
A New Fine Tuning Method for FHEW/TFHE Bootstrapping with IND-CPA^D Security

Anonymous Authors

ABSTRACT Fully homomorphic encryption (FHE) schemes enable computation on encrypted data, making them a crucial component of privacy-enhancing technologies. The availability of encrypted computation arises a new security notion, so-called (c, s) -security, proposed by Li et al. (CRYPTO'22), which considers both computational and statistical security. The key challenge is that, unlike computational security (commonly fixed at 128, 192, or 256 bits), statistical security requirements depend heavily on the application (e.g., circuit shape or number of queries). Thus, smooth control over the statistical security parameter s is essential. FHEW-like schemes offer the best latency and smallest key sizes. However, Cheon et al. (CCS'24) showed a key-recovery attack that exploits failure events during legitimate evaluation queries. Consequently, achieving a sufficiently low bootstrapping failure probability, proportional to 2^{-s} , is critical for the statistical security of FHEW-like schemes. A major limitation is that FHEW's parameter sets are sparse, and suitable parameters achieving the exact target failure probability often do not exist. This typically forces the use of unnecessarily large parameters, leading to excessive computational costs. Recently, Bernard et al. (Eurocrypt'25) introduced a method that significantly reduces bootstrapping failure probability with minimal overhead, but their parameter sets remain sparse. In this work, we propose a new bootstrapping technique, *cutoff blind rotation*, which provides an additional axis of parameter optimization by enabling a smooth trade-off between computation and failure probability. This technique allows nearly continuous control of the bootstrapping failure probability in FHEW-like schemes without extra client-side overhead. We further provide concrete parameter sets achieving various failure probabilities (2^{-64} , 2^{-96} , and 2^{-128}) in our OpenFHE implementation. The implementation results show that the bootstrapping runtime can be reduced by refining the failure probability.

INDEX TERMS Blind rotation, bootstrapping, homomorphic encryption, key recovery attack.

I. Introduction

A Fully Homomorphic Encryption (FHE) scheme enables computations to be performed directly on encrypted data, thereby preserving privacy during data processing. The Brakerski-Gentry-Vaikuntanathan (BGV) and Brakerski-Fan-Vercauteren (BFV) schemes support arithmetic operations on integers [1]–[3], while the FHEW and TFHE schemes [4]–[6] enable operations on logic circuits. These FHE schemes are categorized as *Exact FHE*. In addition, the Cheon-Kim-Kim-Song (CKKS) scheme, introduced in [7], allows approximate operations on complex numbers, thereby classifying it as *Approximate FHE*.

In privacy-preserving applications, FHE is typically used in a client-server architecture, where a client encrypts their data and transmits it to a server for computation. The server performs the requested operations on the encrypted data and returns the encrypted results to the client. Designing an efficient FHE system requires selecting optimal parameters that balance security and computational performance. The

selected FHE scheme must also ensure security against chosen-plaintext attacks (IND-CPA security) so that the server cannot learn any information about the client's data from the ciphertext.

Li and Micciancio extended the conventional notion of IND-CPA security by introducing indistinguishably under chosen-plaintext attacks with a decryption oracle (IND-CPA^D) [8]. In contrast to the traditional IND-CPA model, this variant allows an adversary to observe the results of legitimate computations performed on the ciphertext. It has been shown that the IND-CPA^D model exposes *Approximate FHE* schemes, such as CKKS, to attacks. The standard countermeasure to this vulnerability is a technique known as noise flooding, which involves increasing the noise level in the ciphertext to render it statistically indistinguishable from random noise [8].

Cheon et al. [9] demonstrated that Exact FHE schemes, including BGV/BFV and FHEW/TFHE, also fail to meet IND-CPA^D security due to evaluation failure events. More-

over, they conducted a key-recovery attack on widely used libraries such as TFHE-rs, emphasizing the need to revise the current parameters.

Since Cheon et al.'s attack exploits evaluation failure events, the failure probability of homomorphic operations must be minimized to reduce the adversary's advantage. The failure probability is primarily determined by the parameters of the FHE scheme, such as the ciphertext modulus. Although many bootstrapping methods have been proposed for FHEW-like schemes, the number of viable parameter sets remains limited. Due to the inherent trade-off between performance and failure probability, these restricted options often force the use of parameter sets with unnecessarily low failure probabilities, leading to inefficient runtime.

A. Failure Probability of FHEW-like Schemes and IND-CPA^D Security

The FHEW-like schemes are based on the Learning with Errors (LWE) problem [10] and its ring variant, Ring-LWE (RLWE) [11]. Bootstrapping in FHEW-like schemes is based on a novel homomorphic decryption circuit called blind rotation. Since ciphertexts inherently contain noise that accumulates with each homomorphic operation, decryption will fail if the noise exceeds a certain threshold. To date, FHEW-like schemes have been designed to maintain a low failure probability (preferably less than 2^{-40} – 2^{-50} [4], [5], [12]–[14]) while ensuring acceptable performance.

Cheon et al. proposed a key-recovery (KR^D) attack, which is a relaxation of IND-CPA^D, against FHEW-like schemes in the presence of a decryption oracle that exploits bootstrapping failures [9]. When a failure occurs, the distribution of LWE ciphertext elements exhibits a noticeable difference depending on whether the corresponding secret key is 0 or 1. An adversary can leverage this information to recover the secret key, and a polynomial-time attack becomes feasible given a sufficient number of failure events.

Li et al. proposed a finer-grained definition of bit-security that distinguishes between a *computational* security parameter c and a *statistical* one s [15]; a primitive achieves (c, s) -security if for any adversary \mathcal{A} , either \mathcal{A} has statistical advantage bounded by 2^{-s} (regardless of \mathcal{A} 's running time or computational assumptions), or the running time of the attack is at least 2^c times larger than the advantage achieved. It is important to note that the choice of statistical security parameter s should be *application dependent*, while a commonly accepted value for the computational security parameters in most applications is $c = 128$ bits. To meet (c, s) -security, the bootstrapping failure probability should be determined by considering both the statistical security s and the number of queries made by the adversary.

Reducing the bootstrapping failure probability involves adjusting parameters such as ring dimension, LWE dimension, and ciphertext modulus. However, there is a significant gap between feasible values because the ring dimension must be a power of two for efficient number theoretic transforma-

tion (NTT) and modular arithmetic in the exponent. While the LWE dimension does not need to be a power of two, it directly affects the security level and cannot be drastically altered. Another and most flexible option is changing the digit of decomposition d , typically a *small integer such as 3 or 4* [12], [13]. Unfortunately, even minor changes in d can significantly impact both the failure probability and computational complexity, thereby limiting viable parameters. It is therefore challenging to precisely tune the bootstrapping failure probability to a specific target such as 2^{-80} without incurring unnecessary efficiency loss.

B. Related Works and Our Contribution

We propose a new bootstrapping method that provides a finer-grained statistical security parameter s while introducing a trade-off in runtime complexity. As discussed previously, existing parameter sets for FHEW-like schemes are limited in number due to the significant influence of parameter choices on the overall failure probability. Consequently, identifying an appropriate parameter configuration that achieves arbitrary (c, s) -security is challenging, particularly since the statistical security parameter s is application-dependent. Unfortunately, s also affects the performance of FHE applications, making it essential to determine an appropriate security parameter to ensure practical usability.

Not only for performance, but also for security, a finer-grained statistical parameter s is essential to construct a safe FHE system. For example, several attacks on FHE schemes exploit weaker statistical security parameters s [8], [9], [16], [17]. To mitigate such threats, Alexandru et al. proposed an application-aware homomorphic encryption scheme (AAHE). This scheme incorporates application awareness, thereby rejecting invalid queries such as disallowed operations or encryptions of messages outside the valid domain. Notably, it determines the statistical security parameter s during the key generation phase. This implies that the parameter sets of an FHE system should be fine-tunable with respect to s in order to construct practical systems.

Recently, Bernard et al. introduced a transform function and a quality test function [18]. The transform function modifies the nonce part of a ciphertext, while the quality test function estimates the expected noise of a ciphertext through the modulus switching procedure. This approach suggests a way to improve the statistical security parameter s with negligible complexity. As a result, parameter sets previously considered statistically insecure become secure.

However, each parameter component continues to exert a strong influence on the overall value of s . To address this limitation, we propose a new bootstrapping method that increases the density of viable parameter sets while introducing controlled trade-offs between runtime and failure probability.

Blind rotation involves the process of constructing a monomial $X^{-\langle \vec{a}, \vec{s} \rangle}$ by using homomorphic multiplication. Note that homomorphic multiplication is a costly operation

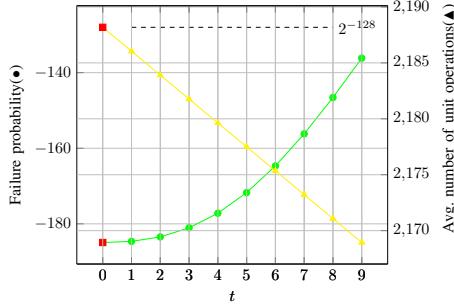


FIGURE 1. The number of unit operations and failure probability by cutoff parameter t . Note that prior arts can only achieve failure probability and runtime marked in red squares. The parameter set is borrowed from OpenFHE (LPF_STD128).

in FHE systems. We emphasize that omitting the factors X^{a_i} corresponding to small norm of a_i has minimal impact on the correct construction of $X^{-\langle \vec{a}, \vec{s} \rangle}$. To reduce the number of multiplications required during blind rotation, we introduce a cutoff value t and omit multiplications for a_i such that $|a_i| \leq t$.

Figure 1 illustrates how the failure probability and the number of multiplications change as the cutoff value t varies. We note that it is straightforward to apply the proposed method to other FHEW-like schemes as well [19], [20].

While the introduction of a cutoff value t may increase the probability of decryption failure, it also reduces computational complexity by omitting homomorphic multiplications. Importantly, t can be freely adjusted within the range $[0, q/2]$, and its effect on failure probability is considerably less pronounced than that of other parameters, such as decomposition digits. Additionally, it does not require any extra key generation, which keeps client-side overhead minimal. By expanding the set of viable parameters, the proposed method enables the selection of parameters that achieve the desired failure probability (e.g., 2^{-64} , 2^{-96} , or even 2^{-128}) while simultaneously reducing runtime.

We also suggest three variations of the proposed method. First, to improve the density of parameter sets, we propose a square-sum cutoff T , which omits a_i such that $\sum_{i \in I} |a_i| \leq T$ for some index set I . Second, to achieve a better trade-off between runtime and failure probability, we utilize the transform function proposed in [18]. Additionally, we introduce a new method that reduces the failure probability when the secret key distribution is binary.

As discussed in Section A, the failure probability of bootstrapping in FHEW-like schemes is critical not only for security but also for maintaining computational integrity. The total failure probability scales with the number of gates in a given circuit, and certain applications, such as financial systems or robot control systems, require extremely low failure probabilities. Thus, it is essential to achieve low failure probabilities while minimizing computational complexity to preserve the integrity of the FHE system. Additionally, the proposed parameter t can be adjusted after key generation,

providing flexibility in parameter tuning without necessitating key regeneration, making it suitable for a wide range of application scenarios. We have implemented the proposed method in OpenFHE [21], and our experimental results demonstrate a reduction in runtime using this approach. Although the focus of this paper is primarily on parameter sets for binary and ternary keys, the proposed technique is also applicable to larger key spaces.

C. Organization

The rest of the paper is organized as follows. Section II introduces the basics of lattice-based HE and reviews prior FHEW-like bootstrapping techniques. Section III discusses the importance and difficulty of achieving finely tunable parameter sets. In Section IV, we present a new optimization technique for FHEW-like bootstrapping that enables fine-tuning of the failure probability. Section V details improvements to the proposed method. The implementation results are shown in VI. Finally, Section VII concludes the paper with summary remarks.

II. Preliminaries

We denote the inner product between two vectors as $\langle \cdot, \cdot \rangle$. Let N be a power of two, and define the $2N$ -th polynomial ring as $\mathbb{Z}[X]/(X^N + 1)$, while the corresponding quotient ring is represented as $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$. Elements of \mathcal{R}_Q are represented in boldface, such as $\mathbf{a}(X)$, and X is omitted when the context is clear. The i -th coefficient of ring element \mathbf{a} is denoted as a_i . Vectors are denoted as \vec{v} , and the i -th element of a vector \vec{v} is denoted by v_i . The L_2 norm of ring elements or vectors is represented as $\|\cdot\|$, and the infinity norm as $\|\cdot\|_\infty$. We use the notation $x \leftarrow \chi$ to indicate that x is sampled from a distribution χ . When x is sampled uniformly from a set S , this is denoted as $x \leftarrow S$.

A. Lattice-based Encryption

Let q and n be positive integers. The LWE encryption of a message $m \in \mathbb{Z}_q$ under a secret key \vec{s} is defined as follows:

$$\text{LWE}_{\vec{s}}(m) = (\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + m + e) \in \mathbb{Z}_q^{n+1}$$

where the secret key is sampled as $\vec{s} \leftarrow \chi_{sk}$, the error term is sampled as $e \leftarrow \chi_{err}$, and the public key component \vec{a} is sampled uniformly from \mathbb{Z}_q^n . Note that χ_{err} typically represents a discrete Gaussian distribution with zero mean and standard deviation σ . The decryption of LWE ciphertext $\text{LWE}_{\vec{s}}(m) = (\vec{a}, b)$ is defined as taking the inner product with $(-\vec{s}, 1)$. In other words,

$$\langle (\vec{a}, b), (-\vec{s}, 1) \rangle = \langle \vec{a}, \vec{s} \rangle + m + e - \langle \vec{a}, \vec{s} \rangle = m + e \approx m.$$

We define RLWE encryption of \mathbf{m} with secret key $\mathbf{z} \leftarrow \chi_{sk}$ as follows:

$$\text{RLWE}_{Q,z}(\mathbf{m}) = (\mathbf{a}, \mathbf{a} \cdot \mathbf{z} + \mathbf{m} + \mathbf{e}) \in \mathcal{R}_Q^2,$$

where $\mathbf{a} \leftarrow \mathcal{R}_Q$, and $\mathbf{e} \leftarrow \chi_{err}$. The decryption of RLWE is defined as follows:

$$\langle (\mathbf{a}, \mathbf{b}), (-\mathbf{z}, 1) \rangle = \mathbf{a} \cdot \mathbf{z} + \mathbf{m} + \mathbf{e} - \mathbf{a} \cdot \mathbf{z} = \mathbf{m} + \mathbf{e} \approx \mathbf{m}.$$

The basic building block of FHEW bootstrapping is RLWE' and Ring-GSW (RGSW) and their multiplication with ring element and RLWE ciphertext [4], [22]. We follow the definitions of $\text{RLWE}'_z(\mathbf{m})$ and $\text{RGSW}_z(\mathbf{m})$ from [12]:

$$\begin{aligned}\text{RLWE}'_z(\mathbf{m}) &:= (\text{RLWE}_z(g_0\mathbf{m}), \dots, \text{RLWE}_z(g_{d_g-1}\mathbf{m})) \\ \text{RGSW}_z(\mathbf{m}) &:= (\text{RLWE}'(-z\mathbf{m}), \text{RLWE}'(\mathbf{m})),\end{aligned}$$

where $\vec{g} = (g_0, g_1, \dots, g_{d_g-1})$ is a gadget vector, which is used in gadget decomposition. We say $h(\mathbf{a}) = (\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{d_g-1})$ is gadget decomposition of $\mathbf{a} \in \mathcal{R}_Q$ if $\mathbf{a} \approx \sum_{i=0}^{d_g-1} g_i \cdot \mathbf{a}_i$, where $\|\mathbf{a}_i\|_\infty < B_g$ and B_g is base of gadget decomposition satisfying $B_g^{d_g} \leq Q$. The gadget decomposition naturally extends to \mathbb{Z}_q and \mathbb{Z}_q^n .

RLWE' provides the following multiplication ($\odot : \mathcal{R}_Q \times \text{RLWE}' \rightarrow \text{RLWE}$):

$$\mathbf{a} \odot \text{RLWE}' = \sum_{i=0}^{d_g-1} \mathbf{a}_i \cdot \text{RLWE}_z(g_i \cdot \mathbf{m}) = \text{RLWE}_z(\mathbf{a} \cdot \mathbf{m}).$$

The multiplication between RLWE and RGSW ($\otimes : \text{RLWE} \otimes \text{RGSW} \rightarrow \text{RLWE}$) is defined as follows:

$$\begin{aligned}\text{RLWE}_z(\mathbf{m}_0) \otimes \text{RGSW}_z(\mathbf{m}_1) &= \mathbf{a} \odot \text{RLWE}'_z(-z \cdot \mathbf{m}_1) + \mathbf{b} \odot \text{RLWE}'_z(\mathbf{m}_1) \\ &= \text{RLWE}_z(\mathbf{m}_0 \cdot \mathbf{m}_1 + e_1 \cdot \mathbf{m}_1).\end{aligned}$$

This operation outputs an RLWE encryption of $\mathbf{m}_0 \cdot \mathbf{m}_1 + e_1 \cdot \mathbf{m}_1$. It is worth noting that the noise term $\mathbf{m}_1 \cdot e_1$ remains small because \mathbf{m}_1 is usually selected as a monomial, and thus consecutive RGSW multiplications primarily accumulate additive noise. The variance of the additive noise due to \odot RLWE' and \otimes RGSW operations are given by $\sigma_{\odot}^2 = d_g N \frac{B_g^2}{12} \sigma^2$ and $\sigma_{\otimes}^2 = 2\sigma_{\odot}^2$, respectively [12]. Note that the \otimes operation constitutes a performance bottleneck in FHEW-like schemes.

B. Bootstrapping in FHEW/TFHE

There are three main bootstrapping methods employed in FHEW-like schemes. The first, introduced by Ducas and Micciancio, builds on the Alperin-Sheriff and Peikert (AP) method [23], which they incorporated into their FHEW scheme [4]. This method delivers consistent performance regardless of the key distribution. The second method, used in the TFHE scheme [5] proposed by Chillotti et al., employs the Gama-Izabachene-Nguyen-Xie (GINX) technique [24], which offers superior performance but is limited to binary key distributions. Although GINX/TFHE bootstrapping can be generalized to support arbitrary secret keys [12], [25], its performance significantly degrades when larger key distributions are used. Lee, Micciancio, Kim, Choi, Deryabin, Eom, and Yoo (LMKCDEY) proposed an efficient bootstrapping technique using ring automorphisms, which achieves performance comparable to GINX/TFHE even with arbitrary secret keys [13].

Note that we only provide the analysis of the GINX method. There are five building blocks of bootstrapping

in FHEW-like scheme: key/modulus switching, accumulator initialization, blind rotation, and LWE extraction.

1) LWE Key Switching

LWE key switching is the process of converting an LWE encryption under a secret key $\vec{z} \in \mathbb{Z}_q^N$ into an LWE encryption under another secret key $\vec{s} \in \mathbb{Z}_q^N$. To perform key switching, one must define a key-switching key (ksk) as follows:

$$\text{ksk} = \left\{ \text{ksk}_{i,j,v} = \text{LWE}_{\vec{s}}(-v \cdot z_i \cdot B_{\text{ks}}^j) \mid i \in [0, N), j \in [0, d_{\text{ks}}), v \in [0, B_{\text{ks}}) \right\},$$

where B_{ks} and d_{ks} are the base and the digit length of the gadget decomposition for key switching, respectively.

The key-switching operation is defined as:

$$\text{KeySwitch}((\vec{a}, b), \text{ksk}) = (\vec{0}, b) + \sum_{i,j} \text{ksk}_{i,j,a_{i,j}},$$

where the gadget decomposition of a_i , denoted as $h(a_i)$, is given by $(a_{i,0}, a_{i,1}, \dots, a_{i,d_{\text{ks}}-1})$.

2) Modulus Switching

Modulus switching is an operation that converts the original modulus Q_1 of an LWE ciphertext to a new modulus Q_2 using a rounding function. Modulus switching, denoted by $[\cdot]_{Q_1}^{Q_2} : \mathbb{Z}_{Q_1} \rightarrow \mathbb{Z}_{Q_2}$ is defined as:

$$[t]_{Q_1}^{Q_2} = \left\lfloor \frac{Q_2 \cdot t}{Q_1} \right\rfloor.$$

The application of modulus switching to an LWE ciphertext $\text{LWE}_{\vec{s}}(m) = (\vec{a}, b)$ is given by:

$$[(\vec{a}, b)]_{Q_1}^{Q_2} = \left(([a_0]_{Q_1}^{Q_2}, [a_1]_{Q_1}^{Q_2}, \dots, [a_{n-1}]_{Q_1}^{Q_2}), [b]_{Q_1}^{Q_2} \right).$$

3) Accumulator Initialization

A cryptographic accumulator, as used in FHEW-like bootstrapping [4], is defined as:

$$\text{acc} = \text{RLWE} \left(\sum_{i=0}^{q/2-1} f(b-i) X^i \right),$$

where f is a mapping function [12]. Let $u = -\langle \vec{a}, \vec{s} \rangle$. Then, the u -th coefficient of acc corresponds to the decryption result of the ciphertext (\vec{a}, b) .

4) Blind Rotation

Blind rotation is the process of homomorphically extracting the u -th coefficient of acc as a constant term by multiplying it with an encryption of the monomial X^u . This operation requires special evaluation keys known as blind rotation keys, denoted by brk . The procedure of applying these keys to update the accumulator is referred to as the *update* step.

The blind rotation keys for GINX method are defined as follows:

$$\text{GINX/TFHE: } \left\{ \begin{array}{l} \text{brk}_{i,u} = \text{RGSW}(x_{i,u}) \\ |x_i \in \{0,1\}^{|U|} \text{ s.t. } \sum_{u \in U} u \cdot x_{i,u} = s_i \end{array} \right\},$$

where $U \subset \mathbb{Z}_q$. For example, $U = \{1, -1\}$ is suitable for ternary secret keys, while $U = \{1\}$ is used for binary secret keys.

The update steps can be expressed as follows:

$$\text{GINX/TFHE: acc} \leftarrow \text{acc} + (X^{u \cdot a_i} - 1)(\text{acc} \otimes \text{brk}_{i,u}).$$

In the GINX/TFHE method, the operation needs to be repeated $2|U|n$ times.¹

5) Extraction of LWE Ciphertext from RLWE Ciphertext

A single coefficient of the message embedded in an RLWE ciphertext can be extracted as an individual LWE ciphertext [4]. This procedure, referred to as LWE extraction, is employed in the FHEW and TFHE schemes to transform the RLWE ciphertext produced by blind rotation into an LWE ciphertext. The LWE extraction operation is formally defined as follows:

$$\begin{aligned} \text{LWEExtract} : \mathcal{R}_Q^2 &\mapsto \mathbb{Z}_Q^{N+1}, \\ \text{LWEExtract}((a, b)) &= (\vec{a}', b_0) \in \mathbb{Z}_Q^{N+1}, \end{aligned}$$

where $\vec{a}' = (a_0, -a_1, -a_2, \dots, -a_{N-1})$. Note that the LWE extraction is a noiseless operation.

C. Noise Analysis

Our noise analysis is based on the methodologies described in [4], [12], [13]. The noise generated during bootstrapping can generally be categorized into three additive sources: noise from blind rotation, noise from key switching, and noise from modulus switching.

1) Key Switching Noise

The sum of key switching keys can be expressed as:

$$\sum_{i,j} \text{ksk}_{i,j,a_{i,j}} = \sum_{i,j} \text{LWE}_{\vec{s}}(-a_{i,j} \cdot B_{\text{ks}}^j \cdot z_i) = \text{LWE}_{\vec{s}}(-\langle \vec{a}, \vec{z} \rangle)$$

and $(0, b)$ is a transparent ciphertext of b . Then, the key-switching operation is equal to

$$\begin{aligned} \text{LWE}_{\vec{s}}(b) + \text{LWE}_{\vec{s}}(-\langle \vec{a}, \vec{z} \rangle) &= \text{LWE}_{\vec{s}}(b - \langle \vec{a}, \vec{z} \rangle) \\ &\approx \text{LWE}_{\vec{s}}(m). \end{aligned}$$

As a result, the total noise variance introduced by the key-switching operation is given as $Nd_{\text{ks}} \cdot \sigma^2$, where the noise in each $\text{ksk}_{i,j,k}$ is independent and has variance σ^2 .

¹For a more generalized approach to GINX/TFHE bootstrapping, see [25].

2) Modulus Switching Noise

The decryption of $[(\vec{a}, b)]_{Q_1}^{Q_2} \in \mathbb{Z}_{Q_2}^{n+1}$ yields:

$$\begin{aligned} [b]_{Q_1}^{Q_2} - \left\langle [\vec{a}]_{Q_1}^{Q_2}, \vec{s} \right\rangle &= \frac{Q_2}{Q_1} \cdot b + r_n - \frac{Q_2}{Q_1} \cdot \langle \vec{a}, \vec{s} \rangle - \sum_{i=0}^{n-1} r_i \cdot s_i \\ &= \frac{Q_2}{Q_1} \cdot m + r_n - \sum_{i=0}^{n-1} r_i \cdot s_i, \end{aligned}$$

where r_i denotes rounding errors; $[a_i]_{Q_1}^{Q_2} = \frac{Q_2}{Q_1} \cdot a_i + r_i$ and $[b]_{Q_1}^{Q_2} = \frac{Q_2}{Q_1} \cdot b + r_n$. When the secret key follows a uniform ternary distribution, the variance of the noise introduced by modulus switching is $\frac{\|\vec{s}\|^2 + 1}{12}$.

3) Blind rotation Noise

The variance of noise introduced by blind rotation (σ_{ACC}^2) is given by:

$$\text{GINX/TFHE: } 2|U|n \cdot \sigma_{\otimes}^2.$$

D. Computational Complexity Analysis

The \odot RLWE' operation constitutes a computational bottleneck in FHEW-like schemes in terms of runtime. To analyze the computational complexity, we count the total number of \odot operations. The computational complexity for GINX blind rotation is given by:

$$\text{GINX/TFHE: } 2n \cdot |U|.$$

E. Failure Probability of Bootstrapping

The total noise variance of bootstrapping can be expressed as follows:

$$\sigma_{total}^2 = \frac{q^2}{Q_{ks}^2} \left(2 \frac{Q_{ks}^2}{Q^2} \sigma_{ACC}^2 + \sigma_{MS_1}^2 + \sigma_{KS}^2 \right) + \sigma_{MS_2}^2,$$

where

$$\sigma_{MS_1}^2 = \frac{\|s_N\|^2 + 1}{12}, \sigma_{MS_2}^2 = \frac{\|\vec{s}_n\|^2 + 1}{12}, \text{sigma}_{KS}^2 = \sigma^2 N d_{\text{ks}}.$$

Note that $s_N \in R_Q$ and $\vec{s}_n \in \mathbb{Z}_q^n$ denote the RLWE and LWE keys, respectively. Here, $\sigma_{MS_1}^2$ and $\sigma_{MS_2}^2$ represent the noise generated by modulus switching operations before and after key switching, respectively, and σ_{KS}^2 represents the noise introduced by the key switching procedure. When calculating the probability of failure, we consider $\|s_N\|^2$ and $\|\vec{s}_n\|^2$ under the assumption that if the secret key is binary or ternary, then $\|\vec{s}_n\| \leq \sqrt{n/2}$ and $\|s_N\| \leq \sqrt{N/2}$, respectively. If the noise of $\text{LWE}_{\vec{s}}(\frac{q}{4} \cdot m)$ exceeds $\frac{q}{8}$, decryption failure occurs. Therefore, the failure probability can be defined as $1 - \text{erf}\left(\frac{q/8}{\sqrt{2} \cdot \sigma_{total}}\right)$, where erf denotes the error function.

III. IND-CPA^D and Density of Parameter Sets

IND-CPA^D security, introduced by Li and Micciancio, represents a stronger notion than standard IND-CPA security, as it considers scenarios in which an adversary can observe the outcomes of computations performed on ciphertexts [8].

Cheon et al. [9] showed that the parameter sets used in FHEW-like schemes have a non-negligible failure probability, thereby preventing these schemes from achieving IND-CPA^D security. The probability that an adversary can successfully distinguish the message encoded in a ciphertext is given by:

$$\Pr[b = b'] = (1 - p) \cdot \Pr[b = b' | \bar{F}] + p \cdot \Pr[b = b' | F] = \frac{1}{2} + \frac{p}{2},$$

where F denotes the event of decryption failure, b' is the message predicted by the adversary, and p represents the probability of the event F . As p increases, the adversary can accumulate a sufficient number of oracle queries to mount an attack on Exact FHE. Cheon et al. further demonstrate that an adversary can perform a key-recovery attack using a decryption oracle, referred to as the KR^D attack. We refer readers to [9] for further details on IND-CPA^D and the KR^D attack on FHEW-like schemes.

A. Dense Parameter Sets Requirements in FHEW-like Schemes

Li et al. extended the classical computational security definition discussed in [15] to the (c, s) -security framework, as outlined in Definition 1. Traditionally, the computational complexity parameter c is chosen to be 128, 192, or 256 bits. However, it is important to note that the appropriate choice of statistical complexity parameter s is *application dependent* [15].

Definition 1 ($((c, s) - \text{security}$ [15]). Let Π be a cryptographic primitive, and \mathcal{G} be an indistinguishable game. Let $\text{adv}^{\mathcal{A}}$ denote the advantage of an adversary \mathcal{A} in breaking Π in \mathcal{G} . We say that Π is (c, s) -secure if for any adversary \mathcal{A} , either

$$\log_2 \frac{T(\mathcal{A})}{\text{adv}^{\mathcal{A}}} \leq c, \text{ or } \log_2 \frac{1}{\text{adv}^{\mathcal{A}}} \leq s,$$

where $T(\mathcal{A})$ is the time complexity of \mathcal{A} .

The advantage of the adversary in the attack proposed by Cheon et al. [9], as well as in other similar attacks, is directly proportional to both the failure probability and the number of queries made. Therefore, achieving a sufficiently low failure probability is essential, and dense parameter sets are required for various values of the statistical parameter s .

Alexandru et al. proposed a secure and practical approach for using FHE in applications by introducing an application-aware homomorphic encryption scheme (AAHE) [26]. AAHE requires additional information about the circuit, such as the types of operations, the order in which they are performed, and the domain of valid messages. It rejects the evaluation of unknown operations and the encryption of messages outside the known message space. This design implies that the statistical parameter s can be determined during the key generation phase. To build a practical FHE application, it is therefore necessary to fine-tune the parameters with respect to the statistical parameter s .

B. Challenges in Achieving Dense Parameter Sets

To reduce the probability of operation failure, various parameters such as n , q , N , and Q can be adjusted. However, since N must be powers of two, there is often a substantial gap between the permissible values. Although n , q , and Q do not need to be powers of two, these parameters directly influence the security level and, therefore, cannot be significantly decreased. The parameters that can be more flexibly modified include the decomposition digits d_g , and d_{ks} . However, even minor changes in these values can have a profound effect on the probability of operation failure, making it challenging to achieve satisfactory parameter configurations.

Additionally, reducing the failure probability by adjusting the parameters will inevitably result in increased runtime due to the associated rise in computational complexity. Given the limited set of feasible parameters, there may be instances where it becomes necessary to use less efficient parameters, compromising performance.

In Section IV, we introduce a blind rotation method that enables the construction of dense parameter sets. Figure 5 in Section VI illustrates the spectrum of failure probabilities achievable with our proposed blind rotation technique, which will be discussed in the following section. Furthermore, as the failure probability increases, a corresponding improvement in runtime can be expected due to the reduction in computational complexity.

C. Reliability Requirement

In AAHE, the statistical security parameter s is determined prior to the key generation phase. An additional advantage of our proposed blind rotation technique is its ability to allow the computing party to dynamically control the statistical parameter s even after key generation. This is achieved by introducing a tunable cutoff parameter t , which enables flexible trade-offs between failure probability and runtime. Such adaptability allows the computing party to optimize either performance or reliability for different circuits using the same key set, without requiring key regeneration.

We emphasize that the failure probability of homomorphic circuits scales with the number of gates in the computation. For example, in large-scale computations, such as those required for large language models, the number of gates can reach billions. In contrast, in small-scale computations the number of gates is significantly smaller. Assuming each gate failure is independent and leads to complete computational failure, the overall failure probability is given by $1 - (1 - p)^G$, where p is the failure probability per gate and G is the number of gates.

This implies that the failure probability p should be adjusted according to G . Moreover, if the circuit size varies depending on certain conditions, it is important to ensure that controlling p remains independent of the key generation phase. For example, this is the case when a server handles multiple circuits depending on client queries, or when

additional key generation is prohibitively expensive, as in multiparty scenarios.

IV. New Blind Rotation Technique

To construct practical FHE systems, fine adjustment of the statistical security parameter is necessary. However, in current FHEW-like schemes, it is difficult to adjust the failure probability—which directly affects the statistical security parameter—due to its inherent sensitivity. To address this issue and enable finer control over the failure probability while reducing computational complexity, we propose a modification of the blind rotation technique, referred to as *cutoff blind rotation*.

The proposed algorithm introduces a new parameter, the cutoff value t , to regulate the bootstrapping process. This cutoff value enables the algorithm to bypass certain $\text{RLWE} \otimes \text{RGSW}$ operations during blind rotation when $|a_i| \leq t$. While this modification may slightly increase the failure probability, it offers a practical mechanism for fine-tuning the failure probability by reducing computational complexity, thereby minimizing performance loss while maintaining IND-CPA^D security.

A. The Cutoff Blind Rotation Algorithm

As discussed in Section 2, the monomial $X^{-\langle \vec{a}, \vec{s} \rangle}$ is computed homomorphically during bootstrapping. The core idea of the cutoff blind rotation technique is to ignore terms for which a_i is sufficiently small, such that omitting the contribution of $a_i \cdot s_i$ does not significantly affect the overall value of $\langle \vec{a}, \vec{s} \rangle$. This approach leads to approximate decryption and results in increased noise. As an example, we illustrate the blind rotation and the cutoff blind rotation in Algorithms 1 and 2, respectively. In lines 4–6 of Algorithm 2, the terms with small $|a_i|$ are ignored.

Algorithm 1 Blind rotation (GINX/TFHE)

Require: Ciphertexts ct_1, ct_2 ; blind rotation key $\text{brk} = \text{RGSW}(s_i)$ with $s_i \in \{0, 1\}$

Ensure: Accumulator acc

```

1:  $ct_{\text{add}} \leftarrow ct_1 + ct_2$ 
2:  $\text{acc} \leftarrow \text{ACC\_init}(ct_{\text{add}})$ 
3: for  $i \leftarrow 1$  to  $n$  do                                 $\triangleright$  blind rotation
4:   for all  $u \in \{0, 1\}$  do
5:      $\text{acc} \leftarrow \text{acc} + (X^{ua_i} - 1) (\text{acc} \otimes \text{brk}_{i,u})$ 
6:   end for
7: end for
8: return  $\text{acc}$ 
```

However, reducing the number of RGSW multiplications lowers computational complexity and minimizes the additive noise introduced by \otimes . This implies that adjusting the cutoff value can increase the failure probability in cases where the original parameter settings yield a failure probability lower than necessary, while also improving computational efficiency. This technique increases flexibility in configura-

Algorithm 2 Cutoff blind rotation (GINX/TFHE)

Require: Ciphertexts ct_1, ct_2 ; blind rotation key $\text{brk} = \text{RGSW}(s_i)$ with $s_i \in \{0, 1\}$; cutoff value t

Ensure: Accumulator acc

```

1:  $ct_{\text{add}} \leftarrow ct_1 + ct_2$ 
2:  $\text{acc} \leftarrow \text{ACC\_init}(ct_{\text{add}})$ 
3: for  $i \leftarrow 1$  to  $n$  do                                 $\triangleright$  blind rotation
4:   if  $|a_i| \leq t$  then                                 $\triangleright$  cutoff
5:     continue
6:   end if
7:   for all  $u \in \{0, 1\}$  do
8:      $\text{acc} \leftarrow \text{acc} + (X^{ua_i} - 1) (\text{acc} \otimes \text{brk}_{i,u})$ 
9:   end for
10: end for
11: return  $\text{acc}$ 
```

ing parameters related to failure probability, allowing for a broader selection of parameter choices. Moreover, this approach affects only the statistical security parameter s , without impacting the computational security parameter c .

B. Computational Complexity Analysis

The computational complexity can be reduced by applying the cutoff blind rotation technique. The fundamental principle of cutoff blind rotation lies in omitting \odot operations. Our focus is on determining how many operations can be skipped. Given a cutoff value t , computations are omitted for a_i falling within the range $[-t, t]$. Consequently, the expected number of \odot operations during bootstrapping, considering the applied cutoff value, can be expressed as follows:

$$\text{GINX/TFHE: } 2n \cdot |U| \cdot \Omega.$$

C. Noise Analysis

Our noise analysis follows the approach outlined in [4], [12], and [13]. Using a cutoff blind rotation affects the number of \otimes operations during the blind rotation process. The noise introduced by the cutoff blind rotation consists of two components: the reduced noise from the fewer RGSW multiplication in blind rotation and the increased noise due to omitting small a_i values.

a: Reduced noise - blind rotation noise

Before conducting the noise analysis, the average number of a_i satisfying $|a_i| > t$, where t is the cutoff threshold, can be expressed as $n \cdot \left(1 - \frac{2t+1}{q}\right)$. We will denote the term $\left(1 - \frac{2t+1}{q}\right)$ as Ω for convenience throughout the rest of this work.

Theorem 1. Let (\vec{a}, b) be an LWE encryption with the secret key \vec{s} , and let \vec{a}^* be the vector whose elements are defined as $a_i^* = a_i$ if $|a_i| > t$, and $a_i^* = 0$ otherwise. Then, $\text{acc} \otimes \text{RGSW}(X^{-\langle \vec{a}^*, \vec{s} \rangle})$ can be found using the GINX blind rotation, and the noise introduced by the cutoff blind rotation

:

is

$$\sigma_{ACC^*-GINX}^2 = 2|U|n\sigma_{\otimes}^2 \cdot \Omega.$$

Proof:

Since (\vec{a}, b) is an LWE ciphertext, the coefficients of \vec{a} are uniformly distributed over the interval $[-q/2, q/2]$ [10]. Therefore, the average number of zero elements in \vec{a}^* is $n \cdot \frac{2t+1}{q}$. Consequently, the probability of skipping RGSW multiplications in the AP blind rotation is $\frac{2t+1}{q}$.

The values encrypted in the RGSW evaluation key are monomials, and the error variance is σ^2 . Thus, the noise introduced by each RGSW multiplication is σ_{\otimes}^2 , and it accumulates additively. Since the GINX blind rotation for (\vec{a}^*, b) requires $n|U|\Omega$ RGSW multiplications, the total noise introduced by the AP blind rotation is given by: $n|U|\sigma_{\otimes}^2 \cdot \Omega$. \square

b: Increased noise - cutoff noise

The noise introduced by approximating \vec{a} to \vec{a}^* given by:

$$\sigma_{CUT}^2 = \frac{t^2}{3} \cdot \|\vec{s}_n\|^2 \cdot (1 - \Omega).$$

We can consider $a_i - a_i^*$ as a random variable uniformly sampled from the interval $[-t, t]$ given that $|a_i| \leq t$. The variance of these omitted elements is $\frac{t^2}{3}$, and the probability of being omitted is $\frac{2t+1}{q} = 1 - \Omega$. We note that this noise is not generated during blind rotation or modulus switching operations. Instead, it is considered as noise that the ciphertext always inherently possesses.

c: Total failure probability of FHEW-like bootstrapping

The resulting total noise is expressed as follows:

$$\sigma_{total}^2 = \frac{q^2}{Q_{ks}^2} \left(2 \frac{Q_{ks}^2}{Q^2} \sigma_{ACC^*}^2 + \sigma_{MS_1}^2 + \sigma_{KS}^2 \right) + \sigma_{MS_2}^2 + \sigma_{CUT}^2.$$

V. Further Improvement and Application to Existing Optimization

Cutoff blind rotation is naturally compatible with various other optimization techniques. In particular, it enables the construction of practical and dense parameter sets when combined with different variants of FHEW-like schemes. We propose four applications of cutoff blind rotation in conjunction with existing optimization methods: (1) shifted mapping, (2) square-sum cutoff, (3) application with approximate gadget decomposition [5], [13], and (4) application with LWE transformation [18].

A. Shifted Mapping

We propose methods to reduce noise when the secret key is binary using cutoff blind rotation. In the cutoff blind rotation, an LWE ciphertext (\vec{a}, b) is approximated to (\vec{a}^*, b) , where $a_i^* = 0$ if $|a_i| \leq t$, and $a_i^* = a_i$ otherwise. It is important to note that the vector \vec{a}^* can be anticipated in advance. Let $\vec{a}_{diff} = \vec{a}^* - \vec{a}$, then the noise introduced by the cutoff approximation is given by:

$$(b - \langle \vec{a}, \vec{s} \rangle) - (b - \langle \vec{a}^*, \vec{s} \rangle) = \langle \vec{a}^*, \vec{s} \rangle - \langle \vec{a}, \vec{s} \rangle = \langle \vec{a}_{diff}, \vec{s} \rangle.$$

We can *precompute* the expected mean and variance of $\langle \vec{a}_{diff}, \vec{s} \rangle$ before performing blind rotation. By incorporating the expected mean into the mapping function, the failure probability can be reduced. However, it is noted that this method is only applicable when the secret key distribution is binary; specifically, when the mean of the secret key elements is non-zero.

Recall that $acc = RLWE(\sum_{i=0}^{q/2-1} f(b-i)X^i)$. For convenience, we denote $acc_{\#}$ as $RLWE(\sum_{i=0}^{q/2-1} f(b+\#-i)X^i)$. When the secret vector \vec{s} is binary, the mean of the inner product $\langle \vec{a}_{diff}, \vec{s} \rangle$ is given by $\mu = \sum_i a_{diff,i}/2$. Given μ , we perform blind rotation using $acc_{\mu} \cdot X^{-\langle \vec{a}^*, \vec{s} \rangle}$ instead of $acc \cdot X^{-\langle \vec{a}^*, \vec{s} \rangle}$.

1) Noise Analysis with Shifted Mapping

Let e denote the noise resulting from blind rotation without the proposed cutoff technique. In other words, the final decryption results in $m + e$, and a failure occurs when $|e| \geq q/8$. When the cutoff technique is applied, additional noise is introduced by approximating \vec{a} to \vec{a}^* , which we denote as e_c . Thus, decryption now fails when $|e+e_c| \geq q/8$, or equivalently, decryption succeeds when $-q/8 < e+e_c < q/8$. An important observation is that the posterior mean of e_c becomes nonzero when \vec{e} is fixed and the secret key is binary. Its mean is $\mu = \sum_i e_i/2$ as explained above.

We may assume that $X = e + e_c$ follows a Gaussian distribution of mean μ and variance σ^2 . The failure probability is then given by:

$$1 - \Pr[-q/8 < X < q/8] = \frac{1}{2} - \frac{1}{2} \operatorname{erf}\left(\frac{q/8 + \mu}{\sqrt{2}\sigma}\right) + \frac{1}{2} - \frac{1}{2} \operatorname{erf}\left(\frac{q/8 - \mu}{\sqrt{2}\sigma}\right).$$

However, when the mapping function is shifted by μ , the failure probability becomes:

$$1 - \Pr[-q/8 < X - \mu < q/8] = 1 - \operatorname{erf}\left(\frac{q/8}{\sqrt{2}\sigma}\right).$$

It is evident that $\operatorname{erf}\left(\frac{q/8}{\sqrt{2}\sigma}\right) > \frac{1}{2} \operatorname{erf}\left(\frac{q/8+\mu}{\sqrt{2}\sigma}\right) + \frac{1}{2} \operatorname{erf}\left(\frac{q/8-\mu}{\sqrt{2}\sigma}\right)$, which implies that the failure probability is reduced by applying the shifted mapping function.

It is noted that the average value of μ is zero when \vec{a} is not given. However, there is a higher probability that μ takes a non-zero value, allowing us to reduce the failure probability in most cases. In fact, μ follows a (shifted) Irwin-Hall distribution, and the probability of $\mu = 0$ is significantly low when the number of ignored coefficients is large.

B. Square-sum Cutoff

In this subsection, we enhance the proposed cutoff blind rotation to provide a more precise choice of failure probability. Instead of independently omitting elements based on a cutoff value, we use the fact that \vec{a} is given in advance, as

discussed in Subsection A, and determine the cutoff value by considering all the elements in \vec{a} collectively.

The detailed algorithm for determining the elements to be skipped is as follows. Let $(a_{(0)}, \dots, a_{(n-1)})$ be the sorted vector of $\vec{a} \in [-q/2, q/2]^n$, arranged in increasing order of a_i^2 . In other words, $a_{(k)}$ has the k -th smallest square value among all a_i s. Given a square-sum cutoff value T , we ignore $a_{(0)}, a_{(1)}, \dots, a_{(k-1)}$ for the largest k satisfying

$$\sum_{i=0}^{k-1} a_{(i)}^2 \leq T < \sum_{i=0}^k a_{(i)}^2.$$

1) Noise Analysis

Let \vec{a}^* be a vector derived from \vec{a} where the ignored terms are replaced by zero. The difference is defined as $\vec{a}_{\text{diff}} = \vec{a} - \vec{a}^*$, where the elements of \vec{a}_{diff} include $a_{(0)}, a_{(1)}, \dots, a_{(k-1)}$ and $n - k$ zero entries. Consequently, the expected noise variance σ_{CUT}^2 for a given \vec{a} can be determined as stated in the following theorem. This analysis can be straightforwardly extended to other secret key distributions.

Theorem 2. *For a given square-sum cutoff value T , the noise variance introduced by approximating \vec{a} to \vec{a}^* using square-sum cutoff is bounded by $\sigma_{CUT}^2 \leq \frac{2T}{3}$ when the secret key is ternary.*

Proof:

We aim to find the variance of the random variable

$$\langle \vec{a}_{\text{diff}}, \vec{s} \rangle = \sum_{i=0}^{k-1} a_{(i)} \cdot s_{(i)},$$

where $s_{(i)}$ represents the corresponding secret key element for $a_{(i)}$. When $s_{(i)}$ are i.i.d. random variables uniformly selected from $\{-1, 0, 1\}$, the variance is:

$$\begin{aligned} \sigma_{TH}^2 &= \text{VAR}[\langle \vec{a}_{\text{diff}}, \vec{s} \rangle] = \text{VAR} \left[\sum_{i=0}^{k-1} a_{(i)} \cdot s_{(i)} \right] \\ &= \sum_{i=0}^{k-1} a_{(i)}^2 \cdot \frac{2}{3} \leq \frac{2T}{3}. \end{aligned}$$

□

sum of the first k values of $A_{(i)}$, that is, $Z_k = \sum_{i=0}^{k-1} A_{(i)}$, for $k \geq 1$, and $Z_0 = 0$.

We can then provide the probability that the number of ignored RGSW multiplications is k as follows:

$$\begin{aligned} \Pr[K_T = k] &= \Pr[(Z_k \leq T) \wedge (Z_{k+1} > T)] \\ &= \Pr[Z_k \leq T] - \Pr[Z_{k+1} \leq T], \end{aligned}$$

for $k < n$, and $\Pr[K_T = n] = \Pr[Z_n \leq T]$. The expectation can be simplified as: $E[K_T] = \sum_{k=0}^{n-1} k (\Pr[Z_k \leq T] - \Pr[Z_{k+1} \leq T]) + n \Pr[Z_n \leq T]$. Simplifying further, we have

$$E[K_T] = \sum_{k=0}^{n-1} \Pr[Z_{k+1} \leq T] = \sum_{k=1}^n \Pr[Z_k \leq T] \quad (1)$$

Equation (1) provides the expected number of RGSW multiplications that can be ignored using the square-sum cutoff with a given cutoff value T .

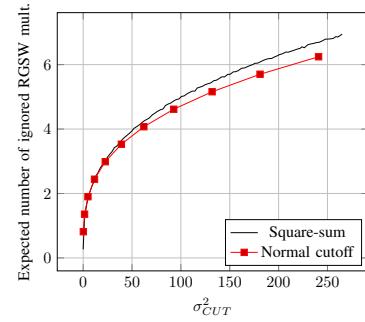


FIGURE 2. The numerical comparison of normal cutoff bootstrapping and square-sum cutoff. We used the parameter LPF STD128 in OpenFHE for this experiment.

While Equation (1) offers an analytical approach, it does not appear to yield a straightforward closed-form solution for $E[K_T]$. Therefore, we performed a numerical analysis to examine the relationship between the cutoff value T and the number of skipped elements, as shown in Figure 2. The results indicate that the square-sum cutoff achieves a superior (or comparable) tradeoff between the number of skipped RGSW multiplications and the noise introduced, compared to the standard cutoff method. Additionally, the square-sum cutoff offers more fine-grained control over the noise, making it ideal for scenarios requiring further runtime optimization.

C. Approximate Gadget Decomposition

Cutoff blind rotation can be combined with an approximate gadget decomposition technique, where the approximation $\delta B_g^{d_g^*} \approx Q$ holds for some approximation factor δ . This technique was proposed in [6], and a simplified variant operating over integers was introduced in [13]. Here, we use the gadget vector $\vec{g}_\delta = (\delta B_g^0, \delta B_g^1, \dots, \delta B_g^{d_g^*-1})$ instead of the standard gadget vector $\vec{g} = (B_g^0, B_g^1, \dots, B_g^{d_g-1})$ as used in [27]. Then, we redefine RLWE' multiplication with

approximate gadget decomposition \odot^* as follows:

$$\begin{aligned} \text{RLWE}'_\delta(\mathbf{m}) &= (\text{RLWE}(\delta B_g^0 \mathbf{m}), \dots, \text{RLWE}(\delta B_g^{d_g^*-1} \mathbf{m})) \\ \odot^* : \langle h(\mathbf{a}), \text{RLWE}'_\delta(\mathbf{m}) \rangle &= \sum_{i=0}^{n-1} \sum_{j=0}^{d_g^*-1} \text{RLWE}(\mathbf{a}_i B_g^j \delta \mathbf{m}) \\ &\approx \text{RLWE}(\mathbf{a} \cdot \mathbf{m}). \end{aligned}$$

This approach effectively ignores $\log_2 \delta$ bits during the \odot^* operation, which increases the probability of failure but reduces the number of *gadget decomposition digits* $d_g^* = d_g - 1$. Therefore, it is essential to determine an optimized value of δ that balances the trade-off between increased noise and improved computational efficiency.

1) Noise Analysis with Approximate Gadget Decomposition

The essence of the approximate gadget decomposition lies in the blind rotation process, where $\log_2 \delta$ bits are disregarded, affecting only σ_{ACC}^2 and also being dependent on the cutoff value t . By disregarding $\log_2 \delta$ bits, the noise term $\frac{\delta^2}{12} (||s_N||^2 + 1)$ arises, and for convenience, we refer this term as Λ_δ . In this case, we can define σ_{ACC}^2 based on δ as follows:

$$\sigma_{ACC_\delta^* \text{-GINX}}^2 = 2u \cdot 2n (\sigma_\odot^2 + \Lambda_\delta) \cdot \Omega.$$

2) Computational Complexity with Approximate Gadget Decomposition

The parameter d_g has a significant impact on both noise and computational complexity. Additionally, the approximate gadget decomposition technique helps minimize the value of d_g while maintaining noise levels at a minimum. By properly adjusting this parameter, d_g can be reduced to $d_g^* = d_g - 1$ with minimal noise increase. Now we can define the expected number of NTTs as follows:

$$\text{GINX} : 2n \cdot |U| \cdot d_g^* \cdot \Omega.$$

D. Cutoff Blind Rotation with Bernard et al.'s Transformation

Recently, Bernard et al. proposed a method to reduce the noise introduced by modulus switching by introducing two functions: *transform* and *quality_test* [18]. The *transform* function regenerates the nonce component of a ciphertext \vec{a} by either adding encryptions of zero or multiplying encryptions of one. Then, the *quality_test* function evaluates the expected noise variance introduced by modulus switching on the transformed nonce. These steps are repeated until the expected noise variance falls below a predefined threshold.

We emphasize that the *transform* function can be utilized in cutoff blind rotation to achieve a better trade-off between failure probability and runtime. By applying the *transform* function to \vec{a} multiple times, the expected number of a_i satisfying $|a_i| \leq t$ can be increased. Thus, the number of available parameters are reduced, as the parameter t has a

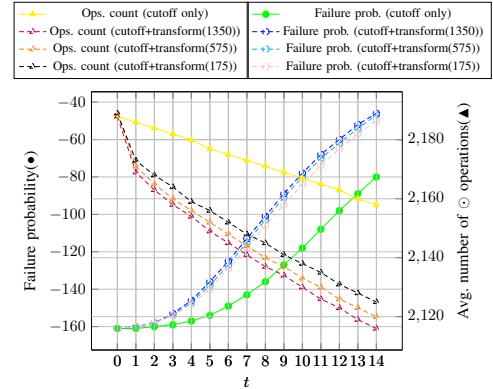


FIGURE 3. This figure illustrates the variation in failure probability and the number of \odot operations. Note that circular markers represent the failure probability, while triangular markers indicate the number of \odot operations. Solid-filled markers correspond to the case where only cutoff blind rotation is applied, whereas unfilled markers represent the case where cutoff blind rotation is applied together with the transform function in [18]. The number # in transform(#) represents the number of trials of the transform function.

greater influence on the overall noise. However, it improves performance by allowing the omission of more \otimes operations.

We illustrate in Figure 3 the variation in failure probability and the number of \odot operations, with and without the transform function. Note that the expected failure probability and the number of \odot operations can be easily estimated using the binomial distribution, since sampling \vec{a} with the transform function is i.i.d.

VI. Implementation Result and Expansion of Parameter Sets

In this section, we implement the cutoff blind rotation technique and analyze its performance. By applying this technique, we demonstrate that there are now more diverse options for selecting efficient parameters compared to previous methods. The cutoff blind rotation was implemented using the open-source library OpenFHE [21]. The parameters primarily covered in this section are four specific configurations: LPF_STD128, LPF_STD128Q, LPF_STD128_LMKCDEY, and LPF_STD128Q_LMKCDEY.

A. Bootstrapping Runtime

A significant number of iterations are necessary to observe how the bootstrapping runtime varies with different cutoff values. Instead, we fixed the cutoff value and observed the difference in bootstrapping runtime based on the number of omitted a_i that satisfied the $|a_i| \leq t$. The results are presented in Figure 4. It can be observed that as the number of omitted a_i increases, the bootstrapping runtime decreases. Note that while OpenFHE currently uses approximate gadget decomposition in a rough manner, we set the parameter δ more precisely to minimize the noise generated during the blind rotation process.

TABLE 1. This table presents the values of δ , B_g , and t for parameter sets that satisfy failure probabilities of $\text{FP} \leq 2^{-64}$, $\text{FP} \leq 2^{-96}$, and $\text{FP} \leq 2^{-128}$.

	STD128(FP64 Ours)			STD128(FP96 Ours)			STD128(FP128 Ours)			LPF_STD128 [21]		
	AP	GINX	LMK+	AP	GINX	LMK+	AP	GINX	LMK+	AP	GINX	LMK+
δ	2^{11}	2^{11}	2^{11}	2^{11}	2^{11}	2^{11}	2^9	2^9	2^{11}	2^7	2^7	2^9
B_g	2^8	2^8	2^8	2^8	2^8	2^8	2^6	2^6	2^8	2^7	2^7	2^9
t	13	13	15	0	0	11	12	12	6	0	0	0
FP	2^{-65}	2^{-65}	2^{-66}	2^{-96}	2^{-96}	2^{-96}	2^{-128}	2^{-128}	2^{-127}	2^{-267}	2^{-267}	2^{-134}
runtime	65.5 ms	44.9 ms	59.3 ms	66.0 ms	45.2 ms	60.0 ms	80.1 ms	56.1 ms	60.3 ms	80.6 ms	56.3 ms	60.8 ms

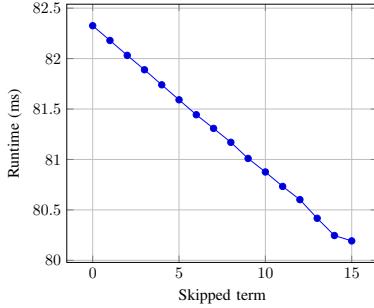


FIGURE 4. This figure shows the changes in bootstrapping runtime according to the number of a_i satisfying $|a_i| \leq t$. This is the average result obtained from performing the NAND gate operation 60,000 times. We used the AP method with the parameter LPF_STD128 with cutoff value $t = 9$, which is currently provided by OpenFHE.

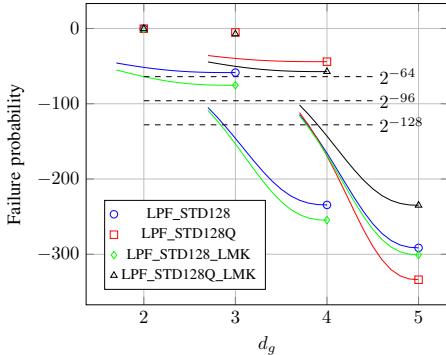


FIGURE 5. This figure illustrates the changes in failure probability as d_g and cutoff value vary. The dashed lines represent cases where the failure probability satisfies 2^{-128} , 2^{-96} , and 2^{-64} . The points marked with an empty shape represent the failure probability according to d_g , i.e., parameters that the previous method can achieve. The solid lines indicate the failure probability achievable by applying the cutoff technique.

B. Diversity of Parameter Selection

The key advantage of the cutoff blind rotation technique is that, unlike previous parameters such as d_g and d_{ks} , it allows fine-grained control over the failure probability. Moreover, it introduces greater flexibility in selecting the statistical security parameter s , which is typically determined by the specific application of FHE [26]. This flexibility enables practical deployments by offering a tunable trade-off between computational complexity and failure probability. For instance, Figure 5 shows how failure probability varies

with changes in d_g , and how it can be further adjusted by incorporating the cutoff technique.

The three failure probabilities, 2^{-128} , 2^{-96} , and 2^{-64} , are target thresholds that we have arbitrarily set as reference points. As shown in Figure 5, without the proposed technique, it is often challenging to tune parameters to satisfy a desired failure probability (e.g., $\text{FP} \leq 2^{-96}$) by adjusting only a single parameter such as d_g . However, by applying the cutoff technique, it becomes possible to modify the cutoff value to meet a given failure probability (e.g., $\text{FP} \leq 2^{-96}$), while also reducing computational complexity. For instance, when d_g is set to 4 or 5, the failure probability is already below 2^{-128} . In such cases, adjusting the cutoff value upward can bring the failure probability closer to 2^{-128} , while simultaneously reducing unnecessary computational cost.

Table 1 presents the parameters that satisfy failure probabilities of $\text{FP} \leq 2^{-64}$, $\text{FP} \leq 2^{-96}$, and $\text{FP} \leq 2^{-128}$, respectively. Note that all parameters, except for δ , B_g , and t are identical to those used in OpenFHE. Additionally, the parameter δ has been finely adjusted to efficiently utilize approximate gadget decomposition.

VII. Conclusion

We enable fine-grained parameter choices in FHEW to achieve an exact target bootstrapping failure probability, by introducing the cutoff blind rotation. Since the bootstrapping failure probability directly determines the statistical security of the scheme, achieving the desired probability is crucial for both security and system reliability. Unlike other homomorphic encryption schemes, FHEW has so far offered only limited parameter sets, often forcing the use of unnecessarily inefficient parameters to ensure statistical security. By providing a new axis of parameter optimization, our method introduces a desirable feature for FHEW-like HE schemes, balancing runtime efficiency with failure probability.

Alexandru et al. propose an AAHE scheme to build practical and secure FHE systems [26]. This approach highlights the necessity of finely tuned statistical security parameters s , which depend on the specific application, to achieve practical performance. Our method introduces a dense parameter set that allows flexible choices of the statistical security parameter s without affecting the computational security parameter c .

- Bernard et al. proposed the transform technique to reduce the bootstrapping failure probability of FHEW-like schemes from 2^{-s} to 2^{-2s} without affecting the runtime, and emphasized the importance of low failure probability for security. The failure probabilities are shifted to lower values; however, the parameter sets remain sparse. Our technique provides continuous parameter choices, bridging the gaps between parameter sets in prior works.
- The proposed method can also be applied to Torus-based [5] and NTRU-based [20], [28] variants. While these algorithms have different parameter sets and are susceptible to attacks such as the one described in [9], finding parameters with negligible failure probability remains crucial. Additionally, our technique can be extended to blind rotations with higher bits, as seen in [14], [29]. Future work could explore applying the proposed method to amortized bootstrapping techniques [30]–[33] for further optimization.
- We enable continuous fine-tuning of the failure probability; however, its impact on runtime is not significant, as shown in Table 1 (with up to a 2.5% reduction in runtime). Nevertheless, we emphasize that the critical challenge lies in identifying dense parameter sets—particularly with respect to the statistical security parameter s —to enable practical FHE systems across diverse applications. Developing new methods for fine-grained parameter selection with improved trade-offs remains an important direction for future work.

REFERENCES

- [1] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) Fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.
- [2] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical GapSVP,” in *Advances in Cryptology – CRYPTO 2012*. Springer, 2012, pp. 868–886.
- [3] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” *IACR Cryptol. ePrint Arch.*, p. 144, 2012.
- [4] L. Ducas and D. Micciancio, “FHEW: bootstrapping homomorphic encryption in less than a second,” in *Advances in Cryptology - EUROCRYPT*. Springer, 2015, pp. 617–640.
- [5] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, “Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE,” in *Advances in Cryptology – ASIACRYPT 2017*. Springer, 2017, pp. 377–408.
- [6] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “TFHE: Fast fully homomorphic encryption over the torus,” *Journal of Cryptology*, pp. 34–91, 2020.
- [7] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology – ASIACRYPT 2017*. Springer, 2017, pp. 409–437.
- [8] B. Li and D. Micciancio, “On the security of homomorphic encryption on approximate numbers,” in *Advances in Cryptology – EUROCRYPT 2021*. Springer, 2021, pp. 648–677.
- [9] J. H. Cheon, H. Choe, A. Passelègue, D. Stehlé, and E. Suvanto, “Attacks against the ind-cpd security of exact fhe schemes,” in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2505–2519. [Online]. Available: <https://doi.org/10.1145/3658644.3690341>
- [10] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.
- [11] V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and learning with errors over rings,” *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–35, 2013.
- [12] D. Micciancio and Y. Polyakov, “Bootstrapping in FHEW-like cryptosystems,” in *WAHC’21*, 2021, pp. 17–28.
- [13] Y. Lee, D. Micciancio, A. Kim, R. Choi, M. Deryabin, J. Eom, and D. Yoo, “Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption,” in *Advances in Cryptology – EUROCRYPT 2023*. Springer, 2023, pp. 227–256.
- [14] L. Bergerat, A. Boudi, Q. Bourgerie, I. Chillotti, D. Ligier, J.-B. Orfila, and S. Tap, “Parameter optimization and larger precision for (T)FHE,” *Journal of Cryptology*, vol. 36, 2023.
- [15] B. Li, D. Micciancio, M. Schultz-Wu, and J. Sorrell, “Securing approximate homomorphic encryption using differential privacy,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 560–589.
- [16] M. Checri, R. Sirdey, A. Boudguiga, and J.-P. Bultel, “On the practical CPAD security of “exact” and threshold FHE schemes and libraries,” in *Annual International Cryptology Conference*. Springer, 2024, pp. 3–33.
- [17] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson, “Key recovery attacks on approximate homomorphic encryption with Non-Worst-Case noise flooding countermeasures,” in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 7447–7461. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity24/presentation/guo-qian>
- [18] O. Bernard, M. Joye, N. P. Smart, and M. Walter, “Drifting towards better error probabilities in fully homomorphic encryption schemes,” in *Advances in Cryptology – EUROCRYPT 2025*. Springer, 2025, pp. 181–211.
- [19] Zama, “TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data,” 2022, <https://github.com/zama-ai/tfhe-rs>.
- [20] C. Bonte, I. Iliašenko, J. Park, H. V. L. Pereira, and N. P. Smart, “Final: Faster the instantiated with NTRU and LWE,” in *Advances in Cryptology – ASIACRYPT 2022*, 2022, pp. 188–215.
- [21] OpenFHE, “Open-Source Fully Homomorphic Encryption Library,” <https://github.com/openfheorg/openfhe-development>, 2022.
- [22] C. Gentry, A. Sahai, and B. Waters, “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based,” in *Advances in Cryptology – CRYPTO 2013*. Springer, 2013, pp. 75–92.
- [23] J. Alperin-Sheriff and C. Peikert, “Faster bootstrapping with polynomial error,” in *Advances in Cryptology – CRYPTO*. Springer, 2014, pp. 297–314.
- [24] N. Gama, M. Izabachene, P. Q. Nguyen, and X. Xie, “Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems,” in *Advances in Cryptology - EUROCRYPT*. Springer, 2016, pp. 528–558.
- [25] M. Joye and P. Paillier, “Blind rotation in fully homomorphic encryption with extended keys,” in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2022, pp. 1–18.
- [26] A. Alexandru, A. Al Badawi, D. Micciancio, and Y. Polyakov, “Application-aware approximate homomorphic encryption: Configuring fhe for practical use,” *Cryptology ePrint Archive*, 2024.
- [27] A. Kim, Y. Lee, M. Deryabin, J. Eom, and R. Choi, “LFHE: Fully homomorphic encryption with bootstrapping key size less than a megabyte,” *Cryptology ePrint Archive*, 2023.
- [28] B. Xiang, J. Zhang, Y. Deng, and D. Feng, “Fast blind rotation for bootstrapping fhes,” in *Advances in Cryptology – CRYPTO 2023*, H. Handschuh and A. Lysyanskaya, Eds., 2023, pp. 3–36.
- [29] Z. Liu, D. Micciancio, and Y. Polyakov, “Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping,” in *Advances in Cryptology-ASIACRYPT 2022*. Springer, 2022, pp. 130–160.
- [30] F.-H. Liu and H. Wang, “Batch bootstrapping i: A new framework for simd bootstrapping in polynomial modulus,” in *Advances in Cryptology – EUROCRYPT 2023*, 2023, p. 321–352.
- [31] ———, “Batch bootstrapping ii: Bootstrapping in polynomial modulus only requires o(1) FHE multiplications in amortization,” in *Advances in Cryptology – EUROCRYPT 2023*, 2023, p. 353–384.
- [32] Z. Liu and Y. Wang, “Amortized functional bootstrapping in less than 7 ms, with o(1) polynomial multiplications,” in *Advances in Cryptology - ASIACRYPT*, 2023, pp. 101–132.
- [33] G. D. Micheli, D. Kim, D. Micciancio, and A. Suhl, “Faster amortized FHEW bootstrapping using ring automorphisms,” *Cryptology ePrint Archive*, 2023.