Faster TFHE Bootstrapping with Block Binary Keys *

Changmin Lee[1], Seonhong Min[2], Jinyeong Seo[2], and Yongsoo Song[2]

[1] Korea Institute for Advanced Study, Seoul, Republic of Korea
[2] changminlee@kias.re.kr
Seoul National University, Seoul, Republic of Korea
[3] {minsh, jinyeong.seo, y.song}@snu.ac.kr

Abstract. Fully Homomorphic Encryption over the Torus (TFHE) is a homomorphic encryption scheme which supports efficient Boolean operations over encrypted bits. TFHE has a unique feature in that the evaluation of each binary gate is followed by a bootstrapping procedure to refresh the noise of a ciphertext. In particular, this gate bootstrapping involves two algorithms called the blind rotation and key-switching.

In this work, we introduce several optimization techniques for the TFHE bootstrapping. We first define a new key distribution, called the block binary distribution, where the secret key can be expressed as a concatenation of several vectors of Hamming weight at most one. We analyze the hardness of (Ring) LWE with a block binary secret and provide candidate parameter sets which are secure against the best-known attacks. Then, we use the block key structure to simplify the inner working of blind rotation and reduce its complexity. We also modify the RLWE key generation and the gadget decomposition method to improve the performance of the key-switching algorithm in terms of complexity and noise growth.

Finally, we use the TFHE library to implement our algorithms and demonstrate their benchmarks. Our experimentation shows that the execution time of TFHE bootstrapping is reduced from 10.5ms down to 6.4ms under the same security level, and the size of the bootstrapping key decreases from 109MB to 60MB.

Keywords: Homomorphic Encryption, Bootstrapping

1 Introduction

Homomorphic encryption (HE) is a cryptosystem that allows us to evaluate arbitrary functions on encrypted data without decryption. After the first construction by Gentry [18], this technology has been considered to be one of the most promising cryptographic primitives for secure computation protocol construction. Currently the best-performing HEs are lattice-based schemes such as BGV [7], B/FV [6, 17], GSW [19], CKKS [10] and FHEW/TFHE [16, 11] where the security relies on the hardness of the Learning with Errors (LWE) problem [31] and its ring variant (RLWE) [29]. While most HE schemes support arithmetic operations over encrypted data such as addition and multiplication, the TFHE cryptosystem is notable for its differentiated functionality which supports an arbitrary Boolean gate evaluation over encrypted bits such as AND, OR, XOR and NAND. In addition to fast evaluation of Boolean circuits, there have been several follow-up studies to develop TFHE in various directions. For example, programmable bootstrapping (PBS) [8, 14, 21] enhances the computational capability of TFHE by encrypting multiple bits in a ciphertext and evaluating a look-up table (LUT) without extra complexity. Meanwhile, multi-key TFHE schemes [9, 26] can perform homomorphic computation on ciphertexts under different keys. This enables us to use the cryptosystem in more general scenarios when multiple parties aggregate and analyze their data without a trusted third party.

Due to its distinctive advantages, TFHE can be a useful toolkit for privacy-preserving machine learning where a client can obtain an inference result from a server's machine learning model without revealing its input data. For example, recent research demonstrates the efficiency of TFHE in secure inference of deep neural networks, decision trees and clustering algorithms [13, 28, 15].

* This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-TB2103-01.

1.1 Our Contribution.

One primary disadvantage of TFHE is that an expensive procedure called the *gate bootstrapping* is followed by each gate evaluation. In this work, we improve the efficiency of gate bootstrapping by modifying its major building blocks, called the *blind rotation* and the *key-switching*. We are motivated from the fact that most HE schemes achieve performance improvements by making additional assumptions on the secret key distribution (e.g. sparsity) but it remains unknown how this approach can be applied to TFHE. Hence, we aim to propose a new method of secret key generation that the TFHE scheme can take advantage of.

More specifically, the TFHE scheme has two secret keys for LWE and RLWE. We first propose a new secret distribution for LWE which outputs a sparse binary vector with a special 'block' structure. This property enables us to reduce the complexity of blind rotation substantially. We also modify the key generation procedure of RLWE so that the LWE secret is reused as part of the RLWE secret. As a result, we obtain better performance of key-switching in terms of both time and space complexity.

We provide a formal cryptanalysis on (R)LWE with block binary secrets to ensure the security of our construction. We consider the state-of-the-art lattice algorithms as well as custom-designed attacks, and provide parameter sets achieving a security level equivalent to the previous work [11].

We provide a proof-of-concept implementation of our construction based on the TFHE library [12] and demonstrate its concrete performance. Our experiments show that the latency of gate bootstrapping is reduced from $10.53ms$ down to $6.49ms$, yielding a speed-up of 1.48x. The size of the bootstrapping key is also reduced by about a half, from 109MB down to 60MB.

We remark that our optimization techniques can be naturally extended and bring major performance improvements to the line of research on TFHE [28, 13, 9, 26, 8, 14, 21]. Furthermore, this work may have a greater impact when combined with hardware accelerators since our method significantly reduces the number of polynomial operations which are the main bottleneck in hardware implementations [22, 34].

1.2 Technical Overview

The gate bootstrapping consists of two building blocks: blind rotation and key-switching. For the gate bootstrapping, firstly the blind rotation algorithm iteratively multiplies monomials to the *test vector*, which is the LUT in a form of the ring polynomial so that the constant term of the encryption becomes the value in the LUT corresponding to the message of the input ciphertext. In order to multiply monomials homomorphically to the test vector, we make use of the *external product* which is a homomorphic operation between TRLWE and ring GSW over torus (TRGSW) ciphertexts. By extracting the constant term of the resulting ciphertext of the blind rotation algorithm, we obtain a TLWE ciphertext under a vectorized version of the ring key. Finally, the key-switching algorithm switches the secret of this TLWE ciphertext back to the TLWE key, exploiting the gadget decomposition technique. We aim to reduce the number of these expensive operations such as external products and gadget operations. The technical overview of our improvements are given below.

Block Binary Distribution. First, we introduce a binary distribution with a special algebraic structure. We define the *block binary distribution* of parameter ℓ and k as the distribution on $n = \ell \cdot k$ dimensional binary vectors such that its instance is represented by the concatenation of k sub-vectors of dimension ℓ each of which has at most one nonzero component. We analyze the LWE problem with block binary secret and estimate its security level against the best-known attacks such as the meet-in-the-middle (MitM) algorithm. We conclude that, in terms of the attack complexity, the use of block binary secret distribution does not affect the hardness of LWE in the usual parameter setup of TFHE. Based on our analysis, we provide candidate parameter sets which satisfy both security and correctness conditions of our new bootstrapping method.

Blind Rotation. Next, we propose a faster blind rotation method based on the block binary distribution. Given a binary secret $\mathbf{s} = (s_0, \ldots, s_{n-1})$ and a ciphertext $\mathbf{c} = (b, \mathbf{a}) = (b, a_0, \ldots, a_{n-1}) \in \mathbb{T}^{n+1}$, let $\overline{b} = \lfloor 2Nb \rceil$ and $\overline{a}_i = \lfloor 2Na_i \rceil$ for $0 \leq i < n$. In the blind rotation, we compute $v \cdot X^{\overline{b}+\sum_{i=0}^{n-1} \overline{a}_i s_i} = v \cdot X^{\overline{b}} \cdot \prod_{i=0}^{n-1} \left[1 + s_i \cdot (X^{\overline{a}_i} - 1)\right]$ where v denotes the test vector. This polynomial can be evaluated with n external products, one per each iteration.

However, leveraging the block binary distribution allows us to reduce the number of the external products. We remark that if the secret key is sampled from the block binary distribution, we can rewrite the formula as $v \cdot X^{\bar{b}} \cdot \prod_{j=0}^{k-1} \left[1 + \sum_{i=j\ell}^{(j+1)\ell-1} s_i \cdot (X^{\bar{a}_i} - 1)\right]$. In a nutshell, we can process ℓ inputs simultaneously. Then, the blind rotation step can be represented using only k external products and linear combinations of TRGSW ciphertexts. Therefore, the overall time complexity decreases by a factor of ℓ compared to the previous work [11]. We also note that the structure of bootstrapping key remains unchanged, hence the size of the bootstrapping key does not increase as well.

Key-switching. We also present a new key-switching method with decreased key size and computational cost. This can be achieved by re-using the LWE key when generating the RLWE key. More precisely, given the binary LWE key $\mathbf{s} = (s_0, \ldots, s_{n-1})$, we set the RLWE key $t = s_0 + \cdots + s_{N-1}X^{N-1}$ where $s_n, \ldots, s_{N-1}$ are uniformly sampled bits. This simple yet novel idea allows us to skip the key-switching procedure for the shared part of the ring key. In consequences, we reduce the computational cost of the process by a factor of $\frac{N}{N-n}$. Not only that, this technique also reduces the noise bound and the size of the key-switching key. In addition, we introduce a new gadget decomposition for the key-switching operation which uses the concept of the balanced form [23]. This further reduces the size of key-switching key by half.

1.3 Related Work

There have been several studies for improving the performance of TFHE/FHEW style bootstrapping. Recently, Kim et al. [24] briefly presented an idea of using binary secrets of a small Hamming weight h in TFHE. The authors modified the gate bootstrapping and reduced the number of external products from n to h, however, this approach does not lead to a performance improvement since it requires a huge computational cost to generate h linear combinations of RGSW ciphertexts and the size of the bootstrapping key is increased by a factor of h.

Bonte et al. [5] and Kluczniak [25] concurrently presented a variant of TFHE enhanced performance of external product from the NTRU assumption. Lee et al. [27] improved the bootstrapping procedure of FHEW using homomorphic automorphisms in FHEW-style bootstrapping which enables compact public key size and faster FHEW-style blind rotation.

2 Background

2.1 Notations

The real torus $\mathbb{R}/\mathbb{Z}$ is written as $\mathbb{T}$, and $\mathbb{T}[X]/(X^N + 1)$ is denoted by $\mathbb{T}_N[X]$ where N is a power of two. We note that $\mathbb{T}$ and $\mathbb{T}_N[X]$ are not rings, but $\mathbb{Z}$-module and $\mathbb{Z}[X]$-module respectively. Thus inner product between $\mathbb{T}$ (or $\mathbb{T}_N[X]$)-vector and $\mathbb{Z}$ (or $\mathbb{Z}[X]$)-vector is well-defined as multiplication between $\mathbb{T}$ (or $\mathbb{T}_N[X]$) and $\mathbb{Z}$ (or $\mathbb{Z}[X]$) is well-defined. The set $\{0, 1\}$ is denoted by $\mathbb{B}$ and the set of polynomials whose degrees are less than N and coefficients are in $\mathbb{B}$ by $\mathbb{B}_N[X]$.

For a distribution $\mathcal{D}$, we denote by $x \leftarrow \mathcal{D}$ to represent a random sampling from $\mathcal{D}$ and $\mathbf{x} \leftarrow \mathcal{D}^N$ to represent a polynomial in $\mathbb{T}_N[X]$ of which coefficients are random samples obtained from $\mathcal{D}$. For a set S, we write the uniform distribution over S as $\mathcal{U}(S)$. A Gaussian distribution of mean 0 and variance α^2 is denoted by $\mathcal{D}_\alpha$. An infinity norm $\|\cdot\|_\infty$ over polynomials is defined as the maximum size of coefficients. When it is used for a vector or a matrix, it denotes the maximum value among the infinity norm of its components. We regard vectors as row matrices.

2.2 TLWE and TRLWE

In the following sections, we will describe the TFHE scheme. Its security relies on the hardness of the T(R)LWE problems. The T(R)LWE problem is a variant of (R)LWE defined over the real torus [11].

Definition 1 (Decisional TLWE). *Let n be an integer, $\alpha > 0$ a real number, and χ a distribution over $\mathbb{Z}^n$. For $\mathbf{s} \in \mathbb{Z}^n$, we define the TLWE distribution, denoted by $\mathcal{D}^{\mathsf{TLWE}}_{\chi,\alpha}(\mathbf{s})$, is a distribution over $\mathbb{T}^{n+1}$ obtained as follows: sample $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{T}^n)$ and $e \leftarrow \mathcal{D}_\alpha$, and output $(b, \mathbf{a})$ where $b = -\langle \mathbf{s}, \mathbf{a} \rangle + e \pmod 1$. The decisional TLWE calls for distinguishing $\mathcal{D}^{\mathsf{TLWE}}_{\chi,\alpha}(\mathbf{s})$ from $\mathcal{U}(\mathbb{T}^{n+1})$ for $\mathbf{s} \leftarrow \chi$.*

Definition 2 (Decisional TRLWE). *Let N be a power of two, $\beta > 0$ a real number, and ψ a distribution over $\mathbb{Z}[X]/(X^N+1)$. For $t \in \mathbb{Z}[X]/(X^N+1)$, we define the TLWE distribution, denoted by $\mathcal{D}^{\mathsf{TRLWE}}_{\psi,\beta}(t)$, is a distribution over $\mathbb{T}_N[X]^2$ obtained as follows: sample $a \leftarrow \mathcal{U}(\mathbb{T}_N[X])$ and $e \leftarrow \mathcal{D}^N_\beta$, and output (b,a) where $b = -t \cdot a + e \pmod 1$. The decisional TRLWE calls for distinguishing $\mathcal{D}^{\mathsf{TRLWE}}_{\psi,\beta}(t)$ from $\mathcal{U}(\mathbb{T}_N[X]^2)$ for $t \leftarrow \psi$.*

Throughout the paper, we use n and N to denote the dimensions of TLWE and TRLWE problems, respectively. We can construct the T(R)LWE cryptosystems upon the hardness of T(R)LWE problems. The TLWE ciphertexts and TRLWE ciphertexts are in the form of $(b, \mathbf{a}) \in \mathbb{T}^{n+1}$ and $(b,a) \in \mathbb{T}_N[X]^2$. We denote η_α and η_β an overwhelming probability upper bound of the distribution $\mathcal{D}_\alpha, \mathcal{D}_\beta$ respectively. We review the TFHE scheme below.

2.3 TRGSW and External product

We describe the RGSW scheme [19] over the torus (TRGSW) below. First, we define the notion of *gadget decomposition*. Let $\mathbf{g} \in \mathbb{T}^d$ be so-called *gadget vector*. Then we call a map $h : \mathbb{T}_N[X] \to \mathbb{Z}[X]^d$ a gadget decomposition if the following holds for some small constants ε, δ and for all $a \in \mathbb{T}_N[X]$:

- $\|\langle h(a), \mathbf{g}\rangle - a\|_\infty \le \varepsilon$
- $\|h(a)\|_\infty \le \delta$

An example of gadget decomposition is the digit decomposition. In the digit decomposition, a gadget vector is given as $\mathbf{g} = (B^{-1}, \dots, B^{-d})$ and the corresponding gadget decomposition is $h(a) = (\tilde{a}_1, \cdots \tilde{a}_d)$ where B is the base and $\tilde{a}_j$ is the polynomial whose coefficients are the j-th digit of the base-B representation of a satisfying $0 \le \|\tilde{a}_j\|_\infty < B$ for $1 \le j \le d$. Then, $\|a - \sum_{1\le j\le d} \tilde{a}_j \cdot B^{-j}\|_\infty \le B^{-d}$ holds, so we obtain $\varepsilon = B^{-d}$ and $\delta = B$.

Now we denote $\mathbf{G} = \begin{bmatrix} \mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \end{bmatrix} \in \mathbb{T}^{2d\times 2}$. Note that the gadget decomposition can be naturally extended to a tuple of ring elements. More precisely, $h : \mathbb{T}_N[X]^2 \mapsto \mathbb{T}_N[X]^{2d}$ is defined by $h(b,a) = (h(b), h(a))$ for $a, b \in \mathbb{T}_N[X]$. We now present the TRGSW scheme and the external product below.

• $\underline{\mathsf{TRGSW.Enc}(t,\mu)}$: Given the TRLWE secret key t and a message $\mu \in \mathbb{Z}_N[X]$, sample $\mathbf{a} \leftarrow \mathbb{T}_N[X]^{2d}$ and $\mathbf{e} \leftarrow \mathcal{D}^{2d}_\beta$, and let $\mathbf{b} = -\mathbf{a}\cdot t + \mathbf{e} \pmod 1$. Return the TRGSW ciphertext $\mathbf{C} = [\mathbf{b} \mid \mathbf{a}] + \mu \cdot \mathbf{G} \in \mathbb{T}_N[X]^{2d\times 2}$.

Definition 3 (External Product). *Let $\mathbf{c}$ be a TRLWE ciphertext and $\mathbf{C}$ be a TRGSW ciphertext. The external product, denoted by $\mathbf{c} \boxdot \mathbf{C}$ is defined as $(\mathbf{c}, \mathbf{C}) \mapsto \mathbf{c} \boxdot \mathbf{C} := h(\mathbf{c})^\top \cdot \mathbf{C}$.*

We first introduce some terminology of TLWE and TRLWE ciphertexts. We define the *phase* of $\mathbf{c}$ with respect to $\mathbf{s}$ as $\varphi_{\mathbf{s}}(b, \mathbf{a}) = b + \langle \mathbf{s}, \mathbf{a}\rangle \pmod 1$. Similarly, for a TRLWE ciphertext (b,a), we define the phase of (b,a) as $\varphi_t(b,a) = b + t \cdot a \pmod 1$. Now, for a TRGSW encryption $\mathbf{C}$ of μ, we have:

$$\begin{aligned}\varphi_t(\mathbf{c} \boxdot \mathbf{C}) &= \langle h(\mathbf{c}), \mathbf{C} \cdot (1,t)\rangle \\ &\approx \langle h(\mathbf{c}), \mu \cdot \mathbf{G} \cdot (1,t)\rangle \approx \mu \cdot \varphi_t(\mathbf{c}) \pmod 1.\end{aligned}$$

2.4 Key-switching

Key-switching is one of the major building blokcs of TFHE. It can be used to convert a TLWE ciphertext under a secret $\mathbf{t} = (t_0, \dots, t_{N-1})$ into another ciphertext under a different key $\mathbf{s} = (s_0, \dots, s_{n-1})$ while almost preserving its phase. This process requires a key-swiching key which can be viewed as special encryptions of t_i under $\mathbf{s}$. In the TFHE scheme, we make use of key-switching algorithm to switch the TRLWE key into the TLWE key.

• $\underline{\mathsf{SwitchKeyGen}(\mathbf{s}, \mathbf{t})}$: Given TLWE secrets $\mathbf{s} = (s_0, \dots, s_{n-1})$ and $\mathbf{t} = (t_0, \dots, t_{N-1})$, generate the key-switching key as follows:

- Sample $\mathbf{a}_{i,j}[k] \leftarrow \mathbb{T}^n$ and $e_{i,j}[k] \leftarrow \mathcal{D}_\alpha$ for $0 \le i < N, 0 \le j < f, 0 \le k < D$, and set $\mathsf{KSK}_{i,j}[k] = [b_{i,j}[k], \mathbf{a}_{i,j}[k]]$ where $b_{i,j}[k] = -\langle \mathbf{a}_{i,j}[k], \mathbf{s}\rangle + k \cdot D^{-j-1} \cdot t_i + e_{i,j}[k] \pmod 1$.
- Return $\mathsf{KSK} = \{\mathsf{KSK}_{i,j}[k] \mid 0 \le i < N, 0 \le j < f, 0 \le k < D\}$.

• $\underline{\mathsf{KeySwitch}(\mathsf{KSK}, \mathbf{c})}$: Given a TLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{N+1}$, let $a_{i,j} \in \mathbb{Z} \cap [0, D)$ be the integers such that $|a_i - \sum_{j=0}^{f-1} a_{i,j} \cdot D^{-j-1}| \le D^{-f}$. Return $(b, 0, \ldots, 0) + \sum_{i=0}^{N-1} \sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{i,j}] \in \mathbb{T}^{n+1}$.

Note that $a_{i,j}$ is the j-th digit after the decimal point of a base-D representation of a_i, *i.e.*, $a_i = a_{i,1} \cdot D^{-1} + \cdots + a_{i,f} \cdot D^{-f} + \ldots$ for $0 \le i < N$.

The correctness of key-switching can be shown as follows. For $i = 0, \ldots, N-1$, we have

$$\begin{aligned}\varphi_{\mathbf{s}}\left(\sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{i,j}]\right) &= \sum_{j=0}^{f-1} \varphi_{\mathbf{s}}(\mathsf{KSK}_{i,j}[a_{i,j}]) \\ &\approx \sum_{j=0}^{f-1} (a_{i,j} \cdot D^{-j-1} \cdot t_i) \approx a_i \cdot t_i \pmod 1.\end{aligned}$$

Then, the phase of output ciphertext under $\mathbf{s}$ is approximately $b + \sum_{i=0}^{N-1} a_i t_i$, as desired.

2.5 The TFHE Scheme

In this subsection, we briefly review the TFHE scheme [11]. We remark that we represent a bit m as an element of $\{\pm 1\}$ instead of $\{0, 1\}$.

• $\underline{\mathsf{Setup}(1^\lambda)}$: Take the security parameter λ as input and generate the following parameters: TLWE dimension n, TRLWE dimension N, error parameters $\alpha, \beta > 0$, the gadget decomposition base and dimension B and d, and the key-switching base D.

• $\underline{\mathsf{KeyGen}(1^\lambda)}$:

- Sample $s_i \leftarrow \mathcal{U}(\mathbb{B})$ for $0 \le i < n$. Return the TLWE key $\mathbf{s} = (s_0, \ldots, s_{n-1})$.
- Sample $t_i \leftarrow \mathcal{U}(\mathbb{B})$. Return the TRLWE key $t = t_0 + \cdots + t_{N-1} X^{N-1}$. Write $\mathbf{t} = (t_0, \ldots, t_{N-1})$.
- Set $\mathsf{BRK}_i \leftarrow \mathsf{TRGSW.Enc}(t, s_i)$ for $0 \le i < n$. Return the blind rotation key $\mathsf{BRK} = \{\mathsf{BRK}_i\}_{0 \le i < n}$.
- $\mathsf{KSK} \leftarrow \mathsf{SwitchKeyGen}(\mathbf{s}, \mathbf{t})$ and return KSK.

• $\underline{\mathsf{Enc}(\mathbf{s}, m)}$: Given the secret key $\mathbf{s} = (s_0, \ldots, s_{n-1})$ and a message $m \in \{\pm 1\}$, sample $\mathbf{a} = (a_0, \ldots, a_{n-1}) \leftarrow \mathcal{U}(\mathbb{T}^n)$ and $e \leftarrow \mathcal{D}_\alpha$. Return $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$ where $b = -\langle \mathbf{s}, \mathbf{a}\rangle + \frac{1}{8} m + e \pmod 1$.

• $\underline{\mathsf{Dec}(\mathbf{s}, \mathbf{c})}$: Given the secret key $\mathbf{s}$ and a ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$, let $\mu = b + \langle \mathbf{s}, \mathbf{a}\rangle \pmod 1$. Output 1 if $\mu > 0$, 0 otherwise.

• $\underline{\mathsf{Boot}(\mathsf{BRK}, \mathsf{KSK}, \mathbf{c})}$: Given the blind rotation key BRK, the key-switching key KSK and a TLWE ciphertext $\mathbf{c} \in \mathbb{T}^{n+1}$, run Alg. 1 and output the ciphertext $\mathbf{c}' \in \mathbb{T}^{n+1}$.

• $\underline{\mathsf{HomNAND}(\mathsf{BRK}, \mathsf{KSK}, \mathbf{c}_1, \mathbf{c}_2)}$:Given the blind rotation key BRK, the key-switching key KSK and TLWE ciphertexts $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{T}^{n+1}$, return $\mathsf{Boot}(\mathsf{BRK}, \mathsf{KSK}, (\frac{1}{8}, \mathbf{0}) - \mathbf{c}_1 - \mathbf{c}_2) \in \mathbb{T}^{n+1}$.

We briefly explain the correctness of the homomorphic NAND operation as follows. Suppose that $\mathbf{c}_1$ and $\mathbf{c}_2$ are TLWE ciphertexts such that $\varphi_{\mathbf{s}}(\mathbf{c}_i) = \frac{1}{8} m_i + e_i \pmod 1$ for some small e_i. Then, it is easy to show that $\mathbf{c} := (\frac{1}{8}, \mathbf{0}) - \mathbf{c}_1 - \mathbf{c}_2$ satisfies that $\varphi_{\mathbf{s}}(\mathbf{c}) = \frac{1}{4} m + e \pmod 1$ for the message $m = m_1 \overline{\wedge} m_2$ and the error $|e| \le \frac{1}{8} + |e_1| + |e_2|$.

Then, it is followed by the bootstrapping procedure (Alg. 1), which consists of three steps called blind rotation (Lines 1-6), sample extraction (Lines 7-8) and key-switching (Line 9) and outputs a TLWE ciphertext $\mathbf{c}' \leftarrow \mathsf{Boot}(\mathsf{BRK}, \mathsf{KSK}, \mathbf{c})$ such that $\varphi_{\mathbf{s}}(\mathbf{c}') = \frac{1}{8} m + e' \pmod 1$ for some small e'. Below we give a brief overview on these procedures.

Algorithm 1 TFHE Bootstrapping

Input: The blind rotation key BRK, the key-switching key KSK and a TLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) = (b, a_0, \ldots, a_{n-1}) \in \mathbb{T}^{n+1}$

Output: A TLWE ciphertext $\mathbf{c}'' \in \mathbb{T}^{n+1}$

1: $v := -\frac{1}{8} \cdot (1 + X + \cdots + X^{N-1}) \in \mathbb{T}_N[X]$
2: Let $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a}_i = \lfloor 2Na_i \rceil$ for $0 \le i < n$
3: $\mathsf{ACC} \leftarrow (X^{\bar{b}} \cdot v, 0) \in \mathbb{T}_N[X]^2$
4: **for** $0 \le j < n$ **do**
5: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + \left[(X^{\bar{a}_j} - 1) \cdot \mathsf{ACC}\right] \boxdot \mathsf{BRK}_j$
6: **end for**
7: Parse ACC as $\left(\sum_{i=0}^{N-1} b'_i X^i, \sum_{i=0}^{N-1} a'_i X^i\right)$
8: $\mathbf{a}' \leftarrow (a'_0, -a'_{N-1}, \ldots, -a'_1)$
9: $\mathbf{c}' \leftarrow \mathsf{KeySwitch}(\mathsf{KSK}, (b'_0, \mathbf{a}'))$

In the blind rotation step, we homomorphically compute $X^{\langle \bar{\mathbf{c}}, (1,\mathbf{s}) \rangle} \cdot v \in \mathbb{T}_N[X]$ for the scaled ciphertext $\bar{\mathbf{c}} = \lfloor 2N \cdot \mathbf{c} \rceil \in \mathbb{Z}_{2N}^{n+1}$ where v is a fixed polynomial called the *test vector*. We initialize the coefficients of the test vector as $v = -\frac{1}{8} \cdot (1 + X + \cdots + X^{N-1})$ so that the constant term of $X^{\langle \bar{\mathbf{c}}, (1,\mathbf{s}) \rangle} \cdot v$ is $\frac{1}{8}$ if $\langle \bar{\mathbf{c}}, (1, \mathbf{s}) \rangle > 0$, or $-\frac{1}{8}$ otherwise. In the sample extraction step, we convert the TRLWE ciphertext obtained from the blind rotation step to a TLWE ciphertext whose phase is approximately $\frac{1}{8}m$. Lastly, the key-switching procedure switches the TLWE secret from $\mathbf{t}$ into $\mathbf{s}$ and returns the ciphertext $\mathbf{c}'$ such that $\varphi_{\mathbf{s}}(\mathbf{c}') \approx \frac{1}{8}m \pmod 1$, as desired.

Finally, we note that every Boolean gate can be instantiated in a similar manner. We refer the reader to [11] for further details.

3 Block Binary Distribution

In this section, we introduce a key distribution for TLWE and provide a formal cryptanalysis to estimate its concrete security against state-of-the-art attacks. In one aspect, the bottleneck hindering the performance of TFHE bootstrapping is the blind rotation that involves n iterative external products. We observe that multiple external products can be consolidated into a single operation if the corresponding secret components have a Hamming weight of at most one (more details will be explained in the next section). Hence, we propose a new key distribution, called the block binary distribution, to improve the performance of TFHE bootstrapping.

Block Binary Distribution. Let ℓ and k be positive integers. We first define $\mathbf{B}_\ell$ as a distribution on $\mathbb{Z}^\ell$ which samples one of the standard unit vectors in $\mathbb{Z}^\ell$ or the zero vector with the equal probability. In other words, it returns one of the $(\ell+1)$ vectors $(0, 0, \ldots, 0), (1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1) \in \mathbb{B}^\ell$ with probability $\frac{1}{\ell+1}$ each. We then extend $\mathbf{B}_\ell$ and define the block binary distribution over $\mathbb{Z}^n$ for $n = \ell \cdot k$ by concatenating k instances of $\mathbf{B}_\ell$. That is, we denote by $\mathbf{B}_{\ell,k}$ the distribution over $\mathbb{Z}^n$ which samples $\mathbf{a}_1, \ldots, \mathbf{a}_k$ independently from $\mathbf{B}_\ell$ and returns $(\mathbf{a}_0, \ldots, \mathbf{a}_{k-1}) \in \mathbb{Z}^n$.

Note that a vector from the block binary distribution has at most one nonzero component in each block of length ℓ. In addition, the block binary distribution can be regarded as a generalization of a uniform binary distribution since $\mathbf{B}_{\ell,k}$ is identical to $\mathcal{U}(\mathbb{B}^n)$ when $\ell = 1$.

There has been a series of work studying the hardness of LWE with various key distributions. For example, Goldwasser *et al.* [20] showed that the LWE problem with binary secrets is secure. Furthermore, they stated that the security proof holds for LWE with an arbitrary binary key distribution with a sufficiently large min-entropy.

Theorem 1 ([20]). *Let $n, q \ge 1$ be integers, let $\mathcal{D}$ be any distribution over $\{0,1\}^n$ having min-entropy at least k, and let $\alpha, \beta > 0$ be such that $\alpha/\beta = \mathsf{negl}(n)$. Then for any $\ell \le \frac{k - \omega(\log n)}{\log q}$, there is a PPT reduction from $DLWE_{\ell,q,\alpha}(\mathcal{U}(\mathbb{Z}_q^\ell))$ to $DLWE_{n,q,\beta}(\mathcal{D})$.*

From this perspective, the hardness of LWE with a block binary distribution can be guaranteed if its min-entropy is larger than what is required in the proof.

On the other hand, the security level is estimated by concrete attacks rather than security reduction to choose optimal parameter sets in practice. Indeed, the existing schemes based on the binary secret LWE problem even consider tailor-made attacks [1, 2, 4, 32] to ensure that they achieve a desired security level (e.g. 128-bit). We also follow the same approach to provide a secure parameter set.

3.1 Security Analysis

To justify whether our parameter regime for the LWE problem with block binary secret is appropriate or not, we basically intend to measure the security level using a well-known lattice estimator [1]. Although the lattice estimator considers most of the known lattice algorithms to date, it may be insufficient to say that valid security levels have been measured, given that it does not actively take advantage of the distribution characteristics we use. To overcome this, we introduce how to utilize the specificity of the secret to previously known attacks. We then present a lattice estimator to measure the security level that takes into account this approach. Moreover, we also consider the recent combinatorial attack algorithm proposed by May [30], which is not involved in lattice estimator as it is highly effective when the secret distribution size is small. Lastly, we provide block binary specific attacks.

Analysis Setup We first clarify the security goal corresponding our scheme. When the following (R)LWE samples are given to the attacker

$$(b_i, \mathbf{a}_i) \leftarrow \mathcal{D}^{\mathsf{TLWE}}_{\mathbf{B}_{\ell,k},\alpha}(\mathbf{s}) \text{ for } \mathbf{s} \leftarrow \mathbf{B}_{\ell,k}$$
$$(b_i, a_i) \leftarrow \mathcal{D}^{\mathsf{TRLWE}}_{\psi,\beta}(t) \text{ for } t \leftarrow \mathbf{B}_{\ell,k} \times \mathcal{U}(\mathbb{B}^{N-n}),$$

We intend to count the time complexity of algorithms that distinguish these pairs from random pairs. We consider each problem as an independent LWE problem and RLWE problem and have measured the security level using a (modified) lattice estimator, respectively. The whole security level of our scheme is then estimated as a lower value of them.

To determine the parameters, we fix the α and β as 2^{-15} and 2^{-25}, respectively. To apply the known algorithms, we also regard the TLWE samples (resp. TRLWE) as usual LWE samples over a discrete modulus space $\mathbb{Z}_q$ by scaling up $1/\alpha^2$ (resp. $1/\beta^2$).

We now briefly describe algorithms that leverage block secret distributions in this setting.

Hybrid lattice reduction algorithm Currently, a lattice based algorithm such as primal and dual attacks are known to be competitive for solving the LWE problem. Since these attacks are very sensitive to the secret dimension, such a significant dimension reduction allows the attacker to maximize the effect of the existing attacks. We explain how both algorithms can be improved for the block binary secret (R)LWE. For simplicity, we only describe dual attacks on LWE samples, but the same strategy applies to primal attacks.

By concatenating, the given LWE samples, it can be written as a matrix form: $\mathbf{b} = -\mathbf{A}\mathbf{s} + \mathbf{e}$. Writing $\mathbf{s} = \begin{bmatrix}\mathbf{s}_0\\ \mathbf{s}_1\end{bmatrix}$ and $\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1]$ with $\mathbf{s}_0 \in \mathbb{Z}_q^{n-k}$, $\mathbf{s}_1 \in \mathbb{Z}_q^k$ $\mathbf{A}_0 \in \mathbb{Z}_q^{m\times(n-k)}$, and $\mathbf{A}_1 \in \mathbb{Z}_q^{m\times k}$, it holds that

$$\mathbf{b} + \mathbf{A}_0\mathbf{s}_0 = -\mathbf{A}_1\mathbf{s}_1 + \mathbf{e}.$$

Then, the dual lattice attack finds a short vector $(\mathbf{x}, \mathbf{y})$ in the lattice

$$\mathcal{L} = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^k \mid \mathbf{x}^\top \mathbf{A}_1 = \mathbf{y}^\top \pmod q\},$$

which gives an $(n-k)$ dimensional LWE sample such that $\mathbf{x}^\top(\mathbf{b} + \mathbf{A}_0\mathbf{s}_0) = -\langle \mathbf{y}, \mathbf{s}_1\rangle + \langle \mathbf{x}, \mathbf{e}\rangle := e'$. Applying the combinatorial approach such as an exhaustive search or meet-in-the-middle algorithm, one can recover the secret $\mathbf{s}_0$.

Since the current combinatorial part of the lattice estimator is designed to count against the binary distribution, we consider two different "guess" algorithms in the hybrid lattice attack to measure the

security level of LWE with block binary secrets more precisely. The original hybrid algorithm considered reducing the dimension by taking a zero position. In the case of block binary distribution, through one guessing instead of zero, more dimension (exactly ℓ) can be reduced compared to a normal binary distribution. Let $\mathbf{s}_i \in \mathbb{B}^\ell$ be the i-th sub-vector of $\mathbf{s}$ with regard to each block. By the setup, each $\mathbf{s}_i$ equals to one of the $(\ell+1)$ candidates - a zero vector or ℓ different vectors with Hamming weight 1. It implies that one can recover ℓ-dimensions of the secret vector via guessing $(\ell+1)$-cases. This modification has the potential to make the dual hybrid attack more competitive. While guessing one in the block might appear to be a more effective attack, we will demonstrate in the following paragraph that the block binary specific guessing is less successful for a broader range of large parameter sets. Consequently, this guessing strategy is only useful in reducing small dimensions.

Meet-in-the-Middle Algorithm Recently, May [30] shows that a standard MitM algorithm for LWE problems can be improved. Because the attack heavily depends on the size of secret space and the LWE with block binary distribution only has fewer candidates for the secret key compared to the entire dimension, the advanced MitM may be the most effective for the LWE problem.

A main strategy of the standard MitM algorithm is that: split the secret vector $\mathbf{s}$ into $\mathbf{s} = \mathbf{s}_0 + \mathbf{s}_1$ with $\mathbf{s}_0 \in \{0,1\}^n$ and $\mathbf{s}_1 \in \{0,1\}^n$ such that $\|\mathbf{s}_0\|_1$ and $\|\mathbf{s}_1\|_1$ are (approximately) equal to the half of $\|\mathbf{s}\|_1$. Since $b_i + \langle \mathbf{s}, \mathbf{a}_i \rangle$ is small, $b_i + \langle \mathbf{s}_0, \mathbf{a}_i \rangle$ is approximately equal to $-\langle \mathbf{s}_1, \mathbf{a}_i \rangle$ for $\mathbf{s} = \mathbf{s}_0 + \mathbf{s}_1$. The algorithm thus builds up two sets;

$$\mathcal{R}_0 = \{b_i + \langle \mathbf{x}_0, \mathbf{a}_i \rangle \mid \mathbf{x}_0 \in \mathbb{B}^n \wedge \|\mathbf{x}_0\|_1 = \|\mathbf{s}\|_1/2\}$$
$$\mathcal{R}_1 = \{-\langle \mathbf{x}_1, \mathbf{a}_i \rangle \mid \mathbf{x}_1 \in \mathbb{B}^n \wedge \|\mathbf{x}_1\|_1 = \|\mathbf{s}\|_1/2\}.$$

One may expect that a collision from these sets happens with a correct pair $(\mathbf{s}_0, \mathbf{s}_1)$. Namely, this collision enables to recover the secret $\mathbf{s}$ as $\mathbf{s}_0 + \mathbf{s}_1$. With this idea, the standard MitM can recover the secret vector $\mathbf{s}$ in time $\mathcal{S}^{0.5}$ where $\mathcal{S}$ is the cardinality of key space for $\mathbf{s}$.

A key idea to improve the former MitM thanks to May [30] is to split both target vectors $\mathbf{s}_0$ and $\mathbf{s}_1$ inductively. To be precise, both vectors are divided into $\mathbf{s}_i = \mathbf{s}_{i,0} + \mathbf{s}_{i,1}$ with a vector $\mathbf{s}_{i,j}$ of fewer Hamming weights. At a high level, this can be viewed as a generalization of tree-based search, and secret vectors are further decomposed repeatedly until they have sufficiently small Hamming weights and can be found easily.

Based on the improved MitM algorithms, the secret vector $\mathbf{s}$ can be obtained in time (asymptotically) $\mathcal{S}^{0.25}$, however, this asymptotic result ignores the guessing part. According to the paper [30], when the algorithm is applied to the concrete instantiation such as NTRU, BLISS, and GLP, the overall cost including the guessing part is in the range $[\mathcal{S}^{0.28}, \mathcal{S}^{0.3}]$. We therefore estimate the time complexity of MitM algorithm as $\mathcal{S}^{0.28}$. Plugging a fact that the size of key space $\mathcal{S}$ for the block binary distribution $\mathbf{B}_{\ell,k}$ is equal to $(\ell+1)^k$ into the complexity, one can recover the secret in time $O(2^{0.28k\log(\ell+1)})$.

Block binary specific analysis. One property of the block binary distribution is that each block has a Hamming weight at most 1. Thus, there are two ways to reduce the dimension of the secret vector in the block binary LWE problem. The first one is to use the block structure and guess the location of the 1 in each block, which reduces ℓ dimensions with probability $1/(\ell+1)$. By repeating this process k times, we can reduce $\ell \cdot k$ dimensions at a cost of $(\ell+1)^k = 2^{O(k\log(\ell+1))}$.

Another way is to treat the given distribution as a binary distribution with low Hamming weights, and reduce one dimension with probability $1 - 1/\ell$. By guessing several zeros, we can reduce $\ell \cdot k$ dimensions at a cost of $1/\big((1-1/\ell)^{\ell\cdot k}\big)$. It can be (asymptotically) represented by $2^{O(k)}$ applying the equality $\lim_{\ell\to\infty}(1-1/\ell)^\ell = 1/\mathfrak{e}$, where $\mathfrak{e}$ is an Euler constant. This method is more efficient than the first one, while it does not utilize the block structure. Nevertheless, we consider both algorithms for concrete cryptanalysis.

As another block binary specific attack, suppose that we are given an LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$ satisfying $\mathbf{b} + \mathbf{A}\mathbf{e} = \mathbf{s}$, which can be obtained by converting the usual LWE samples. Here the secret key $\mathbf{s}$ plays a role of "noise vector" of LWE. Our goal is then to recover $\mathbf{e}$ by eliminating the noise vector. To

do this, we consider the multivariate polynomials F_i and $G_{i,j}$ defined as follows:

$$F_i(\mathbf{x}) = (b_i + \langle \mathbf{a}_i, \mathbf{x} \rangle) \cdot (b_i + \langle \mathbf{a}_i, \mathbf{x} \rangle - 1)$$
$$G_{i,j}(\mathbf{x}) = (b_i + \langle \mathbf{a}_i, \mathbf{x} \rangle) \cdot (b_j + \langle \mathbf{a}_j, \mathbf{x} \rangle),$$

where $\mathbf{a}_i$ (resp. b_i) is the i-th row of $\mathbf{A}$ (resp. $\mathbf{b}$). Note that each $F_i(\mathbf{e})$ is equal to zero since s_i is binary. In addition, if the indices $i \neq j$ are in the same block, then either s_i or s_j is zero and consequently $G_{i,j}(\mathbf{e}) = 0$. Thus, this polynomial system has a root $\mathbf{e}$, which we call a "noise-free polynomial".

Sun, Tibouchi, and Abe [33] presented a refined result on the hardness of binary error LWE showing that the common root can be found in a subexponential time $2^{\tilde{O}(n^{1-\alpha})}$, given $n^{1+\alpha}$-algebraic independent noise-free polynomials. For our purposes, we assume that all given polynomials $\{F_i, G_{i,j}\}$ are algebraic independent. Note that there are n polynomials F_i, and there are $\binom{\ell}{2} \approx \frac{\ell^2}{2}$ pairs (i,j) of distinct indices in each block. In total, we have approximately $\frac{n \cdot \ell}{2}$ noise-free polynomials for $G_{i,j}$. Putting it together, the total number of noise-free polynomials is approximately $\frac{\ell+2}{2} \cdot n$, which we denote by $n^{1+\alpha}$. From [33], the common root $\mathbf{e}$ can be found in a subexponential time $2^{\tilde{O}(n^{1-\alpha})} = 2^{\tilde{O}(\frac{2n}{\ell+2})}$.

3.2 Parameter setup

We provide our concrete parameter setup and their expected security level. For comparison, we consider 1) Esti: the original lattice estimator [3], 2) Modified Est: the modified lattice estimator and 3) MitM: May's advanced MitM algorithm.

To determine the parameters, we fix the α as 2^{-15}. We next compute the minimized n such that $2^{0.28k\log(\ell+1)} \approx 2^{128}$ at each $\ell \geq 2$. Even though the constraint holds, if the concrete parameter setup does not achieve the desired security level against a modified lattice attack, we raise n up until the security level measured from the guessing algorithm is around 128.

Table 1. Recommended Parameter Sets and Security Estimates. The error parameters of TLWE and TRLWE are fixed by $\alpha = 2^{-15}$ and $\beta = 2^{-25}$, respectively.

n	N	ℓ	Esti	Modified Est	MitM
630	1024	2	128.8	128.8	139.793
687	1024	3	128.3	126.7	128.24
788	1024	4	128.6	127.4	128.078
885	1024	5	127.8	126.2	128.111
978	1024	6	127.2	125.4	128.128

4 A Faster Bootstrapping of TFHE

In this section, we modify the bootstrapping algorithm of TFHE to obtain better efficiency. We first recall that there are two main building blocks in the bootstrapping algorithm, namely blind rotation and key-switching. Both of two operations require a considerably large number of polynomial arithmetic, bit operations and scalar additions. We achieve a better performance by reducing the number of these operations.

Firstly, we reduce the number of external products in the blind rotation. Our improvement is mostly based on the observation that the number of iterations in the blind rotation can be reduced if the TLWE key is sampled according to the block binary distribution. More precisely, we compress iterations of the previous blind rotation corresponding to each block of the secret key so that it involves only one external

product operation. In consequences, we can reduce the number of the external products from n, the length of the ciphertext to k, the number of the blocks.

Secondly, we improve the key-switching in terms of both the key size and the speed of key-switching procedure by re-using the TLWE key when generating the TRLWE secret. More precisely, given a TLWE key $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{B}^n$, we randomly sample $s_n, \ldots, s_{N-1}$ from $\mathbb{B}$ and set the ring key as $t = s_0 + \cdots + s_{N-1}X^{N-1} \in \mathbb{B}_N[X]$. Then, we can omit generating key-switching key components for the shared part, which reduces the size of the key-switching key by a factor of $\frac{N}{N-n}$ compared to the previous construction. Furthermore, we can skip the switching operations for the shared part, which also reduces computational cost.

We will elaborate on these techniques in the following sections, and then describe the full scheme in Sec 4.3.

4.1 Blind Rotation

We first present a new blind rotation method with better complexity leveraging the block binary distribution. Let us first briefly review the functionality the blind rotation algorithm. Let $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$ be a TLWE ciphertext under secret $\mathbf{s} \in \mathbb{B}^n$ and $v = v(X)$ be a test vector. Recall that the blind rotation procedure aims to generate an RLWE encryption of $v \cdot X^{\varphi_{\mathbf{s}}(\bar{b},\bar{\mathbf{a}})}$ where $\varphi_{\mathbf{s}}(\bar{b}, \bar{\mathbf{a}}) = \bar{b} + \langle \mathbf{s}, \bar{\mathbf{a}} \rangle \pmod{2N}$ is the phase of $(\bar{b}, \bar{\mathbf{a}}) = \lfloor 2N \cdot (b, \mathbf{a}) \rceil$ with respect to $\mathbf{s}$. In the original scheme, this computation is done by initializing the accumulator ACC as the trivial TRLWE encryption of $X^{\bar{b}} \cdot v$, then homomorphically multiplying $X^{\bar{a}_0 \cdot s_0}, \ldots, X^{\bar{a}_{n-1} \cdot s_{n-1}}$ recursively. In particular, each multiplication is expressed as one external product with BRK_i from the equation $X^{\bar{a}_i \cdot s_i} = 1 + (X^{\bar{a}_i} - 1) \cdot s_i$.

Now let us suppose that the TLWE key $\mathbf{s} = (s_0, \ldots, s_{n-1})$ is sampled according to $\mathbf{B}_{\ell,k}$ so that there is at most one nonzero component in each block $(s_i)_{i \in I_j}$ where $I_j = \{j\ell, j\ell + 1, \ldots, j(\ell+1) - 1\}$ for $0 \le j < k$. Then, it holds that

$$X^{\sum_{i \in I_j} \bar{a}_i \cdot s_i} = 1 + \sum_{i \in I_j} (X^{\bar{a}_i} - 1) \cdot s_i$$

since both sides of the equation equal $X^{\bar{a}_i}$ if there exists $i \in I_j$ such that $s_i = 1$; otherwise they are equal to 1 when $s_i = 0$ for all $i \in I_j$.

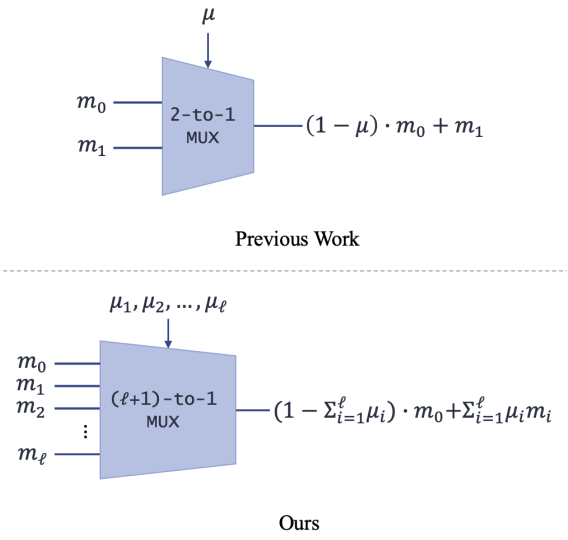

Fig. 1. Generalized Multiplexer Gate

Algorithm 2 NewBlindRotate

Input: The blind rotation key BRK and a TLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$
Output: A TRLWE ciphertext $\mathsf{ACC} \in \mathbb{T}_N[X]^2$
1: $v \leftarrow -\frac{1}{8} \cdot (1 + X + \cdots + X^{N-1}) \in \mathbb{T}_N[X]$
2: Let $\overline{b} = \lfloor 2Nb \rceil$ and $\overline{a}_i = \lfloor 2Na_i \rceil$ for $0 \le i < n$
3: $\mathsf{ACC} \leftarrow (X^{\overline{b}} \cdot v, 0) \in \mathbb{T}_N[X]^2$
4: **for** $0 \le j < k$ **do**
5: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{ACC} \boxdot \left[\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot \mathsf{BRK}_i\right]$
6: **end for**
7: **return** ACC

Our algorithm is derived from the previous blind rotation by replacing the accumulator with k iterations which homomorphically multiply $X^{\sum_{i \in I_j} \overline{a}_i \cdot s_i}$ to ACC for $0 \le j < k$ using the equation above. The new algorithm takes k external products, n scalar multiplications and additions on TRGSW ciphertexts, while the original blind rotation requires n external products. The overall performance improvement of our blind rotation is due to the reduction of external products and FFT conversions for gadget decomposition. More discussions on optimization and implementation techniques will be given in Sec. 5.1.

Lemma 1 (New Blind Rotation). *Given a* TLWE *ciphertext* $(b, \mathbf{a}) \in \mathbb{T}^{n+1}$ *under secret* $\mathbf{s}$ *sampled from* $\mathbf{B}_{\ell,k}$*, let* $\overline{b} = \lfloor 2Nb \rceil$*,* $\overline{a}_i = \lfloor 2Na_i \rceil$ *for* $0 \le i < n$ *and* $\varphi_{\mathbf{s}}(\overline{b}, \overline{\mathbf{a}}) = \overline{b} + \langle \mathbf{s}, \overline{\mathbf{a}} \rangle \pmod{2N}$*. Then, the output of Alg. 2 is an TRLWE ciphertext* $(b', a') \in \mathbb{T}_N[X]^2$ *whose phase is* $X^{\overline{b} + \sum_{0 \le i < n} \overline{a}_i \cdot s_i} \cdot v + e \pmod 1$ *for some* $\|e\|_\infty \le 2k(1+N)\varepsilon + 4\delta\eta_\beta ndN$*.*

Proof. Let ACC_j be the TRLWE ciphertext obtained from the j-th iteration of the accumulator. In other words, $\mathsf{ACC}_0 = (X^{\overline{b}} \cdot v, 0)$ and $\mathsf{ACC}_{j+1} = \mathsf{ACC}_j + \mathsf{ACC}_j \boxdot \left[\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot \mathsf{BRK}_i\right]$ for $0 \le j < k$. We also denote by $\varepsilon_j = \langle h(\mathsf{ACC}_j) \cdot \mathbf{g} - \mathsf{ACC}_j, (1, t) \rangle$. Note that $\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot \mathsf{BRK}_i$ is a TRGSW encryption of $m_j := \sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot s_i = \begin{cases} X^{\overline{a}_i} - 1, & \text{if } s_i = 1 \text{ for some } i \in I_j \\ 0, & \text{otherwise } s_i = 0 \text{ for all } i \in I_j \end{cases}$, and its noise term will be denoted by $\mathbf{e}_j$.

Then, we obtain the following recurrence relation:

$$\begin{aligned} \varphi_t(\mathsf{ACC}_{j+1}) &= \varphi_t(\mathsf{ACC}_j) + \varphi_t\Big(\mathsf{ACC}_j \boxdot \Big[\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot \mathsf{BRK}_i\Big]\Big) \\ &= (m_j + 1) \cdot \varphi_t(\mathsf{ACC}_j) + m_j \cdot \varepsilon_j + \langle h(\mathsf{ACC}_j), \mathbf{e}_j \rangle \\ &= X^{\sum_{i \in I_j} \overline{a}_i s_i} \cdot \varphi_t(\mathsf{ACC}_j) + \mathsf{Err}_j \pmod 1 \end{aligned}$$

where $\mathsf{Err}_j = m_j \cdot \varepsilon_j + \langle h(\mathsf{ACC}_j), \mathbf{e}_j \rangle$.

By combining the above equations for $0 \le j < k$, we obtain

$$\begin{aligned} \varphi_t(\mathsf{ACC}_k) &= X^{\sum_{i=0}^{n-1} \overline{a}_i s_i} \cdot \varphi_t(\mathsf{ACC}_0) + e \pmod 1 \\ &= X^{\overline{b} + \sum_{i=0}^{n-1} \overline{a}_i s_i} \cdot v + e \pmod 1 \end{aligned}$$

for some $e \in \mathbb{R}[X]$ which is bounded by $\|e\|_\infty \le \sum_{j=0}^{k-1} \|\mathsf{Err}_j\|_\infty \le 2k(1+N)\varepsilon + 4\delta\eta_\beta ndN$.

One disadvantage of our blind rotation is that its error bound is slightly larger than that of the prior method. In practical terms, it is not much of an issue since the total bootstrapping noise heavily depends on the key-switching error in the usual parameter setting of TFHE. Nevertheless, if we need to minimize the noise growth under certain circumstance, then this issue can be addressed simply by generating one

more bootstrapping key for each block which indicates where secret components in the interval I_j are all zero or not. Specifically, we define $s'_j = 1 - \sum_{i \in I_j} s_i$ and obtain

$$1 + \sum_{i \in I_j} s_i (X^{\overline{a}_i} - 1) = s'_j + \sum_{i \in I_j} s_i X^{\overline{a}_i}.$$

Then, each iteration of the blind rotation can be written as $\mathsf{ACC} \leftarrow \mathsf{ACC} \boxdot \left[\mathsf{BRK}'_j + \sum_{i \in I_j} X^{\overline{a}_i} \cdot \mathsf{BRK}_i\right]$ where BRK'_j is a TRGSW encryption of s'_j under t.

We remark that our idea is naturally applicable to homomorphic multiplexer [11] and extend its functionality from 2-to-1 gate to $(\ell+1)$-to-1 multiplexer. The previous 2-to-1 multiplexer gate takes two inputs m_0, m_1 and a selector $\mu \in \{0,1\}$, and outputs m_μ. This gate can be evaluated homomorphically with formula $(1-\mu) \cdot m_0 + m_1$. In a similar manner, one can also design a $(\ell+1)$-to-1 multiplexer gate, which takes $(\ell+1)$ inputs $m_0, m_1, \ldots, m_\ell$ and ℓ selectors $\mu_1, \ldots, \mu_\ell \in \{0,1\}$, at most one of which is nonzero. Then, it outputs m_i if $\mu_i = 1$ for some i, and m_0 when $\mu_i = 0$ for all i, as described in Fig. 1. Similar to 2-to-1 MUX, this generalized multiplexer gate also can be evaluated homomorphically using the formula $(1 - \sum_{i=1}^{\ell} \mu_i) \cdot m_0 + \sum_{i=1}^{\ell} \mu_i m_i$.

In this point of view, the previous blind rotation algorithm homomorphically evaluates 2-to-1 multiplexer gates for n times. In the new algorithm, we squash each ℓ multiplexer gates into a single $(\ell+1)-to-1$ multiplexer gate. As a result, our algorithm evalutes only k multiplexer gates in total.

4.2 Key-Switching

In this section, we present our improved key-switching procedure. Recall that given a TLWE ciphertext $(b, \mathbf{a}) = (b, a_0, \ldots, a_{N-1})$ under secret $\mathbf{t}$, the key-switching operation of TFHE aims to generate a TLWE ciphertext $(b', \mathbf{a}') = (b', a'_0, \ldots, a'_{n-1})$ under secret $\mathbf{s}$ such that $b + \langle \mathbf{a}, \mathbf{t} \rangle \approx b' + \langle \mathbf{a}', \mathbf{s} \rangle \pmod 1$. To accelerate this algorithm, we use TLWE secret as a part of TRLWE secret. More precisely, the vectorized TRLWE key $\mathbf{t} = (s_0, \ldots, s_{N-1})$ has the TLWE secret $\mathbf{s} = (s_0, \ldots, s_{n-1})$ as a part of it. Then, we can write it as a concatenation of $\mathbf{s}$ and some binary vector $\mathbf{s}' = (s_n, s_{n+1}, \ldots, s_{N-1})$, *i.e.,* $\mathbf{t} = (\mathbf{s}, \mathbf{s}')$. This allows us to skip computations for the shared part during the key-switching procedure, resulting in a speed-up of $\frac{N}{N-n}$. Additionally, it reduces the size of the key-switching key by a factor of $\frac{N}{N-n}$.

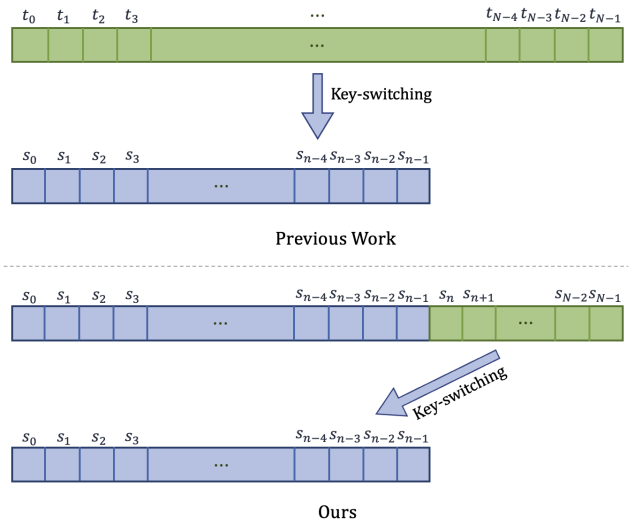

Fig. 2. New Key-Switching Algorithm

The comparison between the previous and our key-switching algorithms is described in Fig. 2. As a result, our approach has advantages in terms of both space and time complexity, as well as the smaller noise growth. A formal description of our key-switching algorithm is given below.

• $\underline{\mathsf{NewSwitchKeyGen}(\mathbf{t})}$: Given the TLWE secret $\mathbf{t} = (s_0, \ldots, s_{N-1}) \in \mathbb{B}^N$, we generate the key-switching key as follows:

– Sample $\mathbf{a}_{i,j}[k] \leftarrow \mathbb{T}^n$ and $e_{i,j}[k] \leftarrow \mathcal{D}_\alpha$ for $0 \le i < N-n, 0 \le j < f, 0 \le k < D$, and set $\mathsf{KSK}_{i,j}[k] = (b_{i,j}[k], \mathbf{a}_{i,j}[k])$ where $b_{i,j}[k] = -\langle \mathbf{a}_{i,j}[k], \mathbf{s}\rangle + k \cdot D^{-j-1} \cdot s_{n+i} + e_{i,j}[k] \pmod 1$.
– Return $\mathsf{KSK} = \{\mathsf{KSK}_{i,j}[k] \mid 0 \le i < N-n, 0 \le j < f, 0 \le k < D\}$.

• $\underline{\mathsf{NewKeySwitch}(\mathsf{KSK}, \mathbf{c})}$: Given a TLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{N+1}$, let $a_{i,j} \in \mathbb{Z} \cap [0, D)$ be the integers such that $|a_i - \sum_{j=0}^{f-1} a_{i,j} \cdot D^{-j-1}| \le D^{-f}$ $(n \le i < N)$. Return the ciphertext $(b, a_0, \ldots, a_{n-1}) + \sum_{i=0}^{N-n-1} \sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{n+i,j}] \in \mathbb{T}^{n+1}$.

Lemma 2. *Given a TLWE ciphertext $\mathbf{c} \in \mathbb{T}^{N+1}$, our key-switching algorithm outputs a TLWE ciphertext $\mathbf{c}' \in \mathbb{T}^{n+1}$ with $|\varphi_{\mathbf{s}}(\mathbf{c}') - \varphi_{\mathbf{t}}(\mathbf{c})| \le (N-n)f\eta_\alpha + (N-n)D^{-f}$.*

Proof. Let us use the same notation as the explanation of idea and the algorithm above. First, note that

$$\begin{aligned}
\varphi_{\mathbf{s}}\Big(\sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{n+i,j}]\Big) &= \sum_{j=0}^{f-1} \varphi_{\mathbf{s}}(\mathsf{KSK}_{i,j}[a_{n+i,j}]) \\
&= \sum_{j=0}^{f-1} (a_{n+i,j} \cdot D^{-j-1} \cdot s_{n+i} + e_{i,j}[a_{n+i,j}]) \\
&= a_{n+i} \cdot s_{n+i} + e_i \cdot s_{n+i} + \sum_{j=0}^{f-1} e_{i,j}[a_{n+i,j}]
\end{aligned}$$

where $e_i = \langle (a_{i,0}, \ldots, a_{i,f-1}), (D^{-1}, \ldots, D^{-f})\rangle - a_i$. Then we obtain

$$\begin{aligned}
\Big|\varphi_{\mathbf{s}}\Big(\sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{n+i,j}]\Big) - a_{n+i} \cdot s_{n+i}\Big| &= |e_i + \sum_{j=0}^{f-1} e_{i,j}[a_{n+i,j}]| \\
&\le D^{-f} + f \cdot \eta_\alpha
\end{aligned}$$

for $0 \le i < N-n$. Consequently, we have

$$\begin{aligned}
|\varphi_{\mathbf{s}}(\mathbf{c}') - \varphi_{\mathbf{t}}(\mathbf{c})| &= |\sum_{i=0}^{N-n-1} a_{n+i} s_{n+i} - \sum_{i=0}^{N-n-1} \varphi_{\mathbf{s}}\Big(\sum_{j=0}^{f-1} \mathsf{KSK}_{i,j}[a_{n+i,j}]\Big)| \\
&= \Big|\sum_{i=0}^{N-n-1} (e_i + \sum_{j=0}^{f-1} e_{i,j}[a_{n+i}, j])\Big| \le (N-n) \cdot (f\eta_\alpha + D^{-f}).
\end{aligned}$$

We note that our key-switching method reduces the key size, computational cost, and noise bound by a factor of $\frac{N}{N-n}$ since it requires $(N-n)$ iterations compared to N of the previous approach. In addition, we can further improve the performance of key-switching by making a trade-off between key size and computational cost, which will be described in detail in the next section.

4.3 Scheme description

Finally, we combine new building blocks for blind rotation and key-switching to demonstrate the gate bootstrapping with better efficiency. The exact scheme description of our scheme is as follows.

Algorithm 3 New TFHE Bootstrapping

Input: The blind rotation key BRK, the key-switching key KSK and a TLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) = (b, a_0, \ldots, a_{n-1}) \in \mathbb{T}^{n+1}$

Output: A TLWE ciphertext $\mathbf{c}' \in \mathbb{T}^{n+1}$

1: $\mathsf{ACC} \leftarrow \mathsf{NewBlindRotate}(\mathsf{BRK}, \mathbf{c})$
2: Parse ACC as $\left(\sum_{i=0}^{N-1} b'_i X^i, \sum_{i=0}^{N-1} a'_i X^i\right)$
3: $\mathbf{c}' \leftarrow \mathsf{NewKeySwitch}(\mathsf{KSK}, (b'_0, a'_0, -a'_{N-1}, \ldots, -a'_1))$

• $\underline{\mathsf{Setup}(1^\lambda)}$:

– Take the security parameter λ as input and generate the following parameters: the block length ℓ, the number of blocks k, TLWE dimension $n = \ell \cdot k$, TRLWE dimension N, error parameters $\alpha, \beta > 0$, the gadget decomposition base and dimension B and d, and the key-switching parameters D and f.

• $\underline{\mathsf{KeyGen}(1^\lambda)}$:

– Sample $(s_0, \ldots, s_{n-1}) \leftarrow \mathbf{B}_{\ell,k}$ and $s_i \leftarrow \mathcal{U}(\mathbb{B})$ for $n \le i < N$. Set the TLWE secret as $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{B}^n$ and the TRLWE key as $t = s_0 + \cdots + s_{N-1}X^{N-1}$. Write $\mathbf{t} = (s_0, \ldots, s_{N-1})$.
– Generate $\mathsf{BRK}_i \leftarrow \mathsf{TRGSW.Enc}(t, s_i)$ for $0 \le i < n$. Return the blind rotation key $\mathsf{BRK} = \{\mathsf{BRK}_i\}_{0 \le i < n}$.
– Generate a key-switching key $\mathsf{KSK} \leftarrow \mathsf{NewSwitchKeyGen}(\mathbf{t})$.

• $\underline{\mathsf{Enc}(\mathbf{s}, m)}$: Given the secret key $\mathbf{s} = (s_0, \ldots, s_{n-1})$ and a message bit $m \in \{\pm 1\}$, sample $\mathbf{a} = (a_0, \ldots, a_{n-1}) \leftarrow \mathcal{U}(\mathbb{T}^n)$ and $e \leftarrow \mathcal{D}_\alpha$. Return $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$ where $b = -\langle \mathbf{s}, \mathbf{a} \rangle + \frac{1}{8}m + e \pmod 1$.

• $\underline{\mathsf{Dec}(\mathbf{s}, \mathbf{c})}$: Given the secret key $\mathbf{s}$ and a ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$, let $\mu = b + \langle \mathbf{s}, \mathbf{a} \rangle \pmod 1$. Output 1 if $\mu > 0$, 0 otherwise.

• $\underline{\mathsf{NewBoot}(\mathsf{BRK}, \mathsf{KSK}, \mathbf{c})}$: Given the blind rotation key BRK, the key-switching key KSK and a TLWE ciphertext $\mathbf{c} \in \mathbb{T}^{n+1}$, run Alg. 3 and return $\mathbf{c}' \in \mathbb{T}^{n+1}$.

• $\underline{\mathsf{NewHomNAND}(\mathsf{BRK}, \mathsf{KSK}, \mathbf{c}_1, \mathbf{c}_2)}$: Given the blind rotation key BRK, the key-switching key KSK and TLWE ciphertexts $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{T}^{n+1}$, return $\mathsf{NewBoot}(\mathsf{BRK}, \mathsf{KSK}, (\frac{1}{8}, \mathbf{0}) - \mathbf{c}_1 - \mathbf{c}_2) \in \mathbb{T}^{n+1}$.

Correctness. Similar to the original TFHE, $\mathbf{c} := (\frac{1}{8}, \mathbf{0}) - \mathbf{c}_1 - \mathbf{c}_2$ is a TLWE ciphertext such that $\varphi_{\mathbf{s}}(\mathbf{c}) \approx \frac{1}{4}m \pmod 1$ if $\varphi_{\mathbf{s}}(\mathbf{c}_i) \approx \frac{1}{8}m_i \pmod 1$ and $m = m_1 \barwedge m_2$. Below we prove the correctness of our new bootstrapping procedure by showing that if $\mathbf{c}$ is an input TLWE ciphertext such that $\varphi_{\mathbf{s}}(\mathbf{c}) \approx \frac{1}{4}m \pmod 1$, then the output ciphertext $\mathbf{c}'$ satisfies $\varphi_{\mathbf{s}}(\mathbf{c}') \approx \frac{1}{8}m \pmod 1$.

From Lem. 1, the phase of ACC in Line 1 of Alg. 3 is approximately $X^{\overline{b} + \langle \overline{\mathbf{a}}, \mathbf{s} \rangle} \cdot v$, where $\overline{b} = \lfloor 2Nb \rceil$, $\overline{\mathbf{a}} = \lfloor 2N\mathbf{a} \rceil$, and $v = -\frac{1}{8}(1 + X + \cdots + X^{N-1})$. If $m = 1$, then it holds that $0 < \overline{b} + \langle \overline{\mathbf{a}}, \mathbf{s} \rangle \le N$, and the constant term of the phase of ACC is close to $1/8$. Conversely, if $m = -1$, then the constant term is close to $-1/8$. Therefore, the extracted TLWE ciphertext in Line 2 of Alg. 3 has a phase close to either $1/8$ or $-1/8$ depending on the sign of m. Finally, when we apply the key-switching operation in Line 3, we obtain the output ciphertext $\mathbf{c}'$, and its phase is still close to either $1/8$ or $-1/8$ due to Lem. 2. Consequently, our bootstrapping algorithm $\mathsf{NewBoot}$ works correctly.

Security. Our scheme is IND-CPA secure under the decisional TLWE assumption of parameter (n, α) with a block binary distribution $\mathbf{B}_{\ell,k}$ since the usual LWE encryption is used. It requires an additional assumption that our scheme remains to be secure even if the bootstrapping key is given to an adversary.

Recall that the bootstrapping key consists of the blind rotation key and key-switching keys. Roughly speaking, the key-switching key is a collection of special TLWE encryptions of $s_n, \ldots, s_{N-1}$ under $\mathbf{s}$, which are indistinguishable from uniformly random samples over $\mathbb{T}^{n+1}$ from the same decisional TLWE assumption.

Meanwhile, our TRLWE key $t = s_0 + \cdots + s_{N-1}X^{N-1}$ (or its vector form $\mathbf{t} = (s_0, \ldots, s_{N-1})$) is obtained by concatenating $\mathbf{s} = (s_0, \ldots, s_{n-1})$ with a random binary vector $\mathbf{s}' = (s_n, \ldots, s_{N-1}) \leftarrow$

$\mathcal{U}(\mathbb{B}^{N-n})$. In addition, the blind rotation key is a collection of TRGSW encryptions of $s_1, \ldots, s_n$ under the TRLWE key t. Hence, our scheme also requires the decisional TRLWE assumption of the parameter (N, β) and the key distribution $\mathbf{B}_{\ell,k} \times \mathcal{U}(\mathbb{B}^{N-n})$. We conduct a similar cryptanalysis as in Sec. 3.1 to estimate the concrete security level of this hardness assumption problem. After analyzing various lattice attacks, we conclude that there is no efficient attack which exploits the dependency of TLWE and TRLWE keys.

Finally, we note that our construction requires an additional circular security assumption similar to the original TFHE scheme since the blind rotation and key-switching keys can be viewed as encryptions of $\mathbf{s}$ under $\mathbf{t}$ and vice versa.

Below we summarize the changes in our construction and compare it with the previous scheme [11].

- We sample the TLWE key from a block binary distribution and reuse it when generating the RLWE key. As a result, the size of the key-switching key is reduced by a factor of $\frac{N}{N-n}$ compared to the original bootstrapping method proposed in [11]. The TLWE dimension n is increased to ensure the hardness of the TLWE problem under block binary distribution, so the size of the bootstrapping key is also increased by a small factor.
- In blind rotation, we reduce the number of required external products by a factor of ℓ using the block structure of the TLWE secret. For each block, we compute a linear combination of ℓ bootstrapping keys and perform only one external product between this linear combination and the accumulator. Consequently, we reduce the number of external products from n to k. However, this change yields a larger noise growth by a factor of at most two, from the multiplication of the coefficients $X^{\bar{a}_i} - 1$ and the bootstrapping key.
- In the key-switching procedure, the number of scalar multiplications between torus elements and the TLWE ciphertexts is reduced to $f(N - n)$ times. This gives us a speed up on the key-switching step by a factor of $\frac{N}{N-n}$, and the error bound is also reduced by the same factor. In consequence, the extra margin for the key-switching error complements the increased error from blind rotation, and thus we can use the error parameter settings that is analogous to those in [11]. In the usual parameter setting with $N = 1024, n \geq 630$, the key-switching procedure becomes at least three times faster.

With all these improvements throughout the bootstrapping procedure, we have reduced the time complexity of the bootstrapping. Moreover, space complexity is also decreased compared to the original TFHE scheme since we reduced the size of key-switching key. Thorough analysis of the result with concrete parameters and the optimization techniques that can be used for better performance will be presented in the following section.

5 Implementation

In this section, we implement our scheme and demonstrate its performance. We present optimization techniques used in our implementation, concrete parameter sets, and benchmark results of our scheme. Our implementation is based on the TFHE library ver 1.1 [12] with `SPQLIOS` FFT processor. All benchmarks are conducted on a single-core of a desktop machine with Intel(R) i5-12400 @ 2.50GHz CPU and 8GB memory. Our source code is available at `https://github.com/SNUCP/blockkey-tfhe`.

5.1 Optimization techniques

We notice that there is a room for improvement when it comes to actual implementation. We present how to optimize blind rotation and key-switching operation in our bootstrapping algorithm Alg. 3.

Blind Rotation. We first review implementation of external product in the TFHE library [12]. The implementation consists of gadget decomposition, scalar multiplications, and Fast Fourier Transform (FFT) operations. We use the coefficient representation of polynomials by default, but they are transformed into

	#FFT	#Mult
TFHE [12]	$(2d+2)n$	$(4d+2)nN$
Ours	$(2d+2)k$	$(4d+2)nN$

Table 2. Complexity of blind rotation algorithms

the FFT form and vice versa before and after polynomial multiplications for the efficiency. In particular, we exploit FFT conversions for homomorphic operations of TRLWE or TRGSW ciphertexts such as external product.

We first analyze the complexity of previous bootstrapping procedure (Alg. 1) in terms of FFT operations and scalar multiplications. At each iteration of the blind rotation

$$\mathsf{ACC} \leftarrow \mathsf{ACC} + \left[(X^{\overline{a}_j} - 1) \cdot \mathsf{ACC}\right] \boxdot \mathsf{BRK}_j,$$

we first compute $(X^{\overline{a}_j} - 1) \cdot \mathsf{ACC}$ and its gadget decomposition $h((X^{\overline{a}_j} - 1) \cdot \mathsf{ACC}))$ without FFT operations. After converting its components into the FFT form using $2d$ FFT operations, we compute $h((X^{\overline{a}_j} - 1) \cdot \mathsf{ACC})) \cdot \mathsf{BRK}_j$ and transform the result back to the coefficient form, which takes $4dN$ scalar multiplications and 2 FFT-operations. Note that no FFT operation is required for BRK_j since their FFT form can be precomputed. Therefore, the blind rotation of Alg. 1 takes $(4d+2)nN$ multiplications and $(2d+2)n$ FFT operations as the blind rotation algorithm has n iterations.

Meanwhile, our blind rotation algorithm has only k iterations so reduces the number of external products (and FFT operations) by a factor of ℓ. To be precise, Alg. 2 takes $(2d+2)k$ FFT operations. On the other hand, our algorithm may require more scalar multiplications since it computes linear combinations of TRGSW ciphertexts. We observe that this issue can be addressed using an optimization technique of reusing the FFT of an accumulator during the blind rotation. Our blind rotation algorithm computes

$$\mathsf{ACC} \leftarrow \mathsf{ACC} + \mathsf{ACC} \boxdot \left[\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot \mathsf{BRK}_i\right]$$

in Line 5 of Alg. 2. It can be rewritten as

$$\mathsf{ACC} \leftarrow \mathsf{ACC} + \left[\sum_{i \in I_j} (X^{\overline{a}_i} - 1) \cdot (\mathsf{ACC} \boxdot \mathsf{BRK}_i)\right]$$

which does not involve linear combination of TRGSW ciphertexts but requires the same number of scalar multiplications as in Alg. 1. Furthermore, since the term $h(\mathsf{ACC})$ commonly appears in the external products $\mathsf{ACC} \boxdot \mathsf{BRK}_i = h(\mathsf{ACC}) \cdot \mathsf{BRK}_i$ for all $i \in I_j$, we can precompute and reuse its FFT form. As a result, the number of FFT operations remains unchanged. In conclusion, the total complexity of our blind rotation algorithm is $(4d+2)nN$ scalar multiplications and $(2d+2)k$ FFT operations (see Table 2 for comparison).

Key-switching. We now present some optimization techniques on the key-switching procedure. Recall that to perform key-switching, the algorithm constructs an encryption of $a_i t_i$ for $0 \le i < N$ via encryptions of $D^{-j} \cdot k (1 \le j \le f, 0 \le k < D)$. Note that it is unnecessary to consider the case $k = 0$, one has to publish $f(D-1)N$ TLWE ciphertexts as the key-switching key in the original TFHE scheme. Although our new key-switching method reduces the key-switching key size by a factor of $\frac{N}{N-n}$, there is still a room for further optimizations with respect to the size of the key-switching key.

Recently, Joye [23] presented the balanced non-adjacent form and its advantages which can be leveraged to the key-switching technique. We shall adopt the idea of balanced representation to further reduce the storage of key-switching key. We remark that the balanced base-D representation takes $-\lfloor D/2 \rfloor, -\lfloor D/2 \rfloor + 1, \ldots, \lfloor D/2 \rfloor$ as a possible digit values and therefore one requires TLWE encryptions of $D^{-j} \cdot k (1 \le j \le f, -\lfloor D/2 \rfloor \le k \le \lfloor D/2 \rfloor)$. However, we can simply convert an encryption of $D^{-j} \cdot k$

	#Add	#TLWE
TFHE [12]	fnN	$f(D-1)N$
Ours	$fn(N-n)$	$f\lfloor D/2 \rfloor (N-n)$

Table 3. Relation between key switching cost and key-switching key size. # Add and # TLWE represent the computational cost and size of the key-switching key, respectively.

	ℓ	α	n	N	f	D
[12]	$\cdot$	2^{-15}	630	1024	8	2^2
Ours	2	2^{-15}	630	1024	4	2^4
	3	2^{-15}	687	1024	4	2^4
	4	2^{-15}	788	1024	4	2^4
	5	2^{-15}	885	1024	4	2^4
	6	2^{-15}	978	1024	4	2^4

Table 4. Parameters for TLWE

	ℓ	n	Blind rotation	Key-switching	Total
[12]	$\cdot$	630	9.40 ms	1.13 ms	10.53 ms
Ours	2	630	6.82 ms	0.23 ms	7.05 ms
	3	687	6.29 ms	0.20 ms	6.49 ms
	4	788	6.56 ms	0.14 ms	6.70 ms
	5	885	6.75 ms	0.07 ms	6.82 ms
	6	978	7.10 ms	0.02 ms	7.12 ms

Table 5. Bootstrapping performance

to $D^{-j} \cdot (-k)$ by multiplying -1 to the encryption. Hence, the key-switching key can be represented by TLWE encryptions of $D^{-j} \cdot k (1 \leq j \leq f, 1 < k \leq \lfloor D/2 \rfloor$, only $f\lfloor D/2 \rfloor (N-n)$ total.

The performance of key-switching procedure is summarized in Table 3. In our implementation, we choose a smaller f by taking a larger D with similar error to obtain better computational cost while maintaining similar unit key size to the existing implementation.

5.2 Parameters and Benchmarks

We present some recommended parameter sets in Table 4, which achieve at least 128-bit security level (see Sec. 3.2 for security analysis). For TRLWE gadget decomposition, our implementation uses the same digit decomposition parameter sets $B = 2^7, d = 3$ as in the TFHE library [12]. For TRLWE and TRGSW schemes, we use the error parameter $\beta = 2^{-25}$ which is also the same as in the TFHE library.

We measure the performance of blind rotation and key-switching operation, and the experimental results are summarized in Table 5. As shown in the table, the execution time of blind rotation does not necessarily decrease as ℓ increases even if the number of FFT operations is reduced. This is due to the cost of scalar multiplications which grows linearly with n. For the key-switching operation, its complexity mainly depends on the value of $(N-n)$ as expected from our analysis. In terms of the total execution time of gate bootstrapping, our implementation achieves 1.48 to 1.62x speed-up compared to the previous implementation [12]. While we reduce the number of FFT operations by a factor of ℓ, the complexity of scalar multiplications is unchanged so it takes about 45% of blind rotation in our experiments.

We also measure the size of public keys for our parameter sets in Table 6. We note that the key-switching key consists of $N-n$ TLWE ciphertext so it gets smaller as n increases when other parameters are fixed. Compared to [12], we require about half the size of the bootstrapping and key-switching keys.

	ℓ	n	Blind rotation	Key-switching	Total
[12]	·	630	30 MB	79 MB	109 MB
Ours	2	630	30 MB	30 MB	60 MB
	3	687	32 MB	28 MB	60 MB
	4	788	37 MB	23 MB	60 MB
	5	885	41 MB	15 MB	56 MB
	6	978	46 MB	6 MB	52 MB

Table 6. Size of bootstrapping key

References

1. Albrecht, M.R.: On dual lattice attacks against small-secret lwe and parameter choices in helib and seal. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II. pp. 103–129. Springer (2017)
2. Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving usvp and applications to lwe. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 297–322. Springer (2017)
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
4. Bai, S., Galbraith, S.D.: Lattice decoding attacks on binary lwe. In: Australasian Conference on Information Security and Privacy. pp. 322–337. Springer (2014)
5. Bonte, C., Iliashenko, I., Park, J., Pereira, H.V., Smart, N.P.: Final: Faster FHE instantiated with NTRU and LWE. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II. pp. 188–215. Springer (2023)
6. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Annual Cryptology Conference. pp. 868–886. Springer (2012)
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
8. Carpov, S., Izabachène, M., Mollimard, V.: New techniques for multi-value input homomorphic evaluation and applications. In: Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019. pp. 106–126. Springer (2019)
9. Chen, H., Chillotti, I., Song, Y.: Multi-key homomorphic encryption from tfhe. In: Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part II 25. pp. 446–472. Springer (2019)
10. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 409–437. Springer (2017)
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption library (August 2016), https://tfhe.github.io/tfhe/
13. Chillotti, I., Joye, M., Paillier, P.: Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In: Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8–9, 2021, Proceedings 5. pp. 1–19. Springer (2021)
14. Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In: Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27. pp. 670–699. Springer (2021)
15. Cong, K., Das, D., Park, J., Pereira, H.V.: Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 563–577 (2022)

16. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 617–640. Springer (2015)
17. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference. pp. 75–92. Springer (2013)
20. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: International Conference on Supercomputing (2010)
21. Guimarães, A., Borin, E., Aranha, D.F.: Revisiting the functional bootstrap in tfhe. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 229–253 (2021)
22. Jiang, L., Lou, Q., Joshi, N.: Matcha: A fast and energy-efficient accelerator for fully homomorphic encryption over the torus. In: Proceedings of the 59th ACM/IEEE Design Automation Conference. pp. 235–240 (2022)
23. Joye, M.: Balanced non-adjacent forms. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 553–576. Springer (2021)
24. Kim, A., Deryabin, M., Eom, J., Choi, R., Lee, Y., Ghang, W., Yoo, D.: General bootstrapping approach for rlwe-based homomorphic encryption. Cryptology ePrint Archive (2021)
25. Kluczniak, K.: Ntru-\$nu\$-um: Secure fully homomorphic encryption from ntru with small modulus. Cryptology ePrint Archive (2022)
26. Kwak, H., Min, S., Song, Y.: Towards practical multi-key tfhe: Parallelizable, key-compatible, quasi-linear complexity. Cryptology ePrint Archive (2022)
27. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III. pp. 227–256. Springer (2023)
28. Lu, W.j., Huang, Z., Hong, C., Ma, Y., Qu, H.: Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1057–1073. IEEE (2021)
29. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 1–23. Springer (2010)
30. May, A.: How to meet ternary lwe keys. In: Annual International Cryptology Conference. pp. 701–731. Springer (2021)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)
32. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse secret lwe and application to he parameters. p. 11–20. WAHC'19, Association for Computing Machinery (2019)
33. Sun, C., Tibouchi, M., Abe, M.: Revisiting the hardness of binary error lwe. In: Information Security and Privacy: 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30–December 2, 2020, Proceedings. pp. 425–444. Springer (2020)
34. Van Beirendonck, M., D'Anvers, J.P., Verbauwhede, I.: Fpt: a fixed-point accelerator for torus fully homomorphic encryption. arXiv preprint arXiv:2211.13696 (2022)