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
12191837@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 a crucial component of privacy-enhancing technologies. Ducas and Micciancio introduced FHEW (Eurocrypt '15), and Chillotti et al. improved it in TFHE (Asiacrypt '16), both of which provide homomorphic binary (or larger) gate evaluations with fast latency due to their small parameters. However, their evaluation failure probability is highly sensitive to parameter selection, resulting in a limited set of viable parameters and a trade-off between failure probability and runtime.

Recently, Cheon et al. proposed a key recovery attack against FHEW/TFHE schemes based on a new security model for FHE, called IND-CPA$^{\mathrm{D}}$ security, which was first introduced by Li and Micciancio (Eurocrypt '21). To prevent this attack, it is necessary to make the failure probability negligible (e.g., 2^{-128}). However, due to limited choice parameters, it is forced to use a parameter set with unnecessarily low failure probabilities than needed, causing inefficiencies in runtime.

We propose a new bootstrapping method for FHEW/TFHE, providing a precise balance between runtime and failure probability, and easy to implement. The proposed methods enable the selection of parameter sets that achieve negligible failure probabilities for each desired security level while optimizing runtime.

Keywords. Homomorphic encryption, key recovery attack, bootstrapping

1 Introduction

A Fully Homomorphic Encryption (FHE) scheme enables various computations to be performed on encrypted data, preserving privacy during data processing. The Brakerski-Gentry-Vaikuntanathan (BGV) and Brakerski-Fan-Vercauteren (BFV) schemes support 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*. Additionally, the Cheon-Kim-Kim-Song (CKKS) scheme, introduced in [CKKS17], allows operations on complex numbers approximately, and thus is categorized as *Approximate FHE*.

The practical application of FHE in privacy-preserving models typically involves a client encrypting data and sending it to a server, which performs the requested computations and returns

the (encrypted) result of evaluation. Since homomorphic evaluations require substantial resources, they are usually executed on high-performance servers. Designing an efficient model involves selecting optimal parameters that balance security and performance. The chosen FHE scheme must be secure against chosen plaintext attacks (IND-CPA security), ensuring that servers cannot infer any information about the client's data from the ciphertext.

Li and Micciancio broadened the scope of IND-CPA by introducing indistinguishability under chosen plaintext attacks with a decryption oracle (IND-CPA$^{\mathrm{D}}$) [LM21]. Unlike the traditional IND-CPA, this variant exposes the computation results of the ciphertext. It is also shown that there exists an attack on *approximate HE* (CKKS scheme) in the IND-CPA$^{\mathrm{D}}$ model, and the known workaround is so-called noise flooding, which increases the noise level and makes the noise statistically indistinguishable [LM21]. Cheon et al. [CCP$^+$24] demonstrated that Exact FHE (i.e., BGV/BFV and FHEW/TFHE) also falls short of satisfying IND-CPA$^{\mathrm{D}}$ security due to operation failure. Moreover, they conducted a key-recovery attack on a widely used library, such as TFHE-rs, highlighting the necessity to revise the current parameter set.

There are two main methods for bootstrapping in FHEW-like schemes. Ducas and Micciancio introduced the Alperin-Sheriff and Peikert (AP) method [AP14] in their FHEW scheme [DM15], which maintains consistent performance regardless of the key distribution. Alternatively, the TFHE scheme [CGGI17], proposed by Chillotti et al., uses the Gama-Izabachene-Nguyen-Xie (GINX) method [GINX16], which offers superior performance but is limited to binary key distributions. Although the GINX/TFHE bootstrapping can be generalized to arbitrary secret keys [MP21, JP22], its performance degrades when the secret key is sampled from a larger distribution. Recently, Lee-Micciancio-Kim-Choi-Deryabin-Eom-Yoo (LMKCDEY) proposed efficient bootstrapping techniques using ring automorphism, which is equally efficient as GINX/TFHE even when using an arbitrary secret [LMK$^+$23]. HE with small key sizes by packing evaluation keys in a smaller number of ciphertexts and reconstructing them server-side is proposed in [KLD$^+$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, which accumulates with each homomorphic operation, decryption may 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^{-50} [DM15, CGGI17, MP21, LMK$^+$23, BBB$^+$23]) while ensuring acceptable performance.

Cheon et al. proposed a key-recovery (KR$^{\mathrm{D}}$) attack against FHEW/TFHE schemes in the existence of a decryption oracle for the result of a homomorphic operation model [CCP$^+$24]. When an adversary has access to 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.

Let Pr be the probability of obtaining a constant number of failure ciphertexts and C be the computational cost of the attack. To meet the λ-bit security level, work factor WF must satisfy

$$WF = C \cdot Pr \leq 2^{-\lambda}$$

The event of failure is independent, thus Pr is approximated to a constant time to the probability

of a single failure of the homomorphic operation. Also, as C is constant, to securely employ Exact FHE, the probability of single operation failure must be negligible as $2^{-\lambda}$.

Reducing this probability involves adjusting parameters such as ring dimension, learning with error (LWE) ciphertext dimension, and ciphertext modulus. However, because the ring dimension must be a power of two for efficient number theoretic transformation (NTT) and modular arithmetic in the exponent, there is a significant gap between feasible values. 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 like 3 or 4* [MP21, LMK+23]. Unfortunately, even minor changes in d can significantly impact failure probability and computational complexity, limiting viable parameters and making it challenging to achieve satisfactory configurations.

1.2 Our Contribution

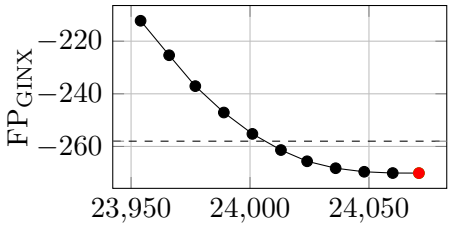
To reduce the probability of operation failure to less than 2^{-128}, modifying parameters unavoidably leads to inefficiencies such as increased computational complexity and key size. Furthermore, as previously mentioned, the range of values that can be modified for parameters is very limited. For instance, parameters like ring dimension may have a significant gap between permissible values, leading to an operation failure probability higher than necessary. Similarly, parameters such as digits of decomposition d can drastically reduce the probability of operation failure and increase the computational complexity with only slight adjustments. Thus, we have to use parameters with an unnecessarily low failure probability and high computational complexity for homomorphic operations.

To minimize the inefficiencies caused by limited parameters, we introduce a new blind rotation technique that omits basic operations with a small effect on noise while bootstrapping FHEW-like HE. Blind rotation is essentially the homomorphic execution of the decryption operation, $\boldsymbol{f} \cdot X^{b-\langle \vec{a}, \vec{s} \rangle}$, on the given ciphertext $(b, \vec{a}) \in \mathbb{Z}_q^{n+1}$. During the proposed bootstrapping, the operation corresponding to $\langle \vec{a}, \vec{s} \rangle$ is carried out for the secret key $\vec{s}$, with the threshold value t set to omit operations for $\langle a_i, s_i \rangle$ where $|a_i| \leq t$, thereby effectively reducing the number of heavy ciphertext multiplications per blind rotation. Naturally, the newly proposed parameter t increases the probability of operation failure. However, it also reduces the computational complexity by omitting partial operations of blind rotation that account for a significant portion of the bootstrapping runtime. Furthermore, t can be freely set within the range $[0, q-1]$, and its impact on the probability of operation failure is much lower compared to decomposition digits. Therefore, it assists in achieving a parameter setting that satisfies a probability of operation failure of 2^{-128} (or lower depending on target security) while minimizing computational complexity.

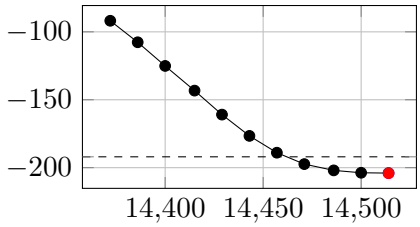
Figure 1 shows the failure probability and runtime (the number of NTTs) as a function of t when our technique is applied. Here, NTT refers to the operations that occur during blind rotation.

An additional important contribution of this work is the new parameter sets of FHEW-like HE with negligible failure probability. We propose optimized parameter sets for 128, 192, and 256-bit security for all existing bootstrapping methods (AP and GINX) considering advanced techniques such as approximate gadget decomposition. This is crucial as the bootstrapping with non-negligible failure leads to a key recovery attack [CCP+24]. Further, we improve the runtime based on the new parameter sets and our proposed technique.

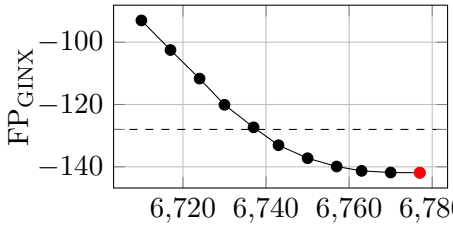
We implement the proposed method in the OpenFHE [Ope22], and the result shows that the runtime can be reduced by applying the proposed method. Although this article targets parameter



(a) The number of operations (NTTs)



(b) The number of operations(NTTs)



(c) The number of operations(NTTs)

Figure 1: The changes in failure probability $\mathrm{FP}_{\mathrm{GINX}}$ and the number of NTTs as a function of t. The red point is when our technique is not applied; it has an unnecessarily low failure probability and a high number of NTTs. (a), (b), and (c) represent $\lambda = 256$, $\lambda = 192$, and $\lambda = 128$, respectively.

sets for binary gates, we would like to note that the proposed technique can also be applied to larger message 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 6. The IND-CPA$^{\mathrm{D}}$ and key recovery attack proposed by Cheon et al. 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 $\langle \cdot, \cdot \rangle$. N is a power of two, and we denote that $2N$-th polynomial ring as $\mathcal{R}[X]/(X^N + 1)$ and $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$. We denote ring elements of $\mathcal{R}_Q$ in bold, such as $\boldsymbol{a}(X)$, and X is omitted when it is obvious. The i-th coefficient of ring element $\boldsymbol{a}$ is denoted as a_i. A vector is also denoted by boldface $\boldsymbol{v}$ and its i-th element is denoted as v_i. We denote the $L2$ norm of ring elements or vectors as $||\cdot||$, and the infinity norm as $||\cdot||_{\mathrm{inf}}$. We use $x \leftarrow \chi$ to indicate that x is sampled from distribution χ. When x is uniformly sampled from a set S, we denote this as $x \leftarrow S$.

2.1 Lattice-based Encryption

Note that integers q and n are positive. We can define LWE encryption of message $m \in \mathbb{Z}q$ under secret key $\vec{s}$ as follows:

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

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

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

Q and N, which are powers of two, are positive integers. We can define RLWE encryption as follows:

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

where $\boldsymbol{a} \leftarrow \mathcal{R}_Q$, and $\boldsymbol{e} \leftarrow \chi_{\mathrm{err}}$. This represents the message $\boldsymbol{m}$ encrypted with RLWE under the secret key $\boldsymbol{z} \leftarrow \chi_{sk}$. As in LWE, 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 $\mathrm{RLWE}'_{\boldsymbol{z}}(\boldsymbol{m})$ and $\mathrm{RGSW}_{\boldsymbol{z}}(\boldsymbol{m})$ from [MP21]:

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

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

RLWE$'$ provides the following multiplication operation:

$$\odot : \mathcal{R}_Q \times \mathrm{RLWE}' \to \mathrm{RLWE}$$

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

The multiplication operation between RLWE and RGSW is defined as follows:

$$\otimes : \mathrm{RLWE} \times \mathrm{RGSW} \to \mathrm{RLWE}$$

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

This operation makes the RLWE encryption with the message $\boldsymbol{m}_0 \cdot \boldsymbol{m}_1 + \boldsymbol{e}_1 \cdot \boldsymbol{m}_1$. It is worth noting that the noise term $\boldsymbol{m}_1 \cdot \boldsymbol{e}_1$ is small because $\boldsymbol{m}_1$ is usually selected as monomial, and thus, the consecutive RGSW multiplication only accumulates additive noise.

The gadget decomposition technique with RLWE$'$ and RGSW multiplication reduces multiplication in homomorphic operations. The variance of additive noise due to $\odot$RLWE$'$ and $\otimes$RGSW are $d_g N \frac{B_g^2}{N}$ and $2 d_g N \frac{B_g^2}{N}$, respectively [MP21].

2.1.2 Ring Automorphism

We can find a ciphertext of ring autormorphism $\psi_k : \mathcal{R}_Q \mapsto \mathcal{R}_Q$, $\psi_k(m(X)) = m(X^k)$ using RLWE$'$. Given RLWE ciphertext $RLWE_{\boldsymbol{z}}(\boldsymbol{m}) = (\boldsymbol{a}, \boldsymbol{b})$, we can find $RLWE_{\boldsymbol{z}(X^k)}(\boldsymbol{m}(X^k))$, by $(\boldsymbol{a}(X^k), \boldsymbol{b}(X^k))$. However, its secret key is now $\boldsymbol{z}(X^k)$, not $\boldsymbol{z}(X)$. We can switch the key back to $\boldsymbol{z}(X)$ when $\mathrm{RLWE}'_{\boldsymbol{z}(X)}(-\boldsymbol{z}(X^k))$ is given using following equation:

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

$\mathrm{RLWE}'_{\boldsymbol{z}(X)}(-\boldsymbol{z}(X^k))$ is often given as a public evaluation key for automorphism, and we call it ak_k.

2.1.3 LWE Key Switching

LWE key switching is the operation that converts LWE encryption under the secret key $\vec{z} \in \mathbb{Z}_q^N$ into another secret key $\vec{s} \in \mathbb{Z}_q^n$. This operation introduces some noise. To perform the key switching operation, one must define a key called the key switching key:

$$\mathrm{ksk} = \left\{ \mathrm{ksk}_{i,j,v} = \mathrm{LWE}_{\vec{s}}\left(-v \cdot z_i \cdot B_{\mathrm{ks}}^j\right) | i = 0, \ldots, B_{\mathrm{ks}} - 1, j = 0, \ldots, N-1, v = 0, \ldots, d_{ks} - 1 \right\},$$

where $\mathrm{ksk}_{i,j,v} = \mathrm{LWE}_{\vec{s}}\left(-v \cdot z_i \cdot B_{\mathrm{ks}}^j\right)$ and B_{ks} and d_{ks} is the base and digits of gadget decomposition for key switching.

We define the key-switching operation as follows:

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

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

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

Total noise after key switching e_{res} is given as follows:

$$e_{\text{res}} = e_0 + \sum e_{i,j,a_{i,j}},$$

where $e_{i,j,a_{i,j}}$ is noise of ciphertext $\text{ksk}_{i,j,a_{i,j}}$. Hence, we the variance of e_{res} is

$$\text{VAR}\left(e_{\text{res}}\right) = \beta^2 = \alpha^2 + N \cdot d_{\text{ks}} \cdot \sigma^2$$

where α^2 is variance of e_0, d_{ks} is a small integer satisfying $B_{\text{ks}}^{d_{\text{ks}}} \geq q$, and σ^2 is variance of χ_{err}.

2.1.4 Extraction of LWE Ciphertext from RLWE Ciphertext

We can extract an LWE ciphertext that contains only the constant term of message polynomial from an RLWE ciphertext [DM15]. This operation, called LWE extraction, is used in FHEW-like bootstrapping, to convert the resulting RLWE ciphertext of blind rotation to 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, $(\boldsymbol{a'}, b)$ is the ciphertext containing the constant term of the RLWE ciphertext $(\boldsymbol{a}, \boldsymbol{b})$.

2.2 Bootstrapping in FHEW/TFHE

Initially, we execute the operation determined by the gate on two ciphertexts encrypted under LWE. Subsequently, we engage in a process known as blind rotation. Blind rotation involves multiplying ring elements $\boldsymbol{f}$ and the monomial X^u, where $u = b - \langle \vec{a}, \vec{s} \rangle$ is determined by an LWE ciphertext $(\vec{a}, b) \in \mathbb{Z}_q^{n+1}$. Blind rotation operates on the accumulator initialized as $\text{ACC} = \text{RLWE}\left(\boldsymbol{f} \cdot X^b\right)$, as it iteratively performs RLWE $\otimes$ RGSW operations. Afterward, we perform the LWE extraction, which involves extracting the constant term $\left(\vec{a'}, b_0\right) \in \mathbb{Z}_Q^{N+1}$ in the RLWE polynomial. Then, the extracted polynomial is a encryption of $-u$-th coefficient of $\boldsymbol{f}$, i.e., $\text{LWE}_z(f_{-u})$. The first mod switching performs $\mathbb{Z}_Q^{N+1} \xrightarrow{\text{mod switching}} \mathbb{Z}_{Q_{ks}}^{N+1}$, followed by key-switching executing $\mathbb{Z}_{Q_{ks}}^{N+1} \xrightarrow{\text{key switching}} \mathbb{Z}_{Q_{ks}}^{n+1}$, and finally, the last mod switching carries out $\mathbb{Z}_{Q_{ks}}^{n+1} \xrightarrow{\text{mod switching}} \mathbb{Z}_q^{n+1}$.

The full procedure is given in Figure 2. In fact, the input LWE ciphertext of blind rotation is a sum of two input ciphertexts. It is noted that the noise analysis in this paper assumes that the summation is done right after LWE extraction, rather than right before blind rotation as in [LMK+23]. By doing so, we can reduce the noise and failure probability with negligible computational overhead.

The full procedure is given in Figure 2. Notably, the input LWE ciphertext for blind rotation is the sum of two ciphertexts. It is important to mention that the noise analysis in this paper assumes that the summation is performed immediately after LWE extraction, rather than just before blind rotation as in [LMK+23]. This approach helps to reduce noise and failure probability with negligible computational overhead.

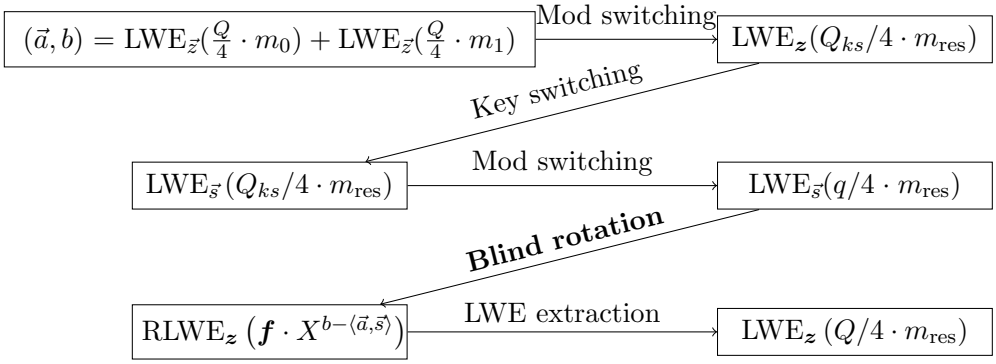


Figure 2: FHEW-like bootstrapping procedure

2.3 Noise Analysis

Our noise analysis refers to [DM15], [MP21], and [CGGI20]. The noise generated by bootstrapping can be broadly categorized into three types: noise due to blind rotation, noise due to key switching, and finally, noise due to modulus switching.

2.3.1 Modulus Switching Noise

Modulus switching is the operation that changes the original modulus Q_1 of the existing LWE ciphertext to a new modulus Q_2 through a randomized rounding function. Randomized rounding function $[\cdot]_{Q_1}^{Q_2} : \mathbb{Z}_{Q_1} \to \mathbb{Z}_{Q_2}$ can be defined as:

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

where $\mathcal{B} \in \{0, 1\}$ is a Bernoulli random variable. The modulus switch defined on the LWE ciphertext can be defined as follows:

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

consider that $[a_i]_{Q_1}^{Q_2} = \frac{Q_2}{Q_1} \cdot a_i + r_i = a_i''$ and $[b]_{Q_1}^{Q_2} = \frac{Q_2}{Q_1} \cdot b + r_n = b_i''$, where r_i are rounding errors. When the distribution of noise originally present in the ciphertext is denoted by σ, we can derive the distribution of noise resulting from modulus switching as follows:

$$\mathrm{err}\left((\vec{a_i''}, b'')\right) = b'' - \left\langle \vec{a_i''}, \vec{s} \right\rangle - \frac{Q_1}{t} \cdot m = \frac{Q_2}{Q_1} \cdot \mathrm{err}\left((\vec{a_i}, b)\right) + \sum_{i=0}^{n-1} r_i + r_n$$

In implementations, as the secret key distribution χ_{sk} follows a uniform ternary distribution, we can express the standard deviation of the noise resulting from modulus switching as follows:

$$\beta = \sqrt{\frac{Q_2^2}{Q_1^2}\sigma^2 + \frac{||\vec{s}||^2 + 1}{3}}$$

2.3.2 Blind Rotation Noise

Blind rotation enables performing operations on encrypted data, represented as $\boldsymbol{f} \cdot X^{b-\langle \vec{a}, \vec{s} \rangle}$. To perform the blind rotation, the $\otimes$ operation is needed, as described in Section 2.1. The brk, known as blind rotation key, is an RGSW encryption. It is involved in the process of performing decryption while separating the coefficients and exponents of polynomials, thus generating ciphertexts with independent noise. There exist three competing blind rotation methods: AP/FHEW, GINX/TFHE, and LMKCDEY. These methods exhibit different performances depending on the key distribution, and for more detailed information we refer to [MP21, LMK+23]. The blind rotation key for each type of blind rotation can be defined as follows:

$$\text{AP/FHEW: } \left\{ \text{brk}_{i,v,j} = \text{RGSW}\left(X^{v \cdot B_r^j \cdot s_i}\right) | v \in \mathbb{Z}_{B_r}, 0 \leq j < \log_{B_r} q \right\}$$

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

$$\text{LMKCDEY: } \left\{ \text{brk}_i = \text{RGSW}\left(X^{s_i}\right) | i \in [0, n) \right\}, \left\{ \text{ak}_{-1}, \text{ak}_{5^k} | 1 \leq k < w \right\}$$

where $w \sim \log(n)$ and $U \subset \mathbb{Z}_q$. For example, we can use $U = \{1, -1\}$ for the ternary secret key and $U = \{1\}$ for the binary secret key. Using the blind rotation key defined above, the blind rotation is done by accumulating $X^{a_i s_i}$ to ACC using the following operations:

$$\text{AP/FHEW:ACC} \longleftarrow \text{ACC} \otimes \text{brk}_{i,v,j} \text{ for all } 0 \leq j < \log_{B_r} q$$

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

where ACC is initialized as RLWE($\boldsymbol{f} \cdot X^b$). We note that in the case of AP/FHEW, the $\otimes$ operation needs to be repeated $nd_r \cdot Nd_g$ times, while in the case of GINX/TFHE, it needs to be repeated $2|U|n \cdot Nd_g$ times, where $|U|$ is the cardinality of U. The blind rotation key is an encryption of the RGSW scheme, so it is decomposed by base B_g.

LMKCDEY bootstrapping, unlike the previous FHEW/AP and GINX/TFHE, uses ring automorphism ψ_t and an automorphism key $\mathbf{ak}_t$, where t is an odd number. In summary, LMKCDEY bootstrapping supports arbitrary secret key distribution without additional runtime and generates less noise compared to AP and GINX. It uses two basic building blocks: constant multiplication to exponent using automorphism and adding s_i to exponent by multiplying RGSW(X^{s_i}). For readers who wish to know more details, please refer to [LMK+23].

As a result, we can calculate the standard deviation of noise caused by blind rotation as follows:

$$\text{AP/FHEW:} \beta = \sqrt{2nd_r \cdot d_g N \frac{B_g^2}{12} \sigma^2}$$

$$\text{GINX/TFHE:} \beta = \sqrt{2|U| \cdot 2n \cdot d_g N \frac{B_g^2}{12} \sigma^2}$$

$$\text{LMKCDEY:} \beta = \sqrt{d_g N \frac{{B_g}^2}{12} \left(2n \cdot \sigma^2 + \left(k + \frac{N-k}{w}\right) \cdot \sigma^2\right)}$$

where 2 represents the factor due to the RGSW scheme being composed of tuples, and $\frac{B_g}{12}$ appears as a factor because the RGSW scheme represents the message in a gadget decomposed form, which is equivalent to uniform sampling from the interval $\left[-\frac{B_g}{2}, \frac{B_g}{2}\right]$.

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

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

Cheon et al. proposed key-recovery attacks under existence of decryption oracle (KR$^{\mathrm{D}}$ attacks) exploiting computational failures caused by to noise in Exact FHE [CCP+24]. It is revealed that the parameters used in many libraries for Exact FHE do not satisfy IND-CPA$^{\mathrm{D}}$, and it underscored the need to reduce the noise generated during homomorphic operation. 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 detailed understanding of the methods involved, please refer 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 can be defined as in the following algorithms, where G is any binary circuit such as NAND, DB is a database to store oracles, and I represents input wires as indexes.

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

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

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

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

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

$\mathrm{ct} \longleftarrow \mathrm{EVAL}(G, \mathrm{DB}_{i \in I}.\mathrm{ct})$
$\mathrm{result}_0 \longleftarrow \mathrm{EVAL}(G, \mathrm{DB}_{i \in I}.m_0)$
$\mathrm{result}_1 \longleftarrow \mathrm{EVAL}(G, \mathrm{DB}_{i \in I}.m_1)$
$\mathrm{DB}_i \longleftarrow \{\mathrm{result}_0, \mathrm{result}_1, \mathrm{ct}\}$
$i \leftarrow i + 1$
return ct

Through these oracles, IND-CPA$^{\mathrm{D}}$ attacks on FHEW/TFHE are feasible. Figure 3 represents generic IND-CPA$^{\mathrm{D}}$ attacks on binary FHE. First, an adversary queries to the challenger with $\mathcal{O}_{\mathrm{ENC}}(\{0, 1\})$ and $\mathcal{O}_{\mathrm{ENC}}(\{1, 1\})$. Then the challenger chooses challenge bit b and encrypts messages $\{0, 1\}_b$ and $\{1, 1\}_b$. Denote that if $b = 0$, then $\{A, B\}_b = A$, and if $b = 1$, then $\{A, B\}_b = B$. Now, the adversary takes two ciphertexts, ct_0, and ct_1, where ct_1 always represents the encryption of 1. Afterward, the adversary requests an evaluation oracle to the challenger. Through this, they 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 always yields 1. Following this, the adversary requests a decryption oracle from the challenger to verify the result of $\mathrm{ct}_{\mathrm{res}}$. In this case, if the operations were performed correctly, the result of $\mathrm{ct}_{\mathrm{res}}$ will always be 1 regardless of the value of b. However, in cases of operation failure due to noise inherent in Exact FHE, the result will be 0, making it evident that the failure originated from ct_1.

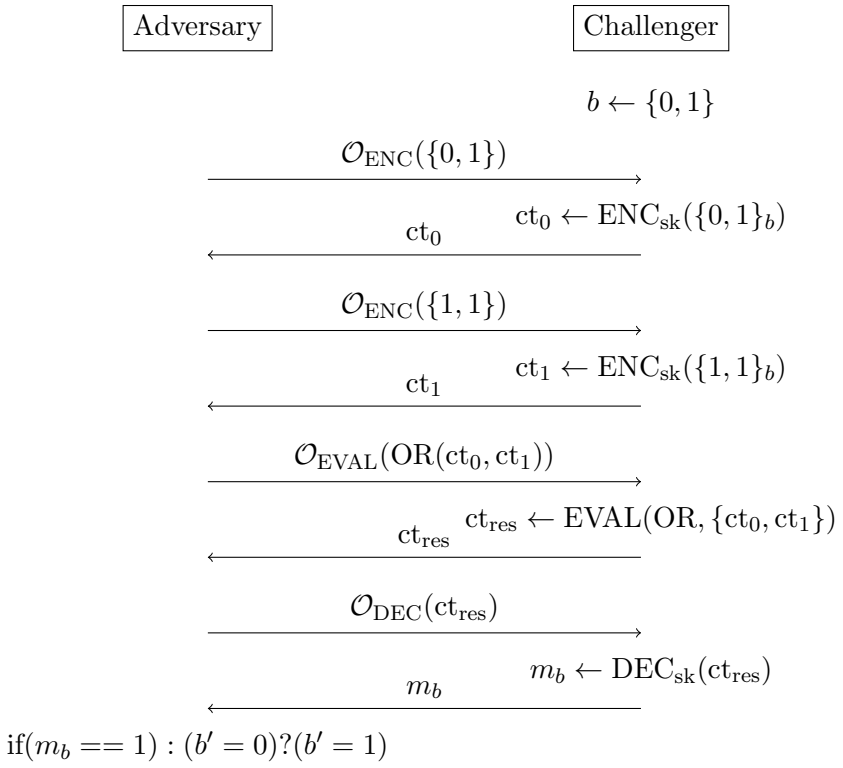


Figure 3: IND-CPA$^{\mathrm{D}}$ attacks on binary FHEW/TFHE

In such a scenario, the adversary can define the probability that $b' = b$ as follows:

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

where F is the event of decryption failure and p is the probability of the event F. Therefore, if p is significantly large, the adversary can obtain sufficient oracles to attack Exact FHE. According to [CCP+24], through various methods, the adversary can increase the attack probability by $\frac{1}{2} + \frac{Np^*}{2}$, where $p^* > p$.

By leveraging the failure probability, one can extend the KR$^{\mathrm{D}}$ attack to Exact FHE. To increase the failure probability, operations are performed using the ciphertext after the final mod switch, as depicted in Figure 2. We described KR$^{\mathrm{D}}$ attack to FHEW/TFHE in Algorithm 4, where $\mathcal{K}$ represents the number of operations that need to be performed to collect failed results for performing KR$^{\mathrm{D}}$ attacks and cnt is the number of failed results. If enough failed results with noise are collected and examined, KR$^{\mathrm{D}}$ attacks become feasible as the distribution of noise varies depending on the secret key value.

3.2 Failure Probability and Performance in FHEW

For the Exact FHE scheme to satisfy IND-CPA$^{\mathrm{D}}$, the attack success probability must satisfy $\frac{1}{2} + \frac{p}{2} \approx \frac{1}{2}$, where p denotes the failure probability of operation. Therefore, the failure probability of the operation, p, should be negligible. This needs to be adjusted according to the security level. For

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

$\mathrm{cnt} = 0$
for $i = 0; i < \mathcal{K}; i = i + 1$ **do**
  $\mathrm{ct}_0 \longleftarrow \mathrm{ENC}_{\mathrm{pk}}(0)$
  $\mathrm{ct}_1 \longleftarrow \mathrm{EVAL}(\mathrm{OR}, \mathrm{ct}_0, \mathrm{ct}_0)$
  $\mathrm{ct}_2 \longleftarrow \mathrm{EVAL}(\mathrm{OR}, \mathrm{ct}_1, \mathrm{ct}_1)$
  $m \longleftarrow \mathrm{DEC}_{\mathrm{sk}}(\mathrm{ct}_2)$
  if $m == 1$ **then**
    $\boldsymbol{e}_{\mathrm{cnt}} \longleftarrow$ evaluate noise from ct_1
    $\mathrm{cnt} = \mathrm{cnt} + 1$
  end
end
$\boldsymbol{e} = \frac{1}{\mathrm{cnt}} \cdot \sum_{j=0}^{\mathrm{cnt}-1} \boldsymbol{e}_j$
if $\left(\boldsymbol{e}_i > \frac{\alpha}{2}\right) ?(s_i = 1) : (s_i = 0)$ for all $i < n$
return s

STD128, the failure probability must satisfy $p \leq 2^{-128}$, while for STD192 and STD256, it must satisfy $p \leq 2^{-192}$ and $p \leq 2^{-256}$, respectively. For special circumstances where the number of queries is very limited, we provide parameter sets for 192 and 256-bit security but failure probability 2^{-128} in Section 6.

To reduce the probability of operation failure, one can consider changing parameters such as n, q, N, and Q. However, since N must be powers of two, there is often a significant gap between the values that can be utilized. While n, q, and Q do not necessarily need to be a power of two, it is a parameter directly affecting the security level. Therefore, they cannot be drastically reduced. Here, the options we can consider include changing digits of decomposition, d_g, d_r, and d_{ks}. However, for d_g, d_r, and d_{ks}, even small changes in values can significantly impact the probability of operation failure. Hence, achieving satisfactory parameter settings may be challenging.

Additionally, to reduce the failure probability, adjusting the parameters will necessitate accepting an increase in runtime due to the increased computational complexity. Due to limited parameter choices, there may be instances where the use of less efficient parameters becomes unavoidable. Figure 4 illustrates the variation in failure probability with respect to d_g at different security levels. As d_g varies, it is evident that the failure probability also undergoes rapid changes within certain ranges. This represents an unnecessary reduction in failure probability, and one must also accept the increase in computational complexity due to the increment in d_g.

In this paper, we propose a blind rotation method that provides a find-grained trade-off between operation failure probability and runtime. Figure 1 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, one can also expect an improvement in runtime due to the reduction in computational complexity.

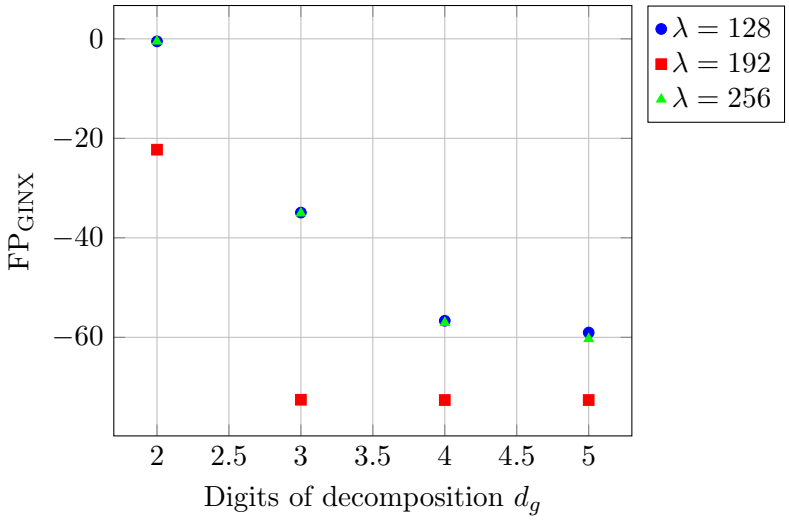


Figure 4: The failure probability varying according to the changes in d_g based on the security level presented in [MP21]. $\mathrm{FP}_{\mathrm{GINX}}$ represents the failure probability when using the GINX method.

4 New Blind Rotation Technique

As discussed in the previous section, a high probability of failure can provide the adversary with more information than necessary, thereby increasing their advantage. To mitigate this risk, it is crucial to reduce the failure probability during bootstrapping to a negligible level. However, the current parameters result in a sufficiently high failure probability that enables successful attacks, necessitating updates to the default parameter sets for FHEW/TFHE. Modifications 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 fine adjustments challenging.

To address this issue and achieve a finer control over the failure probability while optimizing computational complexity, we propose a modification of the blind rotation technique, which we call *cutoff blind rotation*.

In the proposed algorithm, a new parameter, the threshold t, is introduced to influence the bootstrapping process. This threshold allows us to skip certain RLWE $\otimes$ RGSW operations during blind rotation when $|a_i| \leq t$. Consequently, although the failure probability may increase, this approach provides a useful mechanism for fine-tuning the failure probability by reducing computational complexity, thereby minimizing the performance loss for IND-CPA$^{\mathrm{D}}$ security.

4.1 The Cutoff Blind Rotation Algorithm

The proposed cutoff blind rotation algorithm is straightforward. The detailed operation of cutoff blind rotation can be found in Algorithm 5. Although we have only presented the algorithm for the AP method, it can also be applied to the GINX [CGGI17] and LMKCDEY [LMK+23]. As described in Section 2, the bootstrapping procedure enables homomorphic decryption. Specifically, during bootstrapping, we compute $\boldsymbol{f} \cdot X^{\langle \mathbf{a}, \mathbf{s} \rangle}$ homomorphically. The main idea behind cutoff blind rotation is to ignore terms where a_i is sufficiently small, so that omitting $(a_i \cdot s_i)$ does not significantly affect the result of $\langle \mathbf{a}, \mathbf{s} \rangle$. This approach results in performing decryption approximately, which increases

the noise.

However, by reducing the number of RGSW multiplications, the computational complexity decreases and the additive noise from RGSW multiplication is also reduced. This means that in scenarios where parameter choices are limited and lead to a lower failure probability than necessary, adjusting the threshold value can increase the failure probability while simultaneously improving computational efficiency. This method enhances the flexibility of parameter settings concerning the failure probability, providing a broader range of parameter choices.

Algorithm 5: NAND with cutoff blind rotation

Data: ciphertext ct_1, ciphertext 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 $\mathtt{LWEExtract}(\mathrm{Acc}) + Q/8$

4.2 Noise Analysis

Our approach to noise analysis follows [DM15], [MP21], and [LMK+23]. Note that using a cutoff blind rotation affects the number of RLWE $\otimes$ RGSW operations during blind rotation. 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.

Theorem 1. *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, RGSW multiplications in the AP blind rotation are skipped by the probability of $\frac{2t+1}{q}$.

The values encrypted in the RGSW evaluation key here are monomials and the error variance σ^2. Thus, the noise introduced by each RGSW multiplication is $\sigma^2_{ACC} = 2N\frac{B_g^2}{12}\sigma^2$, and it is additive. Since the AP blind rotation for $(\boldsymbol{a}^*, b)$ requires $nd_r\left(1 - \frac{2t+1}{q}\right)$ RGSW multiplications, the noise introduced by the AP blind rotation is given by

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

□ □

Corollary 1. *The noise occurring during GINX cutoff blind rotation is given as follows:*

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

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

$$\sigma^2_{ACC-LMK^+} = d_g N \frac{{B_g}^2}{12} \left(2n \cdot \sigma^2 + \left(k + \frac{N-k}{w} \right) \cdot \sigma^2 \right)$$

where the left term $2n \cdot \sigma^2$ corresponds to mutiplication of RGSW(X^{s_i}) and the right term $\left(k + \frac{N-k}{w}\right) \cdot \sigma^2$ corresponds to automorphsim.

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

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

$$\sigma^2_{ACC-LMK^+} = d_g N \frac{{B_g}^2}{12} \left(2n \cdot \sigma^2 \cdot \left(1 - \frac{2t+1}{q} \right) + \left(k + \frac{N-k}{w} \right) \cdot \sigma^2 \right)$$

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

$$\sigma^2_{TH} = \frac{t^2}{3} \cdot (n \cdot \frac{2t+1}{q}) = \frac{2nt^3 + t^2}{3q}$$

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 those ignored elements is $\frac{t^2}{3}$, and the probability of being ignored is

$\frac{2t+1}{q}$. We note that this noise is not generated during blind rotation or modulus switch operations. Instead, it is considered as noise that the ciphertext always inherently possesses.

In other operations except for blind rotation, the threshold value does not have any impact; therefore we have the following variances of noise:

$$\sigma^2_{MS_1} = \frac{||s_N||^2 + 1}{3}, \quad \sigma^2_{MS_2} = \frac{||s_n||^2 + 1}{3}, \quad \sigma^2_{KS} = \sigma^2 N d_{\mathrm{ks}}$$

$\sigma^2_{MS_1}$ and $\sigma^2_{MS_2}$ represent the noise generated by modulus switches before and after key switching, respectively, and σ^2_{KS} represents the noise that is generated by the key switching method. When calculating the probability of failure, we consider $||s_N||^2$ and $||s_n||^2$ under the assumption that if the secret key is binary or ternary, then $||s_n|| \leq \sqrt{n/2}$ and $||s_N|| \leq \sqrt{N/2}$, respectively.

The final noise that emerges is given as follows:

$$\sigma^2_{total} = \frac{q^2}{Q^2_{ks}} \left(2\frac{Q^2_{ks}}{Q^2}\sigma^2_{ACC} + \sigma^2_{MS_1} + \sigma^2_{KS} \right) + \sigma^2_{MS_2} + \sigma^2_{TH}$$

If the noise of $\mathrm{LWE}_{\boldsymbol{s}}\left(\frac{q}{4} \cdot m\right)$ 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 $1 - \mathrm{erf}\left(\frac{q/8}{\sqrt{2}\cdot\sigma_{total}}\right)$.

4.3 Runtime Analysis

According to [MP21, LMK+23], the number of NTTs while bootstrapping can be defined as follows:

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

This computational complexity can be reduced by applying cutoff blind rotation. As the NTT takes most of the operation in FHEW bootstrapping, we measure the computational complexity by the number of NTTs or RLWE′ multiplications as in [LMK+23]. We recall that the fundamental aspect of cutoff blind rotation is the omission of computations. Now we focus on how many operations can be skipped. If we fix the value of the threshold t, it means that we omit computations for the portion within the range [-t, t], which is uniformly defined within the interval [-q/2, q/2]. Therefore, we can easily calculate the number of NTTs for bootstrapping with the applied threshold:

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

5 Further Improvement and Application to Existing Optimization

5.1 Shifted Mapping

Here, we propose methods to reduce noise when the secret key is binary using threshold blind rotation. In the proposed method, when $(\boldsymbol{a}, b)$ is approximted to $(\boldsymbol{a}^*, b)$, where $a^*_i = 0$ if $|a_i| \leq t$,

$a_i^* = a_i$, other wise. It is noteworthy that we can anticipate $\boldsymbol{a}^*$ in advance. Let $\boldsymbol{e} = \boldsymbol{a}^* - \boldsymbol{a}$, then, noise added by the cutoff approximation is given as

$$(b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle) - (b - \langle \boldsymbol{a^*}, \boldsymbol{s} \rangle) = \langle \boldsymbol{a^*}, \boldsymbol{s} \rangle - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = \langle \boldsymbol{e}, \boldsymbol{s} \rangle.$$

We can precompute $\boldsymbol{e}^*$ prior to blind rotation and estimate the expected mean and variance of $\langle \boldsymbol{e}^*, \boldsymbol{s} \rangle$. In this case, incorporating the expected mean into the mapping function can reduce the failure probability. However, it is noted that this method can only be applied when the secret key distribution is binary. In the case of a binary secret key distribution, the mean of key elements $E[s_i]$ is nonzero, whereas, for a ternary secret key distribution, the mean is zero.

The mean of $\langle \boldsymbol{e}, \boldsymbol{s} \rangle$ is given as $\mu = \sum_i e_i/2$ when $\boldsymbol{s}$ is binary. Given μ we find blind rotation to find RLWE($\boldsymbol{f}^* \cdot X^{b - \langle \boldsymbol{a^*}, \boldsymbol{s} \rangle}$), instead of RLWE($\boldsymbol{f} \cdot X^{b - \langle \boldsymbol{a^*}, \boldsymbol{s} \rangle}$), where $f^*_{-u} = f_{-(u-\mu)}$ and f is mapping polynomial used in Algorithm 5. The detailed algorithm is represented in Algorithm 6; we note that for other gates, simply proper mapping polynomial $\boldsymbol{f}$ needs to be selected as described in [MP21].

Algorithm 6: NAND with cutoff blind rotation and shifted mapping

Data: ciphertext ct_1, ciphertext 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

$(\vec{a}, b) \longleftarrow \mathrm{ct}_1 + \mathrm{ct}_2$
$\vec{a}^* \longleftarrow \texttt{cutoff}(\vec{a})$
$\mu \longleftarrow \sum_i (a_i - a_i^*)/2$
for $i \in [0, q/2)$ **do**
  $k \longleftarrow b - i$
  if $k \in [3q/8 + \mu, 7q/8 + \mu)$ **then**
    $m_{i*2N/q} = -Q/8$
  else
    $m_{i*2N/q} = Q/8$
  end
end
$\mathrm{Acc} \longleftarrow \boldsymbol{m}$
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 $\texttt{LWEExtract}(\mathrm{Acc}) + Q/8$

5.1.1 Noise Analysis with Shifted Mapping

Let e denote the blind rotation without the proposed cutoff technique. In other words, the final decryption results in $m + e$, and the failure occurs when $|e| \geq q/8$. Applying the cutoff technique,

there is more noise introduced by approximating $\boldsymbol{a}$ to $\boldsymbol{a}^*$, and we denote this noise e_c. Now, the decryption fails when $|e + e_c| \geq q/8$, in other words, the decryption succeeds when $-q/8 < e + e_c < q/8$. The important observation here is that the posterior mean of e_c is nonzero when $\boldsymbol{e}$ is determined and the secret is binary. Its mean is $\mu = \sum_i e_i/2$ as explained above.

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

$$\begin{aligned} 1 - \Pr[-q/8 < X < q/8] &= \Pr[X - \mu < -q/8 - \mu] + \Pr[X - \mu > q/8 - \mu] \\ &= \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\frac{q/8 + \mu}{\sqrt{2}\sigma}\right) + \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\frac{q/8 - \mu}{\sqrt{2}\sigma}\right). \end{aligned}$$

However, when we shift the mapping function by μ, the failure probability is

$$\begin{aligned} 1 - \Pr[-q/8 < X - \mu < q/8] &= \Pr[X - \mu < -q/8] + \Pr[X - \mu > q/8] \\ &= 1 - \mathrm{erf}\left(\frac{q/8}{\sqrt{2}\sigma}\right). \end{aligned}$$

It is obvious that

$$\mathrm{erf}\left(\frac{q/8}{\sqrt{2}\sigma}\right) > \frac{1}{2}\mathrm{erf}\left(\frac{q/8 + \mu}{\sqrt{2}\sigma}\right) + \frac{1}{2}\mathrm{erf}\left(\frac{q/8 - \mu}{\sqrt{2}\sigma}\right),$$

and thus the failure probability is reduced.

It is noted that the average of μ is zero when $\boldsymbol{a}$ is not given. However, there is also a higher probability that μ has a nonzero value, and thus we can reduce the failure probability in most cases. In fact, μ follows the (shifted) Irwin-Hall distribution, and the probability of $\mu = 0$ is significantly low when the number of ignored coefficients is large. For example, the probability for nonzero μ is 0.86 for our 256-bit security parameter for GINX.

5.2 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 ignoring elements based on a threshold value, we use the fact that $\vec{a}$ is given in advance, as discussed in Subsection 5.1, and determine the cutoff by considering all the elements in $\vec{a}$. This approach enables a more precise examination of the expected noise due to approximation, resulting in the ability to skip more values.

The detailed algorithm for determining the skipping array is as follows. Let $(a_{(0)}, a_{(1)}, \ldots, a_{(n-1)})$ be the sorted vector of $\vec{a} \in [-q/2, q/2 - 1)^n$ in increasing order of a_i^2. In other words, $a_{(k)}$ has the k-th smallest square value among all a_is.

Given a threshold T, we ignore $a_{(0)}, a_{(1)}, \ldots, a_{(k-1)}$ for the largest k satisfying

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

5.2.1 Noise Analysis

As in the previous section, let $\vec{a}^*$ be a vector derived from $\vec{a}$ where the ignored terms are replaced by zero. The difference is defined as $\vec{e} = \vec{a} - \vec{a}^*$, where the elements of $\vec{e}$ include $a_{(0)}, a_{(1)}, \ldots, a_{(k-1)}$ and $n - k$ zeros. Thus, we can easily find the expected noise varaince σ_{TH}^2 for a given $\boldsymbol{a}$ as in the following theorem. This analysis is straightforward to extend to other secret key distributions.

Theorem 2. *For a given threshold T, the noise introduced by approximating $\boldsymbol{a}$ to $\boldsymbol{a}^*$ using square-sum cutoff has a variance of $\sigma^2_{TH} \leq \frac{2T}{3}$ when the secret key is ternary.*

Proof. We are finding the variance of the random variable

$$\langle \boldsymbol{e}, \boldsymbol{s} \rangle = \sum_{i=0}^{k-1} a_{(i)} \cdot s_{(i)},$$

where $s_{(i)}$ is the element of the key vector corresponding to $a_{(i)}$. When $s_{(i)}$ are i.i.d. random variables uniformly selected from $\{-1, 0, 1\}$, its variance is given as follows:

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

The equation $\mathrm{VAR}[a_{(i)} \cdot s_{(i)}] = a^2_{(i)} \cdot \mathrm{VAR}[s_{(i)}]$ holds because $a_{(i)}$ are constants (since $\boldsymbol{a}$ is given) and not random variables. □ □

Corollary 3. *For a given threshold T, the noise introduced by approximating $\boldsymbol{a}$ to $\boldsymbol{a}^*$ using square-sum cutoff has a variance of $\frac{T}{4}$ when the secret key is binary.*

We note that the mean of the noise can be found as in Subsection 5.1, and thus shifted mapping can also be applied.

5.2.2 Complexity Analysis: The Number of Ignored RGSW Multiplications

We analyze the expected number of ignored RGSW multiplications. As we are calculating the expectation of k, we assume that $\boldsymbol{a}$ is not given. Instead, $\boldsymbol{a}_i$ are i.i.d. random variables uniformly sampled from $\{-q/2, -q/2+1, \ldots, q/2-1\}$.

We define a random variable K_T as the number of ignored elements using square-sum cutoff blind rotation with threshold T. Our goal is to find its expectation, $E[K_T]$. Let $A_i = a_i^2$ be a random variable and $A_{(i)}$ be its i-th order statistic. Using the order statistics, we define another random variable Z_k as the sum of the first k values of $A_{(i)}$, that is,

$$Z_k = \sum_{i=0}^{k-1} A_{(i)}.$$

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

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

if $k < n$, and $\Pr[K_T = n] = \Pr[Z_n \leq T]$. Then, we can simplify its expection as follows:

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

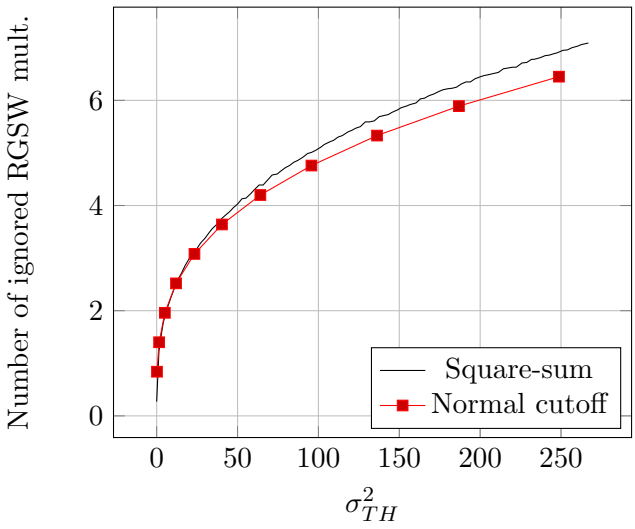


Figure 5: The numerical comparision of normal cutoff bootstrapping and square-sum cutoff.

Simplying further, we have

$$\begin{aligned} E[K_T] &= \sum_{k=0}^{n-1} \left(k \Pr\left[Z_k \leq T\right] - k \Pr\left[Z_{k+1} \leq T\right]\right) + n \Pr[Z_n \leq T] \\ &= \sum_{k=0}^{n-1} \left(k \Pr\left[Z_k \leq T\right]\right) - \sum_{k=1}^{n} \left(k \Pr\left[(Z_k \leq T)\right]\right) + \sum_{k=0}^{n-1} \Pr\left[Z_{k+1} \leq T\right] + n \Pr[Z_n \leq T] \\ &= \sum_{k=0}^{n-1} \Pr\left[Z_{k+1} \leq T\right] \end{aligned} \tag{2}$$

Equation (2) gives us the expected number of RGSW multiplications that can be ignored using the square-sum cutoff with threshold T.

Although the above probability may give us an analytic solution, but it doesn't seem to have a simple closed-from solution for $\Pr\left[(Z_k \leq T)\right]$. Rather, we ran a numerical analysis to check the relationship between threshold T and the number of skipped elements as given in Figure 5. It can be seen that the square-sum cutoff provides a better (or similar) tradeoff between the number of ignored RGSW multiplication and the noise introduced compared to the normal cutoff. Besides, the square-sum cutoff provides a more find-grained control over the noise, and thus suitable if one wants to do even finer optimization on the runtime.

5.2.3 Complexity Analysis: Finding Ignored Elements

The computational overhead of finding $a_{(0)}, a_{(1)}, \ldots, a_{(k-1)}$ is minimal compared to the blind rotation runtime. It requires sorting n integers and performing a linear search of the sorted array to accumulate the sum $\sum_{i=0}^{k} a_{(i)}^2$. Blind rotation involves $O(n)$ RGSW multiplications, and a single RGSW has $d_g + 1$ NTTs. A single NTT has a time complexity of $O(\log N)$ and an overall time complexity of $O(n \log n)$, where $n < N$. Additionally, finding $a_{(0)}, a_{(1)}, \ldots, a_{(k-1)}$ does not require modular reduction, making the computation quite inexpensive.

5.3 Approximate Gadget Decomposition

To efficiently set parameters for optimizing computational complexity, utilizing approximate gadget decomposition can also be a good option. This method is suggested in [CGGI20], and [LMK+23] introduces its simple variant that is working on the integer. In our research, we use the gadget vector $\mathbf{g}_\delta = (\delta B_g^0, \delta B_g^1, \cdots, \delta B_g^{d_g-1})$ instead of $\mathbf{g} = (B_g^0, B_g^1, \cdots B_g^{d_g-1})$ like used in [KLD+23]. So we can redefine RLWE′ and its multiplication operation $\odot$ as follows:

$$\text{RLWE}'(\mathbf{m}) = \Big(\text{RLWE}(\delta B_g^0 \mathbf{m}), \text{RLWE}(\delta B_g^1 \mathbf{m}), \cdots, \text{RLWE}(\delta B_g^{d_g-1} \mathbf{m})\Big) \in \mathcal{R}_Q^{d_g \times 2}$$

$$\odot : \langle \mathbf{a}, \text{RLWE}'(\mathbf{m}) \rangle = \sum_{i=0}^{n} \sum_{j=0}^{d_g-1} \text{RLWE}(a_i B_g^j \delta \mathbf{m}) \approx \text{RLWE}(\mathbf{a} \cdot \mathbf{m})$$

This means it ignores $\log_2 \delta$ bits while performing $\odot$ operation, so it increases the probability of failure and decreases the *digits of gadget decomposition* d_g. In situations where the failure probability is sufficiently low, it is necessary to find an optimized value of δ compared to the increasing noise, which can most effectively improve computational speed.

5.3.1 Noise Analysis with Approximate Gadget Decomposition

We referred to [KLD+23] for noise analysis. Those who desire detailed noise analysis are encouraged to refer to Section 4 of [KLD+23]. The essence of the approximate gadget decomposition lies in the blind rotation process, where $\log_2 \delta$ bits are disregarded, affecting only σ_{ACC} and also being dependent on the threshold value t. In this case, we can define σ_{ACC} based on δ as follows:

$$\sigma^2_{ACC-AP} = d_r \cdot 2n \left(d_g N \frac{B_g^2}{12} \sigma^2 + \frac{\delta^2}{12} (||s_N||^2 + 1) \right) \cdot (1 - \frac{2t+1}{q})$$

$$\sigma^2_{ACC-GINX} = 2u \cdot 2n \left(d_g N \frac{B_g^2}{12} \sigma^2 + \frac{\delta^2}{12} (||s_N||^2 + 1) \right) \cdot (1 - \frac{2t+1}{q})$$

The additional factor introduced here is $\frac{\delta^2}{12}(||s_N||^2 + 1)$. This factor arises from the approximate gadget decomposition, originating from the characteristic of disregarding $\log_2 \delta$ bits during the blind rotation process. Furthermore, this factor is equivalent to the noise generated during the process of secret key $\boldsymbol{s}$ inner product with δ^*, which is extracted from a uniform distribution with range $[-\delta, \delta]$.

5.3.2 Computational Complexity with Approximate Gadget Decomposition

The parameter d_g significantly influences both the noise and computational complexity. Additionally, the approximate gadget decomposition technique serves to minimize the value of d_g while maintaining the noise at a minimum. As mentioned earlier, applying the approximate gadget decomposition results in ignoring values up to $\log_2 \delta$ bits from the least significant bit. In addition, by appropriately adjusting the parameter, we can reduce d_g to $d_g^* = d_g - 1$ with minimal noise.

Now we can define the number of NTTs as follows:

$$\text{AP} : 2n \cdot d_r(1 - \frac{1}{B_r}) \cdot (d_g^* + 1) \cdot (1 - \frac{2t}{q}) = 2n \cdot d_r(1 - \frac{1}{B_r}) \cdot d_g \cdot (1 - \frac{2t+1}{q})$$
$$\text{GINX} : 2n \cdot |U| \cdot (d_g^* + 1) \cdot (1 - \frac{2t}{q}) = 2n \cdot |U| \cdot d_g \cdot (1 - \frac{2t+1}{q})$$

6 Paramter Sets and Implementation Results

In this section, we present the runtime for bootstrapping using our technique and compare it with previous methods. We implemented our blind rotation using OpenFHE, an open-source HE library [Ope22]. Additionally, we provide suitable parameter sets for AP/FHEW and GINX/TFHE when using a ternary secret key distribution.

To demonstrate the flexible applicability of our technique, we propose two parameter sets: one using only our technique and another combining our technique with approximate gadget decomposition [CGGI20, LMK+23]. Our evaluation was conducted using OpenFHE v.1.1.2 on an Intel(R) Core(TM) i9-11900 @ 2.50GHz processor. The code was compiled with clang++ 14, using CMake flags NATIVE_SIZE=32 for ciphertext modulus less than 31 bits, and NATIVE_SIZE=64 otherwise.

6.1 IND-CPA$^{\text{D}}$-secure Parameter Sets

6.1.1 Basic Parameter Sets Without Approximate Gadget Decomposition

The proposed cutoff blind rotation significantly increases the flexibility of parameter choices. Depending on the requirements, the parameter set can be configured to optimize computational and space complexity, or a balanced trade-off between the two. For example, increasing the value of d_{ks} slightly increases the noise but allows for a reduction in key generation time and key size. Conversely, decreasing the value of d_g slightly increases the noise but can decrease runtime. After these adjustments, the threshold t is used to fine-tune the failure probability and runtime.

The proposed parameter sets for 128, 192, and 256-bit security are given in Table 1. The security is measured using the Lattice estimator [APS15]. In configuring these parameter sets, we first reduced Q_{ks} and n. Then, we set d_g and d_{ks} to achieve the best possible performance, and finally, we adjusted the threshold value t to finely tune the failure probability.

Additionally, we provide parameter sets that satisfy different (higher) failure probabilities according to the security level. For specific scenarios where 192 and 256-bit security is required, but the adversary's computational capability is the only threat (without a decryption oracle), we provide additional parameters: Param192* and Param256*, which ensure a failure probability of 2^{-128}.

6.1.2 Parameter Sets with Approximate Gadget Decomposition

Our technique can be applied to various bootstrapping optimizations, facilitating advantageous parameter set configurations. We present parameter sets with efficient computational complexity by utilizing approximate gadget decomposition in conjunction with our proposed technique. These parameter sets are shown in Table 2.

Due to the use of approximate gadget decomposition, the decomposition is given as $d_g = \left\lceil \log_{B_g}(Q/\delta) \right\rfloor$, where δ denotes the approximation factor. We prioritized minimizing d_g by using

Table 1: Proposed optimized parameter sets without approximate gadget decomposition.

	n	q	N	$\log_2 Q$	$\log_2 Q_{ks}$	B_g	B_{ks}	B_r	t	FP_{AP}	FP_{GINX}	FP'_{AP}	FP'_{GINX}
Param128	574	2048	2048	54	15	2^{27}	2^5	2^6	6	2^{-128}	2^{-128}	2^{-141}	2^{-141}
Param192	922	2048	2048	37	16	2^{13}	2^4	2^6	3	2^{-197}	2^{-196}	2^{-203}	2^{-203}
$\text{Param256}_{\text{GINX}}$	1223	4096	2048	29	16	2^6	2^4	2^6	9	-	2^{-267}	-	2^{-334}
$\text{Param256}_{\text{AP}}$	1223	4096	2048	29	16	2^6	2^4	2^6	10	2^{-256}	-	2^{-346}	-
Param192*	922	2048	2048	37	16	2^{13}	2^3	2^6	7	2^{-128}	2^{-128}	2^{-174}	2^{-174}
Param256*	1223	2048	2048	29	16	2^6	2^4	2^6	6	2^{-129}	2^{-128}	2^{-166}	2^{-163}
Param128_bin	620	2048	2048	54	15	2^{27}	2^5	2^6	5	2^{-129}	2^{-129}	2^{-138}	2^{-138}
Param192_bin	998	2048	2048	37	16	2^{13}	2^4	2^6	2	2^{-192}	2^{-192}	2^{-193}	2^{-193}
Param256_bin	1322	4096	2048	29	16	2^6	2^4	2^6	9	2^{-264}	2^{-264}	2^{-335}	2^{-335}
Param192_bin*	998	2048	2048	37	16	2^{13}	2^4	2^6	7	2^{-135}	2^{-135}	2^{-193}	2^{-193}
Param256_bin*	1322	2048	2048	29	16	2^6	2^4	2^6	5	2^{-134}	2^{-134}	2^{-157}	2^{-157}

the smallest possible δ. Subsequently, we minimized Q_{ks} and n, and finally set the threshold value t to ensure a failure probability of 2^{-128} or to meet the respective conditions according to the security level. The notation A/B in the threshold value t indicates that A represents the value for the AP method, while B represents the value for the GINX method.

Table 2: Proposed parameter sets with approximate gadget decomposition and optimization.

	n	q	N	$\log_2 Q$	$\log_2 Q_{ks}$	B_g	B_{ks}	B_r	δ	t	FP_{AP}	FP_{GINX}	FP'_{AP}	FP'_{GINX}
Param128_ag	574	2048	2048	54	15	2^{27}	2^5	2^6	2^0	6	2^{-128}	2^{-128}	2^{-141}	2^{-141}
Param192_ag	992	2048	2048	37	16	2^{13}	2^4	2^6	2^0	3	2^{-197}	2^{-196}	2^{-203}	2^{-203}
Param256_ag	1223	4096	2048	29	16	2^7	2^4	2^6	2^1	9/7	2^{-258}	2^{-263}	2^{-320}	2^{-289}
$\text{Param192\_ag}^*{}_{\text{AP}}$	922	2048	2048	37	16	2^{17}	2^6	2^6	2^3	6	2^{-129}	-	2^{-154}	-
$\text{Param192\_ag}^*{}_{\text{GINX}}$	922	2048	2048	37	16	2^{12}	2^6	2^6	2^1	8	-	2^{-131}	-	2^{-223}
$\text{Param256\_ag}^*{}_{\text{AP}}$	1151	2048	2048	29	15	2^7	2^8	2^6	2^1	2	2^{-129}	-	2^{-130}	-
$\text{Param256\_ag}^*{}_{\text{GINX}}$	1223	2048	2048	29	16	2^7	2^4	2^6	2^1	5	-	2^{-132}	-	2^{-152}
Param128_bin_ag	620	2048	2048	54	15	2^{27}	2^5	2^6	2^0	5	2^{-129}	2^{-129}	2^{-138}	2^{-138}
Param192_bin_ag	998	2048	2048	37	16	2^{13}	2^4	2^6	2^0	2	2^{-192}	2^{-192}	2^{-193}	2^{-193}
Param256_bin_ag	1322	4096	2048	29	16	2^6	2^4	2^6	2^1	7	2^{-264}	2^{-264}	2^{-293}	2^{-293}
Param192_bin_ag^*	998	2048	2048	37	16	2^{17}	2^6	2^6	2^3	5	2^{-130}	2^{-130}	2^{-145}	2^{-145}
Param256_bin_ag^*	1322	2048	2048	29	16	2^7	2^8	2^6	2^1	5	2^{-130}	2^{-130}	2^{-151}	2^{-151}

For reference, we also provide runtime results for the parameter sets proposed in [MP21]. These parameter sets are shown in Table 3, but it should be noted that these parameters are not IND-CPA^{D} secure.

6.2 Bootstrapping Runtime

We compare the key generation time and runtime for different security levels using the parameter sets with approximate gadget decomposition proposed earlier and the parameter sets currently used in the OpenFHE library. The parameter sets currently used in the OpenFHE library do not satisfy

Table 3: Parameter sets proposed in [MP21].

	n	q	N	$\log_2 Q$	$\log_2 Q_{ks}$	B_g	B_{ks}	B_r	$\mathrm{FP}_{\mathrm{AP}}$	$\mathrm{FP}_{\mathrm{GINX}}$
STD128_OPT	502	1024	1024	27	14	2^7	2^7	2^5	2^{-52}	2^{-48}
STD128_APOPT	502	1024	1024	27	14	2^9	2^7	2^5	2^{-36}	-
STD192_OPT	755	1024	2048	37	15	2^{13}	2^5	2^5	2^{-63}	2^{-63}
STD256_OPT	1225	1024	2048	27	16	2^7	2^4	2^5	2^{-37}	2^{-33}

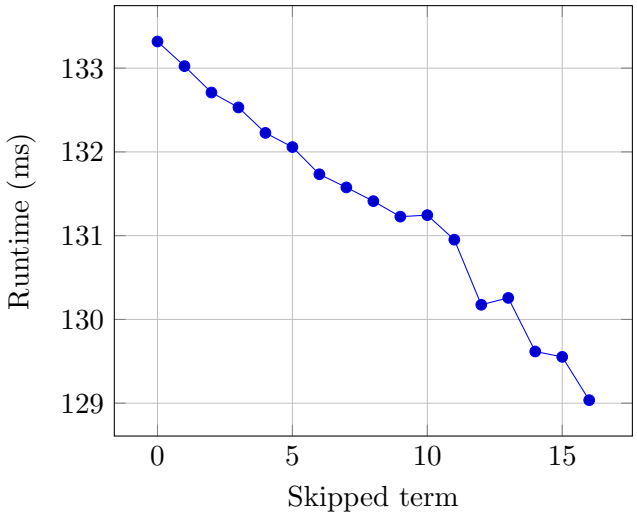


Figure 6: Number of skipped LWE elements vs GINXbootstrapping runtime result with parameter Param128_ag.

the condition $\mathrm{FP} \leq 2^{-128}$ for IND-CPA$^{\mathrm{D}}$ security.

The runtime was measured by performing the NAND gate 1,000 times. Here, the OpenFHE cmake option NATIVE_SIZE was set to 32 when $Q < 30$, and NATIVE_SIZE was set to 64, otherwise. The experimental results are presented in Table 4.

Additionally, we performed a performance comparison between parameter sets. We compared the conventional parameter set with the proposed parameter sets, for which both versions with and without the proposed cutoff blind rotation. The results is given in Tables 5 and 6. The symbol # represents the number of $\odot$ operations.

The threshold value affects runtime, but its impact on noise and runtime is smaller compared to d_g. Consequently, the change in runtime by to the threshold value is difficult to detect with a small (e.g., $1,000$) number of experiments. To illustrate the effect of the threshold value on runtime, we measured the average runtime based on the number of omitted RGSW multiplications. We set the threshold value to $t = 8$ and conducted experiment using the Param128_ag parameter set with the GINX method applied. Additionally, we included the results obtained from performing 60,000 NAND gate operations, as shown in Figure 6.

Table 4: Comparison of bootstrapping runtime with previous parameter sets with failure probability $\approx 2^{-40}$

	Previous [MP21]					INC-CPA^{D} secure (w/o cutoff)			
	AP		GINX			AP		GINX	
Param	KeyGen	Boot	KeyGen	Boot	Param	KeyGen	Boot	KeyGen	Boot
STD128_OPT	2.64s	78.84ms	1.49s	61.43ms	Param128	22.58s	172.39ms	18.27s	143.00ms
STD192_OPT	8.55s	259.80ms	4.83s	197.18ms	Param192	38.59s	281.97ms	27.36s	213.01ms
STD256_OPT	43.38s	387.48ms	15.12s	291.78ms	Param256	48.50s	531.42ms	34.01s	399.52ms

Table 5: Comparison of failure probability (FP) and computational complexity with and without cutoff. # is the number of $\odot$ operations.

	without cutoff				cutoff			
	AP		GINX		AP		GINX	
	FP	#	FP	#	FP	#	FP	#
Param128	2^{-141}	2259	2^{-141}	2294	2^{-128}	**2245**	2^{-128}	**2281**
Param192	2^{-203}	3628	2^{-203}	3686	2^{-197}	**3617**	2^{-196}	**3675**
Param256	2^{-346}	4814	2^{-334}	4890	2^{-256}	**4790**	2^{-267}	**4869**
Param192*	2^{-174}	3628	2^{-174}	3686	2^{-128}	**3603**	2^{-128}	**3660**
Param256*	2^{-166}	4784	2^{-166}	4860	2^{-128}	**4813**	2^{-128}	**4889**

Table 6: Comparison of failure probability (FP) and computational complexity with and without cutoff when approximate gadget decomposition is applied. # is the number of $\odot$ operations.

	without cutoff				cutoff			
	AP		GINX		AP		GINX	
	FP	#	FP	#	FP	#	FP	#
Param128_ag	2^{-141}	2259	2^{-141}	2294	2^{-128}	**2245**	2^{-128}	**2281**
Param192_ag	2^{-203}	3628	2^{-203}	3686	2^{-197}	**3617**	2^{-196}	**3675**
Param256_ag	2^{-320}	4814	2^{-320}	4890	2^{-258}	**4793**	2^{-263}	**4874**
Param192_ag*	2^{-154}	3628	2^{-223}	3686	2^{-129}	**3607**	2^{-131}	**3657**
Param256_ag*	2^{-130}	4529	2^{-152}	4889	2^{-129}	**4520**	2^{-132}	**4865**

7 Conclusion

We propose a new optimization technique applicable to existing blind rotation methods with minimal changes to the algorithm. By introducing the cutoff blind rotation, we enable more flexible parameter settings, achieving an appropriate trade-off between failure probability and runtime. Unlike traditional parameters, the threshold 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} significantly affected the failure probability, making parameter fine-tuning for very low failure probabilities challenging. As discussed in [CCP+24], FHEW/TFHE HE schemes are insecure against IND-CPA$^{\mathrm{D}}$ adversaries. However, when lowering the failure probability lower than 2^{-128} for security, the sparse parameter choices could lead to significant inefficiencies in runtime. Our proposed technique allows for the precise tuning of parameters to achieve the desired failure probability and improve runtime.

The proposed method can also be applied to Torus [CGGI17] and NTRU [BIP+22, XZDF23] variants. While these algorithms have different parameter sets and are subject to attacks such as that in [CCP+24], finding parameters with negligible failure probability is crucial. Additionally, our technique can be extended to blind rotations with higher bits, as seen in [LMP22, BBB+23]. Future work could explore the application of the proposed method to amortized bootstrapping [LW23a, LW23b, LW23c, MKMS23] to improve runtime.

References

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

[APS15] Martin Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[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. Cryptology ePrint Archive, 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.

[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 $\tilde{o}(1)$ 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.

[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.