A New Fine Tuning Method for FHEW/TFHE Bootstrapping with IND-CPA$^{\mathrm{D}}$ Security

Deokhwa Hong[1], Young-Sik Kim[2], Yongwoo Lee[1], and Eunyoung Seo[2]

[1]Inha University, Incheon, Republic of Korea
`deokhwa@inha.edu, yongwoo@inha.ac.kr`
[2]Daegu Gyeongbuk Institute of Science and Technology, Daegue, Republic of Korea
`eunyoung00@gmail.com, ysk@dgist.ac.kr`

Abstract

Fully homomorphic encryption (FHE) schemes enable computations on encrypted data, making them as a crucial component of privacy-enhancing technologies. Ducas and Micciancio introduced the FHEW scheme (Eurocrypt '15), which was further enhanced by Chillotti et al. with TFHE (Asiacrypt '17). These schemes support low-latency homomorphic evaluations of binary (or larger) gates due to their small parameter size. However, the evaluation failure probability in these schemes is highly sensitive to the choice of parameters, resulting in a limited range of viable parameters and a trade-off between failure probability and runtime.

Recently, Cheon et al. proposed a key recovery attack on the FHEW/TFHE schemes based on a novel security model for FHE, known as IND-CPA$^{\mathrm{D}}$ security (CCS '24). Mitigating this attack requires achieving a negligible failure probability (e.g., 2^{-64}). However, the limited range of parameter options in FHEW/TFHE necessitates the adoption of parameter sets with unnecessarily low failure probabilities, leading to inefficient runtime.

We propose a new bootstrapping method for the FHEW/TFHE shcemes that optimizes the trade-off between runtime and failure probability while maintaining ease of implementation. The proposed method allows selecting parameter sets that achieve the desired failure probabilities at various security levels, thereby maximizing runtime efficiency.

Keywords. Homomorphic encryption, key recovery attack, bootstrapping, cutoff Blind Rotation

1 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 [BGV14, Bra12, FV12], while the FHEW and TFHE schemes [DM15, CGGI17, CGGI20] enable operations on logic circuits. These FHE schemes are categorized as *Exact FHE*. In addition, the Cheon-Kim-Kim-Song (CKKS) scheme, introduced in [CKKS17], 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$^{\mathrm{D}}$) [LM21]. In contrast to the traditional IND-CPA model, this variant allows an adversary to observe the results of computations performed on the ciphertext. It has been shown that the IND-CPA$^{\mathrm{D}}$ 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[LM21].

Cheon et al. [CCP^{+}24] demonstrated that Exact FHE schemes, including BGV/BFV and FHEW/TFHE, also fail to meet IND-CPA$^{\mathrm{D}}$ security due to evaluation failure events. Moreover, 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.

There are three main bootstrapping methods in FHEW-like schemes. The first, introduced by Ducas and Micciancio, builds on the Alperin-Sheriff and Peikert (AP) method [AP14], which they incorporated into their FHEW scheme [DM15]. This method delivers consistent performance regardless of the key distribution. The second method, used in the TFHE scheme [CGGI17] proposed by Chillotti et al., employs the Gama-Izabachene-Nguyen-Xie (GINX) technique [GINX16], which offers superior performance but is limited to binary key distributions. Although GINX/TFHE bootstrapping can be generalized to support arbitrary secret keys [MP21, JP22], 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 [LMK^{+}23].

1.1 Failure Probability of FHEW/TFHE and IND-CPA$^{\mathrm{D}}$ Security

The FHEW/TFHE schemes are based on the Learning with Errors (LWE) problem and its ring variant, Ring-LWE (RLWE)[LPR13]. Since ciphertexts inherently contain noise that accumulates with each homomorphic operation, decryption will fail if the noise exceeds a certain threshold. To date, FHEW/TFHE schemes have been designed to maintain a low failure probability (preferably less than 2^{-40}-2^{-50} [DM15, CGGI17, MP21, LMK^{+}23, BBB^{+}23]) while ensuring acceptable performance.

Cheon et al. proposed a key-recovery (KR$^{\mathrm{D}}$) attack, which is a relaxation of IND-CPA$^{\mathrm{D}}$, against FHEW/TFHE schemes in the existence of a decryption oracle for the result of a homomorphic operation model [CCP^{+}24]. When an adversary can access the decrypted value of a resulting ciphertext from a queried computation, it can detect homomorphic operation failures. There is a noticeable difference in the distribution of LWE ciphertext elements when the corresponding secret key is 0 or 1 upon failure. The adversary can exploit this information to recover the secret key, and a polynomial-time attack becomes feasible with a sufficient (but constant) 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 [LMSWS22]; 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* [LMSWS22], 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 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 ciphertext 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 transformation (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* [LMK+23, MP21]. Unfortunately, even minor changes in d can significantly impact both the failure probability and computational complexity, thereby limiting viable parameters. It is thus challenging to perfectly fit the bootstrapping failure probability to a specific target such as 2^{-64} without uselessly losing efficiency.

1.2 Our Contribution

We propose a novel bootstrapping method for FHEW/TFHE that allows for smooth adjustment of the failure probability, optimizing the balance between runtime and failure probability. In FHEW/TFHE schemes, the range of adjustable parameters is inherently limited, and even minor adjustments can drastically affect failure probability and computational complexity. As a result, homomorphic operations can be performed using parameters that lead to unnecessarily low failure probabilities, causing inefficiencies in both runtime and complexity.

To mitigate these inefficiencies, we introduce a novel blind rotation technique that optimizes bootstrapping in FHEW-like HE schemes by omitting operations with minimal impact on noise. Blind rotation involves the homomorphic execution of the decryption operation, $\boldsymbol{f} \cdot X^{b-\langle \vec{a}, \vec{s} \rangle}$, on a given ciphertext $(b, \vec{a}) \in \mathbb{Z}_q^{n+1}$. In our proposed method, the operation corresponding to $\langle \vec{a}, \vec{s} \rangle$ is performed for the secret key $\vec{s}$, introducing a cutoff value t. This cutoff omits operations for a_i and s_i when $|a_i| \le t$, effectively reducing the number of expensive ciphertext multiplications required during blind rotation.

Figure 1 illustrates how the failure probability and the number of multiplications change as the cutoff value t is varied in the parameter settings used by OpenFHE [Ope22], an open-source library for fully homomorphic encryption. We note that it is straightforward to apply the proposed method to other FHEW/TFHE implementations as well [Zam22, BIP+22].

Although the introduction of the new parameter t may increase the probability of operation failure, it significantly reduces computational complexity by omitting operations that contribute minimally to the overall noise. Importantly, t can be freely adjusted within the range $[0, q/2)$, and

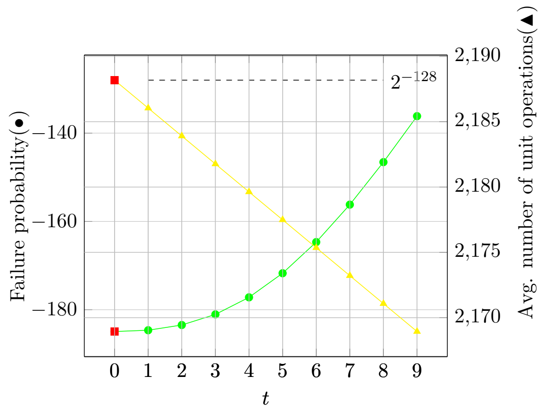

Figure 1: The number of unit operations and failure probability by cutoff parameter t. Note that prior arts can only achieve failure probability and runtime marked in red squares. The parameter set is borrowed from OpenFHE (LPF_STD128).

its effect on failure probability is considerably less pronounced than that of other parameters, such as decomposition digits. 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 minimizing runtime.

As discussed in Section 3.2, the failure probability of bootstrapping in FHEW-like HE 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 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 HE-based 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 [Ope22], 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 gates, the proposed technique is also applicable to larger key spaces.

1.3 Organization

The rest of the paper is organized as follows. The basic lattice-based HE and the prior FHEW/TFHE bootstrapping techniques are presented in Section 2. The IND-CPA$^{\mathrm{D}}$ and key recovery attack proposed on FHEW/TFHE are discussed in Section 3. In Section 4, we propose a new optimization technique for FHEW/TFHE bootstrapping, which is especially efficient for IND-CPA$^{\mathrm{D}}$-secure parameters. Improvements to the proposed method are detailed in Section 5. Implementation results and secure parameters are provided in Section 6. Finally, we conclude the paper with remarks in Section 7.

2 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 $\boldsymbol{a}(X)$, and X is omitted when the context is clear. The i-th coefficient of ring element $\boldsymbol{a}$ is denoted as a_i. Similarly, vectors are denoted using boldface, such as $\boldsymbol{v}$, and the i-th element of a vector $\boldsymbol{v}$ is denoted by v_i. The $L2$ norm of ring elements or vectors is represented as $|| \cdot ||$, and the infinity norm as $|| \cdot ||_{\text{inf}}$. 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 $x \leftarrow S$.

2.1 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 $\vec{s} \leftarrow \chi_{\text{sk}}$, the error term is sampled as $e \leftarrow \chi_{\text{err}}$, 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 $\boldsymbol{m}$ with secret key $\boldsymbol{z} \leftarrow \chi_{sk}$ as follows:

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

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

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

2.1.1 Basic Operations for RLWE Ciphertext

The basic building block of FHEW bootstrapping is RLWE$'$ and Ring-GSW (RGSW) and their multiplication with ring element and RLWE ciphertext [GSW13, DM15]. We follow the definitions of $\text{RLWE}'_{\boldsymbol{z}}(\boldsymbol{m})$ and $\text{RGSW}_{\boldsymbol{z}}(\boldsymbol{m})$ from [MP21]:

$$\begin{aligned}
\text{RLWE}'_{\boldsymbol{z}}(\boldsymbol{m}) &:= \left(\text{RLWE}_{\boldsymbol{z}}(g_0 \cdot \boldsymbol{m}), \text{RLWE}_{\boldsymbol{z}}(g_1 \cdot \boldsymbol{m}), \cdots, \text{RLWE}_{\boldsymbol{z}}(g_{d_g-1} \cdot \boldsymbol{m})\right) \in \mathcal{R}_Q^{2d_g} \\
\text{RGSW}_{\boldsymbol{z}}(\boldsymbol{m}) &:= \left(\text{RLWE}'(-\boldsymbol{z} \cdot \boldsymbol{m}), \text{RLWE}'(\boldsymbol{m})\right) \in \mathcal{R}_Q^{2d_g \times 2},
\end{aligned}$$

where $\boldsymbol{g} = (g_0, g_1, \cdots, g_{d_g-1})$ is a gadget vector, which is used in gadget decomposition. We say $h(\boldsymbol{a}) = (\boldsymbol{a}_0, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_{d_g-1})$ is gadget decomposition of $\boldsymbol{a} \in \mathcal{R}_Q$ if $\boldsymbol{a} \approx \sum_{i=0}^{d_g-1} g_i \cdot \boldsymbol{a}_i$, where $||\boldsymbol{a}||_{\text{inf}} < B_g$ 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 $\odot : \mathcal{R}_Q \times \text{RLWE}' \to \text{RLWE}$:

$$\begin{aligned}\boldsymbol{a} \odot \text{RLWE}' &= \langle (\boldsymbol{a}_0, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_{d_g-1}), \big(\text{RLWE}_{\boldsymbol{z}}(g_0 \cdot \boldsymbol{m}), \cdots, \text{RLWE}_{\boldsymbol{z}}(g_{d_g-1} \cdot \boldsymbol{m})\big)\rangle \\ &= \sum_{i=0}^{d_g-1} \boldsymbol{a}_i \cdot \text{RLWE}_{\boldsymbol{z}}(g_i \cdot \boldsymbol{m}) = \text{RLWE}_{\boldsymbol{z}}\left(\boldsymbol{h} \cdot \boldsymbol{m}\right).\end{aligned}$$

The multiplication between RLWE and RGSW ($\otimes$: RLWE $\times$ RGSW $\to$ RLWE) is defined as follows:

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

This operation outputs an RLWE encryption of $\boldsymbol{m}_0 \cdot \boldsymbol{m}_1 + \boldsymbol{e}_1 \cdot \boldsymbol{m}_1$. It is worth noting that the noise term $\boldsymbol{m}_1 \cdot \boldsymbol{e}_1$ remains small because $\boldsymbol{m}_1$ 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 $d_g N \frac{B_g^2}{12}$ and $2 d_g N \frac{B_g^2}{12}$, respectively [MP21].

2.1.2 Ring Automorphism

We define the ring automorphism $\psi_k : \mathcal{R}_Q \mapsto \mathcal{R}_Q$ as $\psi_k(m(X)) = m(X^k)$ using RLWE$'$. Given RLWE ciphertext $\text{RLWE}_{\boldsymbol{z}}(\boldsymbol{m}) = (\boldsymbol{a}, \boldsymbol{b})$, we can find $\text{RLWE}_{\boldsymbol{z}(X^k)}(\boldsymbol{m}(X^k))$, by $(\boldsymbol{a}(X^k), \boldsymbol{b}(X^k))$. However, its secret key is now $\boldsymbol{z}(X^k)$ rather than $\boldsymbol{z}(X)$. We can revert the secret key back to $\boldsymbol{z}(X)$ when the automorphism key $\text{RLWE}'_{\boldsymbol{z}(X)}(-\boldsymbol{z}(X^k))$ is given using following equation:

$$\begin{aligned}(0, \boldsymbol{b}(X^k)) + \boldsymbol{a}(X^k) \odot \text{RLWE}'_{\boldsymbol{z}(X)}(-\boldsymbol{z}(X^k)) &= (0, \boldsymbol{b}(X^k)) + \text{RLWE}_{\boldsymbol{z}(X)}(-\boldsymbol{a}(X^k) \cdot \boldsymbol{z}(X^k)) \\ &= \text{RLWE}_{\boldsymbol{z}}(\boldsymbol{b} - \boldsymbol{a}(X^k) \cdot \boldsymbol{z}(X^k)) \\ &= \text{RLWE}_{\boldsymbol{z}}(\boldsymbol{m}(X^k)).\end{aligned}$$

The automorphism key $\text{RLWE}'_{\boldsymbol{z}(X)}(-\boldsymbol{z}(X^k))$ is typically provided as a public evaluation key for automorphism operations, denoted as ak_k.

2.1.3 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 $\vec{s} \in \mathbb{Z}_q^n$. This operation introduces additional noise. To perform key switching, one must define a key-switching key (ksk) as follows:

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

where $\text{ksk}_{i,j,v} = \text{LWE}_{\vec{s}}\left(-v \cdot z_i \cdot B_{\text{ks}}^j\right)$ and B_{ks} and d_{ks} are the base and the digit length of the gadget decomposition for key switching, respectively, satisfying $B_{\text{ks}}^{d_{\text{ks}}} \geq q$.

The key-switching operation is defined as:

$$\texttt{KeySwitch}\left((\vec{a}, b), \text{ksk}\right) = (\vec{0}, b) + \sum_{i,j} \text{ksk}_{i,j,a_{i,j}}, \tag{1}$$

where the gadget decomposition of a_i, denoted as $h(a_i)$, is given by $(a_{i,0}, a_{i,1}, \ldots, a_{i,d_{\mathrm{ks}}-1})$. As the sum of these terms can be expressed as $\sum_{i,j} \mathrm{ksk}_{i,j,a_{i,j}} = \sum_{i,j} \mathrm{LWE}_{\boldsymbol{s}}(-a_{i,j} \cdot B_{\mathrm{ks}}^{j} \cdot z_i) = \mathrm{LWE}_{\boldsymbol{s}}\left(-\langle \vec{a}, \vec{z} \rangle\right)$ and $(0, b)$ is a transparent ciphertext of b, (1) is equal to

$$\mathrm{LWE}_{\boldsymbol{s}}(b) + \mathrm{LWE}_{\boldsymbol{s}}\left(-\langle \vec{a}, \vec{z} \rangle\right) = \mathrm{LWE}_{\boldsymbol{s}}\left(b - \langle \vec{a}, \vec{z} \rangle\right) = \mathrm{LWE}_{\boldsymbol{s}}(m).$$

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

2.1.4 Extraction of LWE Ciphertext from RLWE Ciphertext

An LWE ciphertext that contains only the constant term of the message can be extracted from an RLWE ciphertext [DM15]. This operation, known as LWE extraction, is used in FHEW-like HE to convert the resulting RLWE ciphertext from blind rotation into an LWE ciphertext. The LWE extraction operation is defined as follows:

$$\texttt{LWEExtract} : \mathcal{R}_Q^2 \mapsto \mathbb{Z}_Q^{N+1}, \ \texttt{LWEExtract}\left((\boldsymbol{a}, \boldsymbol{b})\right) = \left(\boldsymbol{a'}, b_0\right) \in \mathbb{Z}_Q^{N+1},$$

where $\boldsymbol{a'} = (a_0, -a_1, -a_2, \cdots, -a_{N-1})$. By definition, $(\boldsymbol{a'}, b_0)$ is the LWE ciphertext containing the constant term of the RLWE ciphertext $(\boldsymbol{a}, \boldsymbol{b})$.

2.2 Bootstrapping in FHEW/TFHE

Initially, the operation defined by the gate is performed on two LWE-encrypted ciphertexts. Subsequently, the process known as blind rotation is applied. Blind rotation involves multiplying the ring element $\boldsymbol{f}$ and the monomial X^u, where $u = -\langle \vec{a}, \vec{s} \rangle$ is determined by the input LWE ciphertext $(\vec{a}, b)$.

Blind rotation is performed on the accumulator RLWE ciphertext, which is initialized as: $\mathrm{Acc} = \mathrm{RLWE}\left(\boldsymbol{f} \cdot X^b\right)$, and is iteratively updated using RLWE $\otimes$ RGSW operations. Once the blind rotation is completed, the following steps are applied to transform the result back: LWE extraction, modulus switching, LWE key switching, and another modulus switching step. During LWE extraction, the constant term of the RLWE polynomial is extracted, resulting in an LWE ciphertext that encrypts the $-u$-th coefficient of $\boldsymbol{f}$, i.e., $\mathrm{LWE}_z(f_{-u})$. The first modulus switching operation transforms: $\mathbb{Z}_Q^{N+1} \xrightarrow{\text{mod switching}} \mathbb{Z}_{Q_{ks}}^{N+1}$, which is followed by key switching: $\mathbb{Z}_{Q_{ks}}^{N+1} \xrightarrow{\text{key switching}} \mathbb{Z}_{Q_{ks}}^{n+1}$, and finally, the last modulus switching is performed: $\mathbb{Z}_{Q_{ks}}^{n+1} \xrightarrow{\text{mod switching}} \mathbb{Z}_q^{n+1}$. The complete bootstrapping procedure is illustrated in Figure 2.

This paper follows the procedure outlined in OpenFHE for noise analysis; however, various modifications to this procedure exist [Ope22, MP21]. For instance, the conversion from RLWE to LWE could be delayed as in [LMK$^+$23], which helps reduce both noise and failure probability with minimal computational overhead. Additionally, using Torus LWE [CGGI17], instead of LWE defined over integers, allows for bypassing the first modulus switching. Another option is to employ an LWE key-switching key with non-standard noise variance to bypass the initial modulus switching step [DM15]. For enhanced efficiency, NTRU-based FHEW-like HE [BIP$^+$22] may also be employed.

We note that the proposed method can be seamlessly applied to all of these variants, and the specific approach adopted in this manuscript was chosen for clarity of explanation.

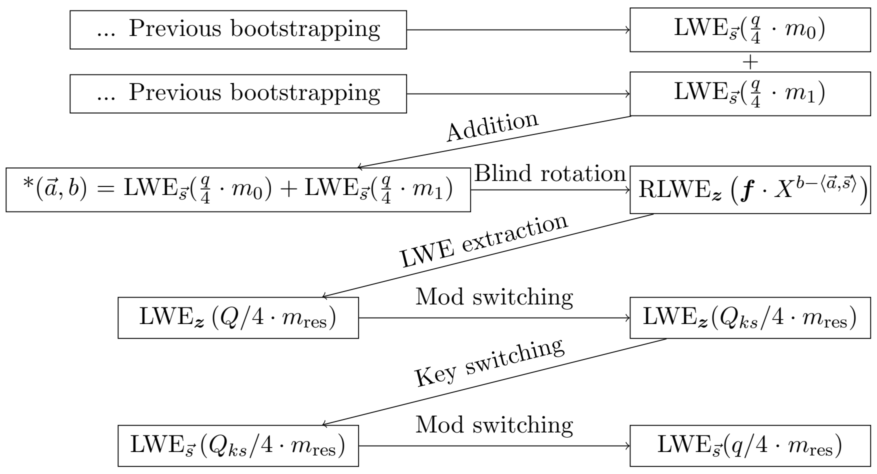
... Previous bootstrapping
$\mathrm{LWE}_{\vec{s}}(\frac{q}{4} \cdot m_0)$
+
... Previous bootstrapping
$\mathrm{LWE}_{\vec{s}}(\frac{q}{4} \cdot m_1)$
Addition
$*(\vec{a}, b) = \mathrm{LWE}_{\vec{s}}(\frac{q}{4} \cdot m_0) + \mathrm{LWE}_{\vec{s}}(\frac{q}{4} \cdot m_1)$
Blind rotation
$\mathrm{RLWE}_{\boldsymbol{z}}\left(\boldsymbol{f} \cdot X^{b-\langle \vec{a}, \vec{s} \rangle}\right)$
LWE extraction
$\mathrm{LWE}_{\boldsymbol{z}}\,(Q/4 \cdot m_{\mathrm{res}})$
Mod switching
$\mathrm{LWE}_{\boldsymbol{z}}(Q_{ks}/4 \cdot m_{\mathrm{res}})$
Key switching
$\mathrm{LWE}_{\vec{s}}\,(Q_{ks}/4 \cdot m_{\mathrm{res}})$
Mod switching
$\mathrm{LWE}_{\vec{s}}(q/4 \cdot m_{\mathrm{res}})$

Figure 2: FHEW-like bootstrapping procedure. It is noted that the relative noise is the greatest in the block marked with a star.

2.3 Noise Analysis

Our noise analysis is based on the methodologies described in [DM15], [MP21], and [CGGI20]. 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. The failure probability should be evaluated at the step where the noise is maximized. As shown in Figure 2, the noise level is highest after the addition of the two input ciphertexts and before the blind rotation (indicated by a star).

2.3.1 Modulus Switching Noise

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 as $[\cdot]_{Q_1}^{Q_2} : \mathbb{Z}_{Q_1} \to \mathbb{Z}_{Q_2}$ is defined as:

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

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

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

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

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

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 given by: $\frac{||\vec{s}||^2+1}{12}$.

2.3.2 Blind Rotation Noise

Blind rotation involves performing the decryption operation on encrypted data, represented as $\boldsymbol{f} \cdot X^{b-\langle \vec{a}, \vec{s} \rangle}$. The blind rotation keys, denoted as brk, consist of RGSW-encrypted values. This process decrypts the input ciphertext while separating the coefficients and exponents of polynomials, resulting in a ciphertext whose noise is independent of that in the input ciphertext.

There are three primary blind rotation methods: AP/FHEW, GINX/TFHE, and LMKCDEY. The performance of each method varies depending on the key distribution, and further details can be found in [MP21, LMK+23]. The blind rotation keys for each method are defined as follows:

$$\begin{aligned} \text{AP/FHEW: } & \left\{ \text{brk}_{i,v,j} = \text{RGSW}\left(X^{v \cdot B_r^j \cdot s_i} \right) | v \in \mathbb{Z}_{B_r}, 0 \leq j < \log_{B_r} q \right\} \\ \text{GINX/TFHE: } & \left\{ \text{brk}_{i,u} = \text{RGSW}\left(x_{i,u} \right) | \boldsymbol{x_i} \in \{0,1\}^{|U|} \text{ s.t. } \sum_{u \in U} u \cdot x_{i,u} = s_i \right\} \\ \text{LMKCDEY: } & \left\{ \text{brk}_i = \text{RGSW}\left(X^{s_i} \right) | i \in [0, n) \right\}, \{ \text{ak}_{-1}, \text{ak}_{5^k} | 1 \leq k < w \}, \end{aligned}$$

where w is a small integer, typically around $\log(n)$, and $U \subset \mathbb{Z}_q$. For example, $U = \{1, -1\}$ is suitable for ternary secret keys, while $U = \{1\}$ is used for binary secret keys.

Using the defined blind rotation keys, the blind rotation is performed by accumulating $X^{a_i s_i}$ into the accumulator (Acc) using the following operations:

$$\begin{aligned} \text{AP/FHEW: Acc} & \longleftarrow \text{Acc} \otimes \text{brk}_{i,v,j} \text{ for all } 0 \leq j < \log_{B_r} q \\ \text{GINX/TFHE: Acc} & \longleftarrow \text{Acc} + (X^{u \cdot a_i} - 1)(\text{Acc} \otimes \text{brk}_{i,u}), \end{aligned}$$

where Acc is initialized as RLWE($\boldsymbol{f} \cdot X^b$). In the AP/FHEW method, the $\otimes$ operation must be repeated nd_r times, where d_r is an AP-specific parameter used to decompose the LWE input $\vec{a}$ such that $B_r^{d_r} \approx q$. In the GINX/TFHE method, the operation needs to be repeated $2|U|n$ times, where $|U|$ is the size of U. [1]

The LMKCDEY bootstrapping method, in contrast to the previous approaches, utilizes ring automorphisms ψ_t and an automorphism key ak_t, where t is an odd number. In summary, the LMKCDEY bootstrapping technique supports arbitrary secret key distributions and achieves a runtime comparable to that of GINX/TFHE while generating less noise than both the AP and GINX methods. This approach relies on two core operations: 1) performing constant multiplication in the exponent using 2) adding s_i to the exponent by multiplying with RGSW(X^{s_i}). For additional information on this bootstrapping technique, readers are referred to [LMK+23].

The variance of noise introduced by blind rotation is thus given by:

$$\begin{aligned} \text{AP/FHEW: } & 2nd_r \cdot d_g N \frac{B_g^2}{12} \sigma^2 \\ \text{GINX/TFHE: } & 2|U| \cdot 2n \cdot d_g N \frac{B_g^2}{12} \sigma^2 \\ \text{LMKCDEY: } & d_g N \frac{{B_g}^2}{12} \left(2n \cdot \sigma^2 + \left(k + \frac{N-k}{w} \right) \cdot \sigma^2 \right), \end{aligned}$$

where the factor of 2 is introduced because the RGSW scheme is composed of tuples, and the term $\frac{B_g}{12}$ appears because the RGSW scheme represents the message in a gadget-decomposed form, which

[1]For a more generalized approach to GINX/TFHE bootstrapping, see [JP22].

corresponds to uniform sampling from the interval $\left[-\frac{B_g}{2}, \frac{B_g}{2}\right]$. Here, k represents the expected number of non-empty index sets $I_\ell^\pm = \{i : a_i = \pm g^\ell\}$ for g, where $\langle g, -1\rangle$ generates $\mathbb{Z}_{2N}^*$, which is expressed as $\min(N, n)$ in the worst-case and as $N\left(1 - \left(1 - \frac{1}{N}\right)^n\right) \approx N\left(1 - e^{-n/N}\right)$ in the average-case [LMK+23].

3 IND-CPA$^{\mathrm{D}}$ and KR$^{\mathrm{D}}$ Attack Exploiting Bootstrapping Failure

IND-CPA$^{\mathrm{D}}$, introduced by Li and Micciancio, is a more stringent security notion than IND-CPA, taking into account scenarios in which an adversary can observe the results of computations performed on ciphertexts [LM21]. Recently, Cheon et al. demonstrated that the current parameter sets used in FHEW/TFHE exhibit a non-negligible failure probability, preventing these schemes from achieving IND-CPA$^{\mathrm{D}}$ security [CCP+24].

This section explains why the non-negligible failure probability leads to the failure of IND-CPA$^{\mathrm{D}}$ security and discusses the key recovery attack under decryption oracle (KR$^{\mathrm{D}}$) that exploits this vulnerability, as well as the necessity of fine-tuning the failure probability to mitigate this issue.

3.1 IND-CPA$^{\mathrm{D}}$ and KR$^{\mathrm{D}}$ Attack

The KR$^{\mathrm{D}}$ attack, proposed by Cheon et al., leverages computational failures induced by noise in Exact FHE schemes [CCP+24]. Their study revealed that the parameters used in many libraries for Exact FHE do not meet IND-CPA$^{\mathrm{D}}$ security, highlighting the need to reduce the noise generated during homomorphic operations. In this section, we briefly explore how KR$^{\mathrm{D}}$ attacks are conducted and why they fail to satisfy IND-CPA$^{\mathrm{D}}$. For a comprehensive analysis of the techniques involved, we refer readers to [CCP+24].

In IND-CPA$^{\mathrm{D}}$, the attacker possesses an oracle not only for encryption and decryption but also for evaluation. Each oracle is defined according to the following algorithms, where G represents any binary circuit (such as NAND), DB is a database used to store oracle outputs, and I denotes input wires as indices.

Algorithm 1: $\mathcal{O}_{\mathrm{ENC}}(m_0, m_1; \mathrm{pk}, b)$ [CCP+24]

```
1 $\mathrm{ct} \longleftarrow \mathrm{ENC}_{\mathrm{pk}}(m_b)$
2 $\mathrm{DB}_i \longleftarrow \{m_0, m_1, \mathrm{ct}\}$
3 $i \leftarrow i + 1$
4 return ct
```

Algorithm 2: $\mathcal{O}_{\mathrm{DEC}}(i; \mathrm{sk})$ [CCP+24]

```
1 if $DB_i.m_0 = DB_i.m_1$ then
2     return $m \longleftarrow \mathrm{DEC}_{\mathrm{sk}}(\mathrm{DB}_i.\mathrm{ct})$
3 end
```

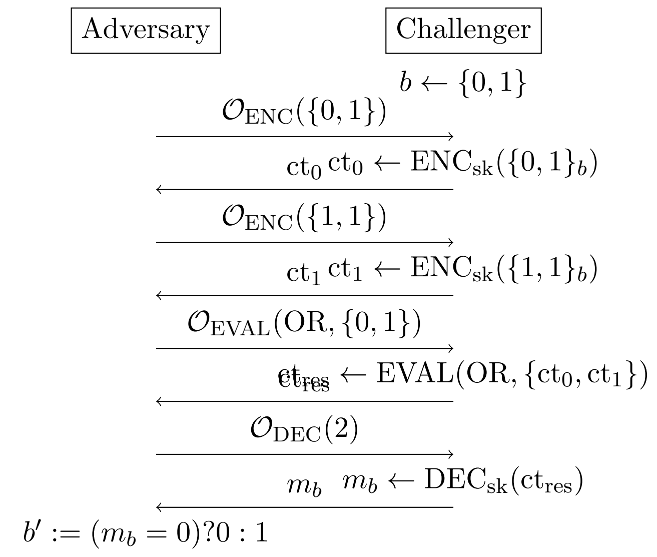
Adversary
Challenger
$b \leftarrow \{0,1\}$
$\mathcal{O}_{\text{ENC}}(\{0,1\})$
ct_0 $\text{ct}_0 \leftarrow \text{ENC}_{\text{sk}}(\{0,1\}_b)$
$\mathcal{O}_{\text{ENC}}(\{1,1\})$
ct_1 $\text{ct}_1 \leftarrow \text{ENC}_{\text{sk}}(\{1,1\}_b)$
$\mathcal{O}_{\text{EVAL}}(\text{OR}, \{0,1\})$
ct_{res} $\text{ct}_{\text{res}} \leftarrow \text{EVAL}(\text{OR}, \{\text{ct}_0, \text{ct}_1\})$
$\mathcal{O}_{\text{DEC}}(2)$
m_b $m_b \leftarrow \text{DEC}_{\text{sk}}(\text{ct}_{\text{res}})$
$b' := (m_b = 0)?0 : 1$

Figure 3: IND-CPA$^{\text{D}}$ attacks on binary FHEW/TFHE

Algorithm 3: $\mathcal{O}_{\text{EVAL}}\,(G, I)$ [CCP$^+$24]

```
1 $\text{ct} \longleftarrow \text{EVAL}(G, \text{DB}_{i\in I}.\text{ct})$
2 $\text{result}_0 \longleftarrow \text{EVAL}(G, \text{DB}_{i\in I}.m_0)$
3 $\text{result}_1 \longleftarrow \text{EVAL}(G, \text{DB}_{i\in I}.m_1)$
4 $\text{DB}_i \longleftarrow \{\text{result}_0, \text{result}_1, \text{ct}\}$
5 $i \leftarrow i + 1$
6 return ct
```

Using these oracles, IND-CPA$^{\text{D}}$ attacks on FHEW/TFHE become feasible. Figure 3 illustrates a generic IND-CPA$^{\text{D}}$ attacks on binary FHE. First, the adversary queries the challenger using $\mathcal{O}_{\text{ENC}}(\{0,1\})$ and $\mathcal{O}_{\text{ENC}}(\{1,1\})$. The challenger then selects a challenge bit b and encrypts the messages $\{0,1\}_b$ and $\{1,1\}_b$, where $\{A,B\}_b = A$ if $b = 0$, and $\{A,B\}_b = B$, otherwise. Now, the adversary receives two ciphertexts, ct_0, and ct_1, where ct_1 always represents the encryption of 1. Afterward, the adversary requests the evaluation oracle from the challenger to perform an OR operation on the previously obtained ciphertexts. Since ct_1 always represents the encryption of 1 (assuming correct execution), the result of the evaluation oracle will always yield 1. Finally, the adversary queries the decryption oracle from the challenger to verify the result of ct_{res}. In this scenario, if all operations are executed correctly, the result of ct_{res} will be 1, regardless of the value of b. However, in cases where computational errors occur due to the inherent noise in Exact FHE schemes, the result will be 0. This discrepancy reveals that the failure is attributable to the noise affecting ct_1.

In such a scenario, the adversary can say $b' = b$ with probability:

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

where F represents the event of decryption failure, and p denotes the probability of event F. Consequently, if p is sufficiently large, the adversary can obtain enough oracle queries to mount an attack on Exact FHE. According to [CCP+24], the adversary can increase the success probability of the attack to $\frac{1}{2} + \frac{\mathsf{q}p}{2}$ using q oracle queries.

By exploiting this failure probability, the $\mathrm{KR^D}$ attack can be extended to Exact FHE schemes. To further increase the failure probability, operations are performed on the ciphertext after the final modulus switch, as shown in Figure 2. Algorithm 4 describes the $\mathrm{KR^D}$ attack on FHEW/TFHE, where $\mathcal{K}$ represents the number of operations required to gather sufficient failed results, and cnt denotes the number of observed failures. If a sufficient number of failed results caused by noise are collected and analyzed, $\mathrm{KR^D}$ attacks become feasible since the distribution of noise varies depending on the secret key value.

Algorithm 4: $\mathrm{KR^D}$ Attacks [CCP+24]

```
1  cnt = 0
2  for $i = 0; i < \mathcal{K}; i = i + 1$ do
3      $\mathrm{ct}_0 \longleftarrow \mathrm{ENC_{pk}}(0)$
4      $\mathrm{ct}_1 \longleftarrow \mathrm{EVAL}(\mathrm{OR}, \mathrm{ct}_0, \mathrm{ct}_0)$
5      $\mathrm{ct}_2 \longleftarrow \mathrm{EVAL}(\mathrm{OR}, \mathrm{ct}_1, \mathrm{ct}_1)$
6      $m \longleftarrow \mathrm{DEC_{sk}}(\mathrm{ct}_2)$
7      if $m == 1$ then
8          $\boldsymbol{e}_{\mathrm{cnt}} \longleftarrow$ evaluate noise from $\mathrm{ct}_1$
9          cnt = cnt + 1
10     end
11 end
12 $\boldsymbol{e} = \frac{1}{\mathrm{cnt}} \cdot \sum_{j=0}^{\mathrm{cnt}-1} \boldsymbol{e}_j$
13 if $\left(\boldsymbol{e}_i > \frac{\alpha}{2}\right)$ ?$(s_i = 1) : (s_i = 0)$ for all $i < n$
14 return s
```

3.2 Smooth Failure Probability Requirements in FHEW-like HE

There are two primary reasons why smooth parameter sets are needed for each targeted failure probability: first, to satisfy the security requirements of the $\mathrm{IND\text{-}CPA^D}$ security requirement, and second, to meet the diverse reliability requirements of HE applications in practical scenarios.

3.2.1 (c, s)-security

Li et al. extended the classical computational security definition discussed in [MW18] to the (c, s)-security framework, as outlined in Theorem 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* [LMSWS22].

Theorem 1 ((c, s) − *security* [LMSWS22])**.** *Let* Π *be a cryptographic primitive, and* $\mathcal{G}$ *be an indistinguishable game. Let* $\mathsf{adv}^{\mathcal{A}}$ *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*

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

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

The advantage of the adversary in the attack proposed by Cheon et al. [CCP+24], as well as 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 it is necessary to have a method that can smoothly adjust the failure probability and performance based on the targeted security requirements and the number of queries.

3.2.2 Hardness of Achieving Desired Failure Probability

For the Exact FHE scheme to satisfy IND-CPA$^{\mathrm{D}}$ security, the probability of a successful attack must be close to $\frac{1}{2} + \frac{qp}{2} \approx \frac{1}{2}$. Therefore, the failure probability of the operation, p, should be negligible and determined by statistical security parameter s.

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, d_r, 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 4, we introduce a blind rotation method that provides a fine-grained trade-off between the bootstrapping failure probability and runtime. Figure 5 illustrates the spectrum of failure probabilities achievable using 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 a reduction in computational complexity.

3.2.3 Reliability Requirement

An additional advantage of making the failure probability independent of the key generation process is that the computing party can avoid revealing information about the number of gates, which could potentially disclose details about the circuit structure, to the key generation party. The technique introduced in Section 4 incorporates a cutoff parameter that balances failure probability and runtime. This parameter can be adjusted after key generation, provided a lower bound for the failure probability is maintained. This flexibility enables the computing party to optimize either runtime or failure probability for different circuits using the same set of keys, without disclosing circuit structures or requiring key regeneration.

Moreover, 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. Assuming each gate failure is independent and results in complete computational failure, the overall failure probability becomes $1-(1-p)^G$, where p is the failure probability per gate and G is the number of gates. Thus, for large computations, it is essential to employ parameters with low failure probabilities.

Finally, certain applications, such as robotic control systems, demand exceptionally high reliability. In such cases, failure is not only a security risk but could potentially result in severe accidents. For these scenarios, it is crucial to use parameters with extremely low failure probabilities, making it necessary to have a mechanism that allows for smooth adjustment of the failure rate.

4 New Blind Rotation Technique

As discussed in the previous section, a high probability of failure can provide the adversary with excessive information, thereby increasing their advantage. To mitigate this risk, reducing the failure probability during bootstrapping to a negligible level is essential. However, the current parameters result in a sufficiently high failure probability, enabling successful attacks and necessitating updates to the default parameter sets for FHEW/TFHE.

Adjustments to parameters such as N, Q, d_g, and d_{ks} are imperative. However, the bootstrapping failure probability is highly sensitive to these parameters, making precise adjustments challenging.

To address this issue and achieve finer control over the failure probability while optimizing computational complexity, we propose modifying 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 RLWE $\otimes$ RGSW operations during blind rotation when $|a_i| \le t$. 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$^{\mathrm{D}}$ security.

4.1 The Cutoff Blind Rotation Algorithm

The detailed operation of cutoff blind rotation can be found in Algorithm 5. Although the algorithm is currently illustrated for the AP method, it can also be applied to the GINX [CGGI17] and LMKCDEY [LMK$^+$23] methods.

As discussed in Section 2, the bootstrapping procedure enables homomorphic decryption. Specifically, during bootstrapping, the term $\boldsymbol{f} \cdot X^{\langle \boldsymbol{a}, \boldsymbol{s} \rangle}$ is computed homomorphically. The core concept of the cutoff blind rotation technique is to disregard terms where a_i is sufficiently small, such that omitting $(a_i \cdot s_i)$ does not substantially impact the overall value of $\langle \boldsymbol{a}, \boldsymbol{s} \rangle$. This approach results in an approximate decryption, increasing the noise.

However, reducing the number of RGSW multiplications decreases the computational complexity, and the additive noise generated from $\circledast$RGSW is also minimized. This implies that adjusting the cutoff value can increase the failure probability in scenarios where the available parameter choices result in a lower-than-required failure probability while simultaneously improving computational efficiency. This technique enhances the flexibility of parameter settings concerning failure probability, enabling a broader range of parameter choices.

Algorithm 5: NAND with cutoff blind rotation

```
Data: ciphertext $\mathrm{ct}_1$, ciphertext $\mathrm{ct}_2$, blind rotation key $\mathbf{ek}$, threshold value $t$
Result: $\mathrm{ct} \longleftarrow \mathrm{ct}_1 \bar{\wedge} \mathrm{ct}_0$ with small noise
$(\mathbf{a}, b) \longleftarrow \mathrm{ct}_1 + \mathrm{ct}_2$
for $i \in [0, q/2)$ do
    $k \longleftarrow b - i$
    if $k \in [3q/8, 7q/8)$ then
        $f_{i \cdot 2N/q} = -Q/8$
    else
        $f_{i \cdot 2N/q} = Q/8$
    end
end
$\mathrm{Acc} \longleftarrow \boldsymbol{f}$
for $i \in [0, n)$ do
    if $|a_i| \leq t$ then continue
    for $j \in [0, d_r)$ do
        $c_j \longleftarrow \lfloor c/B_r^j \rfloor \mod B_r$
        if $c_j \neq 0$ then
            $\mathrm{Acc} \longleftarrow \mathrm{Acc} \otimes \mathbf{ek}_{i,j,c_j}$
        end
    end
end
return LWEExtract(Acc) $+ Q/8$
```

4.2 Noise Analysis

Our noise analysis follows the methodologies from [DM15], [MP21], and [LMK$^+$23]. Using a cutoff blind rotation affects the number of RLWE $\otimes$ RGSW 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.

Before conducting noise analysis, the average number of a_i that satisfies $|a_i| > t$, which is determined by the cutoff value, can be expressed as $n \cdot (1 - \frac{2t+1}{q})$. Here, we will denote $(1 - \frac{2t+1}{q})$ as $\mathcal{N}$ from now on. The noise generated by $\odot$ operation is represented as $d_g N \frac{B_g^2}{12} \sigma^2$ [MP21], and we denote it as $\sigma^2_{\odot}$.

Theorem 2. *Let $(\boldsymbol{a}, b)$ be an LWE encryption with the secret key $\boldsymbol{s}$, and let $\boldsymbol{a}^*$ be the vector whose elements are defined as $a_i^* = a_i$ if $|a_i| > t$, and $a_i^* = 0$ otherwise. Then, RLWE$\left(\boldsymbol{f} \cdot X^{b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle}\right)$ can be found using the AP blind rotation, and the noise introduced by the cutoff blind rotation is*

$$\sigma^2_{ACC-AP} = 2nd_r \left(d_g N \frac{B_g^2}{12} \sigma^2 \right) \cdot (1 - \frac{2t+1}{q}).$$

Proof. Since $(\boldsymbol{a}, b)$ is an LWE ciphertext, the coefficients of $\boldsymbol{a}$ are uniformly distributed over the interval $[-q/2, q/2)$ [Reg09]. Therefore, the average number of zero elements in $\boldsymbol{a}^*$ 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 $\sigma^2_{ACC} = 2Nd_g\frac{B_g^2}{12}\sigma^2 = 2\sigma^2_{\odot}$, and it accumulates additively. Since the AP blind rotation for $(\boldsymbol{a}^*, b)$ requires $nd_r\mathcal{N}$ RGSW multiplications, the total noise introduced by the AP blind rotation is given by

$$\sigma^2_{ACC-AP} = 2nd_r \cdot \sigma^2_{\odot} \cdot \mathcal{N}.$$

□

Corollary 1. *The noise introduced during GINX cutoff blind rotation is given by:*

$$\sigma^2_{ACC-GINX} = 2|U| \cdot 2n \cdot \sigma^2_{\odot} \cdot \mathcal{N}.$$

According to [LMK+23], the maximum noise variance that a ciphertext can have in LMKCDEY blind rotation is given as follows:

$$2n \cdot \sigma^2_{\odot} + \left(k + \frac{N-k}{w}\right) \cdot \sigma^2_{\odot}$$

where the left term corresponds to the multiplication of RGSW(X^{s_i}) and the right term corresponds to the key switching during automorphsim.

Our proposed technique only affects the blind rotation procedure. Specifically, it impacts the value of $\sigma_{ACC-LMK^+}$ and results in a fixed variance depending on the cutoff value t.

Corollary 2. *The noise introduced during LMKCDEY cutoff blind rotation is given as follows:*

$$2n \cdot \sigma^2_{\odot} \cdot \mathcal{N} + \left(k' + \frac{N'-k'}{w}\right) \cdot \sigma^2_{\odot}$$

where $k' = N'\left(1 - \left(1 - \frac{1}{N'}\right)^{n\cdot\mathcal{N}}\right)$ *and* $N' = (N-t)$.

For operations other than blind rotation, the cutoff value does not have any impact. Therefore, the noise variances for these operations are given by:

$$\sigma^2_{MS_1} = \frac{||\boldsymbol{s}_N||^2+1}{12}, \quad \sigma^2_{MS_2} = \frac{||\vec{s}_n||^2+1}{12}, \quad \sigma^2_{KS} = \sigma^2 N d_{\mathrm{ks}},$$

where $\boldsymbol{s}_N \in R_Q$ and $\vec{s}_n \in \mathbb{Z}_q^n$ denote the RLWE and LWE keys, respectively. Here, $\sigma^2_{MS_1}$ and $\sigma^2_{MS_2}$ represent the noise generated by modulus switching operations before and after key switching, respectively, and σ^2_{KS} represents the noise introduced by the key switching procedure. When calculating the probability of failure, we consider $||\boldsymbol{s}_N||^2$ and $||\vec{s}_n||^2$ under the assumption that if the secret key is binary or ternary, then $||\vec{s}_n|| \le \sqrt{n/2}$ and $||\boldsymbol{s}_N|| \le \sqrt{N/2}$, respectively.

Let $\boldsymbol{a}^*$ be a vector such that $a_i^* = a_i$ if $|a_i| > t$, and $a_i^* = 0$, otherwise. The noise introduced by approximating $\boldsymbol{a}$ to $\boldsymbol{a}^*$ given by:

$$\sigma^2_{CUT} = \frac{t^2}{3} \cdot ||\vec{s}_n||^2 \cdot (1 - \mathcal{N}).$$

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 $\frac{2t+1}{q} = 1 - \mathcal{N}$. 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.

The resulting total noise is expressed as follows:

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

If the noise of $\text{LWE}_{\boldsymbol{s}}\left(\frac{q}{4} \cdot m\right)$ exceeds $\frac{q}{8}$, decryption failure occurs. Therefore, the failure probability of (N)AND, (N)OR, and X(N)OR gate operations can be defined as $1 - \text{erf}\left(\frac{q/8}{\sqrt{2}\cdot\sigma_{total}}\right)$, where erf denotes the error function..

4.3 Runtime Analysis

The number of $\odot$ operations for blind rotation is given as follows [MP21, LMK+23]:

$$\begin{aligned} \text{AP/FHEW: } & 2n \cdot d_r(1 - 1/B_r) \\ \text{GINX/TFHE: } & 2n \cdot |U| \\ \text{LMKCDEY: } & 2n + (1 - 1/w)k + N/w + 2. \end{aligned}$$

This computational complexity can be reduced by applying the cutoff blind rotation technique. Since the $\odot$ operation constitutes the majority of the computational workload in FHEW bootstrapping, we measure the overall computational complexity based on the number of RLWE$'$ multiplications, as in [LMK+23].

The fundamental principle of cutoff blind rotation lies in omitting specific computations. Our focus is on determining how many operations can be skipped. Given a cutoff value t, computations are omitted for coefficients 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:

$$\begin{aligned} \text{AP/FHEW: } & 2n \cdot d_r(1 - 1/B_r) \cdot \mathcal{N} \\ \text{GINX/TFHE: } & 2n \cdot |U| \cdot \mathcal{N} \\ \text{LMKCDEY: } & 2n \cdot \mathcal{N} + (1 - 1/w)k' + N'/w + 2. \end{aligned}$$

5 Further Improvement and Application to Existing Optimization

Cutoff blind rotation operates by skipping $a_i \cdot s_i$ operations for a_i values that satisfy $|a_i| \leq t$, making it applicable to commonly used bootstrapping methods such as AP, GINX, and LMKCDEY, as well as compatible with various other optimization techniques. For example, when combined with the approximate gadget decomposition method introduced in [KLD+23], the bootstrapping runtime can be significantly reduced with only a minimal increase in noise.

Additionally, with slight modifications to cutoff blind rotation, it is possible to further fine-tune the trade-off between failure probability and computational complexity, achieving a lower failure probability when the secret key distribution is binary.

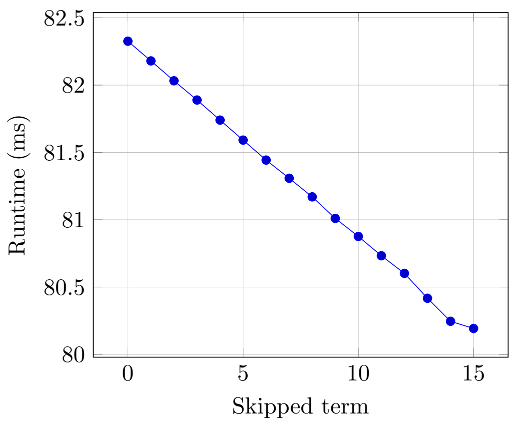

Figure 4: This figure shows the changes in bootstrapping runtime according to the number of a_i satisfying $|a_i| \leq t$. This is the average result obtained from performing the NAND gate operation 60,000 times. We used the AP method with the parameter LPF_STD128 with cutoff value $t = 9$, which is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

6 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 [Ope22]. The parameters primarily covered in this section are four specific configurations: LPF_STD128, LPF_STD128Q, LPF_STD128_LMKCDEY, and LPF_STD128Q_LMKCDEY. The evaluation was conducted using OpenFHE v.1.2.0 on an Intel(R) Core(TM) i9-11900 @ 2.50GHz processor. The code was compiled with clang++ 14, using the CMake flags NATIVE_SIZE=32.

6.1 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 2 shows the changes in failure probability and the number of $\odot$ operations when applying cutoff blind rotation to the parameters currently used in OpenFHE. In Table 2, the cutoff value t is set to 9; however, the cutoff value can be adjusted freely depending on the failure probability requirements of the specific application. As the cutoff value increases, the number of $\odot$ operations decreases linearly.

Table 1: This table provides information about the parameter sets used in Table 2. In cases where the cutoff blind rotation technique is not applied, t is set to 0.

	n	q	N	$\log_2 Q$	$\log_2 Q_{ks}$	B_g	B_{ks}	B_r	δ	t
LPF_STD128	556	2048	1024	27	15	2^7	2^6	2^6	2^7	9
LPF_STD128Q	645	2048	1024	25	16	2^7	2^6	2^5	2^7	9
LPF_STD128_LMKCDEY	556	2048	1024	27	15	2^9	2^5	-	2^9	9
LPF_STD128Q_LMKCDEY	600	2048	1024	25	15	2^7	2^5	-	2^7	9

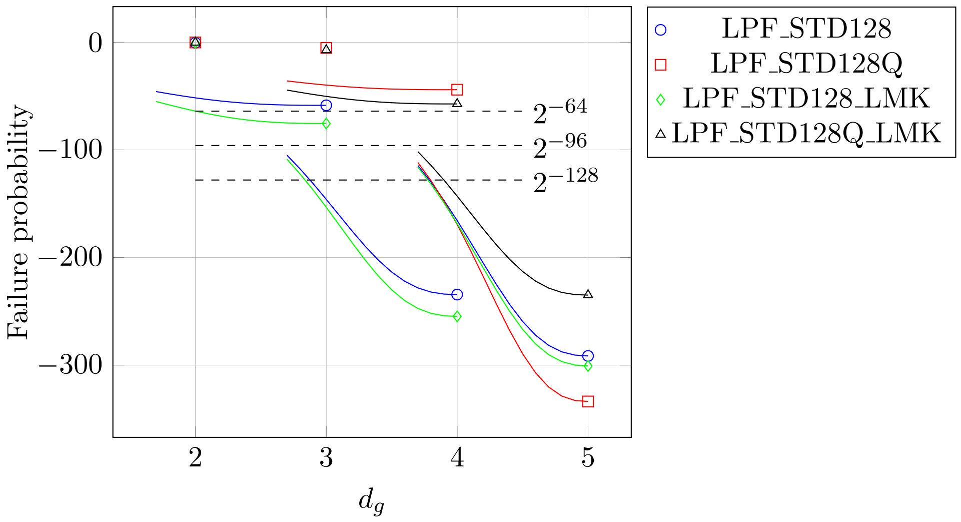

Figure 5: This figure illustrates the changes in failure probability as d_g and cutoff value vary. The dashed lines represent cases where the failure probability satisfies 2^{-128}, 2^{-96}, and 2^{-64}. The points marked with an empty shape represent the failure probability according to d_g, i.e., parameters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

Table 2: This table shows the changes in failure probability and the number of $\odot$ operations when applying the cutoff blind rotation technique to the parameters currently used in OpenFHE. In the case of LPF_STD128(Q), the calculations were based on the AP method. Detailed parameter sets can be found in Table 1.

	without cutoff		with cutoff	
	FP	# of $\odot$	FP	# of $\odot$
LPF_STD128	2^{-185}	2188	2^{-124}	**2168**
LPF_STD128Q	2^{-52}	2538	2^{-47}	**2516**
LPF_STD128_LMKCDEY	2^{-77}	1601	2^{-67}	**1592**
LPF_STD128Q_LMKCDEY	2^{-58}	1712	2^{-52}	**1701**

6.2 Diversity and Precision 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 enables fine-tuning of the failure probability. At the same time, it provides a favorable trade-off between failure probability and computational complexity, facilitating the selection of more efficient parameters. For example, Figure 5 illustrates the changes in failure probability resulting from changes to d_g, as well as the changes in failure probability when the cutoff technique is additionally applied.

The three failure probabilities, 2^{-128}, 2^{-96}, and 2^{-64} are the target failure probabilities that we have set arbitrarily. As shown in Figure 5, it is challenging to configure all parameters to meet the desired failure probability, e.g., $\mathrm{FP} \leq 2^{-96}$ by merely changing d_g. However, by applying the cutoff technique, it becomes straightforward to modify the cutoff value to satisfy $\mathrm{FP} \leq 2^{-96}$, while simultaneously reducing computational complexity. For values of d_g set to 4 or 5, the failure probability is smaller than 2^{-128}. In this case, adjusting the cutoff value allows the failure probability to be brought closer to 2^{-128} while also reducing computational complexity.

Table 3: This table presents the values of δ, B_g, and t for parameter sets that satisfy failure probabilities of $\mathrm{FP} \leq 2^{-64}$, $\mathrm{FP} \leq 2^{-96}$, and $\mathrm{FP} \leq 2^{-128}$. It also shows the theoretical failure probability, $\mathrm{FP}_{\mathrm{th}}$, the experimentally measured noise variance, $\mathrm{VAR}_{\mathrm{exp}}$, and the corresponding experimentally derived failure probability, $\mathrm{FP}_{\mathrm{exp}}$.

	STD128(FP64)			STD128(FP96)			STD128(FP128)			LPF_STD128		
	AP	GINX	LMK+	AP	GINX	LMK+	AP	GINX	LMK+	AP	GINX	LMK+
δ	2^{11}	2^{9}	2^{9}	2^{9}	2^{9}	2^{6}	2^{9}	2^{9}	2^{6}	2^{7}	2^{7}	2^{9}
B_g	2^{8}	2^{6}	2^{9}	2^{6}	2^{6}	2^{7}	2^{6}	2^{6}	2^{7}	2^{7}	2^{7}	2^{9}
t	8	12	5	9	8	8	5	5	5	0	0	0
$\mathrm{FP}_{\mathrm{th}}$	2^{-67}	2^{-65}	2^{-65}	2^{-96}	2^{-102}	2^{-105}	2^{-136}	2^{-128}	2^{-132}	2^{-144}	2^{-127}	2^{-69}
$\mathrm{VAR}_{\mathrm{exp}}$	486.199	392.215	377.1	307.982	264.603	270.829	232.261	244.538	231.259	212.476	203.114	343.736
$\mathrm{FP}_{\mathrm{exp}}$	2^{-101}	2^{-124}	2^{-129}	2^{-157}	2^{-182}	2^{-178}	2^{-207}	2^{-197}	2^{-208}	2^{-226}	2^{-237}	2^{-141}
runtime	65.0 ms	49.7 ms	51.8 ms	65.5 ms	62.4 ms	62.1 ms	80.2 ms	62.9 ms	62.4 ms	80.9 ms	63.3 ms	52.8 ms

Table 3 presents the parameters that satisfy failure probabilities of $\mathrm{FP} \leq 2^{-64}$, $\mathrm{FP} \leq 2^{-96}$, and $\mathrm{FP} \leq 2^{-128}$, 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. The experimentally measured noise variance, $\mathrm{VAR}_{\mathrm{exp}}$, and the unit bootstrapping runtime are based on the results obtained from 1000 NAND gate operations. The difference between the experimental results and theoretical values arises because our experiments focus on events involving very small noise. As a result, obtaining accurate values would require extensive experimentation. Thus, we implemented a failure probability simulator [Noi24].

7 Conclusion

We propose a new optimization technique applicable to existing blind rotation methods with minimal changes to the underlying algorithm. By introducing the cutoff blind rotation, we enable more flexible parameter settings, achieving an appropriate trade-off between failure probability and computational complexity. Unlike conventional parameters, the cutoff value has a relatively small impact on failure probability, allowing for precise parameter adjustments.

This technique operates by partially omitting elements of the LWE ciphertext $(\vec{a}, b)$, making it applicable to various bootstrapping methods. We demonstrated the application of our technique to three existing blind rotation methods—AP, GINX, and LMKCDEY—as well as its integration with approximate gadget decomposition.

Previously, even slight changes to parameters such as d_g and d_{ks} would significantly impact the failure probability, making fine-tuning for very low failure probabilities particularly challenging. As discussed in [CCP$^+$24], FHEW/TFHE HE schemes are vulnerable against IND-CPA$^{\mathrm{D}}$ adversaries. However, achieving low failure probability, such as $2^{-96}, 2^{-128}$ for security, would often result in sparse parameter choices, leading to significant inefficiencies in runtime. Our proposed technique facilitates precise parameter tuning to achieve the desired failure probability while improving runtime efficiency.

The proposed method can also be applied to Torus-based [CGGI17] and NTRU-based [BIP$^+$22, XZDF23] variants. While these algorithms have different parameter sets and are susceptible to attacks such as the one described in [CCP$^+$24], finding parameters with negligible failure probability remains crucial. Additionally, our technique can be extended to blind rotations with higher bits, as seen in [LMP22, BBB$^+$23]. Future work could explore applying the proposed method to amortized bootstrapping techniques [LW23a, LW23b, LW23c, MKMS23] to further optimize runtime performance.

References

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology – CRYPTO*, pages 297–314. Springer, 2014.

[BBB$^+$23] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Parameter optimization and larger precision for (T)FHE. *Journal of Cryptology*, 36, 2023.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[BIP$^+$22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. Final: Faster fhe instantiated with NTRU and LWE. In *Advances in Cryptology – ASIACRYPT 2022*, pages 188–215, 2022.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886. Springer, 2012.

[CCP$^+$24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the INDCPA-D security of exact fhe schemes. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024*, 2024.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *Advances in Cryptology – ASIACRYPT 2017*, pages 377–408. Springer, 2017.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, pages 34–91, 2020.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437. Springer, 2017.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology - EUROCRYPT*, pages 617–640. Springer, 2015.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012.

[GINX16] Nicolas Gama, Malika Izabachene, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Advances in Cryptology - EUROCRYPT*, pages 528–558. Springer, 2016.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013*, pages 75–92. Springer, 2013.

[JP22] Marc Joye and Pascal Paillier. Blind rotation in fully homomorphic encryption with extended keys. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 1–18. Springer, 2022.

[KLD$^+$23] Andrey Kim, Yongwoo Lee, Maxim Deryabin, Jieun Eom, and Rakyong Choi. Lfhe: Fully homomorphic encryption with bootstrapping key size less than a megabyte. Cryptology ePrint Archive, 2023.

[LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology – EUROCRYPT 2021*, pages 648–677. Springer, 2021.

[LMK$^+$23] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2023*, pages 227–256. Springer, 2023.

[LMP22] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In *Advances in Cryptology–ASIACRYPT 2022*, pages 130–160. Springer, 2022.

[LMSWS22] Baiyu Li, Daniele Micciancio, Mark Schultz-Wu, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 560–589, Cham, 2022. Springer Nature Switzerland.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):1–35, 2013.

[LW23a] Feng-Hao Liu and Han Wang. Batch bootstrapping i: A new framework for simd bootstrapping in polynomial modulus. In *Advances in Cryptology – EUROCRYPT 2023*, page 321–352, 2023.

[LW23b] Feng-Hao Liu and Han Wang. Batch bootstrapping ii: Bootstrapping in polynomial modulus only requires o(1) FHE multiplications in amortization. In *Advances in Cryptology – EUROCRYPT 2023*, page 353–384, 2023.

[LW23c] Zeyu Liu and Yunhao Wang. Amortized functional bootstrapping in less than 7 ms, with õ(1) polynomial multiplications. In *Advances in Cryptology - ASIACRYPT*, pages 101–132, 2023.

[MKMS23] Gabrielle De Micheli, Duhyeong Kim, Daniele Micciancio, and Adam Suhl. Faster amortized FHEW bootstrapping using ring automorphisms. Cryptology ePrint Archive, 2023.

[MP21] Daniele Micciancio and Yuriy Polyakov. Bootstrapping in FHEW-like cryptosystems. In *WAHC'21*, pages 17–28, 2021.

[MW18] Daniele Micciancio and Michael Walter. On the bit security of cryptographic primitives. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 3–28, Cham, 2018. Springer International Publishing.

[Noi24] Fhew/tfhe failure probability simulator. `https://gitlab.com/anonymous11112222/noise-calculator-fhew`, April 2024.

[Ope22] OpenFHE. Open-Source Fully Homomorphic Encryption Library. `https://github.com/openfheorg/openfhe-development`, 2022.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[XZDF23] Binwu Xiang, Jiang Zhang, Yi Deng, and Dengguo Feng. Fast blind rotation for bootstrapping fhes. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 3–36, 2023.

[Zam22] 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`.