT-based CBS algorithm proposed by Wang et al. [WWL+, Eurocrypt 2024], accelerating their algorithm by up to 2.6×. Then, observing the suboptimal parallelism and high complexity of modular reduction in NTT under CBS parameters, we extend WWL+ to an FFT-based algorithm by redesigning the pre-processing method and introducing a split FFT technique. This achieves the fastest CBS implementation with the smallest key size, outperforming the open-source WWL+ implementation by up to 12.1× (resp. 5.12× compared to our patched algorithm), and surpassing TFHEpp [MBM+, USENIX 2021] by 3.42× with a key size reduction of 33.2×. Furthermore, we proposed an improved integer input LHE mode by extending our CBS algorithm to support higher precision and combining it with additional optimizations such as multi-bit extraction. Compared to the previous integer input LHE mode proposed by Bergerat et al. [BBB+, JoC 2023], our approach is up to 10.7× faster with a key size reduction of up to 4.4×.

To demonstrate the practicality of our improved LHE mode, we apply it to AES transciphering and general homomorphic look-up table (LUT) evaluation. For AES evaluation, our method is 4.8× faster and reduces the key size by 31.3× compared to the state-of-the-art method, Thunderbird [WLW+, TCHES 2024]. For LUT evaluation, we compare our results with the recent work of Trama et al. [TCBS, ePrint 2024/1201], which constructs a general 8-bit processor of TFHE. Our method not only achieves faster 8-to-8 LUT evaluation but also improves the efficiency of most heavy 8-bit bivariate instructions by up to 21× and the 16-bit sigmoid function by more than 26×.



## Keywords

Homomorphic Encryption, TFHE, Leveled Homomorphic Evaluation, Circuit Bootstrapping, Transciphering, FHE Processor.


[*] Both authors contributed equally to this research.


[†] Ruida Wang was supported by the National Key R&D Program of China under Grant No. 2023YFB4503200, the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDB0690200.


[‡] This work was done while Jincheol Ha was a PhD student at KAIST.


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


[¶] Xianhui Lu and Jooyoung Lee are corresponding authors.


[‖] 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 [12–14], proposed as an extension of FHEW [25], is a homomorphic encryption scheme over torus that supports fast bootstrapping to refresh noise.

TFHE is first proposed as 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 [8] or programmable bootstrapping [15] (PBS). The evaluation strategy based on PBS, referred to as FHE mode, combines function evaluation and noise refreshing 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 since the number of bootstrapping grows exponentially with the circuit depth.

To deal with this issue, Chillotti et al. [13] introduced the Leveled Homomorphic Evaluation (LHE) mode for TFHE, which separates circuit evaluation from noise refreshing. The LHE mode consists of three steps: (1) Eval., an evaluation step to compute circuit, (2) Refr., a refreshing step to reduce noise, and (3) Conv., a conversion step to convert the ciphertext type for next Eval.. The latter two are generally indivisibly sequential, known as circuit bootstrapping (CBS). The number of CBS is always proportional to the circuit depth.

By utilizing a controlled selector gate (CMux gate) operation instead of PBS for Eval., LHE mode significantly enhances circuit evaluation efficiency compared to the FHE mode, as PBS itself consists of hundreds of CMux gates. However, despite the advancement in the evaluation step, the high cost of CBS in the LHE mode becomes a performance bottleneck. There are two milestone breakthroughs among the numerous attempts to improve CBS [15, 28, 35, 41, 44, 45]. First, Chillotti et al. [15] proposed the PBS evaluating multiple look-up tables, known as PBSmanyLUT at Asiacrypt 2021. By enabling mutiple look-up tables to be processed simultaneously, this innovation significantly accelerates the Refr. step , which is conducted through multple PBS, thereby enhancing the efficiency of CBS. More recently, Wang et al. [45] introduced a faster and smaller circuit bootstrapping mode at Eurocrypt 2024 (referred to as the WWL+ method in this paper). This method replaces the heavy private keyswitching operation with homomorphic trace evaluation (HomTrace) [10] and scheme switching (SchemeSwitch) [21], significantly reducing the asymptotic computational and storage complexity of the Conv. step.

## 1.1 Motivation

However, the LHE mode still faces several challenges.

*A. The Missed Multiplicative Error Growth in WWL+*[1]*.* The circuit bootstrapping algorithm proposed by Wang et al. encounters multiplicative error growth issues. Specifically, HomTrace amplifies the input noise by a factor of $N$, where $N$ denotes the ring dimension. This necessitates their work increasing bootstrapping parameters, significantly impacting the efficiency of their CBS algorithm.

*B. The Discrepancy Between Asymptotic Complexity and Concrete Cost.* In homomorphic encryption schemes, polynomial computation is critical. Typically, the Number Theoretic Transform (NTT) is employed when the modulus is prime, while Fast Fourier Transform (FFT) is preferred when modulus is a power of two to accelerate polynomial computations. In the CBS mode proposed by WWL+, to mitigate the phase amplification introduced by (HomTrace), it leveraged the inverse of $N$ to ensure algorithm correctness, restricting this method to NTT-based systems (since $N^{-1}$ does not exist in the FFT setting). However, a gap exists between their theoretical optimizations and actual implementation performance[2]. This discrepancy arises because the concrete cost of NTT is higher than FFT for large modulus sizes in LHE mode (54-bit in NTT setting, 64-bit in FFT setting), due to suboptimal parallelism and the high complexity of modular reduction in NTT calculation.

*C. The Inflexibility and Computational Intensity Problem.* In the current LHE mode [13, 15, 45], the Conv. step is always paired with the heavy Refr. step, executing together in what we known as circuit bootstrapping. This coupling leads to a inflexibility and unnecessary overhead when evaluating small-scale circuits.

*D. User-Unfriendly Concerns.* Compared to the FHE mode, the LHE mode has more complex parametrization. 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 TFHE scheme.

*E. A Lack of Efficient Integer Input LHE Solution.* Real-world applications always encode data into integers rather than single bits. To process such inputs in the LHE mode, Bergerat et al. presented a new programmable bootstrapping without padding algorithm (new WoP-PBS) in JoC23 [4], denoted as the BBB+ method in this paper. This method involves homomorphically extracting multi-bit data into single bits and converting ciphertext type for CMux evaluation, thus supporting higher precision to handle integer messages. Despite its capabilities, the BBB+ method itself is quite heavy and may be up to 100 times slower than single-bit circuit bootstrapping.

## 1.2 Our Contributions

In this paper, we address the aforementioned issues and significantly boost the efficiency of the LHE mode in TFHE. Subsequently, we apply this enhanced mode to two practical applications: AES transciphering and look-up table (LUT) evaluation, aiming for superior performance. The details of the contribution are as follows:

*Patched and Extended FFT-based CBS Algorithm.* To address the error growth issue in WWL+ (Challenge A), we first patch the NTT-based CBS by adopting a newly designed pre-processing method. This approach enhances the WWL+ performance by up to 2.6×.

Furthermore, to improve the implementation performance of CBS in response to Challenge B, we extend the patched algorithm into an FFT-based one. This shift addresses two critical technical

[1]The authors of [45] confirmed this issue and presented adjusted parameter sets and performance at Eurocrypt 2024 conference.

[2]The CBS implementation in WWL+ achieves only a 1.31× speedup compared to the FFT-based CBS of TFHEpp (refer to Table 9 in [45]), rather than their proposed 3.5×.

issues, rather than merely an implementation difference. First, we propose a FFT pre-processing method to tackle Challenge A, as the patch used for WWL+ method is incompatible with FFT parameter setting. Next, we propose a split FFT to handle the FFT error, which is amplified by HomTrace and disrupts the algorithm's correctness. By integrating these approaches and optimizing the parameters, we implement an improved FFT-based CBS algorithm, outperforming the open-source NTT-based WWL+ implementation by up to 12.1×, and surpassing FFT-based CBS of TFHEpp [35, 36] by 3.4× with a key size reduction of 33.2×. In addition, to address the user-unfriendly concerns (Challenge D), we provide an automated parameter evaluation tool as well as some default recommended sets to lower the barrier to using the LHE mode.

*Integer Input LHE Solution.* In addressing Challenge E, our initial strategy involved applying our improved circuit bootstrapping algorithm into the BBB+ dramework. However, the HomTrace algorithm can not support integer inputs. To overcome this, we develop a high-precision HomTrace algorithm based on GLWE keyswitching. By combining it with a novel proposed multi-bit extraction method, we construct an improved integer input LHE solution. This approach is 10.7× faster than BBB+, and reduces the key size by up to 4.4×.

*Application I: AES Transciphering.* Our first application is the AES transciphering. We propose a light and flexible LHE mode tackling Challenge C, which is suitable for the AES S-Box evaluation. This mode is based on a newly designed HalfCBS algorithm which decouples the Refr. and Conv. steps. Then we evaluate the AES circuit by this mode within 11.53 seconds, which outperforms the state-of-the-art, Thunderbird [46], by 4.8× faster and 31.3× smaller key size.

*Application II: LUT and TFHE Processor.* Recently, Trama et al. [43] proposed a general-purpose TFHE processor abstraction supporting 8-bit instructions such as AND/OR/XOR, ADD/SUB, MUL/DIV, and MIN/MAX, based on an PBS-based LUT evaluation method. Under our improved integer input LHE mode, we can boost their general 8-to-8 LUT evaluation by a factor of 1.42×. Furthermore, by evaluating 16-to-8 LUT using our method, we can improve most of the heavy 8-bit bivariate instructions by up to 21× (homomorphic division operator), and the 16-bit sigmoid function by 26×.

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

## 2.2 TFHE

In this section, we briefly review the core concepts of the TFHE scheme. For the details of the operations in TFHE, we refer to Appendix C. 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 by $\mathcal{R}_N[X]$. We also write $\mathcal{R}_{q,N} = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(X^N + 1)$ 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 $\Delta$ such that $\Delta \le q/p$ as follows [7].

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

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 $\chi_\sigma$ 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. If $A_i = 0$ for $i = 1, 2, \ldots, k$ and $B = [\Delta \cdot M]_q$, it is called a trivial GLWE ciphertext. It does not protect the plaintext $M$, but encodes $M$ as a GLWE ciphertext. For simplicity, $B$ is sometimes denoted by $A_{k+1}$.

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 $\Delta$. The decryption works correctly if the error contained in the ciphertext is small enough to be eliminated during the rounding by $\Delta$.

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 gadget length $\ell$ and base $B$ under a GLWE secret key $\mathbf{S} \in \mathbb{B}_N[X]^k$ is defined as a vector of $\ell$ GLWE ciphertexts of $M \in \mathcal{R}_{q,N}$ as follows.

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

where $v_j = \lceil q/B^j \rceil$ for $j = 1, \ldots, \ell$. When $N = 1$, it is called a Lev ciphertext.

*GGSW Ciphertexts.* In the case of nonlinear operations such as multiplication, TFHE uses another type of ciphertext called generalized GSW (GGSW) [27]. Let $B \in \mathbb{N}$ be a power-of-two and $\ell \in \mathbb{N}$. A GGSW ciphertext $\overline{\overline{\mathrm{C}}} \in \mathcal{R}_{q,N}^{(k+1)\ell\times(k+1)}$ of a message $M \in \mathcal{R}_{q,N}$ with a gadget length $\ell$ and base $B$ 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.

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

where $v_j = \lceil q/B^j \rceil$ for $j = 1, \ldots, \ell$, $\mathbf{S} = (S_1, \ldots, S_k)$, $S_{k+1} = -1$. One can also represent $\overline{\overline{\mathrm{C}}}$ 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 length $\ell$ decomposes an input $a \in \mathbb{Z}_q$ into a vector $(a_1, \ldots, a_\ell) \in \mathbb{Z}_q^\ell$ such that

$$a = \sum_{j=1}^{\ell} a_j \cdot v_j + e$$

where $v_j = \lceil q/B^j \rceil$, $a_j \in [\![-B/2, B/2[\![$ for all $j = 1, \ldots, \ell$ and the decomposition error $e$ satisfies $|e| \le \lceil \frac{q}{2B^\ell} \rceil$. The gadget decomposition can be extended to a polynomial in $\mathcal{R}_{q,N}$ 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 $\overline{\overline{\mathrm{C}}}_1$ of $M_1$ and a GLWE ciphertext $\mathbf{C}_2$ of $M_2$ is defined as

$$\overline{\overline{\mathrm{C}}}_1 \boxdot \mathbf{C}_2 = \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A_{i,j} \cdot \mathrm{GLWE}(v_j \cdot (-S_i \cdot M_1))$$

where $\mathrm{GLWE}(v_j \cdot (-S_i \cdot M))$ is each GLWE component of $\overline{\overline{\mathrm{C}}}_1$ for $(i,j) \in [k+1] \times [\ell]$, $(A_{i,1}, \ldots, A_{i,\ell})$ is a gadget decomposition of $A_i$ for $i \in [k+1]$ such that $\mathbf{C}_2 = (A_1, \ldots, A_{k+1})$, and the multiplication between a polynomial and a GLWE ciphertext denotes multiplying the polynomial to each polynomial component of the GLWE ciphertext. Then the result becomes a GLWE ciphertext of $M_1 M_2$.

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

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

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

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

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

*GLWE Keyswitching and Functional Keyswitching.* It is possible to switch the key of GLWE ciphertexts. Such keyswitching operation is called GLWE keyswitching. The core concept of GLWE keyswitching is the same with that of LWE keyswitching, but its polynomial operation can be accelerated by NTT/FFT. The detailed algorithm is given in Algorithm 5 in Appendix C.2.

Functional keyswitching is usually used to denote LWE(s) to GLWE keyswitching with evaluating a function on the input LWE(s). When the evaluating function is public, it is called public functional keyswitching. On the other hand, if the evaluating function contains secret information, it is called private functional keyswitching. We refer to Appendix C.3 for the details.

## 2.3 PBS with Multiple Outputs

*2.3.1 PBSmanyLUT.* Chillotti et al. [15] proposed a generalized version of PBS, and extended it to PBSmanyLUT that evaluates several LUTs in a single blind rotation. PBSmanyLUT takes a non-negative integer $\vartheta$ in the plaintext modulus switching step, where $\vartheta$ denotes the number of empty bottom bits after the modulus switching.[3] At the cost of increased input error, it enables evaluating at most $2^\vartheta$ LUTs on the same input at the cost of a single blind rotation.

Suppose $2^\vartheta$ functions $f_j$, $j = 1, \ldots, 2^\vartheta$ are given. Thanks to the empty $\vartheta$ LSBs after the plaintext modulus switching, one can encode all the $2^\vartheta$ functions in a single polynomial $P_{(f_1,\ldots,f_{2^\vartheta})}$ defined as

[3]There is another parameter $\varkappa$ used in [15] to denote the number of skipped MSBs of the input, while we omit for simplicity as it is the same as multiplying $2^\varkappa$ to the input.

follows.

$$P_{(f_1,\ldots,f_{2^\vartheta})}(X) = X^{\frac{N}{2p}} \sum_{j=0}^{p-1} X^{j\frac{N}{p}} \sum_{k=0}^{\frac{N}{p2^\vartheta}-1} X^{k\cdot 2^\vartheta} \sum_{i=0}^{2^\vartheta-1} f_{i+1}(j) X^i$$

where $p = \frac{q}{2\Delta_{\mathsf{in}}}$ and $\Delta_{\mathsf{in}}$ is the scaling factor of the input LWE ciphertext. Let $(\beta, m') \in \mathbb{B}\times\mathbb{N}$ be the result of the plaintext modulus switching, denoted $(\beta, m') \leftarrow \mathsf{PTModSwitch}_q(m, \Delta, \vartheta)$, where $\beta$ is the MSB of the output and $m'$ is the remaining message part. By extracting the first $2^\vartheta$ coefficients after the blind rotation, one can obtain the function values of $(-1)^\beta \cdot f_j(m')$ for $j = 1, \ldots, 2^\vartheta$ unless the PBSmanyLUT operation fails. We present Algorithm 8 to describe the detailed procedure in Appendix I.

Lemma 2.1 (Theorem 4 in [15]). *Suppose that Algorithm 8 takes an input LWE ciphertext of noise variance $\sigma^2_{\mathsf{in}}$. Then it outputs LWE ciphertexts $\mathbf{c}_j$ encrypting $(-1)^\beta \cdot f_j(m') \cdot \Delta_{\mathsf{out}}$ for $i = 1, \ldots, 2^\vartheta$ if and only if*

$$\sigma^2_{\mathsf{in}} < \frac{\Delta^2_{\mathsf{in}}}{4\Gamma^2} - \frac{q^2}{12w^2} + \frac{1}{12} - \frac{nq^2}{24w^2} - \frac{n}{48}$$

*where $\Gamma$ is a variable depending on the probability of correctness defined as $P = \mathsf{erf}\left(\frac{\Gamma}{\sqrt{2}}\right)$ and $w = 2N \cdot 2^{-\vartheta}$. The output noise variance $V_{\mathsf{pbs}}$ (excluding the FFT error) is estimated as follows.*

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

*where $\sigma^2_{\mathsf{bsk}}$ is the noise variance of the bootstrapping key* BSK.

Unless otherwise stated, the failure probability denotes that of PBSmanyLUT derived from Lemma 2.1 throughout the paper.

*2.3.2 MV-PBS.* There is another method proposed by Carpov et al. [8] to evaluate multiple LUTs on a single input at the cost of a single blind rotation. Given a function $f_j$, let $P_j$ be the polynomial encoding $f_j$ for $j = 1, \ldots, t$. The main idea of MV-PBS is to decompose $P_j$ as $P_j = P'_j \cdot P_0$, sharing a common polynomial $P_0$ for all $j = 1, \ldots, t$. Then, after evaluating only a single polynomial $P_0$ by blind rotation, each $P_j$ can be computed by multiplying $P'_j$. We presented the detailed algorithm in Appendix I (see Algorithm 9).

Unlike PBSmanyLUT, MV-PBS increases the output error because the output of blind rotation is multiplied by $P'_j$. Hence, the output error variance of MV-PBS is obtained by multiplying $\ell_2(P_j)^2$ to $V_{\mathsf{pbs}}$ in Lemma 2.1.

## 2.4 Circuit Bootstrapping

The circuit bootstrapping (CBS) refreshes and converts an LWE ciphertext of a single bit into the corresponding GGSW ciphertext [14]. After obtaining refreshed GGSW ciphertexts by CBS, the LHE mode can evaluate circuits efficiently by CMux gates. In this paper, we describe CBS in two steps: the Refr. and Conv. steps.

The Refr. step refreshes the input LWE ciphertext by PBS and outputs the corresponding Lev ciphertext. Given an LWE ciphertext $\mathsf{LWE}_{\mathbf{s}}(\Delta m)$ of a single bit message $m$ with some scaling factor $\Delta$, one can compute $\mathsf{Lev}^{(B,\ell)}_{\mathbf{s}}(m)$ by gathering its internal LWE ciphertexts $\mathsf{LWE}_{\mathbf{s}}(v_j \cdot m)$ where $v_j = \lceil q/B^j \rceil$ for $j = 1, \ldots, \ell$ using PBS. Since it computes $\ell$ PBS operations on the same LWE input with several output scaling factors, PBSmanyLUT can improve this step without increasing the PBS error.

Next is the Conv. step converting the Lev ciphertext into the GGSW ciphertext. Chillotti et al. [13] used the private functional keyswitching to convert $\mathsf{LWE}(v_j \cdot m)$ to $\mathsf{GLWE}(v_j \cdot (-S_i \cdot m))$ when proposing the CBS algorithm, requiring $(k+1)\ell$ private functional keyswitching operations for each CBS operation.[4]

Since then, there have been many approaches to optimize the Conv. step. The first one is to precompute the multiplication result of the evaluation key to improve the speed of the private keyswitching itself at the cost of increased key size (see Appendix C.3 for the details). For example, the TFHEpp library [35] adopts this technique. Another one is using public functional keyswitching combined with scheme switching to replace private functional keyswitching [9, 46]. The MOSFHET library [28] adopts this technique. Since scheme switching is much faster than functional keyswitching, it can reduce the number of functional keyswitching from $\ell(k+1)$ to $\ell$. The last one is using homomorphic trace evaluation [45] to replace the public functional keyswitching part, improving the Conv. step asymptotically.

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

**Input:** $\mathbf{C} = \mathsf{GLWE}_{\mathbf{S}(X)}(M(X))$, $d \in \mathbb{Z}^\times_{2N}$
**Input:** $\mathsf{AutoKey}_d = \mathsf{KS}_{\mathbf{S}(X^d)\to\mathbf{S}(X)}$ with decomposition base $B$ and length $\ell$
**Output:** $\mathbf{C}' = \mathsf{GLWE}_{\mathbf{S}(X)}(M(X^d))$

$\mathbf{C} = (A_1, \ldots, A_k, B)$
$\mathbf{C}' \leftarrow (A'_1, \ldots, A'_k, B') = (A_1(X^d), \ldots, A_k(X^d), B(X^d))$
$\mathbf{C}' \leftarrow \mathsf{GLWE\_KS}(\mathbf{C}', \mathsf{AutoKey}_d)$
**return** $\mathbf{C}'$

## 2.5 Automorphism and Trace

The automorphism and trace can be defined on the polynomial ring $\mathcal{R}_N$ and its residue ring $\mathcal{R}_{q,N}$. For $d \in \mathbb{Z}^\times_{2N}$, the automorphism $\tau_d$ 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

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

The automorphism can be homomorphically evaluated by GLWE keyswitching as described in Algorithm 1. On top of it, Chen et al. [10] proposed an efficient algorithm to evaluate the trace function as described in Algorithm 2. We refer to Appendix C.4 for the detail.

# 3 Improved Circuit Bootstrapping Algorithms

WWL+ employed HomTrace to construct their faster and smaller circuit bootstrapping [45]. It is crucial to point that the trace evaluation will multiply the phase of the ciphertext by $N$, containing both

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

**Algorithm 2:** Evaluating Trace 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:** $\mathbf{C}' = \mathrm{GLWE}_{\mathbf{S}(X)}(N \cdot m_0)$

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

message and error, as Equation (1). WWL+ set the scaling factor to $N^{-1}\Delta \bmod q$ in their NTT setting to recover the scaling factor, but the error is multiplied by $N$ as Figure 1a. Without handling this, one has to use a larger parameter set to guarantee the algorithm correctness, degrading the overall performance. In this section, we first present a novel pre-processing method to patch WWL+, and further propose a FFT-based extension algorithm to achieve better performance.

## 3.1 Patched NTT-Based CBS

*Novel pre-processing method.* To prevent the error amplification, we first propose a novel pre-processing method. A possible solution in the NTT setting is to execute the efficient conversion algorithm from LWE to GLWE ciphertext proposed by Chen et al. [10] after PBSmanyLUT. The key point is to multiply $N^{-1}$ to both the message and error term, such that only homomorphic trace evaluation error is added after the conversion. We add this method in WWL+ workflow as Figure 1b, and simplify and improve this method as Figure 1c.

*Patched NTT-Based CBS.* Then 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:
- PBSmanyLUT without sample extraction

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

Our method can significantly reduce the noise growth from by removing an $N^2$ multiplicative factor in the error variance. We present Theorem 3.1 to analyze the noise growth in our patched algorithm, and also provide a detailed re-analysis of the WWL+ method in Appendix H.

Theorem 3.1. *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* $V_{\mathsf{cbs}}$ *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* $V_{\mathsf{pbs}}$ *denotes the* PBSmanyLUT *error variance,* $V_{\mathsf{tr}}$ *denotes the* HomTrace *error variance, and* $V_{\mathsf{ss}}$ *denotes the scheme switching error variance.*

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

$$N^{-1}v_j \cdot m + N^{-1}y_1 X + \ldots + N^{-1}y_{N-1}X^{N-1} + N^{-1}E_{\mathsf{pbs}}(X).$$

Then the trace evaluation change the phase to

$$v_j \cdot m + e_{\mathsf{pbs}} + E_{\mathsf{tr}}(X),$$

where $e_{\mathsf{pbs}}$ is the constant term of $E_{\mathsf{pbs}}$ 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 [1, 48]. This is due to the decreased parallelism in NTT and the complexity of the modular reduction algorithm. This key observation promote us to design an FFT-based CBS method to get better performance.

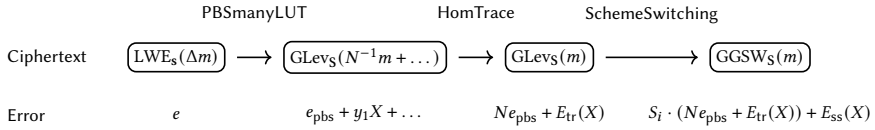
However, there are two technical subtleties that prevent the improved WWL+ method from being applied in the FFT domain: (a) an inverse of $N$ mod $q$ does not exist in FFT domains for the aforementioned pre-processing, and (b) the inherent errors in FFT (i.e., precision errors in FFT floating-point calculations) will be multiplied by $N$ during HomTrace, compromising the correctness of circuit bootstrapping. Our FFT-based CBS algorithm addresses these issues with the following techniques: (a) a new pre-processing method designed for FFT domains, and (b) a split FFT-based calculation method to handle FFT errors.

*New Pre-processing.* We used $N^{-1}$ mod $q$ to mitigate the multiplicative factor of $N$ during HomTrace in the patched NTT-based CBS, but the existence of $N^{-1}$ is guaranteed only when $(N, q)$ are coprime, which it is not the case in the FFT setting where both $N$ and $q$ are power-of-two. Inspired by this limited approach, we propose a new pre-processing method that divides both the scaling factor and the blind rotation error by $N$ using modulus switching and modulus raising, see Figure 2 for the pictorial description.

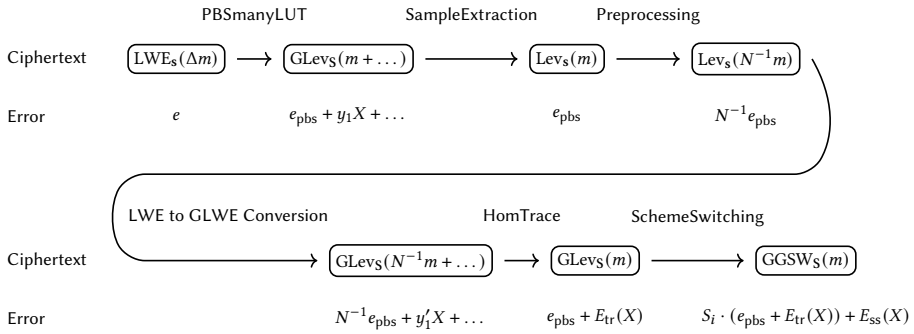
Specifically, let $\mathbf{C} = \mathrm{GLWE}_{q,\mathbf{S}}(v_j \cdot m)$ be a GLWE ciphertext of phase $\mu = \Delta m + e$ modulo $q$ under a GLWE secret key $\mathbf{S} \in \mathbb{B}_N[X]^k$, which corresponds to a GLWE component of the output GLev ciphertext of PBSmanyLUT (see Figure 1c), where $q$ and $N$ are both powers of two. We omit the redundant term $y_1 X + \cdots + y_{N-1}X^{N-1}$ for simplicity as it is vanished by HomTrace. The modulus switching of $\mathbf{C}$ from $q$ to $q/N$ divides its phase by $N$ at the cost of additional modulus switching error $E_{\mathsf{ms}}$, obtaining a GLWE ciphertext $\mathbf{C}' = \mathrm{GLWE}_{\frac{q}{N},\mathbf{S}}(\frac{v_j}{N}m)$ of phase $\mu' = \frac{1}{N}\mu + E_{\mathsf{ms}} = \frac{v_j}{N}m + \frac{1}{N}e + E_{\mathsf{ms}}$ modulo $\frac{q}{N}$. 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,

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

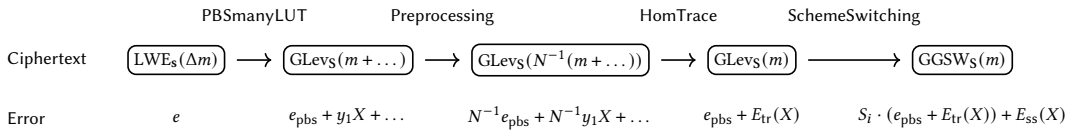
for some $U \in \mathbb{Z}[X]/(X^N + 1)$ since the phase of $\mathbf{C}'$ is $\mu'$ modulo $\frac{q}{N}$ under the secret key $\mathbf{S}$. The modulus raising interprets each coefficient of $\mathbf{C}'$ in $\mathbb{Z}_{q/N}$ as an element of $\mathbb{Z}_q$ of the same value,




**(a) WWL+ method**




**(b) A patched method using Chen et al.'s [10] preprocessing**




**(c) Our improved method**

**Figure 1: The Circuit Bootstrapping Workflow. "$m+\dots$" denote the plaintext polynomial with some redundant power of $X$ terms.**

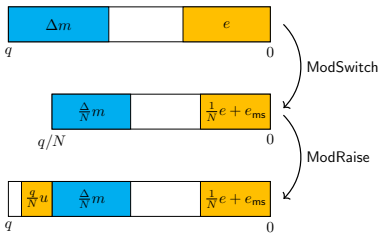



**Figure 2: New pre-processing method on an FFT domain.**

obtaining a GLWE ciphertext $\mathbf{C}'' = (A_1, \dots, A_k, B) \in \mathcal{R}_q^{k+1}$ 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 $\mu$ is changed to the GLWE ciphertext of phase $\mu'' = \frac{1}{N}\mu + E_{\mathrm{ms}} + \frac{q}{N} \cdot U$ under the same GLWE secret key and modulus $q$. Since both the input message and error are divided by $N$, the multiplication by $N$ during HomTrace can be canceled out. The term $\frac{q}{N} \cdot U$ will also vanish through trace evaluation, which multiplies it by $N$ modulo $q$. Additionally, only the constant $e_{\mathrm{ms}}$ of $E_{\mathrm{ms}}$ remains after HomTrace.

This pre-processing takes negligible time compared to HomTrace. Its additional error increment by $Ne_{\mathrm{ms}}$, whose variance is $N^2 \cdot \frac{kN+1}{12}$, is much smaller than the blind rotation error and HomTrace error. For more details, we refer to Appendix C.1.

By adopting this new pre-processing method in the FFT setting, HomTrace after blind rotation outputs a GLWE ciphertext of a phase

$$N\mu'' + E_{\mathrm{tr}} = \mu + Ne_{\mathrm{ms}} + E_{\mathrm{tr}} \tag{3}$$

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

*Split FFT.* Most FFT-based implementations of TFHE, especially for PBS, do not deal with the errors generated from FFT floating-point calculation. But for HomTrace, it is hard to ignore the FFT error because it is multiplied by $N$ during HomTrace, becoming larger than the HomTrace error itself. One naive way to reduce the FFT error is to decrease the gadget basis, but it is not enough. Increasing the precision of floating-point arithmetic for FFT can be another solution, but we do not consider this because double-precision FFT is most commonly used in TFHE.

To reduce FFT errors further, we split a polynomial of 64-bit precision into two parts: 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:

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

We call this strategy 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 $\chi$ be the bit-precision of the floating point representation, which is 53 for double-precision. Bergerat et al. [5] proposed an estimation for the variance of FFT error from the external product, so we used a slightly modified formula for GLWE keyswitching as follows.[5]

$$2^{-2\chi-2.6} \cdot \ell q^2 B^2 N^2 k$$

where $\ell$ and $B$ are gadget length and base of the GLWE keyswitching, respectively. There is also an FFT error estimate proposed by Klemsa et al. [31], we analyze the difference in Appendix B. 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

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

This is consistent with the experimental results. 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 $2^{-256}$, enabling one to ignore the failure probability of the split FFT compared to that of PBS (see Appendix B for the detailed analysis).[6] On the other hand, there might be an FFT error for the lower part that should be added to the final error, but it is reduced significantly by the split FFT.

We then present Theorem 3.2 to analyze the noise growth in our FFT-based circuit bootstrapping algorithm.

Theorem 3.2. *Let* $\mathbf{c}$ *be an LWE ciphertext of phase* $\mu$ *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* $V_{\mathsf{cbs}}$ *of* $E_{\mathsf{cbs}}(X)$ *is given as follows.*

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

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

Proof Sketch. The proof is analogous to that of Theorem 3.1 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

In this section, we provide the performance of our improved CBS algorithms. The benchmark environment is i9-11900K @ 3.50 GHz with 32 GB RAM supproting AVX-512 optimization.

*Implementations.* For our patched NTT-based CBS[7], we employ the same level of security parameters from the WWL+ method for a fair comparison. In addition, we provide multiple parameter sets tailored to support different levels of circuit depth. The parameters were selected using "Lattice Estimator" [8] to ensure 128-bit security under standard security assumptions. Table 1 summarizes the recommended parameter sets for our patched NTT-based CBS. The ciphertext modulus for the blind rotation is a prime of 54 bits (see Appendix A.2 for the detail). $\ell_{\mathsf{op}}$ and $B_{\mathsf{op}}$ for $\mathsf{op} \in \{\mathsf{pbs}, \mathsf{tr}, \mathsf{ss}, \mathsf{ks}\}$ denote the gadget length and base of the corresponding operations. $\ell$ and $B$ are the gadget length and base of circuit evaluation. We also provide reanalyzed parameters for the WWL+ method in Appendix H.

| Sets | $n$ | $N$ | $k$ | $\ell_{\mathsf{pbs}}$ | $B_{\mathsf{pbs}}$ | $\ell_{\mathsf{tr}}$ | $B_{\mathsf{tr}}$ | $\ell_{\mathsf{ss}}$ | $B_{\mathsf{ss}}$ | $\ell$ | $B$ | $\ell_{\mathsf{ks}}$ | $B_{\mathsf{ks}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NTT-CMux$_1$ | 571 | 2048 | 1 | 1 | $2^{26}$ | 3 | $2^{13}$ | 1 | $2^{28}$ | 4 | $2^3$ | 10 | $2^3$ |
| NTT-CMux$_2$ | 571 | 2048 | 1 | 2 | $2^{17}$ | 3 | $2^{13}$ | 2 | $2^{19}$ | 4 | $2^4$ | 10 | $2^3$ |
| NTT-CMux$_3$ | 571 | 2048 | 1 | 2 | $2^{17}$ | 6 | $2^8$ | 2 | $2^{19}$ | 4 | $2^5$ | 10 | $2^3$ |

**Table 1: Recommended parameters for our NTT-based CBS.**

For our FFT-based CBS[9], we increase the LWE dimension $n$ from 571 to 636 to mitigate larger errors on FFT. The recommended parameter sets for our FFT-based circuit bootstrapping are summarized in Table 2. The additional column of $b_{\mathsf{tr}}$ denotes the split FFT base to reduce the FFT error of HomTrace. The ciphertext modulus is $2^{64}$. We also provide a parameter analysis tool for the FFT-based LHE mode that considers the FFT error.[10]

| Sets | $n$ | $N$ | $k$ | $\ell_{\mathsf{pbs}}$ | $B_{\mathsf{pbs}}$ | $\ell_{\mathsf{tr}}$ | $B_{\mathsf{tr}}$ | $b_{\mathsf{tr}}$ | $\ell_{\mathsf{ss}}$ | $B_{\mathsf{ss}}$ | $\ell$ | $B$ | $\ell_{\mathsf{ks}}$ | $B_{\mathsf{ks}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFT-CMux$_1$ | 636 | 2048 | 1 | 1 | $2^{23}$ | 5 | $2^8$ | - | 1 | $2^{25}$ | 4 | $2^3$ | 5 | $2^2$ |
| FFT-CMux$_2$ | 636 | 2048 | 1 | 2 | $2^{15}$ | 6 | $2^7$ | - | 2 | $2^{17}$ | 4 | $2^4$ | 5 | $2^2$ |
| FFT-CMux$_3$ | 636 | 2048 | 1 | 2 | $2^{15}$ | 6 | $2^7$ | 35 | 2 | $2^{17}$ | 4 | $2^4$ | 5 | $2^2$ |

**Table 2: Recommended parameters for our FFT-based CBS. "-" indicates that split FFT is not used.**

*Performance.* We implement our FFT-based CBS (resp. NTT-based CBS) using TFHE-rs [48] (resp. OpenFHE [1]). Table 3 shows the performance of our CBS algorithms and compares them to previous methods. WWL+ denotes NTT-based CBS by the WWL+ method with revised parameters. TFHEpp [35], MOSFEHT [28] and TFHE-rs [48] denote the FFT-based CBS supported by the corresponding libraries. Our patched NTT-based CBS decreases the latency (resp. the keysize) by factors up to 2.60 (resp. 3.43) compared to the revised WWL+. In the case of our FFT-based CBS, it enjoys 3.42× faster running time and 33.2× smaller key size compared to TFHEpp.

Remark 1. *We implemented all NTT-based circuit bootstrapping in the open-source OpenFHE library. Additionally, WWL+ utilized an AVX-512 based NTT library for further optimization. We conducted estimates by testing the efficiency of the underlying NTT operations within the same library, with the results shown in parentheses.*

[5] In [5], $k+1$ is used instead of $k$ since there is FFT-based multiplication for the body part in the external product, which is not in the keyswitching.

[6] We set the failure probabilty of the split FFT to be much smaller than that of PBS, which is set to $2^{-40}$ in this paper, because relaxing the failure probability of the split FFT has no impact on time complexity.

[7] https://github.com/LightFHE/CircuitBootstrap

[8] https://github.com/malb/lattice-estimator

[9] https://github.com/KAIST-CryptLab/refined-tfhe-lhe. It also includes the implementation for Section 4 and 5.

[10] https://github.com/KAIST-CryptLab/refined-tfhe-lhe/tree/main/error_analysis

| Methods | Refr. | Conv. | Max Depth | Time (ms) | | Keysize (MB) |
|---|---|---|---|---|---|---|
| WWL+ ($\mathrm{CMux}_{O1}$) | PBSmanyLUT | HomTrace + SS | 2 | 95.92 | (39.00) | 30.56 |
| Ours (NTT-$\mathrm{CMux}_1$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 30 | 71.02 | (26.66) | 15.50 |
| Ours (FFT-$\mathrm{CMux}_1$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 8 | 14.15 | | 20.84 |
| TFHEpp | PBSmanyLUT | $\mathrm{PrivateKS}_{\mathrm{pre}}$ | 61 | 63.46 | | 1359.43 |
| MOSFHET | PBSmanyLUT | $\mathrm{PublicKS}_{\mathrm{pre}}$ + SS | 1788 | 101.32 | | 2127.31 |
| WWL+ ($\mathrm{CMux}_{O2}$) | PBSmanyLUT | HomTrace + SS | 101 | 119.09 | (52.29) | 45.91 |
| Ours (NTT-$\mathrm{CMux}_2$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 413 | 95.03 | (39.05) | 30.57 |
| Ours (FFT-$\mathrm{CMux}_2$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 2102 | 18.57 | | 40.92 |
| TFHE-rs | PBS | PrivateKS | 5237 | 80.62 | | 168.81 |
| WWL+ ($\mathrm{CMux}_{O3}$) | PBSmanyLUT | HomTrace + SS | 53378 | 250.41 | (102.64) | 106.43 |
| Ours (NTT-$\mathrm{CMux}_3$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 57335 | 96.25 | (40.47) | 31.01 |
| Ours (FFT-$\mathrm{CMux}_3$) | PBSmanyLUT | Preprocessing + HomTrace + SS | 21296 | 20.62 | | 40.92 |

**Table 3: Comparison of CBS performance.**

Remark 2. *We contained the LWE keyswitching error to compute the max depth for accuracy, which was not considered in [45]. Also, we use a smaller failure probability of $2^{-40}$ instead of $2^{-32}$.*

Remark 3. *The key size reduction in our scheme arises from two factors. First, compared to WWL+, reducing noise amplification allows for smaller parameters, leading to a more compact key. Second, compared to TFHEpp and MOSFHET, we replace the key-switching step by using* HomTrace*, which reduces the key from $2N$ to $\log N + 1$ GLWE ciphertexts. Throughout this paper, the reported key size refers to the compressed form, where the mask of each fresh ciphertext is replaced by a seed to generate it. We refer Appendix C.6 for details.*

## 4 Application I: AES Transciphering

Since 2012, homomorphic evaluation of AES has been regarded as an important benchmark for testing FHE [11, 19, 24, 26]. In recent years, it has been further promoted as an application for transciphering, which combines a symmetric cipher with a homomorphic encryption scheme. 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 [39]. Lots of works evaluating AES by TFHE have been proposed [6, 42, 45–47] to achieve low latency. Among them, the fastest method to date (in a single thread) is based on the LHE mode [45, 46].

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. This enables us to enhance AES transciphering through our improved circuit bootstrapping algorithms. Additionally, we propose the following 2 techniques to further improve AES transciphering performance: (a) a flexible LHE mode and (b) a modified AES evaluation workflow.

Remark 4. *There have been many works for transciphering with other symmetric ciphers or other HE schemes [2, 3, 16–18, 20, 22, 23, 29, 30, 37, 38], but we only deal with AES transciphering on TFHE in this section.*

### 4.1 A Flexible LHE Mode

The aforementioned LHE mode is inflexible since the Conv. step is always executed together 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 (e.g. AES S-box) to convert ciphertext types results in wasted depth. We then propose a HalfCBS algorithm, which takes as input $\overline{\overline{\mathrm{C}}} = \mathrm{GGSW}_{\mathrm{S}}(m)$ and outputs $\overline{\overline{\mathrm{C}}}' = \mathrm{GGSW}_{\mathrm{S}}(L[m])$ where $L$ denotes the circuit to be evaluated, in order to achieve circuit composability without refreshing noise as illustrated in Figure 3. Compared to our proposed CBS algorithms in Section 3.1 and Section 3.2, the HalfCBS algorithm (Algorithm 3) does not have a Refr. step.

Specifically, instead of using PBSmanyLUT, HalfCBS employs $\ell$ look-up table circuits $\mathrm{Circuit}_{m \to v_j \cdot L[m]}$, each implemented as a CMux-based binary tree, where $\ell$ is the GGSW gadget length (line 2). In each circuit, $v_j \cdot L[m]$ serves as the test polynomial, and $\mathrm{GGSW}(m)$ as the control bit. After removing the redundant terms appearing on non-constant terms by the preprocessing and HomTrace (line 3-4), the resulting GLWE ciphertexts are gathered into a corresponding GLev ciphertext (line 5), and converted into the desired GGSW ciphertext by scheme switching (line 6).

As HalfCBS does not contain the heavy Refr. step, its computational complexity is lower than CBS. In asymptotic perspective, the complexity of HalfCBS is $O(kN \log^2 N)$ and that of CBS is $O((k+1)nN \log N)$. We remark that the parameters $k$ and $N$ might differ for HalfCBS and CBS. That said, because $n$ is much larger than $\log N$, HalfCBS can make circuit evaluation faster at the cost of increased noise growth. We call an LHE mode using HalfCBS by a flexible LHE mode.

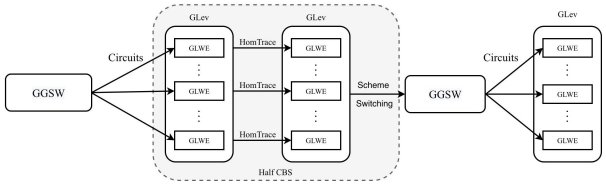



**Figure 3: A Flexible Leveled Homomorphic Evaluation (LHE) Mode.**

**Algorithm 3:** HalfCBS

**Input:** $\overline{\overline{\mathrm{C}}} = \mathrm{GGSW}_{\mathbf{S}}(m)$
**Input:** $\ell$ look up table circuits $\mathrm{Circuit}_{m \to v_j \cdot L[m]}$, $j \in \{1, \ldots, \ell\}$
**Input:** Automorphism keys under $\mathbf{S}$
**Input:** Scheme switching key under $\mathbf{S}$
**Output:** $\overline{\overline{\mathrm{C}}}' = \mathrm{GGSW}_{\mathbf{S}}(L[m])$

1 **for** $j = 1$ *to* $\ell$ **do**
2 $\quad \mathrm{C}'_j \leftarrow \mathrm{Circuit}_{m \to v_j \cdot L[m]}(\overline{\overline{\mathrm{C}}})$ /* $\mathrm{C}'_j = \mathrm{GLWE}_{\mathbf{S}}(v_j \cdot L[m] + \ldots)$ */
3 $\quad \mathrm{C}'_j \leftarrow \mathrm{Preprocess}(\mathrm{C}'_j)$
4 $\quad \mathrm{C}'_j \leftarrow \mathrm{HomTrace}(\mathrm{C}'_j)$ /* $\mathrm{C}'_j = \mathrm{GLWE}_{\mathbf{S}}(v_j \cdot L[m])$ */
5 $\overline{\mathrm{C}}' \leftarrow \{\mathrm{C}'_j\}_{j \in \{1,\ldots,\ell\}}$ /* $\overline{\mathrm{C}}' = \mathrm{GLev}_{\mathbf{S}}(L[m])$ */
6 $\overline{\overline{\mathrm{C}}}' \leftarrow \mathrm{SchemeSwitch}(\overline{\mathrm{C}}')$ /* $\overline{\overline{\mathrm{C}}}' = \mathrm{GGSW}_{\mathbf{S}}(L[m])$ */
7 **return** $\overline{\overline{\mathrm{C}}}'$

Theorem 4.1. *Let $\overline{\overline{\mathrm{C}}}$ be a GGSW ciphertext of $m$ under a secret key $\mathbf{S}$. Then, Algorithm 3 returns a GGSW ciphertext $\overline{\overline{\mathrm{C}}}'$ of $L[m]$ with an error variance $V_{\text{half-cbs}}$ such that*

$$V_{\text{half-cbs}} \leq V_{\text{circuit}} + V_{\text{pre}} + \frac{N}{2} V_{\text{tr}} + V_{\text{ss}}$$

*where $V_{\text{circuit}}$ denotes the output error variance from the circuit $L$ evaluation on the input $\overline{\overline{\mathrm{C}}}$, $V_{\text{pre}}$ denotes the pre-processing error variance such that $V_{\text{pre}} = 0$ for the NTT-based one and $V_{\text{pre}} = N^2 V_{\text{ms}}$ for the FFT-based one, $V_{\text{tr}}$ denotes the trace evaluation error variance, and $V_{\text{ss}}$ denotes the scheme switching error variance. $V_{\text{circuit}}$, $V_{\text{tr}}$, and $V_{\text{ss}}$ should contain their FFT error variances in the FFT setting.*

Proof. $V_{\text{circuit}}$ denotes the error variance of $\mathrm{C}'_j$ in line 2. Then the above inequality can be obtained from the proof of Theorem 3.1 and Theorem 3.2 by replacing $V_{\text{pbs}}$ with $V_{\text{circuit}}$. □

Remark 5. *We consider a single input circuit $L$ in Algorithm 3 for the consistency with CBS, while it can be generalized to circuits having arbitrarily number of inputs. For example, if $L$ can be evaluated by CMux gates of depth $d$, then*

$$V_{\text{circuit}} \leq d\left((1+kN)\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right) + (k+1)\ell N\left(\frac{B^2+2}{12}\right)\sigma_{\text{in}}^2\right)$$

*It can be derived from Lemma C.4 where $(B, \ell)$ is the gadget decomposition parameter for the input GGSW ciphertexts and $\sigma_{\text{in}}^2$ is the input error variance.*

| Sets | $n$ | $N$ | $k$ | $\ell_{\text{ks}}$ | $B_{\text{ks}}$ | $\ell_{\text{pbs}}$ | $B_{\text{pbs}}$ | $\vartheta$ | $\ell_{\text{tr}}$ | $B_{\text{tr}}$ | $b_{\text{tr}}$ | $\ell_{\text{ss}}$ | $B_{\text{ss}}$ | $\ell$ | $B$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HalfCBS | 768 | 1024 | 3 | 3 | $2^4$ | - | - | - | 3 | $2^{15}$ | 42 | 3 | $2^{13}$ | 3 | $2^7$ |
| Set-I | 768 | 1024 | 2 | 3 | $2^4$ | 1 | $2^{23}$ | 3 | 3 | $2^{12}$ | - | 2 | $2^{17}$ | 6 | $2^2$ |
| Set-II | 768 | 1024 | 2 | 3 | $2^4$ | 2 | $2^{15}$ | 2 | 6 | $2^7$ | 34 | 2 | $2^{17}$ | 4 | $2^4$ |

**Table 4: Parameters for the AES evaluation.**

*Hybrid use of HalfCBS and CBS algorithms.* Theorem 4.1 demonstrates that the HalfCBS algorithm does not refresh the ciphertext noise. Hence, we can use both HalfCBS and CBS in a flexible LHE mode to manage the noise. 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 4.2.

## 4.2 AES Transciphering

*A modified AES evaluation workflow.* First, we embed the AES round key into the S-box to generate an encrypted keyed S-box. Specifically, by sending GLWE ciphertexts that encrypt the tables of $S_k$ such that $S_k(x) = S(x \oplus rk)$ where $S$ is the AES S-box and $rk$ is the round key, we can eliminate the homomorphic addition and error accumulation in AddRoundKey at the cost of increased key size. Furthermore, we combine SubBytes, MixColumns, and ShiftRows together into four 8-24 LUTs following Wei et al.'s method [46], reducing the number of homomorphic additions. For a detailed step-by-step evaluation, we refer to Appendix D.1.

*Implementations.* We then implement the AES transciphering based on our flexible LHE mode using our proposed FFT-based CBS and HalfCBS. For the implementations, we use the parameter sets as listed in Table 4. The standard deviation of the fresh error is chosen to guarantee 128-bit security (see Appendix A.1). HalfCBS denotes the parameters for the HalfCBS-based round, and Set-I/II denotes those for the CBS-based round.

For instance, the latency of the HalfCBS round is 423.53 ms, which is **3.27**/**5.19** times faster than the CBS round (1387.49/2198.33 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 bootstrapping failure probability. In our design, the HalfCBS algorithm is applied only in the first round, and the standard CBS algorithm is used for the remaining rounds. This strategy improves efficiency while keeping the decryption failure probability negligible. In this strategy, the Flexible LHE mode offers a 1.23× performance improvement over the traditional LHE mode in AES evaluation.

*Performance.* Table 5 shows the benchmark result and comparison with previous works.

The overall benchmark environment is the same as that of Section 3.3. Compared to the fastest previous result [46], our result based on the flexible LHE mode with Set-I is 4.78× faster with 31.3× smaller keysize. Our results with Set-II provides AES evaluation with very low failure probability, while they are still faster than the previous results.

| Evaluation Modes | Library | Failure Prob. | Time (s) | Key Size (MB) |
|---|---|---|---|---|
| FHE [42][11] (∗) | TFHE-rs | $2^{-23}$ | 249.2 | 160.00 |
| FHE [6] (∗) | TFHE-rs | $2^{-40}$ | 95.9 | 32.88 |
| LHE [47] | TFHEpp | - | 79.3 | 1359.43 |
| LHE [45][12] | OpenFHE | $2^{-32}$ | 68.3 | 46.64 |
| LHE [46] | TFHEpp | - | 55.0 | 734.40 |
| LHE [Set-I] | TFHE-rs | $2^{-34.86}$ | 12.7 | 19.36 |
| Flex. LHE [Set-I] | TFHE-rs | $2^{-34.86}$ | 11.5 | 23.46 |
| LHE [Set-II] | TFHE-rs | $2^{-368.33}$ | 19.4 | 37.83 |
| Flex. LHE [Set-II] | TFHE-rs | $2^{-129.75}$ | 15.8 | 41.93 |

**Table 5: Our AES evaluation performance compared with the state of the art. The modes indicated by (∗) are estimated from their building blocks.**

## 5 Application II: LUT and TFHE Processor

In Section 4, we demonstrate the advantage of LHE mode when processing bit-wise input. However, the bitwise input LHE cannot handle the case of using an integer input for each LWE ciphertext. In this section, we first briefly summarize previous methods to evaluate LUT with integer inputs, and propose our improved method.

### 5.1 Previous Methods for LUT

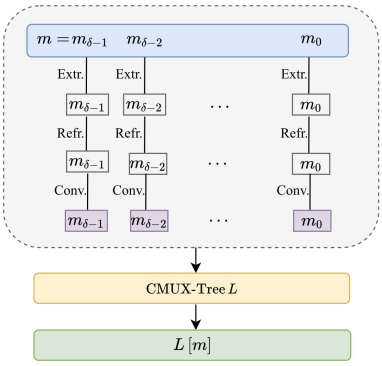
There are LUT evaluation methods for both FHE and LHE modes. For the FHE mode, a method named tree-PBS was proposed by Carpov et al. [8]. For simplicity, we describe a tree-PBS for the 4-bit identity function illustrated in Figure 5 as follows; Initially, the input message, ranging from $[0, ..., 4^2 - 1]$, is encoded into two 2-bit (basis 4) message chunks, each encrypted in an LWE ciphertext $c_{i\in[0,1]}$. Then the LUT is decomposed into four test polynomials, each with four LUT values, accordingly. In the first layer of the PBS-tree, it performs four PBS operations using ciphertext $c_0$ to rotate the test polynomials and extract four LWE results. Subsequently, one can use functional key switching to repack them into a new GLWE ciphertext encrypting the layer-2 test polynomial. Finally, it perform a PBS with $c_1$ to output the final result.

Generally, for a $B^d$-$B$ size integer LUT, the tree-PBS performs $\sum_{i=0}^{d-1} B^i$ PBS operations and $\sum_{i=0}^{d-2} B^i$ key switching. Since the PBS in the first layer can be merged into one MV-PBS, then the number of blind rotations can be reduced to $1 + \sum_{i=0}^{d-2} B^i$.

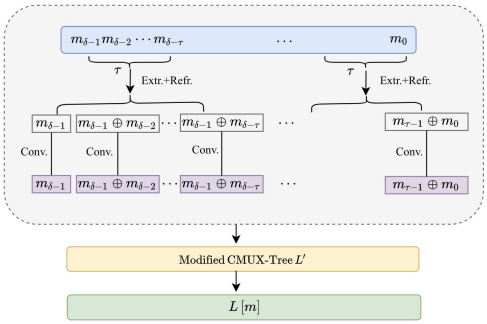
The tree-PBS method leads to a challenge for the LHE mode to deal with integer inputs instead of bits. To respond to this issue, Bergerat et al. [4] proposed the BBB+ method as a new WoP-PBS algorithm, as shown in Figure 4a. This algorithm extracts each bit from the ciphertext chunks, and converts them into GGSWs to perform CMux evaluation.

[11]Trama et al. used TFHElib to implement their result, while the building blocks for their method are not publicly open. We estimate their result using TFHE-rs as it contains all the building blocks.

[12]Wang et al. evaluate AES transciphering with their WWL+ circuit bootstrapping method [45], achieved a latency of 26 seconds on their machine. However, the parameters used in the paper incur a large decryption failure probability. Therefore, we reanalyzed the WWL+ method and provided updated parameter sets in Appendix H, so we used the performance under these parameters as a benchmark.




**(a) The BBB+ Method [4]**




**(b) Our HP-LHE Method**

**Figure 4: The High-Precision Solutions in the LHE Mode.**

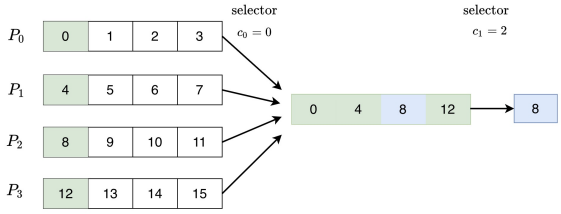



**Figure 5: Diagram for the tree-PBS with a basis 4**

### 5.2 New Integer Input LHE Mode

In this section, we propose a new integer input LHE mode as shown in Figure 4b and Algorithm 4.

The overall idea is to improve the BBB+ method as follows:

- Integrate the Extr. and Refr. steps by single PBSmanyLUT.
- Propose high-precision HomTrace to improve the Conv. step.
- Propose multi-bit extraction to enhance the Extr. step.

Integrating the Extr. and Refr. steps and proposing multi-bit extraction reduce the number of required PBSmanyLUT operations, improving the performance significantly. Adopting HomTrace-based Conv. step is also important as in the bitwise input LHE mode. Because the integer input LHE mode requires a higher precision

**Algorithm 4:** CBS of Integer Input LHE Mode

**Input:** $\mathbf{c} = \mathrm{LWE}_{\mathbf{s}}(\frac{q}{2^\delta} \cdot m)$ where $\tau \mid \delta$

**Input:** $\ell$ test polynomials $F_j$ to compute $\frac{v_j}{2} \cdot (-1)^{x_{\tau-1}+1}$ for $j = 1, \ldots, \ell$ from a $\tau$-bit input $x = \sum_{i=0}^{\tau-1} x_i \cdot 2^i$

**Input:** $\tau - 1$ polynomials $G_k$ such that $G_k \cdot F_j$ becomes a test polynomial to compute $\frac{v_j}{2} \cdot (-1)^{(x_{\tau-1} \oplus x_k)+1}$ from a $\tau$-bit input $x = \sum_{i=0}^{\tau-1} x_i \cdot 2^i$ for $k = 0, \ldots, \tau - 2$

**Input:** Bootstrapping keys under $\mathbf{S}$

**Input:** Automorphism keys under $\mathbf{S}$ for Conv (or keys for HP-Conv if necessary)

**Input:** Scheme switching key under $\mathbf{S}$

**Output:** $\overline{\overline{\mathbf{C}}}_i = \mathrm{GGSW}(m'_i)$ for $i = 0, \ldots, \delta - 1$ where $m'_i = m_i$ if $\tau \mid (i+1)$ and $m'_i = m_{\lceil (i+1)/\tau \rceil \cdot \tau} \oplus m_i$ otherwise.

1 **for** $i = 0$ *to* $\delta/\tau - 1$ **do**

2 $\quad \mathbf{c}' \leftarrow 2^{\delta-(i+1)*\tau} \cdot \mathbf{c}$ /* $\mathbf{c}' = \mathrm{LWE}(\frac{q}{2^\tau} \cdot \sum_{k=0}^{\tau-1} m_{i\cdot\tau+k} \cdot 2^k)$ */

3 $\quad \overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)} \leftarrow \mathsf{PBSmanyLUT}\left(\mathbf{c}'; (F_j)_{j=1}^{\ell}\right)$ (except sample extraction) /* $\overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)}[j] = \mathrm{GLWE}(\frac{v_j}{2} \cdot (-1)^{m_{i\cdot\tau+(\tau-1)}+1} + \ldots)$ */

4 $\quad$ **for** $j = 1$ *to* $\ell$ **do**

5 $\quad\quad$ **for** $k = 0$ *to* $\tau - 2$ **do**

6 $\quad\quad\quad \overline{\mathbf{C}}_{i\cdot\tau+k}[j] \leftarrow G_k \cdot \overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)}$ /* $\overline{\mathbf{C}}_{i\cdot\tau+k}[j] = \mathrm{GLWE}(\frac{v_j}{2} \cdot (-1)^{(m_{i\cdot\tau+(\tau-1)} \oplus m_{i\cdot\tau+k})+1} + \ldots)$ */

7 $\quad\quad\quad \overline{\mathbf{C}}_{i\cdot\tau+k}[j] \leftarrow \overline{\mathbf{C}}_{i\cdot\tau+k}[j] + \mathrm{GLWE}^0(v_j/2)$ /* $\overline{\mathbf{C}}_{i\cdot\tau+k}[j] = \mathrm{GLWE}(v_j \cdot (m_{i\cdot\tau+(\tau-1)} \oplus m_{i\cdot\tau+k}) + \ldots)$ */

8 $\quad\quad \overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)}[j] \leftarrow \overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)} + \mathrm{GLWE}^0(v_j/2)$ /* $\overline{\mathbf{C}}_{i\cdot\tau+(\tau-1)}[j] = \mathrm{GLWE}(v_j \cdot m_{i\cdot\tau+(\tau-1)} + \ldots)$ */

9 $\quad$ **for** $k = 0$ *to* $\tau - 1$ **do**

10 $\quad\quad \overline{\overline{\mathbf{C}}}_{i\cdot\tau+k} \leftarrow \mathsf{Conv}(\overline{\mathbf{C}}_{i\cdot\tau+k})$ (or $\mathsf{HP\text{-}Conv}(\overline{\mathbf{C}}_{i\cdot\tau+k})$) /* $\overline{\overline{\mathbf{C}}}_{i\cdot\tau+k} = \mathrm{GGSW}(m'_{i\cdot\tau+k})$ */

11 $\quad$ **if** $i < \delta/\tau - 1$ **then**

12 $\quad\quad \mathbf{C}'' \leftarrow \mathrm{Circuit}_{(m'_{i\cdot\tau+(\tau-1)}, \ldots, m'_{i\cdot\tau}) \to \frac{q}{2^{\delta-i\cdot\tau}} \cdot L[\sum_{k=0}^{\tau-1} m_{i\cdot\tau+k} \cdot 2^k]}\left((\overline{\overline{\mathbf{C}}}_{i\cdot\tau+k})_{k=0}^{\tau-1}\right)$

13 $\quad\quad \mathbf{c}'' \leftarrow \mathsf{SampleExtract}(\mathbf{C}'')$ /* $\mathbf{c}'' = \mathrm{LWE}(\frac{q}{2^{\delta-i\cdot\tau}} \sum_{k=0}^{\tau-1} m_{i\cdot\tau+k} \cdot 2^k)$ */

14 $\quad\quad \mathbf{c} \leftarrow \mathbf{c} - \mathbf{c}''$ /* $\mathbf{c} = \mathrm{LWE}(\frac{q}{2^{\delta-(i+1)\tau}} \sum_{k=0}^{\delta-(i+1)\tau} m_{(i+1)\tau+k} \cdot 2^k)$ */

15 **return** $(\overline{\overline{\mathbf{C}}}_j)_{j=0}^{\delta-1}$

for this step, we proposed a method to improve the precision of HomTrace at the cost of additional homomorphic evaluation.

We introduce each technique as follow, and give some more detailed explanations and its error analysis in Appendix E.

*Integrate Extr. with Refr.* Suppose that an LWE ciphertext of a $\delta$-bit message $m$ scaled by $\Delta = q/2^\delta$ 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 BBB+ method, each bit is first extracted into a new LWE ciphertext by PBS (Extr.), and the resulting LWE ciphertext is taken into another PBS in the Refr. step of CBS. It even uses $\ell$ PBS for the Refr. step where $\ell$ is the gadget length of the CBS output.

However, we find that both Extr. and Refr. steps take $\mathrm{LWE}(\frac{q}{2} \cdot m_j)$ as an input to PBS. To integrate both steps and remove unnecessary PBS, we use PBSmanyLUT to refresh it, obtain $\mathrm{GGSW}(m_j)$ from the followed Conv. step, and recover $\mathrm{LWE}(\frac{q}{2^{\delta-j}} \cdot m_j)$ from $\mathrm{GGSW}(m_j)$ to repeat the bit extraction.

*Improved Conv. with HP-HomTrace.* HomTrace and scheme switching improves the Conv. step of CBS in terms of both time complexity and keysize compared to private functional keyswitching. However, in terms of error growth, this method cannot support higher precision due to the multiplicative factor of about $N^{3/2}$, which is not in private functional keyswitching. Although the error induced by HomTrace was small enough to be used in the bitwise input LHE mode, we need a 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. Then, after the pre-processing on the switched GLWE ciphertext, we evaluate the trace on the large dimension with high-precision. Finally, we switch it back into the original GLWE dimension $k$. By evaluating HomTrace on a larger GLWE dimension, we can reduce HomTrace error significantly. Although two additional GLWE keyswitching operations to switch the GLWE dimension induce additiona error, it is much smaller than the HomTrace error in the original GLWE dimension. We provide Algorithm 7 for details and Theorem E.1 for error analysis in Appendix E.

*Masked 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 ($\tau$ bits) in a single blind rotation. The basic idea of our masked multi-bit extraction is as follows.

(1) Move the least $\tau$-bit to the MSBs (line 2).
(2) Extract all the $\tau$ bits (masked by the MSB) (line 3-8), and convert them into GGSW ciphertexts (line 10).

(3) Subtract the least $\tau$-bit from the input ciphertext, and repeat the process (line 12-14).

The idea of extracting the least $\tau$-bit for each iteration is similar to the homomorphic digit decomposition proposed by Liu et al. [33]. Since there is no padding bit, they require two blind rotation operations to exactly extract it. On the other hand, to reduce the number of blind rotation, we extract with slightly modified outputs; given an LWE ciphertext of the $\tau$-bit input $\sum_{j=0}^{\tau-1} m_j \cdot 2^j$ without padding, PBSmanyLUT combined with MV-PBS can output $\left(\mathrm{GLev}(m'_j + \dots)\right)_{j=0}^{\tau-1}$ where $m'_{\tau-1} = m_{\tau-1}$ and $m'_j = m_{\tau-1} \oplus m_j$ for $j = 0, \dots, \tau-2$ in a single blind rotation. Although we obtain slightly modified outputs $\left(\mathrm{GGSW}(m'_j)\right)_{j=0}^{\tau-1}$, masked by its MSB, we can evaluate the desired LUT $L$ easily by evaluating a modified LUT $L'$ such that $L(m) = L'(m')$ where $m = \sum_{j=0}^{\tau-1} m_j \cdot 2^j$ and $m' = \sum_{j=0}^{\tau-1} m'_j \cdot 2^j$.

## 5.3 Performance

We implement our new integer input LHE mode using the TFHE-rs library where the overall benchmark environment is the same as that of Section 3.3.

*Parameters.* We choose some of the recommended parameter sets for the BBB+ method in the TFHE-rs library with bases 16, 64, and 256, which corresponds to $2^\delta$. Then, under the same LWE dimension $n$, GLWE dimension $k$ and polynomial size $N$, we propose recommended parameters sets for our integer input LHE mode corresponding to them. Table 6 summarizes our recommended parameter sets. $\ell_{k\to k'}$, $B_{k\to k'}$ and $b_{k\to k'}$ denote the gadget length, base, and split FFT base of the GLWE keyswitching to the larger dimension, and $\ell_{k'\to k}$, $B_{k'\to k}$, and $b_{k'\to k}$ are those of the GLWE keyswitching from the larger dimension. We note that our parameter set for the base 64 uses a smaller PBS gadget length than that of the TFHE-rs library, and the standard deviation of the fresh error is updated to guarantee 128-bit security (see Appendix A.1).

*Comparison with BBB+.* We compare our new integer input LHE mode with the BBB+ method implemented in the TFHE-rs library. The detailed results are shown in Table 7. The maximum CMux depth after circuit bootstrapping is computed according the failure probability of $2^{-40}$. 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 bases, our integer input LHE mode improves the running time of the BBB+ method by factors from 7.5 to 10.7 for the parameters supporting larger max-depth. For $(\delta, \tau) = (6, 3)$, it achieves even 12.7× faster result at the cost of smaller max-depth. In terms of the key size, our method reduces it by factors from 3.4 to 4.4.

*General LUT Evaluation.* We measure the performance of evaluating general 8-to-8 and 16-to-8 LUTs using our LHE modes, and compare it to the state-of-the-art result proposed by Trama et al. [43] using tree-PBS. For our integer LHE mode, we use the parameter set of $(\delta, \tau) = (4, 2)$ with a padding bit, the same plaintext encoding with [43] of basis 16. Although the padding bit decreases its max-depth from 333 to 83, it is enough to evaluate the LUTs and enables PBS-based evaluation interchangeably. Table 8 compares the performance of 8-to-8 and 16-to-8 LUTs. It shows that our integer LHE mode is 1.42× faster than the tree-PBS for evaluating 8-to-8 LUT. It is also faster than the bitwise LHE mode since the number of PBSmanyLUT decreases due to the multi-bit extraction.

*8-bit TFHE Processor.* The work of Trama et al. [43] is not just for evaluating 8-to-8 LUT. They constructed a general-purpose 8-bit TFHE processor containing various instructions using 8-to-8 LUT, and implemented practical functions and algorithms based on the instructions. The 8-bit instructions are evaluated by 4-to-4 (PBS) and 8-to-8 (tree-PBS) LUTs, and the cost of each instruction depends on how complex its circuit is. We can replace heavy 8-bit bivariate instructions with our 16-to-8 LUT evaluation.

Table 9 compares some of the 8-bit instructions given in [43]. Their implementation based on TFHElib is not publicly open, so we estimate their work by measuring PBS and public keyswitching time on TFHE-rs under their parameters, the dominant building blocks of tree-PBS. Not all the instructions are improved, but the performance degradation (0.81× for AND/OR/XOR) is much smaller than the improvement for heavy instructions (21.15× for DIV). The comparison of all the instructions is given in Appendix F.

Furthermore, since our method evaluates a general 16-to-8 LUT, we can evaluate some 16-bit precision functions without decomposition. For example, the sigmoid function of 16-bit precision evaluated by the 8-bit instructions takes more than 6025 ms while our 16-to-8 LUT can evaluate it in 227.05 ms, which is at least 26.55× faster.

*Comparison to Batched LUT.* There are also another works evaluating LUT using other HE schemes supporting batched operation [18, 34]. Contrary to our work focusing on latency, they aims to high throughput at the cost of high latency. The performance is summarized in Table 10. Since the source code of these works are not publicly open, we just borrowed the benchmark results on their paper. We note that the high performance of [18] is affected by its GPU implementation. [18] also applied their batched LUT to batched AES evaluation, achieving 578.83 s to evaluate 2048 blocks in parallel.

# 6 Conclusion

The current designs and applications of TFHE are mainly focused on programmable 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 improved circuit bootstrapping algorithm, a building block of the leveled homomorphic evaluation. Then we refined the workflow of leveled homomorphic evaluation based on TFHE, making it faster and more flexible. 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 HomTrace, FFT multiplication, parameter evaluation are carefully polished. Based on the improvement above, the homomorphic evaluation of AES can be speed up by 4.8× and the evaluation of 8-bit instructions by factors of up to 21×.

| Basis | $n$ | $N$ | $k$ | $\ell_{\mathsf{ks}}$ | $B_{\mathsf{ks}}$ | $\ell_{\mathsf{pbs}}$ | $B_{\mathsf{pbs}}$ | $\ell_{\mathsf{tr}}$ | $B_{\mathsf{tr}}$ | $b_{\mathsf{tr}}$ | $\ell_{\mathsf{ss}}$ | $B_{\mathsf{ss}}$ | $\ell_{k\to k'}$ | $B_{k\to k'}$ | $b_{k\to k'}$ | $\ell_{k'\to k}$ | $B_{k'\to k}$ | $b_{k'\to k}$ | $\ell$ | $B$ | $\vartheta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 | 769 | 2048 | 1 | 3 | $2^4$ | 2 | $2^{15}$ | 7 | $2^7$ | 35 | 2 | $2^{16}$ | - | - | - | - | - | - | 4 | $2^4$ | 2 |
| 64 | 873 | 2048 | 1 | 2 | $2^7$ | 3 | $2^{11}$ | 4 | $2^{12}$ | 40 | 4 | $2^{10}$ | 3 | $2^{15}$ | 42 | 3 | $2^{13}$ | 40 | 4 | $2^5$ | 2 |
| 256 | 953 | 2048 | 1 | 2 | $2^7$ | 4 | $2^9$ | 6 | $2^9$ | 37 | 4 | $2^{10}$ | 3 | $2^{15}$ | 42 | 4 | $2^{10}$ | 38 | 8 | $2^3$ | 3 |

**Table 6: Recommended parameter sets for our new integer input LHE mode.**

| Basis | Method | $\delta$ | $\tau$ | Time (ms) | | | | Max Depth | Key Size (MB) |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Extr. | Refr. | Conv. | Total | | |
| 16 | [4] | 4 | 1 | 53.76 | 191.31 | 173.33 | 418.40 | 124 | 170.01 |
| | Ours | 4 | 1 | 73.33 | | 19.93 | 93.26 | 377 | 49.33 |
| | | 4 | 2 | 36.17 | | 19.95 | 56.12 | 333 | |
| 64 | [4] | 6 | 1 | 164.35 | 572.74 | 512.24 | 1249.33 | 576 | 365.13 |
| | Ours | 6 | 1 | 159.28 | | 44.92 | 204.20 | 1016 | 82.80 |
| | | 6 | 2 | 79.45 | | 45.09 | 124.53 | 897 | |
| | | 6 | 3 | 52.96 | | 45.04 | 98.00 | 140 | |
| 256 | [4] | 8 | 1 | 251.33 | 1672.4 | 1360.4 | 3284.13 | 4851 | 375.13 |
| | Ours | 8 | 1 | 290.70 | | 162.20 | 452.90 | 12418 | 120.45 |
| | | 8 | 2 | 145.03 | | 161.18 | 306.21 | 7107 | |

**Table 7: Our new integer input LHE mode performance.**

| Method | 8-to-8 LUT | 16-to-8 LUT |
|---|---|---|
| Tree-PBS [43] | 160.15 | - |
| Bitwise LHE (FFT-CMux$_1$) | 115.04 | - |
| Bitwise LHE (FFT-CMux$_2$) | 150.41 | 306.84 |
| Our Integer-LHE (Basis 16) | 112.74 | 227.05 |

**Table 8: Performance of 8-to-8 and 16-to-8 LUTs in ms.**

| Instructions | [43] | Ours | Improvement |
|---|---|---|---|
| AND/OR/XOR | 184.52 | 227.05 | 0.81 |
| EQ | 276.79 | 227.05 | 1.22 |
| LT(E)/GT(E) | 436.94 | 227.05 | 1.92 |
| MIN/MAX | 825.12 | 227.05 | 3.63 |
| MUL | 504.82 | 227.05 | 2.22 |
| DIV | 4801.05 | 227.05 | 21.15 |
| MOD | 4393.76 | 227.05 | 19.35 |

**Table 9: Performance of 8-bit instructions in ms.**

| Method | # Batch | 8-to-8 LUT | | 12-to-12 LUT | | Hardware |
|---|---|---|---|---|---|---|
| | | Amortized | Total | Amortized | Total | |
| [32] | 32768 | - | - | 39.1 ms | 1280 s | `e2-standard-4` |
| [18] | 32768 | 0.15 ms | 4.94 s | - | - | NVIDIA A100 GPU |
| Ours | 1 | 112.74 ms | | 170.62 ms | | i9-11900K |

**Table 10: Performance comparison to batched LUT.**

# References


[1] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. 2022. OpenFHE: Open-Source Fully Homomorphic Encryption Library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography* (Los Angeles, CA, USA) *(WAHC'22)*. Association for Computing Machinery, New York, NY, USA, 53–63. https://doi.org/10.1145/3560827.3563379

[2] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. 2015. Ciphers for MPC and FHE. In *Advances in Cryptology – EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9056. Springer, 430–454.

[3] Thibault Balenbois, Jean-Baptiste Orfila, and Nigel P. Smart. 2023. Trivial Transciphering With Trivium and TFHE. Cryptology ePrint Archive, Paper 2023/980. https://eprint.iacr.org/2023/980 to appear WAHC 2023.

[4] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2023. Parameter Optimization and Larger Precision for (T)FHE. *Journal of Cryptology* 36 (2023), 28. https://doi.org/10.1007/s00145-023-09463-5

[5] Loris Bergerat, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, Adeline Roux-Langlois, and Samuel Tap. 2024. New Secret Keys for Enhanced Performance in (T)FHE. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (Salt Lake City, UT, USA) *(CCS '24)*. Association for Computing Machinery, New York, NY, USA, 2547–2561. https://doi.org/10.1145/3658644.3670376

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

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (Cambridge, Massachusetts). ACM, 309–325. https://doi.org/10.1145/2633600

[8] Sergiu Carpov, Malika Izabachène, and Victor Mollimard. 2019. New Techniques for Multi-value Input Homomorphic Evaluation and Applications. In *CT-RSA 2019*, Mitsuru Matsui (Ed.). Springer, 106–126. https://doi.org/10.1007/978-3-030-12612-4_6

[9] Hao Chen, Ilaria Chillotti, and Ling Ren. 2019. Onion Ring ORAM: Efficient Constant Bandwidth Oblivious RAM from (Leveled) TFHE. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. ACM, 345–360. https://doi.org/10.1145/3319535.3354226

[10] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. 2021. Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts. In *Applied Cryptography and Network Security*, Kazue Sako and Nils Ole Tippenhauer (Eds.). Springer, 460–479.

[11] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. 2013. 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*. Springer, 315–335.

[12] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In *Advances in Cryptology – ASIACRYPT 2016*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.), Vol. 10031. Springer, 3–33.

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2017. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 377–408.

[14] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *Journal of Cryptology* 33 (2020), 34–91. Issue 1. https://doi.org/10.1007/s00145-019-09319-x

[15] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2021. Improved Programmable Bootstrapping with Larger Precision and Efficient Arithmetic Circuits for TFHE. In *ASIACRYPT 2021*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, 670–699. https://doi.org/10.1007/978-3-030-92078-4_23

[16] Jihoon Cho, Jincheol Ha, Seongkwang Kim, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon. 2021. Transciphering Framework for Approximate Homomorphic Encryption. In *Advances in Cryptology – ASIACRYPT 2021*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, 640–669.

[17] Mingyu Cho, Woohyuk Chung, Jincheol Ha, Jooyoung Lee, Eun-Gyeol Oh, and Mincheol Son. 2024. FRAST: TFHE-Friendly Cipher Based on Random S-Boxes. *IACR Transactions on Symmetric Cryptology* 2024, 3 (Sep. 2024), 1–43. https://doi.org/10.46586/tosc.v2024.i3.1-43

[18] Heewon Chung, Hyojun Kim, Young-Sik Kim, and Yongwoo Lee. 2024. Amortized Large Look-up Table Evaluation with Multivariate Polynomials for Homomorphic Encryption. Cryptology ePrint Archive, Paper 2024/274. https://eprint.iacr.org/2024/274

[19] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. 2014. 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*. Springer, 311–328.

[20] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. 2022. Towards Case-Optimized Hybrid Homomorphic Encryption. In *ASIACRYPT 2022*, Shweta Agrawal and Dongdai Lin (Eds.). Springer, 32–67. https://doi.org/10.1007/978-3-031-22969-5_2

[21] Gabrielle De Micheli, Duhyeong Kim, Daniele Micciancio, and Adam Suhl. 2024. Faster Amortized FHEW Bootstrapping Using Ring Automorphisms. In *Public-Key Cryptography – PKC 2024*, Qiang Tang and Vanessa Teague (Eds.). Springer Nature Switzerland, Cham, 322–353.

[22] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. 2018. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10991. Springer, 662–692.

[23] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. 2021. Pasta: A Case for Hybrid Homomorphic Encryption. Cryptology ePrint Archive, Report 2021/731. https://ia.cr/2021/731.

[24] Yarkın Doröz, Yin Hu, and Berk Sunar. 2016. Homomorphic AES evaluation using the modified LTV scheme. *Designs, Codes and Cryptography* 80 (2016), 333–358.

[25] Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In *Advances in Cryptology – EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9056. Springer, 617–640.

[26] Craig Gentry, Shai Halevi, and Nigel P Smart. 2012. Homomorphic evaluation of the AES circuit. In *Annual Cryptology Conference*. Springer, 850–867.

[27] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *CRYPTO 2013*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, 75–92. https://doi.org/10.1007/978-3-642-40041-4_5

[28] Antonio Guimarães, Edson Borin, and Diego F. Aranha. 2024. MOSFHET: Optimized Software for FHE over the Torus. *Journal of Cryptographic Engineering* 14, 3 (July 2024), 577–593. https://doi.org/10.1007/s13389-024-00359-z

[29] Jincheol Ha, Seongkwang Kim, Byeonghak Lee, Jooyoung Lee, and Mincheol Son. 2022. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2022*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, 581–610.

[30] Clément Hoffmann, Pierrick Méaux, and Thomas Ricosset. 2020. Transciphering, Using FiLIP and TFHE for an Efficient Delegation of Computation. In *INDOCRYPT 2020*, Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran (Eds.). Springer, Cham, 39–61. https://doi.org/10.1007/978-3-030-65277-7_3

[31] Jakub Klemsa. 2021. Fast and Error-Free Negacyclic Integer Convolution Using Extended Fourier Transform. In *Cyber Security Cryptography and Machine Learning*, Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander Schwarzmann (Eds.). Springer International Publishing, Cham, 282–300.

[32] Fukang Liu and Mohammad Mahzoun. 2023. Algebraic Attacks on RAIN and AIM Using Equivalent Representations. *Cryptology ePrint Archive* (2023).

[33] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. 2022. Large-Precision Homomorphic Sign Evaluation Using FHEW/TFHE Bootstrapping. In *ASIACRYPT 2022*, Shweta Agrawal and Dongdai Lin (Eds.). Springer, 130–160. https://doi.org/10.1007/978-3-031-22966-4_5

[34] Zeyu Liu and Yunhao Wang. 2023. Amortized Functional Bootstrapping in Less than 7 ms, with $\tilde{O}(1)$ Polynomial Multiplications. In *Advances in Cryptology – ASIACRYPT 2023*, Jian Guo and Ron Steinfeld (Eds.). Springer Nature Singapore, Singapore, 101–132.

[35] Kotaro Matsuoka. 2020. TFHEpp: pure C++ implementation of TFHE cryptosystem. https://github.com/virtualsecureplatform/TFHEpp.

[36] Kotaro Matsuoka, Ryotaro Banno, Naoki Matsumoto, Takashi Sato, and Song Bian. 2021. Virtual Secure Platform: A Five-Stage Pipeline Processor over TFHE. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 4007–4024. https://www.usenix.org/conference/usenixsecurity21/presentation/matsuoka

[37] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. 2016. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *EUROCRYPT 2016*, Marc Fischlin and Jean-Sébastien Coron (Eds.), Vol. 9665. Springer, 311–343. https://doi.org/10.1007/978-3-662-49890-3_13

[38] Pierrick Méaux, Jeongeun Park, and Hilder V. L. Pereira. 2024. Towards Practical Transciphering for FHE with Setup Independent of the Plaintext Space. *IACR Communications in Cryptology* 1, 1 (2024). https://doi.org/10.62056/anxrxrxqi

[39] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can Homomorphic Encryption be Practical?. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop* (Chicago, Illinois, USA). ACM, 113–124. https://doi.org/10.1145/2046660.2046682

[40] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.

[41] Shintaro Narisada, Hiroki Okada, Kazuhide Fukushima, and Takashi Nishide. 2023. Time-Memory Trade-off Algorithms for Homomorphically Evaluating Look-up Table in TFHE. In *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 1–10.

[42] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. 2023. 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* (, Copenhagen, Denmark,) *(WAHC '23)*. Association for Computing Machinery, New York, NY, USA, 79–90. https://doi.org/10.1145/3605759.3625260

[43] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. 2024. Designing a General-Purpose 8-bit (T)FHE Processor Abstraction. Cryptology ePrint Archive, Paper 2024/1201. https://eprint.iacr.org/2024/1201

[44] Ruida Wang, Benqiang Wei, Zhihao Li, Xianhui Lu, and Kunpeng Wang. 2024. TFHE Bootstrapping: Faster, Smaller and Time-Space Trade-Offs. In *Australasian Conference on Information Security and Privacy*. Springer, 196–216.

[45] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. 2024. Circuit Bootstrapping: Faster and Smaller. In *Advances in Cryptology – EUROCRYPT 2024*, Marc Joye and Gregor Leander (Eds.). Springer Nature Switzerland, Cham, 342–372.

[46] Benqiang Wei, Xianhui Lu, Ruida Wang, Kun Liu, Zhihao Li, and Kunpeng Wang. 2024. 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 (2024), 530–573.

[47] Benqiang Wei, Ruida Wang, Zhihao Li, Qinju Liu, and Xianhui Lu. 2023. Fregata: Faster Homomorphic Evaluation of AES via TFHE. In *Information Security*, Elias Athanasopoulos and Bart Mennink (Eds.). Springer Nature Switzerland, Cham, 392–412.

[48] Zama. 2022. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data. https://github.com/zama-ai/tfhe-rs.


# A Ciphertext Modulus and Error Standard Deviation

The FFT-based and NTT-based TFHE describe the error standard deviation with respect to the ciphertext modulus in different ways. The FFT-based one describes the error standard deviation by considering the ciphertext modulus as 1, while the NTT-based one first fixes the error standard deviation and determines the maximum ciphertext modulus for a given security level.

## A.1 Standard Deviation of LWE Error for FFT-based TFHE

We set the standard deviations of our recommended parameters for the FFT-based CBS to achieve 128-bit security (with a small security margin) using the lattice estimator.[13] The detailed values are given in Table 11.

## A.2 Ciphertext Modulus for NTT-based TFHE

In the NTT-based TFHE, the error standard deviation $\sigma$ is fixed to 3.2. Then, the maximum ciphertext modulus is determined according to the security. The ciphertext moduli for the NTT-based TFHE used in our paper are the same as the WWL+ method [45]. Concretely, the ciphertext modulus for the input LWE ciphertext is $2^{10}$ and that for the blind rotation is a 54-bit prime.

[13] https://github.com/malb/lattice-estimator

| Sets | LWE Dim. | Std. Dev. | Security |
|---|---|---|---|
| FFT-CMux | 636 | $9.251\,20 \times 10^{-5}$ | 130.7 bit |
| AES | 768 | $8.763\,87 \times 10^{-6}$ | 130.1 bit |
| Integer LHE (Basis 16) | 769 | $8.763\,87 \times 10^{-6}$ | 130.1 bit |
| Integer LHE (Basis 64) | 873 | $1.396\,26 \times 10^{-6}$ | 130.1 bit |
| Integer LHE (Basis 256) | 953 | $3.490\,65 \times 10^{-7}$ | 130.2 bit |
| FFT-CMux / HP-LHE / AES | 2048 | $9.251\,20 \times 10^{-16}$ | 130.7 bit |
| AES (HalfCBS) | 3072 | $2.168\,40 \times 10^{-19}$ | 159.5 bit |
| HP-HomTrace | 4096 | $2.168\,40 \times 10^{-19}$ | 218.5 bit |

**Table 11: LWE error standard deviations used for the recommended parameters of our FFT-based CBS.**

## B Error Analysis of the Split FFT

Klemsa [31] proposed an upper bound for the FFT error of (negacyclic) polynomial multiplication as follows.[14]

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

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 between the worst-case and average-case analysis, since most TFHE parameters are chosen based on the average-case analysis using the independence heuristic [14, 15]. So we opt for Bergerat et al.'s method [5] to estimate the FFT error in this paper.

For the split FFT to work correctly, we have to guarantee its failure probability is small enough. By estimating the standard deviations of the upper part under the parameters used in this paper, we show that the failure probabilities of the split FFT are smaller than $2^{-256}$. The results are summarized in Table 12.

| | $N$ | $k$ | $B$ | $\ell$ | $b$ | Std. Dev. | F.P. |
|---|---|---|---|---|---|---|---|
| FFT-CMux$_3$ | 2048 | 1 | $2^7$ | 6 | 35 | $2^{-6.01}$ | $2^{-751.7}$ |
| Integer LHE (Basis 16) | 2048 | 1 | $2^7$ | 7 | 35 | $2^{-5.90}$ | $2^{-645.0}$ |
| Integer LHE (Basis 64) | 2048 | 1 | $2^{15}$ | 3 | 42 | $2^{-5.51}$ | $2^{-378.0}$ |
| | 2048 | 2 | $2^{12}$ | 4 | 40 | $2^{-5.80}$ | $2^{-564.9}$ |
| | 2048 | 2 | $2^{12}$ | 3 | 40 | $2^{-6.01}$ | $2^{-751.7}$ |
| Integer LHE (Basis 256) | 2048 | 2 | $2^{15}$ | 3 | 42 | $2^{-5.51}$ | $2^{-378.0}$ |
| | 2048 | 1 | $2^9$ | 6 | 37 | $2^{-5.51}$ | $2^{-378.0}$ |
| | 2048 | 2 | $2^{10}$ | 4 | 38 | $2^{-5.80}$ | $2^{-564.9}$ |
| AES (HalfCBS) | 1024 | 3 | $2^{15}$ | 3 | 42 | $2^{-5.72}$ | $2^{-502.6}$ |
| AES (Set-II) | 1024 | 2 | $2^7$ | 6 | 34 | $2^{-5.51}$ | $2^{-378.0}$ |

**Table 12: Standard deviations for the upper part of the split FFT and its failure probabilities for the GLWE keyswitching under the parameters used in this paper.**

[14]The second-order terms are neglected.

## C TFHE Operations

As in most 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 $\sigma$. We denote the noise variance of a key in terms of $\ell_\infty$-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 $\ell$, we assume the decomposition error is uniformly sampled from $[\![-\frac{q}{2B^\ell}, \frac{q}{2B^\ell}[\![$ as analogous to [15]. As mentioned in Section 2.2, we only deal with the binary secret key in this section.[15] The proofs given in this section comes from [9, 10, 15, 21] with a slight modification generalizing GLWE dimension $k$.

### C.1 Modulus Switching

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

Lemma C.1 (Modulus Switching). *Let $\mathbf{C} \in \mathcal{R}_{q,N}^{k+1}$ be a GLWE ciphertext of a phase $\mu$ 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 $V_{\mathsf{ms}}$ of $E_{\mathsf{ms}}$ 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 $e_{\mathsf{ms}}$ has variance bounded above by $\frac{n+1}{12}$. We note that $e_{\mathsf{ms}}$ (and $E_{\mathsf{ms}}$) does not depend on $q$ and $q'$.

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

## C.2 GLWE Keyswitching

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

**Algorithm 5:** GLWE keyswitching GLWE_KS

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

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

Lemma C.2 (GLWE Keyswitching). *Let $\mathbf{C}$ be a GLWE ciphertext of a phase $\mu$ 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 $\mu + E_{\mathrm{ks}}(X)$ under $\mathbf{S}'$ where the variance $V_{\mathrm{ks}}$ of $E_{\mathrm{ks}}(X)$ is given as follows.*

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

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 $E_{\mathrm{ks}} = -\sum_{i=1}^{k} E'_i\cdot S_i + \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j}\cdot E_{i,j}$ is given as follows.

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

□

## C.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 [14, 15].

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 $\mathrm{KS}_i = \mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(s_i)$ for $i = 1, \ldots, n$ where $B$ and $\ell$ 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 $\mathrm{KS}_i$. The LWE to GLWE keyswitching outputs a GLWE ciphertext $\mathbf{C}$ of $m$ given as follows.

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

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

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

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

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

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

where $(\tilde{a}_{i,1}, \ldots, \tilde{a}_{i,\ell})$ is the gadget decomposition of the function 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. $\ell$) 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.

*Precomputing Keyswitching Keys.* The private functional keyswitching uses the keyswitching keys $\mathrm{KS}_{z,i,j}^{(f)}$ to compute $\tilde{a}_{i,j}^{(z)}\, \mathrm{KS}_{z,i,j}^{(f)}$ for all $z, i, j$. To reduce the computational cost and error growth at the cost of increased key size, one can precompute $\tilde{a}\, \mathrm{KS}_{z,i,j}^{(f)}$ by $\mathrm{GLWE}_{\mathbf{S}}(\tilde{a}\cdot q/B^j \cdot f(0, \ldots, 0, s_i, 0, \ldots, 0))$ for all possible $\tilde{a}$ in $[\![1, B/2]\!]$. The same trade-off can be applied to other kinds of keyswitching operations.

## C.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. [10] 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 $\tau_d$ maps $M(X)$ into $M(X^d)$. Given a GLWE secret key $\mathbf{S}(X) \in \mathcal{R}_{q,N}^k$, a GLWE ciphertext $\mathrm{GLWE}_{\mathbf{S}(X)}(M(X))$ of $M(X)$ under $\mathbf{S}(X)$ can be regarded as one $\mathrm{GLWE}_{\mathbf{S}(X^d)}(M(X^d))$ of $M(X^d)$ under $\mathbf{S}(X^d)$. By switching the key of $\mathrm{GLWE}_{\mathbf{S}(X^d)}(M(X^d))$ from $\mathbf{S}(X^d)$ to $\mathbf{S}(X)$, one can obtain the GLWE ciphertext of $M(X^d)$ under the original secret key $\mathbf{S}(X)$. We refer to Appendix C.2 for the details of GLWE keyswitching.

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

Lemma C.3 (HomTrace Evaluation). *Let $\mathbf{C}$ be a GLWE ciphertext of a phase $\mu$ under $\mathbf{S}$. Let $V_{\mathsf{auto}}$ 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 $V_{\mathsf{tr}}$ of $E_{\mathsf{tr}}$ is given as follows.*

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

*where $V_{\mathsf{auto}}$ 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 $V_{\mathsf{auto}}$ can be upper bounded by Lemma C.2 since EvalAuto evaluates a single GLWE keyswitching operation.

## C.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.

Lemma C.4 (External Product). *Let $\overline{\overline{\mathbf{C}}}_1 = \mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M)$ be a GGSW ciphertext of $M$ having variance $\sigma_{\mathsf{ext}}^2$ under $\mathbf{S}$ and $\mathbf{C}_2$ be a GLWE ciphertext of a phase $\mu$ under $\mathbf{S}$. Then, external product $\overline{\overline{\mathbf{C}}}_1 \boxdot \mathbf{C}_2$ outputs a GLWE ciphertext of a phase $\mu \cdot M + E_{\mathsf{ext}}$ under $\mathbf{S}$ where the variance $V_{\mathsf{ext}}$ of $E_{\mathsf{ext}}$ is given as follows.*

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

**Algorithm 6:** SchemeSwitch

**Input:** $\overline{\mathrm{C}} = \mathrm{GLev}_{\mathbf{S}}^{(B,\ell)}(M)$ 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:** $\overline{\overline{\mathrm{C}}}' = \mathrm{GGSW}_{\mathbf{S}}^{(B,\ell)}(M)$ such that $\mathrm{C}'_{i,j} = \mathrm{GLWE}_{\mathbf{S}}(-\frac{q}{B^j} \cdot MS_i)$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$

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

**for** $i = 1$ *to* $k+1$ **do**

  **for** $j = 1$ *to* $\ell$ **do**

    $\mathrm{C}'_{i,j} \leftarrow \mathrm{SS}[i] \boxdot \mathrm{C}_j$

**return** $\overline{\overline{\mathrm{C}}}' = \left(\mathrm{C}'_{i,j}\right)_{(i,j)\in[k+1]\times[\ell]}$

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

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

where $\langle \mathrm{C}_{i,j}, (-\mathbf{S}, 1)\rangle = \frac{q}{B^j}(-S_i \cdot M) + E_{i,j}$ and $\mathrm{Var}(E_{i,j}) = \sigma^2_{\mathsf{ext}}$ for $i = 1, \ldots, k+1$ and $j = 1, \ldots, \ell$. Let $\mathrm{C}_2 = (A_1, \ldots, A_{k+1})$ and

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

be the gadget decomposition of $A_i$ such that $\|A'_{i,j}\|_\infty \le \frac{B}{2}$ and $\|E'_{i,j}\|_\infty \le \frac{q}{2B^\ell}$ for $i = 1, \ldots, k+1$. Then the output of external product $\overline{\overline{\mathrm{C}}}_1 \boxdot \mathrm{C}_2$ can be represented as $\sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j} \cdot \mathrm{C}_{i,j}$. Then, the phase of the output is given as follows.

$$\begin{aligned}
\langle \overline{\overline{\mathrm{C}}}_1 \boxdot \mathrm{C}_2, (-\mathbf{S}, 1)\rangle &= \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j}\left(\frac{q}{B^j}(-S_i \cdot M) + E_{i,j}\right) \\
&= \sum_{j=1}^{\ell} A'_{k+1,j} \cdot \left(\frac{q}{B^j}M + E_{k+1,j}\right) - \sum_{i=1}^{k}\sum_{j=1}^{\ell} A'_{i,j}\left(\frac{q}{B^j}M - E_{i,j}\right) \\
&= \mu \cdot M + \left(E'_{k+1} - \sum_{i=1}^{k} E'_i S_i\right)M + \sum_{i=1}^{k+1}\sum_{j=1}^{\ell} A'_{i,j}E_{i,j}.
\end{aligned}$$

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 $E_{\mathsf{ks}}$ is given as follows.

$$\begin{aligned}
\mathrm{Var}(E_{\mathsf{ks}}) \le{}& (1+kN)\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right)\ell_2(M)^2 \\
&+ (k+1)\ell N\left(\frac{B^2+2}{12}\right)\sigma^2_{\mathsf{ext}}.
\end{aligned}$$

□

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 C.4.

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

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

## C.6 Evaluation Key Size

In this subsection, we describe the size of various evaluation keys used in TFHE according to the parameters, summarizing the result in Table 13. As evaluation keys are encryptions of secret information, we begin with the description of ciphertext sizes.

*Ciphertext Size.* An LWE ciphertext $(a_1, \ldots, a_n, b) \in \mathbb{Z}_q^{n+1}$ consists of $n+1$ elements in $\mathbb{Z}_q$, so its size is given by $(n+1)\log q$ bits. If the LWE ciphertext is a fresh one such that no homomorphic operation is performed on it yet, then one can compress the random mask $a$ into a seed for generating it. Such LWE ciphertexts are called seeded LWE ciphertexts. Ignoring the seed size by assuming that one seed generates all the random masks for multiple seeded ciphertexts, the size of the seeded LWE ciphertext is only $\log q$.[16] 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 $\ell$ and $\ell(k+1)$ GLWE ciphertexts, respectively. Table 13a summarizes the size of each type of 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 $k_{\mathsf{src}}$ to another key $\mathbf{S}_{\mathsf{dst}} \in \mathcal{R}_{q,N}^{k_{\mathsf{dst}}}$ of dimension $k_{\mathsf{dst}}$ with the same polynomial size $N$ is a set of $k_{\mathsf{src}}$ 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 13b 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 $\{\mathrm{Lev}_{\mathbf{s}}^{(B_{\mathsf{ks}},\ell_{\mathsf{ks}})}(s'_i)\}_{i=1}^{kN}$.

[16] 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.

*Circuit Bootstrapping Key.* Let $\mathbf{s} \in \mathbb{B}^n$, $\mathbf{S} \in \mathbb{B}_N[X]^k$ and $\mathbf{s}' \in \mathbb{B}^{kN}$ be defined the same as above. The (previous) circuit bootstrapping takes an input LWE ciphertext under $\mathbf{s}$ and outputs a corresponding GGSW ciphertext under $\mathbf{S}$ using a sequence of PBS operations and private functional keyswitching operations. The private functional keyswitching operation for the circuit bootstrapping, which switches an LWE ciphertext $\mathrm{LWE}_{\mathbf{s}}(m)$ into a GLWE ciphertext $\mathrm{GLWE}_{\mathbf{S}}(-S_i \cdot m)$ for $i = 1, \ldots, k+1$, requires a set of $k+1$ GLev ciphertexts $\{\mathrm{GLev}_{\mathbf{S}}^{(B_{\mathsf{priv}}, \ell_{\mathsf{priv}})}(-S_i)\}_{i=1}^{k+1}$ where $\mathbf{S} = (S_1, \ldots, S_k)$ and $S_{k+1} = -1$.[17] Table 13c summarizes the evaluation keys for the bootstrapping operations in TFHE.

# D 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. [40], 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.

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

## D.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 optimization based on GLWE keyswitching proposed by Bergerat et al. [5], which is an extension of the method proposed by Chen et al. [10].[18]

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 $n_{\mathsf{src}}$ and $n_{\mathsf{dst}}$. 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 $m + u_1 X + \cdots + u_{N-1} X^{N-1}$ under $\mathbf{S}_{\mathsf{src}}$ where $u_1, \ldots, u_{N-1}$ 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 $\mathbf{S}_{\mathsf{dst}}$.

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. One can directly evaluate each output bit of the AES S-box using an 8-to-1 general LUT evaluation. Then, the following ShiftRows, MixColumns, and AddRoundKey require at most 8 additions.

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 integrating AddRoundKey and SubBytes using keyed S-box, i.e., sending GLWE ciphertexts that encode the tables of keyed S-boxes $S_k$ such that $S_k(x) = S(x \oplus k)$ where $S$ is the AES S-box and $k$ is the round key. By transferring the keyed S-box instead of the round key, one can integrate AddRoundKey and SubBytes in LUT evaluation at the cost of increased key size.

*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.

# E Detailed Description for Our Integer LHE Mode

*Integrate Extr. with Refr.* First, we integrate the Extr. step with the Refr. step by a single PBSmanyLUT operation, reducing the number of blind rotation. The Extr. step extracts each bit of the message from

[17] To be precise, the private keyswitching from $\mathrm{LWE}(m)$ to $\mathrm{GLWE}(-S_{k+1} \cdot m)$ is a packing keyswitching since $-S_{k+1} \cdot m = m$.
[18] The work of Bergerat et al. [5] enables using non-power-of-two LWE dimensions based on their new concept of partial GLWE secret key, while we choose power-of-two LWE dimensions for simplicity.

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

**(a) Size of 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 13: Size of FHEW/TFHE ciphertexts and evaluation keys in bits. The size of seeds or auxiliary information is ignored.**

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 PBSmanyLUT operation per each bit.

*Improved Conv. with HP-HomTrace.* 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.

We propose Algorithm 7 for high-precision conversion step, and present Theorem E.1 to bound the conversion noise. We note that HomTrace is evaluated in the larger GLWE dimension $k'$ in Algorithm 7, so one can make $V_{\mathsf{tr}}$ in Theorem E.1 much smaller than that in Theorem 3.1 and 3.2 for the same $N$.

Theorem E.1. *Let $\mathbf{C}_j$ be a GLWE ciphertext of $v_j \cdot m + \dots$ for $j = 1, \dots, \ell$ under a secret key $\mathbf{S} = (S_1, \dots, S_k)$, $\mathbf{S}' = (S'_1, \dots, S'_k)$ be a GLWE secret key of a dimension $k'$ such that $k' > k$, and $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}$ (resp. $\mathrm{KS}_{\mathbf{S}'\to\mathbf{S}}$) be a GLWE keyswitching key from $\mathbf{S}$ to $\mathbf{S}'$ (resp. $\mathbf{S}'$ to $\mathbf{S}$). Then Algorithm 7 outputs a GGSW ciphertext $\overline{\overline{\mathbf{C}}}$ of $m$ under $S$ with additional error variance $V_{\mathsf{hp\text{-}conv}}$ such that*

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

**Algorithm 7:** High-Precision Conversion HP-Conv

**Input:** $\mathbf{C}_j = \mathrm{GLWE}_{\mathbf{S}}(v_j \cdot m + \dots)$ for $j = 1, \dots, \ell$
**Input:** GLWE keyswitching keys $\mathrm{KS}_{\mathbf{S}\to\mathbf{S}'}$ and $\mathrm{KS}_{\mathbf{S}'\to\mathbf{S}}$ and $\mathbf{S}' = (S'_1, \dots, S'_{k'})$ wheres $k' > k$
**Input:** Automorphism keys under $\mathbf{S}'$
**Input:** Scheme switching key under $\mathbf{S}$
**Output:** $\overline{\overline{\mathbf{C}}} = \mathrm{GGSW}_{\mathbf{S}}(m)$

```
for j = 1 to ℓ do
    C'_j ← GLWE_KS(C_j, KS_{S→S'})
    C'_j ← Preprocess(C'_j)
    C'_j ← HomTrace(C')
    C'_j ← GLWE_KS(C', KS_{S'→S})
C̄' ← {C'_j}_{j=1}^ℓ
C̿ ← SchemeSwitch(C̄')
return C̿
```

*where $V_{\mathsf{pre}}$ is the pre-processing noise variance, $V_{\mathbf{S}\to\mathbf{S}'}$ (resp. $V_{\mathbf{S}'\to\mathbf{S}}$) is the GLWE keyswitching noise variance from $\mathbf{S}$ to $\mathbf{S}'$ (resp. $\mathbf{S}'$ to $\mathbf{S}$), $V_{\mathsf{hp\text{-}tr}}$ is the trace evaluation variance under $\mathbf{S}'$, and $V_{\mathsf{ss}}$ is the additive scheme switching error.*

Proof. Compared to the original Conv. step, the high-precision conversion (Algorithm 7) has additional GLWE keyswitching operations. For the first keyswitching error (from $\mathbf{S}$ to $\mathbf{S}'$), the subsequent pre-processing prevents the error amplification by HomTrace and only its constant term is remained after 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 HomTrace error. □

*Masked Multi-Bit Extraction.* Lastly, we propose masked multi-bit extraction to reduce the number of PBSmanyLUT operations, which was the same as the number of the extraction operations. We extract

$\tau$-bit of the message for each extraction step, where $\tau \in \{2, 3\}$, using a single PBSmanyLUT operation combined with the MV-PBS method [8]. For each $\tau$-bit extraction, all the bits are masked by its MSB bit (except the MSB bit itself), while we can evaluate LUT correctly by modifying the evaluating function using the masked input bits.

For simplicity, we describe the case of $\tau = 2$. Suppose an input ciphertext $\mathrm{LWE}(\lceil q/2^2 \rceil \cdot (2m_1 + m_0))$ of a 2-bit message 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 the BBB+ method originally outputs $\mathrm{GGSW}(m_0)$ and $\mathrm{GGSW}(m_1)$ to 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 PBSmanyLUT operation, $\vartheta$ should be increased by 1 to evaluate twice many functions. However, using a larger $\vartheta$ increases the failure probability of the PBSmanyLUT operation. Combined with the small scaling factor $\lceil q/2^2 \rceil$ of the input ciphertext in the multi-bit extraction, increasing $\vartheta$ might lead to a large failure probability. To keep the value of $\vartheta$, we opted for the MV-PBS.

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

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

Blind rotation on it 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 as the first phase 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 $f_{m_1}$. The full algorithm for the CBS algorithm of our integer input LHE mode is described in Algorithm 4.

Theorem E.2. *Provided that the* PBSmanyLUT *does not fail, Algorithm 4 outputs GGSW ciphertexts with error variance*

$$c_\tau V_{\mathsf{pbs}} + V_{\mathsf{conv}}$$

*where* $c_\tau = \max_{0 \le k \le \tau-2} \ell_2(G_k)^2$ *and*

$$V_{\mathsf{conv}} = \begin{cases} V_{\mathsf{pre}} + \frac{N}{2} V_{\mathsf{tr}} + V_{\mathsf{ss}} & \text{if Conv is used,} \\ V_{\mathsf{hp\text{-}conv}} \text{ (see Theorem E.1)} & \text{if HP-Conv is used.} \end{cases}$$

Proof Sketch. The proof is the same as Theorem 3.2 except that $V_{\mathsf{pbs}}$ is multiplied by $c_\tau$ during MV-PBS in line 6. We refer to Appendix G for the details. □

We used $\tau = 2$ and 3 for the multi-bit extraction (see Table 6). To be precise, we note that $c_2 = 1$ and $c_3 = 3$ because $G_0 = -X^{N/2}$ and $G_1 = -(X^{N/4} - X^{N/2} + X^{3N/4})$.

*Max Depth of the Integer Input LHE Mode.* The max depth of the integer input LHE mode is also defined as the maximum number of CMux gates under the failure probability of $2^{-40}$ for the next PBS. But computation is more complicated than that of the bitwise input CBS.

Let $M$ be the number of CMux gates, then the input error variance $V_{\mathsf{in}}$ of the integer input LHE is given by

$$V_{\mathsf{in}} = M \cdot \left( (1 + kN)\left(\frac{q^2 - B^{2\ell}}{24B^{2\ell}} + \frac{1}{12}\right) + (k+1)\ell N \left(\frac{B^2 + 2}{12}\right)(c_\tau V_{\mathsf{pbs}} + V_{\mathsf{conv}}) \right)$$

where $c_\tau V_{\mathsf{pbs}} + V_{\mathsf{conv}}$ is the error variance of the output of Algorithm 4 (see Theorem E.2), and $(B, \ell)$ is the gadget base and length of the output GGSW ciphertext (see Lemma C.4). Then, for the base $2^p$, the $\tau$ LSB bits of the input are moved to the MSB by multiplying $2^{p-\tau}$, and fed to PBSmanyLUT operation. With $\sigma_{\mathsf{in}}^2 = 2^{2(p-\tau)} \cdot V_{\mathsf{in}}$, $\Delta_{\mathsf{in}} = q/2^\tau$, the failure probability of the PBSmanyLUT operation is computed by Lemma 2.1. There are also PBSmanyLUT operations in next iterations, while their input error variances are smaller than the first iteration since the multiplication factor to move the $\tau$ LSB to the MSB decreases.

## F Comparison of Full 8-bit Instructions

In this section, we give a full comparison of the performance for the 8-bit instructions. The list of the instructions proposed in [43] are as follows. If there is 'i' at the end of the instruction, it implies the variant of the instruction taking as inputs an encrypted one and a cleartext one.

- Arithmetic instructions
  - ADD(i)/SUB(i)/MUL(i): addition/subtraction/multiplication of two bytes (modulo 256)
  - ADDZ: addition provided that one of the inputs is zero
  - MULM(i): most significant byte of the product of two bytes
  - DIV4(i): division of a byte by a nibble[19]
  - DIV(i): division of a byte by another one
  - MOD4(i): modulo of a byte by a nibble
  - MOD(i): modulo of a byte by another one
- Bitwise instructions
  - AND(i)/OR(i)/XOR(i): bitwise and/or/xor of two bytes
  - (U)SHL(i)/(U)SHR(i): shift a byte (signed or unsigned) left/right by a byte index
  - ROL(i)/ROR(i): rotate a byte left/right by a byte index
- Test instructions
  - EQ(i): test if two inputs are the same byte
  - GT(E)(i)/LT(E)(i): test if the first input byte is greater/less than (or equal to) the second input byte
- Other instructions
  - MIN(i)/MAX(i): minimum/maximum of two bytes
  - (N)CDUP(i): conditional duplication
  - CSEL: conditional selection
  - ABS: absolute value of a signed byte
  - NEG: opposite of a signed byte
  - XOP: user's defined 8-to-8 LUT

[19] A nibble denotes 4-bit message

| Instructions | # PBS | # PubKS | [43] | Ours | Improvement | Instructions | # PBS | # PubKS | [43] | Ours | Improvement |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ANDi/ORi/XORi | 2 | 0 | 48.75 | 36.17 | 1.35 | AND/OR/XOR | 4 | 2 | 184.52 | 227.05 | 0.81 |
| DC | 2 | 0 | 48.75 | 36.17 | 1.35 | RC | 8 | 0 | 195.00 | 144.67 | 1.35 |
| (U)SHLi/(U)SHRi | 2 | 0 | 48.75 | 36.17 | 1.35 | (U)SHL/(U)SHR | 6 | 4 | 320.30 | 227.05 | 1.41 |
| ROLi/RORi | 4 | 0 | 97.50 | 36.17 | 2.70 | ROL/ROR | 9 | 6 | 480.45 | 227.05 | 2.12 |
| EQi | 2 | 0 | 48.75 | 36.17 | 1.35 | EQ | 6 | 3 | 276.79 | 227.05 | 1.22 |
| LT(E)i/GT(E)i | 2 | 1 | 92.26 | 112.74 | 0.82 | LT(E)/GT(E) | 9 | 5 | 436.94 | 227.05 | 1.92 |
| (N)CDUP | 3 | 1 | 116.64 | 70.71 | 1.65 | CSEL | 9 | 6 | 480.45 | 227.05 | 2.12 |
| NEG/ABS | 2 | 1 | 92.26 | 112.74 | 0.82 | MIN/MAX | 16 | 10 | 825.12 | 227.05 | 3.63 |
| ADDi/SUBi | 2 | 1 | 92.26 | 112.74 | 0.82 | ADD/SUB | 7 | 4 | 344.68 | 227.05 | 1.52 |
| ADDZ | 4 | 2 | 184.52 | 227.05 | 0.81 | MUL(M)i/DIV(4)i/MOD4i | 2 | 1 | 92.26 | 112.74 | 0.82 |
| MODi | 3 | 2 | 160.15 | 112.74 | 1.42 | MUL | 10 | 6 | 504.82 | 227.05 | 2.22 |
| MULM | 32 | 20 | 1650.24 | 227.05 | 7.27 | DIV4 | 21 | 14 | 1121.04 | 227.05 | 4.94 |
| DIV | 97 | 56 | 4801.05 | 227.05 | 21.15 | MOD4 | 10 | 6 | 504.82 | 227.05 | 2.22 |
| MOD | 91 | 50 | 4393.73 | 227.05 | 19.35 | (N)CDUPi | 1 | 0 | 24.38 | 18.08 | 1.35 |
| XOP | 3 | 2 | 160.15 | 112.74 | 1.42 | MINi/MAXi | 2 | 1 | 92.26 | 112.74 | 0.82 |

**Table 14: Performance of all the 8-bit TFHE instructions in ms.**

– DC/RC: binary decomposition / recomposition

As mentioned in Section 5.3, we estimate the work of Trama et al. [43] by measuring the PBS time (24.375 ms) and public keyswitching time (43.512 ms). If the instruction does not require public keyswitching, we also estimate our performance of it only by our PBS time. If not, we estimate our performance by 8-to-8 LUT for univariate instructions and 16-to-8 LUT for bivariate instructions. Especially for (N)CDUP, one of its input is a selection bit so we estimate its performance by 9-to-8 LUT. The result is summarized in Table 14.

# G Proofs

## G.1 Theorem.1

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

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

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

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

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

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

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

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

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 $e_{\mathsf{pbs}}$ is only a constant term. Furthermore, $V(E_{\mathsf{tr}}(X) S_i) \le \frac{N}{2} V_{\mathsf{tr}}$, where $N$ is the ring expansion factor. Substituting these estimate into the above formula, we obtain

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

□

## G.2 Theorem.2

Proof. The proof is analogous to that of Theorem 3.1 except that there is an additional pre-processing error. The Refr. step outputs a GLev ciphertext whose $j$-th GLWE ciphertext is

$$\frac{q}{B^j} m + y_1 X + \cdots + y_{N-1} X^{N-1} + E_{\mathsf{pbs}}(X),$$

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

$$\frac{1}{N}\left(\frac{q}{B^j} m + y_1 X + \cdots + y_{N-1} X^{N-1} + E_{\mathsf{pbs}}(X)\right) + E_{\mathsf{ms}}(X) + \frac{q}{N} U(X)$$

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 HomTrace is

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

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

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

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

## H Re-Analysis of WWL+

This section reanalyzes the circuit bootstrapping proposed in [45] and fixes previous errors. The original WWL+ 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 [15] without sample extraction, or
- automorphism-based multi-value blind rotation [45].

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

We propose Theorem H.1 to provide a detailed re-analysis of the noise growth in the CBS algorithm proposed by Wang et al. [45].

Theorem H.1. *Let* $\mathbf{c}$ *be an LWE ciphertext of phase* $\mu$ *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* $\mu + E_{\mathrm{cbs}}(X)$ *under the GLWE secret key* $\mathbf{S} = (S_1, \ldots, S_k)$ *corresponding to* $\mathbf{s}$ *where the variance* $V_{\mathrm{cbs}}$ *of* $E_{\mathrm{cbs}}(X)$ *is given as follows.*

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

*where* $V_{\mathrm{pbs}}$ *denotes the* PBSmanyLUT [20] *output error variance,* $V_{\mathrm{tr}}$ *denotes the* HomTrace *output error variance, and* $V_{\mathrm{ss}}$ *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

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

where $y_i X^i$ are some redundant terms and $E_{\mathrm{pbs}}(X)$ 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

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

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

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

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

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

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

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

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

□

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

Based on the noise analysis from Theorem H.1, we have adjusted the noise control parameters, as listed in Table 15 and implemented them in OpenFHE [1]. As in Remark 1, the estimations of AVX-512 accelerated results are given in parentheses. We note that the key size in this paper assumes that the evaluation keys are compressed (see Appendix C.6).

## I Algorithms

### I.1 PBSmanyLUT

**Algorithm 8:** PBSmanyLUT

**Input:** $\mathbf{c}_{\mathrm{in}} = \mathrm{LWE}_{\mathbf{s}}(m \cdot \Delta_{\mathrm{in}}) = (a_1, \ldots, a_n, a_{n+1} = b) \in \mathbb{Z}_q^{n+1}$ under $\mathbf{s} = (s_1, \ldots, s_n)$ where $(\beta, m') \leftarrow \mathrm{PTModSwitch}_q(m, \Delta_{\mathrm{in}}, \vartheta)$

**Input:** $\mathrm{BSK} = (\mathrm{GGSW}_{\mathbf{S}'}(s_i))_{1 \le i \le n}$ under $\mathbf{S}' = (S'_1, \ldots, S'_k)$ with decomposition base $B_{\mathrm{pbs}}$ and level $\ell_{\mathrm{pbs}}$

**Input:** $P_{(f_1,\ldots,f_{2^\vartheta})}$: a redundant LUT for $f_1, \ldots, f_{2^\vartheta}$

**Output:** $\mathbf{c}_1, \ldots \mathbf{c}_{2^\vartheta}$ where $\mathbf{c}_j = \mathrm{LWE}_{\mathbf{s}'}((-1)^\beta \cdot f_j(m') \cdot \Delta_{\mathrm{out}})$ and $\mathbf{s}'$ is the LWE secret key corresponding to $\mathbf{S}'$

**for** $i = 1$ *to* $n+1$ **do**
  $a'_i \leftarrow \left\lfloor \left\lfloor \frac{a_i \cdot 2N \cdot 2^{-\vartheta}}{q} \right\rceil \cdot 2^\vartheta \right\rceil_{2N}$ /* PTModSwitch */
$\mathbf{C}_{(f_1,\ldots,f_{2^\vartheta})} \leftarrow \mathrm{GLWE}^0_{\mathbf{S}'}(P_{(f_1,\ldots,f_{2^\vartheta})})$
$\mathbf{C} \leftarrow \mathrm{BlindRotate}\left(\mathbf{C}_{(f_1,\ldots,f_{2^\vartheta})}, \{a'_i\}_{i=1}^{n+1}, \mathrm{BSK}\right)$
**for** $j = 1$ *to* $2^\vartheta$ **do**
  $\mathbf{c}_j \leftarrow \mathrm{SampleExtract}_{j-1}(\mathbf{C})$

### I.2 MV-PBS

**Algorithm 9:** MV-PBS

**Input:** $\mathbf{c}_{\mathrm{in}} = \mathrm{LWE}_{\mathbf{s}}(m \cdot \Delta_{\mathrm{in}}) = (a_1, \cdots, a_n, a_{n+1} = b) \in \mathbb{Z}_q^{n+1}$

**Input:** $\mathrm{BSK} = (\mathrm{GGSW}_{\mathbf{S}'}(s_i))_{1 \le i \le n}$ under $\mathbf{S}' = (S'_1, \cdots, S'_k)$

**Input:** $t+1$ polynomials $P_0$ and $P'_j$ for $j = 1, \ldots, t$ such that $P'_j \cdot P_0 = P_j$ where $P_j$ is the polynomial encoding $f_j$

**Output:** $\mathbf{c}_1, \cdots, \mathbf{c}_t$ where $\mathbf{c}_j = \mathrm{LWE}_{\mathbf{s}'}(f_j(m) \cdot \Delta_{\mathrm{out}})$ and $\mathbf{s}'$ is the LWE secret key corresponding to $\mathbf{S}'$

**for** $i = 1$ *to* $n+1$ **do**
  $\tilde{a}_i = \lfloor a_i \cdot (2N)/q \rceil$ /* ModSwitch */
Let $\mathrm{ACC} \leftarrow \mathrm{GLWE}^0_{\mathbf{S}'}(P_0)$
$\mathrm{ACC} \leftarrow \mathrm{BlindRotate}(\mathrm{ACC}, \{\tilde{a}_i\}_{i=1}^{n+1}, \mathrm{BSK})$
**for** $j = 1$ *to* $t$ **do**
  $\mathrm{ACC}_j \leftarrow P'_j \cdot \mathrm{ACC}$
  $\mathbf{c}_j = \mathrm{SampleExtract}(\mathrm{ACC}_j)$
**return** $\mathbf{c}_1, \cdots, \mathbf{c}_t$

| Sets | $\ell_{\text{pbs}}$ | $B_{\text{pbs}}$ | $\ell_{\text{tr}}$ | $B_{\text{tr}}$ | $\ell_{\text{ss}}$ | $B_{\text{ss}}$ | $\ell$ | $B$ | Max Depth | Key Size (MB) | # of NTTs | Time (ms) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NTT-CMux$_{\text{O1}}$ | 2 | $2^{17}$ | 3 | $2^{13}$ | 1 | $2^{28}$ | 4 | $2^3$ | 2 | 30.56 | 3610 | 95.92 | (39.00) |
| NTT-CMux$_{\text{O2}}$ | 3 | $2^{13}$ | 5 | $2^9$ | 1 | $2^{28}$ | 4 | $2^4$ | 101 | 45.91 | 4840 | 119.09 | (52.29) |
| NTT-CMux$_{\text{O3}}$ | 7 | $2^7$ | 7 | $2^7$ | 2 | $2^{19}$ | 4 | $2^5$ | 53378 | 106.43 | 9500 | 250.41 | (102.64) |

**Table 15: 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.**