Practical Circuit Privacy/Sanitization for TFHE

Intak Hwang, Seonhong Min, and Yongsoo Song

Seoul National University, Seoul, Republic of Korea
{intak.hwang,minsh,y.song}@snu.ac.kr

Abstract. Fully homomorphic encryption (FHE) enables the computation of arbitrary circuits over encrypted data. A widespread application of FHE is a simple two-party computation (2PC) protocol, where the server evaluates a circuit over the client's encrypted data and its private inputs. However, while the security of FHE guarantees that the client's data is protected from the server, there is no inherent support for the privacy of the server's input and the circuit.

One effective solution to this problem is an additional algorithm for FHE called sanitization, introduced by Ducas and Stehlé (Eurocrypt 2016). Roughly speaking, a sanitization algorithm removes any meaningful information contained in the ciphertext, including previous evaluations of circuits. Following their definition, several constructions for sanitization have been proposed, particularly for TFHE. However, all of these methods were impractical, requiring several bootstrappings or an excessive amount of randomized evaluations.

In this work, we construct a novel sanitization algorithm for TFHE that overcomes these issues. Our method only adds two lightweight randomization steps to the original TFHE bootstrapping, without any modifications to the core algorithms. As a result, our algorithm achieves sanitization with a single bootstrapping and minimal randomization, bringing sanitization closer to practicality.

To empirically evaluate the efficiency of our method, we provide concrete benchmark results based on our proof-of-concept implementation. Our algorithm sanitizes a single TFHE ciphertext in 35.88 ms, which is only 3.4% (1.18 ms) slower than the original TFHE bootstrapping with the same parameters. When directly compared to previous works, our method achieves a speedup by a factor of 4.82 to 209.03.

Keywords: Fully Homomorphic Encryption, TFHE, Circuit Privacy, Sanitization

1 Introduction

Fully Homomorphic Encryption (FHE) is a cryptosystem that allows the evaluation of an arbitrary circuit over encrypted data, without any limitation on the circuit depth. Thanks to such functionality, it can be utilized to construct a low-communication two-party computation protocol for scenarios where the client owns private data and the server owns a circuit. In this protocol, the

client first encrypts its private data using an FHE scheme and sends it to the server. Then, the server evaluates some circuit over the given ciphertexts and its own private data, and returns the output ciphertext to the client. Finally, the client decrypts the returned ciphertext and obtains the evaluation result on its data. By the security of the underlying FHE scheme, it is guaranteed that the server learns nothing about the private data of the client nor the evaluation result. However, generally the converse is not true: the client can learn some information about the server's data, namely the circuit being evaluated or the server's private inputs.

To solve this problem, Gentry [Gen09] introduced the concept of circuit privacy. Roughly speaking, an FHE scheme is circuit-private if the output ciphertext does not contain any information about the circuit after being evaluated. There is a line of works that studies using multi-party computation techniques, such as Garbled Circuit or Oblivious Transfer, to achieve circuit privacy [GHV10,OPP14,DD22]. However, they are mainly of theoretical interest, often suffering from high communication costs and computational complexity. A more common approach to achieving circuit privacy is noise flooding [Gen09], which erases the information left over in the ciphertext by adding an encryption of zero with an exponentially large noise term. Naturally, this leads to exponential overhead in the parameters, as one needs to guarantee correct decryption even with the added noise. In RLWE-based FHE schemes such as BFV [Bra12,FV12], BGV [BGV14] and CKKS [CKKS17], this can be a minor problem, as the usual parameters of these schemes are already large, with ciphertext modulus often spanning hundreds of bits. However, some FHE schemes, such as FHEW [DM15] and TFHE [CGGI16], use very compact parameters, making the noise flooding method impossible.

Alternatively, Ducas and Stehlé [DS16] proposed the soak-spin-repeat approach. First, they defined a special algorithm for FHE ciphertexts called sanitization, which transforms a ciphertext in a way that any two ciphertexts that encrypt the same plaintext are indistinguishable. The soak-spin-repeat method achieves sanitization by repeating two operations: adding a moderately sized noise, and bootstrapping. The general idea is that by repetitively adding noise to two ciphertexts, their statistical distance decreases. Therefore, by sufficiently repeating this cycle, two ciphertexts eventually become statistically indistinguishable. The obvious disadvantage of this approach is that one needs to perform several consecutive bootstrappings. For example, in usual TFHE parameters, to achieve a sufficiently low statistical distance, the number of cycles required is around 5 to 10. This implies that soak-spin-repeat sanitization is 5 to 10 times slower than normal TFHE bootstrapping.

Recently, Bourse and Izabachène [BI22] and Kluczniak [Klu22] independently proposed an alternative construction for TFHE that overcomes this issue. The core idea of their work lies in randomizing the bootstrapping, more specifically randomizing the external product in a similar way to [ASP14,BdMW16]. Instead of computing the external product $\mathsf{ACC} \boxdot \mathsf{BRK} = h(\mathsf{ACC})^\top \cdot \mathsf{BRK}$ directly, they replace the gadget decomposition $h(\mathsf{ACC})$ with Gaussian random variables sam-

pled from a coset $h(\mathsf{ACC}) + \Lambda^{\perp}(\mathbf{g})$. Then, using the convolution property of the Gaussian distribution, we can remove the information about ACC in the error term of the output ciphertext by adding a moderately sized Gaussian noise. Finally, to erase the information in the mask, they add a public key encryption of zero (RLWE in the case of [BI22], and LWE in the case of [Klu22]) to the bootstrapped ciphertext. While their construction achieves sanitization with only a single bootstrapping, it practically often performs worse than soak-spin-repeat sanitization. The main culprit of this issue is their heavy use of the discrete Gaussian distribution, which is notoriously hard to sample efficiently. To be precise, they require $O(|\mathbf{g}|nN)$ Gaussian samples, which translates to millions of samples in practical parameters. Therefore, achieving efficient circuit privacy or sanitization in TFHE remains an open problem.

1.1 Our Contribution

In this paper, we construct a novel sanitization algorithm for the TFHE scheme, which solves the aforementioned issues: it only requires a single, unmodified TFHE bootstrapping, in contrast to [DS16], and the required number of Gaussian samples is only $O(N)$, which is significantly lower than the constructions of [BI22,Klu22]. As a result, the performance of our sanitization algorithm is significantly faster than previous works, nearly matching the original TFHE bootstrapping.

The core idea of our sanitization method is derived from the observation that the bootstrapping algorithm of TFHE can be written as a linear combination of a monomial and an RLWE ciphertext, each only containing the information about the body and the mask of the input ciphertext, respectively. In other words, for $(b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$,

$$\begin{aligned}&\mathtt{BlindRotate}(\mathsf{BRK}, \mathtt{KeySwitch}(\mathsf{KSK}, (b, \mathbf{a})))\\ &= X^{\overline{b}} \cdot \mathtt{BlindRotate}(\mathsf{BRK}, \mathtt{KeySwitch}(\mathsf{KSK}, (0, \mathbf{a})))\end{aligned}$$

where $\overline{b}$ only depends on b. Using this relationship, we add a simple post-processing and pre-processing step to the bootstrapping algorithm, each erasing the information about $\mathbf{a}$ and b, respectively. To be precise, we first rerandomize the mask of the input ciphertext $\mathbf{a}$ to some vector $\mathbf{a}' \in \mathbb{Z}_q^N$ which is uniformly random. Then, the distribution of the RLWE ciphertext $\mathsf{ACC} := \mathtt{BlindRotate}(\mathsf{BRK}, \mathtt{KeySwitch}(\mathsf{KSK}, (0, \mathbf{a}')))$ becomes simulatable without any modifications to `BlindRotate` and `KeySwitch`, since all of the inputs $\mathsf{BRK}, \mathsf{KSK}, \mathbf{a}'$ are known constants or follow a known distribution. Then, after bootstrapping, we compute $X^{\overline{b}} \cdot \mathsf{ACC}$ with a randomized linear evaluation algorithm, which hides the information of $X^{\overline{b}}$. As a result, all information of b and $\mathbf{a}$ is removed, achieving sanitization.

For the pre-processing step, we propose a new mask randomization method based on the construction of [BI22]. In [BI22], the authors showed that the mask can be randomized by adding an RLWE public key encryption of zero by

sampling from a moderately sized discrete Gaussian distribution. By doing so, the mask of the resulting ciphertext follows a uniform distribution, independent of the secret key. We improve their method by replacing the discrete Gaussian distribution with rounded Gaussians, which is significantly faster to sample. We stress that our modification is not based on heuristics, as we provide a rigorous security proof for using a rounded Gaussian distribution.

On the other hand, the circuit-private linear HE scheme by de Castro et al. [dCKK+24] can be used for randomized linear evaluation in the post-processing step. Briefly speaking, given an affine function $\mathfrak{a}x + \mathfrak{b}$ over the polynomial ring R_p, we sample $\mathfrak{r}$ from a coset $\mathfrak{a} + p\mathbb{Z}^N$ and homomorphically compute $\mathfrak{r}x + \mathfrak{b}$. Then, we add a moderately sized Gaussian noise to smooth the noise distribution. This allows us to perform the linear evaluation obliviously, without leaking partial information on the coefficients $\mathfrak{a}$ and $\mathfrak{b}$ to the output ciphertext. However, their method was only proven secure when the input ciphertext was a (fresh) public key encryption. We expand on their work so that it supports an arbitrary ciphertext. In addition, we replace the discrete Gaussian distribution with a rounded Gaussian distribution and prove its security, similar to the mask randomization explained above.

Finally, we implement our sanitization algorithm in Go and provide benchmark results. The experimental results show that the latency of sanitization is reduced from 7500ms of [BI22] and 1330ms of [Klu22] to 35.88ms, yielding a speedup with a factor of 209.03 and 37.07, respectively. Compared to the soak-spin-repeat strategy [DS16], it also gives a 4.82× speedup, with a significantly lower failure rate. Moreover, our method merely adds 1.18ms of overhead to the original TFHE bootstrapping, which is only 3.4% of the total runtime of bootstrapping.

While we focus on the TFHE scheme in this paper, our method can be applied to any AP-like cryptosystems that utilize monomials to bootstrap, such as FHEW [DM15] or LMKCDEY [LMK+23], without heavy modifications. We also stress that our sanitization algorithm only uses existing TFHE bootstrapping subroutines, such as `BlindRotate` and `KeySwitch`, as a black box. As a result, recent advances in TFHE bootstrapping [LMSS23,LY23,BCL+23] can be immediately applied to our algorithm, enhancing the performance even further.

2 Preliminaries

2.1 Notation

We denote the quotient polynomial ring $\mathbb{Z}[X]/(X^N + 1)$ by R, for some power of two N. The quotient ring R/qR is written as R_q. For a ring element $a = \sum_{i=0}^{N-1} a_i X^i$, we define its coefficient vector $\mathbf{a}$ and negacyclic matrix $\mathbf{A}$ as

$$\mathbf{a} = (a_0, \ldots, a_{N-1}), \quad \mathbf{A} = \begin{bmatrix} a_0 & -a_{N-1} & \cdots & -a_1 \\ a_1 & a_0 & \cdots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & \cdots & a_0 \end{bmatrix}.$$

We denote the ℓ-norm of a vector $\mathbf{x}$ by $\|\mathbf{x}\|_\ell$, and the ℓ-norm of a matrix $\mathbf{M}$ by $\|\mathbf{M}\|_\ell$. For a positive definite matrix $\mathbf{\Sigma}$, $\sqrt{\mathbf{\Sigma}}$ denotes a matrix that satisfies $\mathbf{\Sigma} = \sqrt{\mathbf{\Sigma}}^\top \sqrt{\mathbf{\Sigma}}$. We write $\mathbf{B} \geq \mathbf{A}$ if $\mathbf{B}\mathbf{B}^\top - \mathbf{A}\mathbf{A}^\top$ is positive semi-definite. For simplicity, we write $\mathbf{B} \geq k$ for $k \in \mathbb{R}$ if $\mathbf{B} \geq k\mathbf{I}$.

For a probability distribution $\mathcal{X}$, we denote sampling x from $\mathcal{X}$ as $x \leftarrow \mathcal{X}$. For sampling from a uniform distribution over a set S, we write $x \xleftarrow{\$} S$. If a distribution $\mathcal{X}$ is multivariate, we denote the distribution of the i-th element as $\mathcal{X}|_i$. The statistical distance between two distributions $\mathcal{X}_1$ and $\mathcal{X}_2$ with the same support Ω is defined as $\Delta_{\mathrm{SD}}(\mathcal{X}_1, \mathcal{X}_2) = \frac{1}{2}\sum_{x\in\Omega}|\mathcal{X}_1(x) - \mathcal{X}_2(x)|$. We write $\mathcal{X}_1 \overset{\varepsilon}{\approx} \mathcal{X}_2$ when $\Delta_{\mathrm{SD}}(\mathcal{X}_1, \mathcal{X}_2) \leq \varepsilon$. We say that two distributions are statistically indistinguishable if the statistical distance between them is negligible, and write $\mathcal{X}_1 \approx_s \mathcal{X}_2$. Similarly, we say that two distributions are computationally indistinguishable if no probabilistic polynomial-time (PPT) algorithm can distinguish between them, and write $\mathcal{X}_1 \approx_c \mathcal{X}_2$.

2.2 Discrete Gaussian over Lattices

The n-dimensional Gaussian function $\rho_{\mathbf{c},\sqrt{\mathbf{\Sigma}}}$ with center $\mathbf{c}$ and parameter $\sqrt{\mathbf{\Sigma}}$ is defined as

$$\rho_{\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c})\right).$$

We define the (discrete) Gaussian distribution over a lattice coset $\mathbf{a}+\mathit{\Lambda}$ and the continuous Gaussian distribution over $\mathbb{R}^n$ with center $\mathbf{c}$ and parameter $\sqrt{\mathbf{\Sigma}}$ as

$$\mathcal{D}_{\mathbf{a}+\mathit{\Lambda},\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) = \frac{\rho_{\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{x})}{\sum_{\mathbf{v}\in\mathbf{a}+\mathit{\Lambda}}\rho_{\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{v})}, \quad \mathcal{D}_{\mathbb{R}^n,\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) = \frac{\rho_{\mathbf{c},\sqrt{\mathbf{\Sigma}}}(\mathbf{x})}{\sqrt{2\pi\det\mathbf{\Sigma}}}$$

respectively. We also define the rounded Gaussian distribution as $\left\lfloor \mathcal{D}_{\mathbb{R}^n,\mathbf{c},\sqrt{\mathbf{\Sigma}}} \right\rceil := \{\lfloor \mathbf{x} \rceil \mid \mathbf{x} \leftarrow \mathcal{D}_{\mathbb{R}^n,\mathbf{c},\sqrt{\mathbf{\Sigma}}}\}$. We simplify the notation in special cases. When $\mathbf{c} = \mathbf{0}$, we omit $\mathbf{c}$. If $\mathbf{\Sigma} = \sigma^2\mathbf{I}$ for some $\sigma > 0$, we denote σ instead of $\sqrt{\mathbf{\Sigma}}$. For a polynomial a and its coefficient vector $\mathbf{a}$, we denote the lattice coset $\mathbf{a} + p\mathbb{Z}^N$ as $a + p\mathbb{Z}^N$.

2.3 LWE and RLWE

In the following sections, we describe the RGSW cryptosystem and the TFHE scheme. Its security relies on the hardness of the LWE problem [Reg09] and its ring variant, the RLWE problem [LPR13].

Definition 1 (LWE). *Let m, n, q be positive integers, and χ, ψ be a distribution over $\mathbb{Z}^n$ and $\mathbb{Z}$ respectively. The decisional LWE problem $\mathsf{LWE}^{m,n,q}_{\chi,\psi}$ is to distinguish the following two distributions.*

$$\{(\mathbf{A}, \mathbf{A}\mathbf{s}+\mathbf{e}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{s} \leftarrow \chi, \mathbf{e} \leftarrow \psi^m\}, \ \{(\mathbf{A}, \mathbf{u}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m\}.$$

If χ or ψ is a Gaussian distribution with parameter σ, we simply denote them as σ in the subscript.

Definition 2 (Ring-LWE). *Let m, n, q be positive integers, and χ, ψ be a distribution over $R = \mathbb{Z}[X]/(X^n + 1)$. The decisional RLWE problem $\mathsf{RLWE}^{m,n,q}_{\chi,\psi}$ is to distinguish the following two distributions.*

$$\{(\mathbf{a}, \mathbf{a}\mathbf{s} + \mathbf{e}) \mid \mathbf{a} \xleftarrow{\$} R_q^m, \mathbf{s} \leftarrow \chi, \mathbf{e} \leftarrow \psi^m\},\ \{(\mathbf{a}, \mathbf{u}) \mid \mathbf{a}, \mathbf{u} \xleftarrow{\$} R_q^m\}.$$

If χ or ψ is a Gaussian distribution with parameter σ, we simply denote them as σ in the subscript.

Under these hardness assumptions, we can construct LWE and RLWE cryptosystems. An LWE ciphertext has the form $(b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$, and an RLWE ciphertext has the form $(b, a) \in R_q^2$. For convenience, we introduce the *phase* for both LWE and RLWE ciphertexts. For the LWE secret $\mathbf{s}$ and RLWE secret t, the LWE phase $\varphi_{\mathbf{s}} : \mathbb{Z}_q^{n+1} \to \mathbb{Z}_q$ is defined by $\varphi_{\mathbf{s}}(b, \mathbf{a}) := b + \langle \mathbf{a}, \mathbf{s} \rangle$, and the RLWE phase $\varphi_t : R_q^2 \to R_q$ is defined by $\varphi_t(b, a) := b + a \cdot t$.

2.4 Homomorphic Encryption

Definition 3 (Homomorphic Encryption [Gen09]). *A homomorphic encryption scheme Π is a tuple of PPT algorithms* $\mathtt{Setup}$, $\mathtt{KeyGen}$, $\mathtt{Enc}$, $\mathtt{Dec}$ *and* $\mathtt{Eval}$, *defined as follows.*

- $\mathtt{Setup}(1^\lambda)$*: Given a security parameter λ, output a public parameter* pp.
- $\mathtt{KeyGen}(\mathsf{pp})$*: Given a public parameter* pp*, output a secret key and evaluation key* $(\mathsf{sk}, \mathsf{evk})$.
- $\mathtt{Enc}(\mathsf{sk}, \mathsf{pt})$*: Given a secret key* sk *and plaintext* pt*, output a ciphertext* ct.
- $\mathtt{Dec}(\mathsf{sk}, \mathsf{pt})$*: Given a secret key* sk *and ciphertext* ct*, output a plaintext* pt.
- $\mathtt{Eval}(\mathsf{evk}, C, (\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell))$*: Give an evaluation key* evk*, circuit C and ciphertexts* $\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell$*, output a ciphertext* $\mathsf{ct}_{\mathrm{out}}$.

Π is said to be correct if for any circuit C with ℓ input wires and $x_1, \ldots, x_\ell$,

$$\mathtt{Dec}(\mathsf{sk}, \mathtt{Eval}(\mathsf{evk}, C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))) = C(x_1, \ldots, x_\ell)$$

with an overwhelming probability where $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$ and $\mathsf{ct}_i \leftarrow \mathtt{Enc}(\mathsf{sk}, x_i)$ for $1 \le i \le \ell$. Π is also said to be compact if ciphertexts are of polynomial size.

Definition 4 (Circuit Privacy [Gen09]). *A homomorphic encryption scheme $\Pi = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Eval})$ is circuit private for a class of circuits $\mathcal{C}$ if there exists a PPT algorithm* $\mathtt{Sim}$ *such that for any circuit $C \in \mathcal{C}$ and $x_1, \cdots, x_\ell$,*

$$\begin{aligned}&\{(\mathsf{sk}, \mathsf{evk}, (\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell), \mathtt{Eval}(\mathsf{evk}, C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)))\} \\ &\quad \approx_c \{(\mathsf{sk}, \mathsf{evk}, \mathtt{Sim}(\mathsf{evk}, (\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell), C(x_1, \ldots, x_\ell)))\}\end{aligned}$$

where $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathtt{KeyGen}(\mathsf{pp})$ *and* $\mathsf{ct}_i \leftarrow \mathtt{Enc}(\mathsf{sk}, x_i)$ *for* $1 \le i \le \ell$.

Definition 5 (Sanitization [DS16]). *For a homomorphic encryption scheme* $\Pi = (\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Eval})$*, a PPT algorithm* $\mathtt{Sanitize}(\mathsf{pk}, \mathsf{ct})$ *is said to be message-preserving if for any ciphertext* ct*,*

$$\mathtt{Dec}(\mathsf{sk}, \mathtt{Sanitize}(\mathsf{evk}, \mathsf{ct})) = \mathtt{Dec}(\mathsf{sk}, \mathsf{ct})$$

with an overwhelming probability where $(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathtt{KeyGen}$*. Moreover, we say* $\mathtt{Sanitize}$ *is sanitizing if for any two ciphertexts* $\mathsf{ct}, \mathsf{ct}'$ *such that* $\mathtt{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathtt{Dec}(\mathsf{sk}, \mathsf{ct}')$*,*

$$\{(\mathsf{sk}, \mathsf{evk}, \mathtt{Sanitize}(\mathsf{evk}, \mathsf{ct})\} \approx_c \{(\mathsf{sk}, \mathsf{evk}, \mathtt{Sanitize}(\mathsf{evk}, \mathsf{ct}')\}.$$

Note that sanitization implies circuit privacy in the semi-honest model, since we can construct a circuit-private evaluation algorithm by sanitizing the output ciphertext after evaluating a circuit.

Circuit Privacy/Sanitization in malicious model, where the keys and ciphertexts may not be well-formed, is also studied. We note that Ostrovsky et al. [OPP14] gives a generic transformation from semi-honest circuit-private FHE scheme to maliciously circuit-private one.

Therefore, in this paper, we only focus on sanitization in the semi-honest model.

2.5 Gadget Decomposition

The gadget toolkit is a widely used technique in HE construction for noise management. A fixed vector $\mathbf{g} \in \mathbb{Z}^d$ is called a *gadget vector*, and the map $h : \mathbb{Z}_q \to \mathbb{Z}^d$ is called a *gadget decomposition* corresponding to $\mathbf{g}$ if h satisfies the following conditions for any $a \in \mathbb{Z}_q$ and some small $\epsilon, \delta > 0$:

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

A common choice of gadget decomposition is the *digit decomposition*. In digit decomposition, we choose the gadget vector $\mathbf{g} = [q/B^d, q/B^{d-1}, \ldots, q/B]$ for some positive integers B, d such that B^d divides q. The decomposition function $h(\cdot)$ is then given by a function that outputs each digit of $\lfloor B^d/q \cdot a \rceil$ for the input a, i.e., $\lfloor B^d/q \cdot a \rceil = \sum_{0 \leq i < d} a_i \cdot B^i$ where $h(a) = (a_0, \ldots, a_{d-1})$. For smaller noise growth, the balanced representation can be utilized, ensuring that $a_i \in (-B/2, B/2]$ for $0 \leq i < d$. In this case, it is straightforward to show that $\delta = \frac{B}{2}$ and $\varepsilon = \frac{q}{2B^d}$. It is worth noting that the distributions of $h(a)$ and $\langle h(a), \mathbf{g}\rangle - a$ can be regarded as uniform distributions over $(-\frac{B}{2}, \frac{B}{2}]$ and $(-\frac{q}{2B^d}, \frac{q}{2B^d}]$, respectively, as long as a is drawn from the uniform distribution over $\mathbb{Z}_q$.

2.6 RGSW and External Product

First, we remark that a gadget decomposition $h : \mathbb{Z}_q \to \mathbb{Z}^d$ can be naturally extended to a tuple of ring elements as $h : R_q^2 \to R^{2d}$ by decomposing each

coefficient of the input polynomials. The RGSW cryptosystem [GSW13] is a cryptosystem that is based on the gadget toolkit. For a gadget decomposition $h : R_q^2 \to R^{2d}$ associated with a gadget vector $\mathbf{g} \in \mathbb{Z}^d$, the encryption algorithm for RGSW is given as follows.

• $\underline{\texttt{RGSW.Enc}(t, \mu)}$: Given a secret key $t \in R$ and a message $\mu \in R$, sample $\mathbf{a} \leftarrow R_q^{2d}$ and $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\beta}^{2d}$. Return $\mathbf{C} \leftarrow [-t \cdot \mathbf{a} + \mathbf{e}, \mathbf{a}] + \mu \cdot \begin{bmatrix} \mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \end{bmatrix} \in R_q^{2d \times 2}$.

We introduce a multiplicative operation between RLWE and RGSW ciphertexts, called the *external product*. An external product between a RLWE ciphertext ct and RGSW ciphertext $\mathbf{C}$ is defined as follows:

$$\mathsf{ct} \boxdot \mathbf{C} = h(\mathsf{ct})^\top \cdot \mathbf{C} \pmod q.$$

We remark that the external product only introduces an additive noise after multiplication. We provide a more detailed analysis below. Let $\mathbf{C}$ be an encryption of $\mu \in R$ under secret t, and $\mathbf{e}$ be the noise vector of $\mathbf{C}$. Then, the following holds:

$$\begin{aligned}
\varphi_t(\mathsf{ct} \boxdot \mathbf{C}) &= \langle h(\mathsf{ct}), \mathbf{C} \cdot \left[1\ t\right]^\top \rangle \\
&= \langle h(\mathsf{ct}), \mu \cdot \begin{bmatrix} \mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \end{bmatrix} \cdot \left[1\ t\right]^\top + \mathbf{e} \rangle \\
&= \mu \cdot \mathsf{ct} \cdot \left[1\ t\right]^\top + \mu \cdot (\langle h(\mathsf{ct}), \mathbf{g} \rangle - \mathsf{ct}) \cdot \left[1\ t\right]^\top + \langle h(\mathsf{ct}), \mathbf{e} \rangle \\
&= \mu \cdot \varphi_t(\mathsf{ct}) + e' \pmod q
\end{aligned}$$

for small e'.

2.7 TFHE scheme

We review the TFHE scheme [CGGI16] below.

• $\underline{\texttt{Setup}(1^\lambda)}$: For the security parameter λ, generate and output the parameter $\mathsf{pp} = (n, N, q, \alpha, \beta, \mathbf{g}, h, \mathbf{g}', h')$ for LWE dimension n, RLWE dimension N, plaintext and ciphertext modulus p, q, error parameters $\alpha, \beta > 0$, the gadget vector and decomposition $\mathbf{g} \in \mathbb{Z}^d$ and $h : \mathbb{Z}_q \to \mathbb{Z}^d$, and the key-switching gadget vector and decomposition $\mathbf{g}' \in \mathbb{Z}^{d'}$ and $\mathbf{h}' : \mathbb{Z}_q \to \mathbb{Z}^{d'}$.

For simplicity, we suppose that the plaintext modulus p divides the ciphertext modulus q. We also define the *scaling factor* $\Delta = \frac{q}{p} \in \mathbb{Z}$.

• $\underline{\texttt{KeyGen}(\mathsf{pp})}$:

- Sample $s_i \xleftarrow{\$} \{0, 1\}$ for $0 \le i < n$ and return the LWE key $\mathbf{s} = (s_0, \ldots, s_{n-1})$.
- Sample $t_i \xleftarrow{\$} \{0, 1\}$ for $0 \le i < N$ and return the RLWE key $t = \sum_{i=0}^{N-1} t_i X^i$.
- Set $\mathsf{BRK}_i \leftarrow \texttt{RGSW.Enc}(t; s_i)$ for $0 \le i < n$ and return the blind rotation key $\mathsf{BRK} = \{\mathsf{BRK}_i\}_{0 \le i < n}$

– Sample $\mathbf{A}_{i,1} \xleftarrow{\$} \mathbb{Z}_q^{d' \times n}$ and $\mathbf{e}_i \leftarrow \mathcal{D}_\alpha^{d'}$ for $0 \le i < n$, and set $\mathsf{KSK}_i \leftarrow (\mathbf{A}_{i,0} | \mathbf{A}_{i,1}) \in \mathbb{Z}_q^{d' \times (n+1)}$ where $\mathbf{A}_{i,0} = -\mathbf{A}_{i,1} \cdot \mathbf{s} + \mathbf{g}' \cdot t_i + \mathbf{e}_i \pmod q$.
– Sample $\mathsf{pk}_1 \xleftarrow{\$} R_q$, $e_{\mathsf{pk}} \leftarrow \mathcal{D}_\alpha^N$. Compute and output $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ where $\mathsf{pk}_0 = -\mathsf{pk}_1 \cdot t + e_{\mathsf{pk}} \pmod q$.

For simplicity, we write $\mathsf{evk} = (\mathsf{pk}, \mathsf{KSK}, \mathsf{BRK})$. Moreover, we denote $\mathbf{t}$ as the vectorized ring secret $(t_0, t_1, \ldots, t_{N-1}) \in \mathbb{Z}^N$.

• $\underline{\texttt{Enc}(\mathbf{t}, m)}$: Given the secret key $\mathbf{t}$ and a message $0 \le m < \frac{p}{2} + 1$, sample $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^N$ and $e \leftarrow \mathcal{D}_{\mathbb{Z}^N, \alpha}$. Return $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ where $b = -\langle \mathbf{a}, \mathbf{t} \rangle + \Delta \cdot m + e \pmod q$.

• $\underline{\texttt{Dec}(\mathbf{t}, \mathbf{c})}$: Given the ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$, output $\mu = \lfloor \frac{1}{\Delta} \cdot [b + \langle \mathbf{a}, \mathbf{t} \rangle]_q \rceil$.

The TFHE bootstrapping consists of three steps: 1. Key-switching, 2. Blind Rotation, and 3. Sample Extraction. In the key-switching step, the ciphertext size is switched from the RLWE dimension N to LWE dimension n in order to enhance the bootstrapping performance, using the key-switching key KSK. It is followed by the blind rotation step, which homomorphically decrypts the input ciphertext over the exponent of a monomial. Finally, the constant term of the phase of the output RLWE ciphertext from blind rotation is extracted during the sample extraction step. We describe the algorithms for each step below.

• $\underline{\texttt{KeySwitch}(\mathsf{KSK}, \mathbf{c})}$: Given an LWE ciphertext $\mathbf{c} = (b, a_0, \ldots, a_{N-1}) \in \mathbb{Z}_q^{N+1}$, return $(b, \mathbf{0}) + \sum_{i=0}^{N-1} h'(a_i)^\top \cdot \mathsf{KSK}_i \in \mathbb{Z}_q^{n+1}$.

• $\underline{\texttt{BlindRotate}(\mathsf{BRK}, \mathbf{c})}$: Given the blind rotation key BRK and LWE ciphertext $\mathbf{c} \in \mathbb{Z}_q^{N+1}$, run Algorithm 1.

Algorithm 1 Blind Rotation

Input: Blind Rotation Key BRK, LWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$
1: Let $\tilde{b} = \lfloor \frac{2N}{q} \cdot b \rceil$, $\tilde{a_i} = \lfloor \frac{2N}{q} \cdot a_i \rceil$ for $0 \le i < n$
2: $\mathsf{tv} := \Delta \cdot \sum_{i=-N/p+1}^{(p-1)N/p} \lfloor \frac{p}{2N} i \rceil \cdot X^{2N-i}$
3: $\mathsf{ACC} \leftarrow (X^{\tilde{b}} \cdot \mathsf{tv}, 0) \in R_q^2$
4: **for** $0 \le i < n$ **do**
5: $\quad \mathsf{ACC} \leftarrow \mathsf{ACC} + (X^{\tilde{a}_i} - 1) \cdot (\mathsf{ACC} \boxdot \mathsf{BRK}_i)$
6: **end for**
7: Return ACC

• $\underline{\texttt{SampleExtract}(\mathsf{ct})}$: Given an RLWE ciphertext $\mathsf{ct} = (b = \sum_{i=0}^{N-1} b_i X^i, a = \sum_{i=0}^{N} a_i X^i) \in R_q^2$, return $(b_0, a_0, -a_{N-1}, \ldots, -a_1) \in \mathbb{Z}_q^{N+1}$.

The entire bootstrapping procedure is provided in Algorithm 2.

One intrinsic feature of the TFHE scheme is that it can evaluate a univariate function while bootstrapping. To be precise, given a function $f : \mathbb{Z}_p \cap [0, p/2) \to$

Algorithm 2 TFHE Bootstrapping

Input: Blind Rotation Key BRK, Key Switching Key KSK, LWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$

1: $\mathbf{c}_{\text{ks}} \leftarrow \texttt{KeySwitch}(\mathsf{KSK}, \mathbf{c})$
2: $\mathsf{ct}_{\text{br}} \leftarrow \texttt{BlindRotate}(\mathsf{BRK}, \mathbf{c}')$
3: $\mathbf{c}_{\text{out}} \leftarrow \texttt{SampleExtract}(\mathsf{ct})$
4: Return $\mathbf{c}_{\text{out}}$

$\mathbb{Z}_p$, the test vector tv is replaced by the following polynomial:

$$\Delta \cdot \sum_{i=-N/p+1}^{(p-1)N/p} f\left(\left\lfloor \frac{p}{2N} i \right\rceil\right) \cdot X^{2N-i}.$$

In the following sections, we define the error function $\texttt{Err}_{\mathbf{s}}(\mathbf{c}) : \mathbb{Z}_q^{n+1} \to \mathbb{Z}$, defined by $\texttt{Err}_{\mathbf{s}}(\mathbf{c}) := [\varphi_{\mathbf{s}}(\mathbf{c})]_{\Delta}$. Naturally, this function outputs the noise term of the input ciphertext. Similarly, we generalize this definition to RLWE ciphertext as well, by $\texttt{Err}_t(\mathsf{ct}) := [\varphi_t(\mathsf{ct})]_{\Delta}$.

2.8 Useful Lemmas

Definition 6 ([MR07, Smoothing Parameter]). *For a n-dimensional lattice Λ and $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is defined as the smallest σ such that $\rho_{1/(\sigma\sqrt{2\pi})}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.*

Lemma 1 ([MR07, Lemma 3.3]). *For a n-dimensional lattice Λ and $\varepsilon > 0$,*

$$\eta_\varepsilon(\Lambda) \leq \lambda_n(\Lambda) \cdot \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}}$$

where $\lambda_n(\Lambda)$ is the n-th successive minimum of Λ.

Lemma 2 ([Pei10, Theorem 3.1]). *Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ be positive definite matrices, and let $\mathbf{\Sigma}^{-1} := \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}$. If $\eta_\varepsilon(\mathbb{Z}^n)/\sqrt{2\pi} \leq \sqrt{\mathbf{\Sigma}}$ for $0 < \varepsilon < 1$, then for any $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$,*

$$\Delta(\mathcal{D}_{\mathbb{Z}^n, \mathbf{c}_1, \sqrt{\mathbf{\Sigma}_1}} + \mathcal{D}_{\mathbb{Z}^n, \mathbf{c}_2, \sqrt{\mathbf{\Sigma}_2}}, \mathcal{D}_{\mathbb{Z}^n, \mathbf{c}_1+\mathbf{c}_2, \sqrt{\mathbf{\Sigma}_1+\mathbf{\Sigma}_2}}) \leq 2\varepsilon.$$

Lemma 3 ([dCKK+24, Lemma 6]). *Let $\mathbf{E} \in \mathbb{Z}^{m \times n}$ be a matrix, $t > 0$ be an integer, and $\sigma, \tau > 0$ be reals such that*

$$\frac{1}{\sigma^2} + \frac{\|\mathbf{E}\|_2^2}{\tau^2} \leq \frac{2\pi}{\eta_\varepsilon(t\mathbb{Z}^n)^2}$$

for $0 < \varepsilon \leq 1/2$. Then, for any $\mathbf{a} \in \mathbb{Z}^n$,

$$\Delta\left(\mathbf{E}\mathcal{D}_{\mathbf{a}+t\mathbb{Z}^n, \sigma} + \mathcal{D}_{\mathbb{Z}^m, \tau}, \mathcal{D}_{\mathbb{Z}^m, \sqrt{\sigma^2 \mathbf{E}\mathbf{E}^\top + \tau^2 \mathbf{I}}}\right) \leq 4\varepsilon.$$

3 Randomization Techniques for Sanitization

In this section, we describe two randomization techniques that serve as building blocks for our sanitization algorithm. The first technique is *mask rerandomization*, which transforms the distribution of the mask in a given ciphertext to a uniform distribution without altering the message. The second technique is *randomized linear evaluation*, which enables the homomorphic evaluation of an affine function $\mathfrak{a}x + \mathfrak{b}$ without revealing $\mathfrak{a}$ and $\mathfrak{b}$ in the output ciphertext.

3.1 Mask Rerandomization

In this subsection, we explain the (modified) mask rerandomization algorithm based on the construction from [BI22], and present the correctness and security proofs. Our mask rerandomization algorithm leverages the fact that the mask of a public key encryption of zero is computationally indistinguishable from a uniformly random variable over R_q under careful parameter choices. By adding this specially crafted encryption of zero to a given ciphertext, we can achieve mask rerandomization.

Below, we provide the algorithm that generates a public key encryption of zero, with Gaussian parameters σ_r and τ_r.

• $\underline{\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk})}$: For a public key $\mathsf{pk} \in R_q^2$, sample $e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r}$ and $e_0 \leftarrow \overline{\lfloor \mathcal{D}_{\mathbb{R}^N,\tau_r} \rceil}$. Output $e_2 \cdot \mathsf{pk} + (e_0, e_1) \pmod q$.

It is straightforward to show that this algorithm outputs an encryption of zero under the secret t as long as the noise parameters σ_r and τ_r are sufficiently small, since

$$\begin{aligned}\varphi_t(\mathsf{ct}_r) &= e_2 \cdot \mathsf{pk}_0 + e_0 + t \cdot (e_2 \cdot \mathsf{pk}_1 + e_1) \\ &= e_2 \cdot e_{\mathsf{pk}} + e_0 + e_1 \cdot t \pmod q\end{aligned}$$

for $\mathsf{ct}_r \leftarrow \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk})$ and $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$. Moreover, observe that the mask of ct_r is $e_2 \cdot \mathsf{pk}_1 + e_1$. When combined with the error of ct_r, the mask can be viewed as an RLWE sample with the key e_2 and error e_1, given with a linear hint $e_{\mathsf{pk}} \cdot e_2 + t \cdot e_1 + e_0$. Therefore, the distribution of the mask can be shown to be computationally indistinguishable from uniform by a reduction to the Hint-RLWE problem [KLSS23,MKMS22]. However, since the Hint-RLWE problem was only proven secure in the discrete Gaussian setting in prior works, we provide a new proof for the rounded Gaussian setting. We provide a detailed analysis of the distribution of ct_r in the following lemma.

Lemma 4 (Distribution of `EncZero`). *Let $\sigma_r, \tau_r > 0$ be reals. Let $B > 0$ be an upper bound of $\|\left[\mathbf{E}_{\mathsf{pk}}\ \mathbf{T}\right]\|_2$ for any t, pk sampled from `KeyGen`, where $\mathbf{E}_{\mathsf{pk}}$ and $\mathbf{T}$ are the negacyclic matrices of $e_{\mathsf{pk}} := \mathtt{Err}_t(\mathsf{pk})$ and t respectively. If for $0 < \varepsilon \le 1/2$,*

$$\frac{1}{\kappa^2} := \frac{1}{\sigma_r^2} + \frac{B^2}{\tau_r^2} \le \frac{\pi}{2\eta_\varepsilon(\mathbb{Z}^{2N})^2}$$

and $\mathsf{RLWE}^{1,N,q}_{\kappa/\sqrt{2},\kappa/\sqrt{2}}$ *is hard, then*

$$\begin{aligned}&\{(t, \mathsf{evk}, \mathsf{ct}_r) \mid \mathsf{ct}_r \leftarrow \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk})\}\\ &\quad\approx_c \{(t, \mathsf{evk}, (-ut + e_r, u)) \mid u \xleftarrow{\$} R_q, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}\}\end{aligned}$$

where $(t, \mathsf{evk}) \leftarrow \mathtt{KeyGen}$ *and*

$$\mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}} := \{(te_1 + e_{\mathsf{pk}}e_2 + e_0) \mid e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r}, e_0 \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_r} \rceil\}.$$

Moreover, the noise of $\mathtt{EncZero}$ *is bounded by*

$$\mathtt{Var}(\mathtt{Err}_t(\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}))) \le V^{\sigma_r,\tau_r}_{\mathtt{EncZero}} := N\left(\beta^2 + \frac{1}{4}\right)\sigma_r^2 + \tau_r^2 + \frac{1}{12}.$$

Proof. For $\mathbf{\Gamma} = \begin{bmatrix}\mathbf{E}_{\mathsf{pk}} & \mathbf{T}\end{bmatrix}$, consider a sample

$$\{(\mathbf{A}, \begin{bmatrix}\mathbf{I} & \mathbf{A}\end{bmatrix}\mathbf{r}, \mathbf{\Gamma}, \mathbf{\Gamma r} + \mathbf{y}) \mid a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r}\}$$

where $\mathbf{A}$ is a negacyclic matrix of a. We claim that this sample is computationally indistinguishable from

$$\{(\mathbf{A}, \mathbf{u}, \mathbf{\Gamma}, \mathbf{\Gamma r} + \mathbf{y}) \mid a, u \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r}\}$$

given the conditions in Lemma 4, where $\mathbf{u}$ is a coefficient vector of u.

Since the conditional distribution of $\mathbf{r}$ given $\mathbf{\Gamma r} + \mathbf{y}$ satisfies

$$\begin{aligned}\Pr_{\substack{\mathbf{r}\leftarrow\mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}\\ \mathbf{y}\leftarrow\mathcal{D}_{\mathbb{R}^N,\tau_r}}}[\mathbf{r} = \mathbf{v} \mid \mathbf{\Gamma r} + \mathbf{y} = \mathbf{w}] &= \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}(\mathbf{v}) \cdot \mathcal{D}_{\mathbb{R}^N,\tau_r}(\mathbf{w} - \mathbf{\Gamma v})\\ &\propto \exp\left[-\pi\left(\frac{1}{\sigma_r^2}\mathbf{v}^\top\mathbf{v} + \frac{1}{\tau_r^2}(\mathbf{w} - \mathbf{\Gamma v})^\top(\mathbf{w} - \mathbf{\Gamma v})\right)\right]\\ &\propto \exp\left[-\pi(\mathbf{v} - \tau_r^{-2}\mathbf{\Sigma E}^\top\mathbf{z})^\top\mathbf{\Sigma}^{-1}(\mathbf{v} - \tau_r^{-2}\mathbf{\Sigma\Gamma}^\top\mathbf{w})\right]\\ &\propto \rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{v} - \mathbf{c})\end{aligned}$$

where $\mathbf{\Sigma} = \left(\sigma_r^{-2}\cdot\mathbf{I} + \tau_r^{-2}\mathbf{\Gamma}^\top\mathbf{\Gamma}\right)^{-1}$ and $\mathbf{c} = \tau_r^{-2}\mathbf{\Sigma\Gamma}^\top\mathbf{w}$, we have

$$\begin{aligned}&\{(\mathbf{A}, \begin{bmatrix}\mathbf{I} & \mathbf{A}\end{bmatrix}\mathbf{r}, \mathbf{\Gamma}, \mathbf{\Gamma r} + \mathbf{y}) \mid a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r}\}\\ &\quad\equiv \left\{(\mathbf{A}, \begin{bmatrix}\mathbf{I} & \mathbf{A}\end{bmatrix}\mathbf{r}', \mathbf{\Gamma}, \mathbf{w}) \;\middle|\; \begin{array}{c} a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r}\\ \mathbf{w} = \mathbf{\Gamma r} + \mathbf{y}, \mathbf{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}}}\end{array}\right\}.\end{aligned}$$

Then, using Lemma 2, we decompose the distribution of $\mathbf{r}'$ as

$$\mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}}} \approx_s \mathcal{D}_{\mathbb{Z}^{2N},\kappa/\sqrt{2}} + \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}-\kappa^2/2\mathbf{I}}}.$$

This decomposition is valid under the condition that $\mathbf{\Sigma} - \kappa^2/2\cdot\mathbf{I}$ is positive definite and

$$\eta_\varepsilon(\mathbb{Z}^{2N})/\sqrt{2\pi} \le \sqrt{(\kappa^2/2\cdot\mathbf{I})^{-1} + (\mathbf{\Sigma} - \kappa^2/2\cdot\mathbf{I})^{-1}}^{-1}.$$

We can check these conditions by noticing

$$\sigma_{\min}(\mathbf{\Sigma}) = \left\| \frac{1}{\sigma_r^2}\mathbf{I} + \frac{\mathbf{\Gamma}^\top \mathbf{\Gamma}}{\tau_r^2} \right\|_2^{-1} \geq \left(\frac{1}{\sigma_r^2} + \frac{B_2^2}{\tau_r^2} \right)^{-1} = \kappa^2.$$

Therefore, $\mathbf{\Sigma} - \kappa^2/2 \cdot \mathbf{I}$ is positive-definite, and

$$\sqrt{(\kappa^2/2 \cdot \mathbf{I})^{-1} + (\mathbf{\Sigma} - \kappa^2/2 \cdot \mathbf{I})^{-1}}^{-1} \geq \frac{\kappa}{2} \geq \frac{\eta_\varepsilon(\mathbb{Z}^{2N})}{\sqrt{2\pi}}.$$

Finally, we conclude that

$$\begin{aligned}
&\left\{ (\mathbf{A}, \begin{bmatrix}\mathbf{I}\ \mathbf{A}\end{bmatrix} \mathbf{r}', \mathbf{\Gamma}, \mathbf{w}) \;\middle|\; \begin{array}{r} a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \\ \mathbf{w} = \mathbf{\Gamma}\mathbf{r} + \mathbf{y}, \mathbf{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}}} \end{array} \right\} \\
&\quad \approx_s \left\{ (\mathbf{A}, \begin{bmatrix}\mathbf{I}\ \mathbf{A}\end{bmatrix} (\mathbf{r}'_1 + \mathbf{r}'_2), \mathbf{\Gamma}, \mathbf{w}) \;\middle|\; \begin{array}{r} a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \\ \mathbf{w} = \mathbf{\Gamma}\mathbf{r} + \mathbf{y}, \mathbf{r}'_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\kappa/\sqrt{2}}, \mathbf{r}'_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}-\kappa^2/2\mathbf{I}}} \end{array} \right\} \\
&\quad \approx_c \left\{ (\mathbf{A}, \begin{bmatrix}\mathbf{I}\ \mathbf{A}\end{bmatrix} \mathbf{r}'_2 + \mathbf{u}, \mathbf{\Gamma}, \mathbf{w}) \;\middle|\; \begin{array}{r} a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \\ \mathbf{w} = \mathbf{\Gamma}\mathbf{r} + \mathbf{y}, u \leftarrow R_q, \mathbf{r}'_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\mathbf{c},\sqrt{\mathbf{\Sigma}-\kappa^2/2\mathbf{I}}} \end{array} \right\} \\
&\quad \equiv \left\{ (\mathbf{A}, \mathbf{u}, \mathbf{\Gamma}, \mathbf{w}) \;\middle|\; \begin{array}{r} a \xleftarrow{\$} R_q, \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^{2N},\sigma_r}, \mathbf{y} \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \\ \mathbf{w} = \mathbf{\Gamma}\mathbf{r} + \mathbf{y}, u \xleftarrow{\$} R_q \end{array} \right\}
\end{aligned}$$

as desired, where the last computational indistinguishability comes from the hardness of $\mathsf{RLWE}^{1,N,q}_{\kappa/\sqrt{2},\kappa/\sqrt{2}}$.

Finally, since

$$\begin{aligned}
\mathsf{ct}_r = e_2 \cdot \mathsf{pk} + (e_0, e_1) &= (-e_2\mathsf{pk}_1 t + e_2 e_{\mathsf{pk}} + e_0, e_2\mathsf{pk}_1 + e_1) \\
&= (-(\mathsf{pk}_1 e_2 + e_1)t + (te_1 + e_{\mathsf{pk}} e_2 + e_0), \mathsf{pk}_1 e_2 + e_1),
\end{aligned}$$

the sample $(t, \mathsf{evk}, \mathsf{ct}_r)$ is equivalent to $(\mathsf{evk}, \mathsf{pk}_1, \mathsf{pk}_1 e_2 + e_1, t, e_{\mathsf{pk}}, te_1 + e_{\mathsf{pk}} e_2 + e_0)$. Let $\mathbf{P}_1$ be negacyclic matrix of pk_1, and $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$ be coefficient vectors of e_0, e_1, e_2. Then the distribution of the sample satisfies

$$\begin{aligned}
&\left\{ \left(\mathsf{evk}, \mathbf{P}_1, \begin{bmatrix}\mathbf{I}\ \mathbf{P}_1\end{bmatrix} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix}, \mathbf{\Gamma}, \mathbf{\Gamma} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix} + \mathbf{e}_0 \right) \;\middle|\; \begin{array}{r} \mathsf{pk}_1 \xleftarrow{\$} R_q, e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r} \\ e_0 \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_r} \rceil \end{array} \right\} \\
&\quad \equiv \left\{ \left(\mathsf{evk}, \mathbf{P}_1, \begin{bmatrix}\mathbf{I}\ \mathbf{P}_1\end{bmatrix} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix}, \mathbf{\Gamma}, \left\lfloor \mathbf{\Gamma} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix} + \mathbf{e}_0 \right\rceil \right) \;\middle|\; \begin{array}{r} \mathsf{pk}_1 \xleftarrow{\$} R_q, e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r} \\ e_0 \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \end{array} \right\} \\
&\quad \approx_s \left\{ \left(\mathsf{evk}, \mathbf{P}_1, \mathbf{u}, \mathbf{\Gamma}, \left\lfloor \mathbf{\Gamma} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix} + \mathbf{e}_0 \right\rceil \right) \;\middle|\; \begin{array}{r} u, \mathsf{pk}_1 \xleftarrow{\$} R_q, e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r} \\ e_0 \leftarrow \mathcal{D}_{\mathbb{R}^N,\tau_r} \end{array} \right\} \\
&\quad \equiv \left\{ \left(\mathsf{evk}, \mathbf{P}_1, \mathbf{u}, \mathbf{\Gamma}, \mathbf{\Gamma} \begin{bmatrix}\mathbf{e}_2 \\ \mathbf{e}_1\end{bmatrix} + \mathbf{e}_0 \right) \;\middle|\; \begin{array}{r} u, \mathsf{pk}_1 \xleftarrow{\$} R_q, e_1, e_2 \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma_r} \\ e_0 \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_r} \rceil \end{array} \right\}.
\end{aligned}$$

We conclude that

$$\begin{aligned}
&\{(t, \mathsf{evk}, \mathsf{ct}_r) \mid \mathsf{ct}_r \leftarrow \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk})\} \\
&\quad \approx_c \{(t, \mathsf{evk}, (-ut + e_r, u)) \mid u \xleftarrow{\$} R_q, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}\}.
\end{aligned}$$

Since $\mathtt{Err}_t(\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}))$ follows $\mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}$, the bound on the variance can be computed as

$$\begin{aligned}\mathtt{Var}(\mathtt{Err}_t(\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}))) &\le N\mathtt{Var}(t)\mathtt{Var}(e_1) + N\mathtt{Var}(e_{\mathsf{pk}})\mathtt{Var}(e_2) + \mathtt{Var}(e_0)\\ &\le N\frac{1}{4}\sigma_r^2 + N\beta^2\sigma_r^2 + \tau_r^2 + \frac{1}{12}\\ &= N\left(\beta^2+\frac{1}{4}\right)\sigma_r^2 + \tau_r^2 + \frac{1}{12}.\end{aligned}$$

□

Now we describe the mask rerandomization algorithm for LWE ciphertexts. In our mask rerandomization method, we utilize the $\mathtt{EncZero}$ procedure described above. Since an RLWE ciphertext can be decomposed into N LWE ciphertexts, each corresponding to a coefficient of the plaintext in the RLWE encryption, we can obtain an LWE encryption of zero by extracting the constant term of the RLWE public key encryption of zero using the sample extraction algorithm. Since the mask remains the same up to the sign and order after the sample extraction, the mask of the extracted LWE ciphertext remains uniform. Consequently, the mask of an LWE ciphertext can be randomized by adding the extracted LWE public key encryption of zero. The exact algorithm and the security proof are provided below.

• $\underline{\mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk},\mathbf{c})}$: For a public key $\mathsf{pk} \in R_q^2$ and ciphertext $\mathbf{c} \in \mathbb{Z}_q^{N+1}$, output $\mathbf{c} + \mathtt{SampleExtract}(\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}))\pmod q$.

Lemma 5 (Mask Rerandomization). *Let $\mathbf{c} = (b = -\langle \mathbf{a}, \mathbf{t}\rangle + \Delta m + e, \mathbf{a}) \in \mathbb{Z}_q^{N+1}$ be an LWE ciphertext. If $\sigma_r, \tau_r > 0$ satisfy the conditions of Lemma 4, then*

$$\begin{aligned}&\{(\mathbf{t}, \mathsf{evk}, \mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk},\mathbf{c}))\}\\ &\quad\approx_s \{(\mathbf{t}, \mathsf{evk}, (-\langle \mathbf{u}, \mathbf{t}\rangle + \Delta m + e + e_r) \mid \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}|_0\}\end{aligned}$$

where $(t, \mathsf{evk}) \leftarrow \mathtt{KeyGen}$. Moreover,

$$\mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk},\mathbf{c}))) \le \mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c})) + V^{\sigma_r,\tau_r}_{\mathtt{EncZero}}.$$

Proof. Let $(\hat{b}, \hat{a}) = \left(\sum_{i=0}^{N-1} \hat{b}_i X^i, \sum_{i=0}^{N-1} \hat{a}_i X^i\right) \leftarrow \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk})$. By Lemma 4, the distribution of $(\hat{b}, \hat{a})$ is computationally indistinguishable from $\{(-\hat{a}t + e_r, \hat{a}) \mid \hat{a} \xleftarrow{\$} R_q, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}\}$. Therefore, the distribution of $\mathtt{SampleExtract}((\hat{b}, \hat{a})) = (\hat{b}_0, \hat{a}_0, -\hat{a}_{N-1}, \ldots, -\hat{a}_1)$ is computationally indistinguishable from $\{(-\langle \mathbf{u}, \mathbf{t}\rangle + e_r, \mathbf{u}) \mid \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}|_0\}$. This implies that

$$\begin{aligned}&\{(\mathbf{t}, \mathsf{evk}, \mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk},\mathbf{c}))\}\\ &\quad\equiv \{(\mathbf{t}, \mathsf{evk}, (b + \hat{b}, \mathbf{a} + \hat{\mathbf{a}})) \mid (\hat{b}, \hat{\mathbf{a}}) \leftarrow \mathtt{SampleExtract}(\mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}))\}\end{aligned}$$

$$\approx_s \{(\mathbf{t}, \mathsf{evk}, (-\langle \mathbf{a}+\hat{\mathbf{a}}, \mathbf{t}\rangle + \Delta m + e + e_r, \mathbf{a}+\hat{\mathbf{a}})) \mid \hat{\mathbf{a}} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}|_0\}$$
$$\equiv \{(\mathbf{t}, \mathsf{evk}, (-\langle \mathbf{u}, \mathbf{t}\rangle + \Delta m + e + e_r) \mid \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}|_0\}$$

as desired. Moreover, the bound of the variance of $\mathtt{Err}_{\mathbf{t}}(\mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk}, \mathbf{c})) = e + e_r$ immediately follows from Lemma 4. □

3.2 Randomized Linear Evaluation

In this subsection, we explain the randomized linear evaluation technique for RLWE ciphertexts based on circuit-private linear homomorphic encryption from de Castro et al. [dCKK+24]. Recall that the RLWE encryption of m is given by $\mathsf{ct} = (b = -as + \Delta m + e, a) \in R_q^2$, where $\Delta = q/p$ for p dividing q. Then, an affine function $f(x) = \mathfrak{a}m + \mathfrak{b}$ for $\mathfrak{a}, \mathfrak{b} \in R_p$ can be evaluated as follows:

$$\mathsf{ct}' := \mathfrak{a} \cdot \mathsf{ct} + (\Delta\mathfrak{b}, 0) = (\mathfrak{a} \cdot b + \Delta\mathfrak{b}, \mathfrak{a} \cdot a) = (-(\mathfrak{a} \cdot a)s + \Delta(\mathfrak{a}m + \mathfrak{b}) + \mathfrak{a}e, \mathfrak{a} \cdot a).$$

However, this resulting ciphertext clearly leaks information about $\mathfrak{a}$ in the mask and the error of the output ciphertext. To address this issue, we use a Gaussian random variable $\mathfrak{r}$ sampled from the coset $\mathfrak{a} + p\mathbb{Z}^N$. By adding a moderately sized Gaussian error, we can erase the information of $\mathfrak{a}$ from the error of the output ciphertext. Finally, we rerandomize the mask by adding a ciphertext from $\mathtt{EncZero}$, similar to the previous section. We provide a full description of our randomized linear evaluation algorithm below.

• $\underline{\mathtt{RandLinEval}_{\sigma_\ell,\tau_\ell,\sigma_r,\tau_r}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b})}$: Given a ciphertext ct and ring elements $\mathfrak{a}, \mathfrak{b} \in R_p$, sample $\mathfrak{r} \leftarrow \mathcal{D}_{\mathfrak{a}+p\mathbb{Z}^N,\sigma_\ell}$ and $\hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil$. Compute and output $\mathfrak{r} \cdot \mathsf{ct} + (\Delta\mathfrak{b} + \hat{e}, 0) + \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}) \pmod q$.

It is worth noting that the additional Gaussian noise is chosen as a rounded Gaussian distribution in our construction, compared to the discrete Gaussian distribution in [dCKK+24]. Now, we prove that the distribution of the output ciphertext of this randomized linear evaluation is independent of $\mathfrak{a}$ and $\mathfrak{b}$, except for the message. Before that, we prove the following lemma, which is an adaptation of the Gaussian lemma Lemma 3 to the rounded Gaussian distribution.

Lemma 6 (Rounded Gaussian Lemma). *Let $\mathbf{E} \in \mathbb{Z}^{m\times n}$ be a matrix, $t > 0$ be an integer, and $\sigma, \tau > 0$ be reals such that*

$$\frac{1}{\sigma^2} + \frac{\|\mathbf{E}\|_2^2}{\tau^2} \le \frac{2\pi}{\eta_\varepsilon(t\mathbb{Z}^n)^2}$$

for $0 < \varepsilon < 1/2$. Then, for any $\mathbf{a} \in \mathbb{Z}^n$,

$$\Delta\left(\mathbf{E}\mathcal{D}_{\mathbf{a}+t\mathbb{Z}^n,\sigma} + \lfloor \mathcal{D}_{\mathbb{R}^m,\tau} \rceil, \left\lfloor \mathcal{D}_{\mathbb{R}^m,\sqrt{\sigma^2\mathbf{E}\mathbf{E}^\top + \tau^2\mathbf{I}}} \right\rceil\right) \le 4\varepsilon.$$

Proof. Let $p > 0$ be an integer. By Lemma 3, we have

$$\mathbf{E}\mathcal{D}_{\mathbf{a}+t\mathbb{Z}^n,\sigma} + \mathcal{D}_{\frac{1}{p}\mathbb{Z}^m,\tau} \equiv \frac{1}{p}\left(\mathbf{E}\mathcal{D}_{p\mathbf{a}+pt\mathbb{Z}^n,p\sigma} + \mathcal{D}_{\mathbb{Z}^m,p\tau}\right)$$

$$\overset{4\varepsilon}{\approx} \frac{1}{p}\mathcal{D}_{\mathbb{Z}^m, p\sqrt{\sigma^2\mathbf{E}\mathbf{E}^\top+\tau^2\mathbf{I}}}$$
$$\equiv \mathcal{D}_{\frac{1}{p}\mathbb{Z}^m, \sqrt{\sigma^2\mathbf{E}\mathbf{E}^\top+\tau^2\mathbf{I}}}.$$

The result follows by taking $p \to \infty$ and rounding both sides. □

Lemma 7 (Randomized Linear Evaluation). *Let $\sigma_r, \tau_r, \sigma_\ell, \tau_\ell > 0$ be reals, and $\mathsf{ct} = (-at + \Delta m + e, a) \in R_q^2$ be an RLWE ciphertext. Let $B > 0$ be an upper bound of $\|\mathbf{E}\|_2$ where $\mathbf{E}$ is the negacyclic matrix of e. If σ_r, τ_r satisfy the condition of Lemma 4 and $\sigma_\ell, \tau_\ell > 0$ satisfy*

$$\frac{1}{\sigma_\ell^2} + \frac{B^2}{\tau_\ell^2} \le \frac{2\pi}{\eta_\varepsilon(p\mathbb{Z}^n)^2}$$

for some $0 < \varepsilon \le 1/2$, then for any $a, b \in R_q$,

$$\{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b}))\}$$
$$\approx_c \{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}, 0, 0) + (\Delta(\mathfrak{a}m + \mathfrak{b}), 0))\}.$$

Moreover,

$$\mathtt{Var}(\mathtt{Err}_t(\mathtt{RandLinEval}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b})))$$
$$\le N\sigma_\ell^2 \mathtt{Var}(\mathtt{Err}_t(\mathsf{ct})) + \tau_\ell^2 + \frac{1}{12} + V^{\sigma_r,\tau_r}_{\mathtt{EncZero}}.$$

Proof. Let $\mathsf{ct}' = \mathfrak{r} \cdot \mathsf{ct} + (\Delta\mathfrak{b} + \hat{e}, 0)$ where $\mathfrak{r} \leftarrow \mathcal{D}_{\mathfrak{a}+p\mathbb{Z}^N,\sigma_\ell}$ and $\hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil$. Then,

$$\begin{aligned}\mathsf{ct}' &= \mathfrak{r} \cdot (b, a) + (\Delta\mathfrak{b} + \hat{e}, 0)\\ &= (-\mathfrak{r}at + \Delta\mathfrak{r}m + \mathfrak{r}e + \Delta\mathfrak{b} + \hat{e}, \mathfrak{r}a)\\ &= (-(\mathfrak{r}a)t + \Delta(\mathfrak{a}m + \mathfrak{b}) + \mathfrak{r}e + \hat{e}, \mathfrak{r}a) \pmod q.\end{aligned}$$

By Lemma 6, it follows that

$$\mathtt{Err}_t(\mathsf{ct}') = \mathfrak{r}e + \hat{e} \sim \mathbf{E}\mathcal{D}_{\mathfrak{a}+p\mathbb{Z}^N,\sigma_\ell} + \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil \overset{4\varepsilon}{\approx} \left\lfloor \mathcal{D}_{\mathbb{R}^N, \sqrt{\sigma_\ell^2\mathbf{E}\mathbf{E}^\top+\tau_\ell^2\mathbf{I}}} \right\rceil$$

where $\mathbf{E}$ is a negacyclic matrix of e. Therefore, the distribution of $\mathtt{RandLinEval}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b})$ can be written as

$$\{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b}))\}$$
$$\equiv \left\{ (t, \mathsf{evk}, \mathsf{ct}' + \mathsf{ct}_r) \;\middle|\; \begin{array}{r} \mathfrak{r} \leftarrow \mathcal{D}_{\mathfrak{a}+p\mathbb{Z}^N,\sigma_\ell}, \hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil \\ \mathsf{ct}' \leftarrow \mathfrak{r} \cdot \mathsf{ct} + (\Delta\mathfrak{b} + \hat{e}, 0) \\ \mathsf{ct}_r \leftarrow \mathtt{EncZero}_{\sigma_r,\tau_r}(\mathsf{pk}) \end{array} \right\}$$
$$\approx_c \left\{ (t, \mathsf{evk}, (-ut + \Delta(\mathfrak{a}m + \mathfrak{b}) + e_\ell + e_r, u) \;\middle|\; \begin{array}{r} \mathfrak{r} \leftarrow \mathcal{D}_{\mathfrak{a}+p\mathbb{Z}^N,\sigma_\ell}, \hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil \\ e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}, e_\ell \leftarrow \mathfrak{r}e + \hat{e}, u \xleftarrow{\$} R_q \end{array} \right\}$$
$$\approx_s \left\{ (t, \mathsf{evk}, (-ut + \Delta(\mathfrak{a}m + \mathfrak{b}) + e_\ell + e_r, u) \;\middle|\; \begin{array}{r} e_\ell \leftarrow \left\lfloor \mathcal{D}_{\mathbb{R}^N, \sqrt{\sigma_\ell^2\mathbf{E}\mathbf{E}^\top+\tau_\ell^2\mathbf{I}}} \right\rceil \\ e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}, u \xleftarrow{\$} R_q \end{array} \right\}$$

where the first computational indistinguishability comes from lemma 4.

Now, let $\mathsf{ct}'' = \mathfrak{r}' \cdot \mathsf{ct} + (\Delta(\mathfrak{a}m + \mathfrak{b}) + \hat{e}, 0)$ where $\mathfrak{r}' \leftarrow \mathcal{D}_{\mathbb{Z}^N, \sigma_\ell}$ and $\hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N, \tau_\ell} \rceil$. Then, analogous to the logic above, we have

$$\mathsf{ct}'' = (-(\mathfrak{r}'a)t + \Delta(\mathfrak{a}m + \mathfrak{b}) + \mathfrak{r}'e + \hat{e}, \mathfrak{r}'a) \pmod q$$

and

$$\mathtt{Err}_t(\mathsf{ct}'') = \mathfrak{r}'e + \hat{e} \sim \mathbf{E}\mathcal{D}_{p\mathbb{Z}^N, \sigma_\ell} + \lfloor \mathcal{D}_{\mathbb{R}^N, \tau_\ell} \rceil \overset{4\varepsilon}{\approx} \left\lfloor \mathcal{D}_{\mathbb{R}^N, \sqrt{\sigma_\ell^2 \mathbf{E}\mathbf{E}^\top + \tau_\ell^2 \mathbf{I}}} \right\rceil.$$

Hence, the distribution of $\mathtt{RandLinEval}(\mathsf{pk}, \mathsf{ct}, 0, 0) + (\Delta(\mathfrak{a}m + \mathfrak{b}), 0)$ can be written as

$$\begin{aligned}
&\{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r, \tau_r, \sigma_\ell, \tau_\ell}(\mathsf{pk}, \mathsf{ct}, 0, 0) + (\Delta(\mathfrak{a}m + \mathfrak{b}), 0))\} \\
&\quad \equiv \left\{ (t, \mathsf{evk}, \mathsf{ct}'' + \mathsf{ct}_r) \;\middle|\; \begin{array}{r} \mathfrak{r}' \leftarrow \mathcal{D}_{p\mathbb{Z}^N, \sigma_\ell}, \hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N, \tau_\ell} \rceil \\ \mathsf{ct}'' \leftarrow \mathfrak{r} \cdot \mathsf{ct} + (\Delta(\mathfrak{a}m + \mathfrak{b}) + \hat{e}, 0) \\ \mathsf{ct}_r \leftarrow \mathtt{EncZero}_{\sigma_r, \tau_r}(\mathsf{pk}) \end{array} \right\} \\
&\quad \approx_c \left\{ (t, \mathsf{evk}, (-ut + \Delta(\mathfrak{a}m + \mathfrak{b}) + e_\ell + e_r, u) \;\middle|\; \begin{array}{r} \mathfrak{r}' \leftarrow \mathcal{D}_{p\mathbb{Z}^N, \sigma_\ell}, \hat{e} \leftarrow \lfloor \mathcal{D}_{\mathbb{R}^N, \tau_\ell} \rceil \\ e_r \leftarrow \mathcal{D}^{\sigma_r, \tau_r}_{\mathtt{EncZero}}, e_\ell \leftarrow \mathfrak{r}e + \hat{e}, u \xleftarrow{\$} R_q \end{array} \right\} \\
&\quad \approx_s \left\{ (t, \mathsf{evk}, (-ut + \Delta(\mathfrak{a}m + \mathfrak{b}) + e_\ell + e_r, u) \;\middle|\; \begin{array}{r} e_\ell \leftarrow \left\lfloor \mathcal{D}_{\mathbb{R}^N, \sqrt{\sigma_\ell^2 \mathbf{E}\mathbf{E}^\top + \tau_\ell^2 \mathbf{I}}} \right\rceil \\ e_r \leftarrow \mathcal{D}^{\sigma_r, \tau_r}_{\mathtt{EncZero}}, u \xleftarrow{\$} R_q \end{array} \right\}
\end{aligned}$$

where the first computational indistinguishability comes from Lemma 4. We conclude that

$$\begin{aligned}
&\{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r, \tau_r, \sigma_\ell, \tau_\ell}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b}))\} \\
&\quad \approx_c \left\{ (t, \mathsf{evk}, (-ut + \Delta(\mathfrak{a}m + \mathfrak{b}) + e_\ell + e_r, u) \;\middle|\; \begin{array}{r} e_\ell \leftarrow \left\lfloor \mathcal{D}_{\mathbb{R}^N, \sqrt{\sigma_\ell^2 \mathbf{E}\mathbf{E}^\top + \tau_\ell^2 \mathbf{I}}} \right\rceil \\ e_r \leftarrow \mathcal{D}^{\sigma_r, \tau_r}_{\mathtt{EncZero}}, u \xleftarrow{\$} R_q \end{array} \right\} \\
&\quad \approx_c \{(t, \mathsf{evk}, \mathtt{RandLinEval}_{\sigma_r, \tau_r, \sigma_\ell, \tau_\ell}(\mathsf{pk}, \mathsf{ct}, 0, 0) + (\Delta(\mathfrak{a}m + \mathfrak{b}), 0))\}.
\end{aligned}$$

Finally, the bound of variance of the output error can be computed as

$$\begin{aligned}
&\mathtt{Var}(\mathtt{Err}_t(\mathtt{RandLinEval}_{\sigma_r, \tau_r, \sigma_\ell, \tau_\ell}(\mathsf{pk}, \mathsf{ct}, \mathfrak{a}, \mathfrak{b}))) \\
&\quad \le N\mathtt{Var}(\mathfrak{r})\mathtt{Var}(e) + \mathtt{Var}(\hat{e}) + \mathtt{Var}(\mathtt{EncZero}_{\sigma_r, \tau_r}(\mathsf{pk})) \\
&\quad \le N\sigma_\ell^2 \mathtt{Var}(\mathtt{Err}_t(\mathsf{ct})) + \tau_\ell^2 + \frac{1}{12} + V^{\sigma_r, \tau_r}_{\mathtt{EncZero}}.
\end{aligned}$$

□

4 Our Sanitization Method

In this section, we provide a detailed analysis of our novel sanitization method for TFHE, leveraging the mask rerandomization and the randomized linear evaluation algorithm described in Section 3.

4.1 Our Algorithm

To achieve sanitizability, we modify the bootstrapping algorithm, as it is the most straightforward way to remove the existing error of the input ciphertext. Note that it is essential to randomize the underlying bootstrapping algorithm, since the noise term of the output ciphertext of the bootstrapping may leak partial information about the input ciphertext. Prior works [BI22,Klu22] focused on the randomization of unit operations in bootstrapping. In particular, they randomized the external products in the blind rotation, exploiting the randomized gadget decomposition technique [ASP14,BdMW16].

In contrast, in our sanitization algorithm, we directly randomize the input and output ciphertexts, rather than the unit operations. The key idea is to randomize the mask of the input ciphertext and run the bootstrapping algorithm only with the rerandomized mask, without the body. After bootstrapping, we perform a randomized linear evaluation to ensure that the bootstrapped result is correct. Let us explain in more detail. Let $\mathbf{c}_{\mathrm{in}} \in \mathbb{Z}_q^{N+1}$ be the input ciphertext. First, we randomize the input ciphertext by computing $\mathbf{c}_{\mathrm{rand}} := (b_{\mathrm{rand}}, \mathbf{a}_{\mathrm{rand}}) \leftarrow \mathtt{ReRand}(\mathbf{c}_{\mathrm{in}})$. By Lemma 5, the mask $\mathbf{a}_{\mathrm{rand}}$ is computationally indistinguishable from a uniformly sampled vector over $\mathbb{Z}_q^N$. Next, we run the key-switching algorithm over $\mathbf{c}_{\mathrm{rand}}$, resulting in $\mathbf{c}_{\mathrm{ks}} := (b_{\mathrm{ks}}, \mathbf{a}_{\mathrm{ks}}) \leftarrow \mathtt{KeySwitch}(\mathsf{KSK}, \mathbf{c}_{\mathrm{rand}})$. Note that the mask $\mathbf{a}_{\mathrm{ks}}$ is fully simulatable as well, since it is identical to the mask of $\mathtt{KeySwitch}(\mathsf{KSK}, (0, \mathbf{a}_{\mathrm{rand}}))$ due to the linearity of LWE ciphertexts. Now, using only the mask $\mathbf{a}_{\mathrm{ks}}$, we perform the blind rotation algorithm as $\mathtt{ct}_{\mathrm{br}} \leftarrow \mathtt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\mathrm{ks}}))$. Since $\mathtt{ct}_{\mathrm{br}}$ only depends on $\mathbf{a}_{\mathrm{rand}}$, it can be simulated by running the same algorithm with a uniformly random vector over $\mathbb{Z}_q^N$. Now, to realize the correct functionality of blind rotation, we homomorphically multiply the monomial $X^{\lfloor 2N/q \cdot b_{\mathrm{ks}} \rceil}$ to the resulting ciphertext $\mathtt{ct}_{\mathrm{br}}$. The correctness of this modification follows from the functionality of blind rotation, since

$$\begin{aligned}
\varphi_t(\mathtt{BlindRotate}(\mathsf{BRK}, (b, \mathbf{a}))) &\approx \mathsf{tv} \cdot X^{\lfloor 2N/q \cdot b \rceil + \sum_{i=0}^{n-1} \lfloor 2N/q \cdot a_i \rceil \cdot s_i} \\
&\approx X^{\lfloor 2N/q \cdot b \rceil} \cdot \mathsf{tv} \cdot X^{\sum_{i=0}^{n-1} \lfloor 2N/q \cdot a_i \rceil \cdot s_i} \\
&\approx \varphi_t\left(X^{\lfloor 2N/q \cdot b \rceil} \cdot \mathtt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}))\right).
\end{aligned}$$

However, a naive multiplication of the monomial will result in the leakage of the value $\lfloor 2N/q \cdot b_{\mathrm{ks}} \rceil$, since it is multiplied to the final noise term. Therefore, the randomized linear evaluation technique from Section 3.2 is utilized to multiply the monomial obliviously. Then, we can simulate the output noise as a composition of the error distribution of blind rotation and the error distribution after the randomized linear evaluation, finally achieving sanitization. We provide the exact algorithm in Algorithm 3.

It is worth noting that our sanitization method can be naturally applied to any AP-like cryptosystems without any modification of key algorithms. For example, FHEW or LMKCDEY schemes realize the blind rotation by directly multiplying the monomials $X^{a_i s_i}$ to the accumulator. Unlike prior works [BI22,Klu22],

Algorithm 3 New Sanitization Method $\mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}_{\mathrm{in}})$

Input: Public key pk, blind rotation key BRK, key-switching key KSK, input LWE ciphertext $\mathbf{c}_{\mathrm{in}} \in \mathbb{Z}_q^{N+1}$ and Gaussian parameters $\sigma_r, \tau_r, \sigma_\ell, \tau_\ell > 0$.

1: $\mathbf{c}_{\mathrm{rand}} \leftarrow \mathtt{ReRand}_{\sigma_r,\tau_r}(\mathsf{pk}, \mathbf{c}_{\mathrm{in}})$
2: $\mathbf{c}_{\mathrm{ks}} := (b_{\mathrm{ks}}, \mathbf{a}_{\mathrm{ks}}) \leftarrow \mathtt{KeySwitch}(\mathsf{KSK}, \mathbf{c}_{\mathrm{rand}}))$
3: $\mathsf{ct}_{\mathrm{br}} \leftarrow \mathtt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\mathrm{ks}}))$
4: $\mathsf{ct}_{\mathrm{lin}} \leftarrow \mathtt{RandLinEval}_{\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}_{\mathrm{br}}, X^{\lfloor 2N/q \cdot b_{\mathrm{ks}} \rceil}, 0)$
5: $\mathbf{c}_{\mathrm{out}} \leftarrow \mathtt{SampleExtract}(\mathsf{ct}_{\mathrm{lin}})$
6: **return** $\mathbf{c}_{\mathrm{out}}$

the blind rotation remains a black-box subroutine and therefore does not need to be modified at all.

We also remark that our sanitization algorithm can be naturally extended to accept arbitrary test vectors. Therefore, to construct a circuit-private evaluation algorithm, one can simply replace the last functional bootstrapping with Algorithm 3, instead of having an additional sanitization step at the end of the evaluation.

4.2 Noise and Security Analysis

In this section, we provide a noise analysis for our sanitization algorithm and its major building blocks, and give a detailed security proof. In our analysis, we assume that each component of LWE and RLWE ciphertexts follows the uniform distribution over $\mathbb{Z}_q$ and R_q due to the (R)LWE assumption. Then, the gadget decomposition result and the decomposition error will also follow uniform distributions. More precisely, for any $a \in \mathbb{Z}_q$, the inner product $\langle h(a), \mathbf{g}\rangle - a$ and the gadget decomposition $h(a)$ follow uniform distributions over $\mathbb{Z} \cap [-\frac{B}{2}, \frac{B}{2}]$ and $\mathbb{Z} \cap [-\frac{q}{2B^d}, \frac{q}{2B^d}]$, respectively. Therefore, their variances are given by $\frac{B^2}{12}$ and $\frac{q^2}{12B^{2d}}$.

Lemma 8 (Key Switching). *For an LWE ciphertext $\mathbf{c} \in \mathbb{Z}_q^{N+1}$, the noise variance of $\mathbf{c}_{\mathrm{ks}} \leftarrow \mathtt{KeySwitch}(\mathsf{KSK}, \mathbf{c})$ is bounded by*

$$\mathtt{Var}(\mathtt{Err}_{\mathbf{s}}(\mathbf{c}_{\mathrm{ks}})) \le \mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c})) + V_{\mathtt{KeySwitch}}$$

for

$$V_{\mathtt{KeySwitch}} := \frac{1}{12}\frac{1}{B'^{2d'}}q^2N + \frac{1}{12}\alpha^2 d' N B'^2.$$

Proof. Let $\mathbf{c} = (b, a_0, \ldots, a_{N-1})$ and $\mathbf{e}_i$ denote the encryption noise vector of i-th key-switching key KSK_i for $0 \le i < N$. Then, we have

$$\begin{aligned}\varphi_{\mathbf{s}}(\mathbf{c}_{\mathrm{ks}}) - \varphi_{\mathbf{t}}(\mathbf{c}) &= b + \sum_{i=0}^{N-1} h'(a_i)^\top \varphi_{\mathbf{s}}(\mathsf{KSK}_i) - (b + \sum_{i=0}^{N-1} a_i \cdot t_i) \\ &= \sum_{i=0}^{N-1} (\langle h'(a_i), \mathbf{g}' \cdot t_i + \mathbf{e}_i\rangle - a_i t_i)\end{aligned}$$

$$= \sum_{i=0}^{N-1} t_i \cdot (\langle h'(a_i), \mathbf{g}' \rangle - a_i) + \sum_{i=0}^{N-1} \langle h'(a_i), \mathbf{e}_i \rangle$$

from definition. Then, since each t_i is either zero or one, the final error term is bounded by $\frac{q^2 N}{12B'^{2d'}} + \frac{1}{12}\alpha^2 d' N B'^2$ and it concludes the proof. □

Lemma 9 (Modulus Switching). *For an LWE ciphertext* $\mathbf{c} = (b, a_0, \ldots, a_{n-1}) \in \mathbb{Z}_q^{n+1}$*, the noise variance of* $\mathbf{c}' = (\lfloor \frac{2N}{q} \cdot b \rceil, \lfloor \frac{2N}{q} \cdot a_0 \rceil, \ldots, \lfloor \frac{2N}{q} \cdot a_{n-1} \rceil) \in \mathbb{Z}_{2N}$ *is bounded by*

$$\mathtt{Var}(\mathtt{Err}_{\mathbf{s}}(\mathbf{c}')) \leq \left(\frac{2N}{q}\right)^2 \mathtt{Var}(\mathtt{Err}_{\mathbf{s}}(\mathbf{c})) + V_{\mathtt{ModSwitch}}$$

for $V_{\mathtt{ModSwitch}} := \frac{n+1}{12}$.

Proof. We have

$$\varphi_{\mathbf{s}}(\mathbf{c}') - \frac{2N}{q}\varphi_{\mathbf{s}}(\mathbf{c}) = (\lfloor \frac{2N}{q} \cdot b \rceil - \frac{2N}{q} \cdot b) + \sum_{i=0}^{n-1} (\lfloor \frac{2N}{q} \cdot a_i \rceil - \frac{2N}{q} \cdot a_i) \cdot s_i.$$

Similar to the previous proofs, we assume that the ciphertext follows the uniform distribution. Therefore, $\lfloor \frac{2N}{q} \cdot b \rceil - \frac{2N}{q} \cdot b$ and $\lfloor \frac{2N}{q} \cdot a_i \rceil - \frac{2N}{q} \cdot a_i$ follow a uniform distribution over $\frac{1}{q}\mathbb{Z} \cap [-\frac{1}{2}, \frac{1}{2}]$ for $0 \leq i < n$. Moreover, as s_i is either one or zero, the variance of the term above is bounded by $\frac{n+1}{12}$. □

Lemma 10 (Blind Rotation). *For an LWE ciphertext* $\mathbf{c} = (b, a_0, \cdots, a_{n-1}) \in \mathbb{Z}_q^{n+1}$*, the noise variance of* $\mathsf{ct}_{\mathrm{br}} \leftarrow \mathtt{BlindRotate}(\mathsf{BRK}, \mathbf{c})$ *is bounded by*

$$\mathtt{Var}(\mathtt{Err}_t(\mathsf{ct}_{\mathrm{br}})) \leq V_{\mathtt{BlindRotate}} := \frac{1}{12}\frac{1}{B^{2d}} n(N+1)q^2 + \frac{1}{6}\beta^2 n d N B^2.$$

Proof. Let ACC_i be the value of the accumulator before i-th iteration of the for loop in Algorithm 1 (lines 4-6). Also, let ACC_n denote the output of Algorithm 1. Then, by definition, it follows that

$$\begin{aligned}
&\varphi_t(\mathsf{ACC}_{i+1}) \\
&= \varphi_t(\mathsf{ACC}_i) + (X^{\tilde{a}_i} - 1) \cdot \varphi_t(\mathsf{ACC}_i \boxdot \mathsf{BRK}_i) \\
&= \varphi_t(\mathsf{ACC}_i) + (X^{\tilde{a}_i} - 1) \cdot \langle h(\mathsf{ACC}_i), \mathbf{g} \cdot \begin{bmatrix} 1 \\ t \end{bmatrix} s_i + \mathbf{e}_i \rangle \\
&= X^{\tilde{a}_i s_i} \varphi_t(\mathsf{ACC}_i) + (X^{\tilde{a}_i} - 1) \cdot \left(s_i \left(\langle h(\mathsf{ACC}_i), \mathbf{g} \rangle - \mathsf{ACC}_i \right) \cdot \begin{bmatrix} 1 \\ t \end{bmatrix} + \langle h(\mathsf{ACC}_i), \mathbf{e}_i \rangle \right)
\end{aligned}$$

where $\mathbf{e}_i$ denotes the noise vector of i-th blind rotation key BRK_i. As s_i is a binary value and t is a polynomial with binary coefficients, the introduced error can be bounded by

$$(N+1)\frac{q^2}{12B^{2d}} + \frac{1}{6} d N \beta^2 B^2.$$

Note that each line introduces an additional error, and monomial multiplication only rotates the coefficient vector (up to sign). So the variance for the final noise term, i.e., $\mathtt{Var}(\mathtt{Err}_t(\mathsf{ACC}_n))$ is given by

$$n(N+1)\frac{q^2}{12B^{2d}} + \frac{1}{6}dnN\beta^2B^2.$$

□

Lemma 11 (Sanitization). *For an LWE ciphertext $\mathbf{c}_{\mathrm{in}} \in \mathbb{Z}_q^{N+1}$, the noise variance of $\mathbf{c}_{\mathrm{out}} \leftarrow \mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}_{\mathrm{in}})$ is bounded by*

$$\mathtt{Var}(\mathtt{Err}_t(\mathbf{c}_{\mathrm{out}})) \le V_{\mathtt{Sanitize}}^{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell} := N\sigma_\ell^2 V_{\mathtt{BlindRotate}} + \tau_\ell^2 + \frac{1}{12} + V_{\mathtt{EncZero}}^{\sigma_r,\tau_r}.$$

Proof. Recall that our sanitization method performs the randomized linear evaluation right after the blind rotation, so the noise variance directly follows from Lemma 10 and Lemma 7. □

Finally, we provide the correctness and security proof of our sanitization algorithm. The correct parameter selection will be discussed in Section 5.1.

Theorem 1 (Correctness of Sanitize). *Let σ_r, τ_r and σ_ℓ, τ_ℓ be reals that satisfy the conditions of Lemma 4 and Lemma 7 respectively. For an LWE encryption $\mathbf{c}_{\mathrm{in}} \in \mathbb{Z}_q^{N+1}$ of $m \in \mathbb{Z}_p \cap [0, p/2)$, let*

$$\mathbf{c}_{\mathtt{Sanitize}} \leftarrow \mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}_{\mathrm{in}}).$$

Then,

$$\varphi_{\mathbf{t}}(\mathbf{c}_{\mathtt{Sanitize}}) = \Delta m_{\mathtt{Sanitize}} + e_{\mathtt{Sanitize}}$$

such that $m_{\mathtt{Sanitize}} := \lfloor m + \frac{p}{2N}e \rceil \in \mathbb{Z}_p$ for some $e \in \mathbb{Z}$ where

$$\mathtt{Var}(e) \le \left(\frac{2N}{q}\right)^2 \cdot \left(\mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c}_{\mathrm{in}})) + V_{\mathtt{EncZero}}^{\sigma_r,\tau_r} + V_{\mathtt{KeySwitch}}\right) + V_{\mathtt{ModSwitch}}.$$

Moreover, the variance of $e_{\mathtt{Sanitize}} \in \mathbb{Z}$ is bounded by $V_{\mathtt{Sanitize}}^{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}$.

Proof. We will follow the notations from Algorithm 3 in the following proof. In our sanitization algorithm, the mask rerandomization and key-switching are performed before the blind rotation. By Lemma 5 and Lemma 10, it follows that the $\mathtt{Err}_{\mathbf{t}}(\mathbf{c}_{\mathrm{ks}})$ is bounded by $\mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c}_i n)) + V_{\mathtt{EncZero}}^{\sigma_r,\tau_r} + V_{\mathtt{KeySwitch}}$. Subsequently, in the blind rotation, the ciphertext modulus is switched from q to $2N$ and it introduces error with variance $V_{\mathtt{ModSwitch}}$, resulting in the error variance with bound

$$\left(\frac{2N}{q}\right)^2 \cdot \left(\mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c}_{\mathrm{in}})) + V_{\mathtt{EncZero}}^{\sigma_r,\tau_r} + V_{\mathtt{KeySwitch}}\right) + V_{\mathtt{ModSwitch}}.$$

Therefore, in the blind rotation, it essentially computes $X^{\frac{2N}{p}\cdot m+e}$ with $\mathtt{Var}(e) \le \left(\frac{2N}{q}\right)^2 \cdot \left(\mathtt{Var}(\mathtt{Err}_{\mathbf{t}}(\mathbf{c}_i n)) + V_{\mathtt{EncZero}}^{\sigma_r,\tau_r} + V_{\mathtt{KeySwitch}}\right) + V_{\mathtt{ModSwitch}}$, and the functionality

of the blind rotation implies that the constant term of the final accumulator is a noisy encoding of $\Delta m_{\texttt{Sanitize}} = \Delta\lfloor m + \frac{p}{2N}e \rceil$. On the other hand, the variance of the output noise is bounded by $V_{\texttt{Sanitize}}^{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}$ by Lemma 11. □

Theorem 2 (Sanitizability of `Sanitize`). *Let $\sigma_r, \tau_r, \sigma_\ell, \tau_\ell$ be reals that satisfy the conditions of Theorem 1. Then, $\texttt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}$ (Algorithm 3) is sanitizing.*

Proof. Let $\mathbf{c}_{\text{in}} = (b_{\text{in}} = -\langle \mathbf{a}_{\text{in}}, \mathbf{t}\rangle + \Delta m + e_{\text{in}}, \mathbf{a}_{\text{in}})$. To prove the sanitization property, we alternatively show that the distribution of the output ciphertext of `Sanitize` only depends on evk and m. We proceed using the hybrid argument, where $\mathbf{Hyb}_0$ corresponds to the output of $\texttt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}_{\text{in}})$.

$\mathbf{Hyb}_1$. In this hybrid, we replace $\mathbf{c}_{\text{rand}}$ with a ciphertext with uniform mask. By Lemma 5, the distribution of $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$ are computationally indistinguishable.

$\mathbf{Hyb}_1(\mathbf{t}, \mathsf{evk}, m, e_{\text{in}})$

$\mathbf{a}_{\text{rand}} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\texttt{EncZero}}|_0$

$\mathbf{c}_{\text{rand}} \leftarrow (b_{\text{rand}} := -\langle \mathbf{a}_{\text{rand}}, \mathbf{t}\rangle + \Delta m + e_{\text{in}} + e_r, \mathbf{a}_{\text{rand}})$

$\mathbf{c}_{\text{ks}} := (b_{\text{ks}}, \mathbf{a}_{\text{ks}}) \leftarrow \texttt{KeySwitch}(\mathsf{KSK}, \mathbf{c}_{\text{rand}})$

$\mathsf{ct}_{\text{br}} \leftarrow \texttt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\text{ks}}))$

$\mathsf{ct}_{\text{lin}} \leftarrow \texttt{RandLinEval}_{\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}_{\text{br}}, X^{\lfloor 2N/q \cdot b_{\text{ks}} \rceil}, 0)$

$\mathbf{c}_{\text{out}} \leftarrow \texttt{SampleExtract}(\mathsf{ct}_{\text{lin}})$

return $\mathbf{c}_{\text{out}}$

$\mathbf{Hyb}_2$. In this hybrid, we replace ct_{lin}. Note that by the correctness of `BlindRotate`, ct_{br} encrypts $\mathsf{tv} \cdot X^{\sum_{i=0}^{n-1} \lfloor 2N/q \cdot a_{\text{ks},i} \rceil s_i}$, where tv is defined as in Algorithm 1 and $\mathbf{a}_{\text{ks}} = (a_{\text{ks},0}, \ldots, a_{\text{ks},n-1})$. Therefore, by Lemma 7, the distribution of $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are computationally indistinguishable.

$\mathbf{Hyb}_2(\mathbf{t}, \mathsf{evk}, m, e_{\text{in}})$

$\mathbf{a}_{\text{rand}} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\texttt{EncZero}}|_0$

$\mathbf{c}_{\text{rand}} \leftarrow (b_{\text{rand}} := -\langle \mathbf{a}_{\text{rand}}, \mathbf{t}\rangle + \Delta m + e_{\text{in}} + e_r, \mathbf{a}_{\text{rand}})$

$\mathbf{c}_{\text{ks}} := (b_{\text{ks}}, \mathbf{a}_{\text{ks}}) \leftarrow \texttt{KeySwitch}(\mathsf{KSK}, \mathbf{c}_{\text{rand}})$

$\mathsf{ct}_{\text{br}} \leftarrow \texttt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\text{ks}}))$

$m' \leftarrow \mathsf{tv} \cdot X^{\lfloor 2N/q \cdot b_{\text{ks}} \rceil + \sum_{i=0}^{n-1} \lfloor 2N/q \cdot a_{\text{ks},i} \rceil s_i}$

$\mathsf{ct}_{\text{lin}} \leftarrow \texttt{RandLinEval}_{\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}_{\text{br}}, 0, 0) + (m', 0)$

$\mathbf{c}_{\text{out}} \leftarrow \texttt{SampleExtract}(\mathsf{ct}_{\text{lin}})$

return $\mathbf{c}_{\text{out}}$

$\mathbf{Hyb}_3$. In this hybrid, we replace m' with Δm. Note that the output ciphertext $\mathbf{c}_{\text{out}}$ only encrypts the constant term of $m' = \mathsf{tv} \cdot X^{\lfloor 2N/q \cdot b_{\text{ks}} \rceil + \sum_{i=0}^{n-1} \lfloor 2N/q \cdot a_{\text{ks},i} \rceil s_i}$,

which is Δm by the correctness of bootstrapping. Therefore, this change is only syntactic, and the distribution of $\mathbf{Hyb}_2$ and $\mathbf{Hyb}_3$ are identical.

$\mathbf{Hyb}_3(\mathbf{t}, \mathsf{evk}, m, e_{\mathrm{in}})$

$\mathbf{a}_{\mathrm{rand}} \xleftarrow{\$} \mathbb{Z}_q^N, e_r \leftarrow \mathcal{D}^{\sigma_r,\tau_r}_{\mathtt{EncZero}}|_0$

$\mathbf{c}_{\mathrm{rand}} \leftarrow (b_{\mathrm{rand}} := -\langle \mathbf{a}_{\mathrm{rand}}, \mathbf{t}\rangle + \Delta m + e_{\mathrm{in}} + e_r, \mathbf{a}_{\mathrm{rand}})$

$\mathbf{c}_{\mathrm{ks}} := (b_{\mathrm{ks}}, \mathbf{a}_{\mathrm{ks}}) \leftarrow \mathtt{KeySwitch}(\mathsf{KSK}, \mathbf{c}_{\mathrm{rand}})$

$\mathsf{ct}_{\mathrm{br}} \leftarrow \mathtt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\mathrm{ks}}))$

$\mathsf{ct}_{\mathrm{lin}} \leftarrow \mathtt{RandLinEval}_{\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}_{\mathrm{br}}, 0, 0) + (\Delta m, 0)$

$\mathbf{c}_{\mathrm{out}} \leftarrow \mathtt{SampleExtract}(\mathsf{ct}_{\mathrm{lin}})$

return $\mathbf{c}_{\mathrm{out}}$

$\mathbf{Hyb}_4$. In this hybrid, we remove b_{ks} entirely. This change is only syntactical, as b_{ks} is not used anywhere in $\mathbf{Hyb}_3$. Note that we also remove b_{rand}, since

$$\mathtt{KeySwitch}(\mathsf{KSK}, (b_{\mathrm{rand}}, \mathbf{a}_{\mathrm{rand}})) = \mathtt{KeySwitch}(\mathsf{KSK}, (0, \mathbf{a}_{\mathrm{rand}})) + (b_{\mathrm{rand}}, 0)$$

by definition. Therefore, the distribution of $\mathbf{Hyb}_3$ and $\mathbf{Hyb}_4$ are identical.

$\mathbf{Hyb}_4(\mathsf{evk}, m)$

$\mathbf{a}_{\mathrm{rand}} \xleftarrow{\$} \mathbb{Z}_q^N$

$(\,\cdot\,, \mathbf{a}_{\mathrm{ks}}) \leftarrow \mathtt{KeySwitch}(\mathsf{KSK}, (0, \mathbf{a}_{\mathrm{rand}}))$

$\mathsf{ct}_{\mathrm{br}} \leftarrow \mathtt{BlindRotate}(\mathsf{BRK}, (0, \mathbf{a}_{\mathrm{ks}}))$

$\mathsf{ct}_{\mathrm{lin}} \leftarrow \mathtt{RandLinEval}_{\sigma_\ell,\tau_\ell}(\mathsf{pk}, \mathsf{ct}_{\mathrm{br}}, 0, 0) + (\Delta m, 0)$

$\mathbf{c}_{\mathrm{out}} \leftarrow \mathtt{SampleExtract}(\mathsf{ct}_{\mathrm{lin}})$

return $\mathbf{c}_{\mathrm{out}}$

We conclude that the output of $\mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}_{\mathrm{in}})$ is computationally indistinguishable from $\mathbf{Hyb}_4(\mathsf{evk}, \mathtt{Dec}(\mathbf{t}, \mathbf{c}_{\mathrm{in}}))$. This implies that for any two ciphertexts $\mathbf{c}, \mathbf{c}'$ where $m = \mathtt{Dec}(\mathbf{t}, \mathbf{c}) = \mathtt{Dec}(\mathbf{t}, \mathbf{c}')$, we have

$$\begin{aligned}
&\{(t, \mathsf{evk}, \mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c})\} \\
&\quad \approx_c \{(t, \mathsf{evk}, \mathbf{Hyb}_4(\mathsf{evk}, m))\} \\
&\quad \approx_c \{(t, \mathsf{evk}, \mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}(\mathsf{evk}, \mathbf{c}')\}.
\end{aligned}$$

Therefore, $\mathtt{Sanitize}_{\sigma_r,\tau_r,\sigma_\ell,\tau_\ell}$ is sanitizing. □

4.3 Performance Analysis

In this subsection, we analyze and compare the performance of our new sanitization method with previous works that utilize randomized external products [BI22,Klu22]. Specifically, we focus on the number of Gaussian samples required for sanitization, as Gaussian sampling is often the primary bottleneck in

practical implementations. For instance, in previous works [BI22,Klu22], Gaussian sampling accounted for up to 80% and 60%, respectively, of the total running time.

In our sanitization method, Gaussian sampling is required in `ReRand` and `RandLinEval`. In `ReRand`, we need N Gaussian samples for $\mathcal{D}_{\mathbb{Z}^N,\sigma_r}$ and N rounded Gaussian samples for $\lfloor \mathcal{D}_{\mathbb{R}^N,\tau_r} \rceil$ when calling `EncZero`. However, since we only use the extracted LWE ciphertext of `EncZero`, we only need to sample the constant term of the rounded Gaussian distribution. This reduces the requirement to N Gaussian samples and 1 rounded Gaussian sample. In `RandLinEval`, we need N Gaussian samples for $\mathcal{D}_{X^{-b}+p\mathbb{Z}^N,\sigma_\ell}$ and N rounded Gaussian samples for $\lfloor \mathcal{D}_{\mathbb{R}^N,\tau_\ell} \rceil$. Additionally, we need N Gaussian samples and N rounded Gaussian samples from `EncZero`, as described earlier. We note that in our sanitization algorithm (Algorithm 3), `SampleExtract` is invoked immediately after `RandLinEval`. This means that only the constant term of the rounded Gaussian samples is needed, similar to `ReRand`. This results in $2N$ Gaussian samples and 2 rounded Gaussian samples. Therefore, our method requires a total of $3N$ Gaussian samples and 3 rounded Gaussian samples.

In contrast, the major source of Gaussian sampling in previous works stems from their heavy use of randomized gadget decomposition. Specifically, they replace a gadget decomposition $h(\mathsf{ACC})$ with samples from $\mathcal{D}_{h(c_{i,j})+\Lambda^\perp(\mathbf{g}),\sigma}$, where $c_{i,j}$ are coefficients of ACC. Therefore, a single decomposition requires $2dN$ Gaussian samples. Since we need to decompose an RLWE ciphertext n times for a single blind rotation, the total amount of Gaussian samples needed is $2dnN$. Previous works also require a post-processing step. Bourse and Izabachène [BI22] use a technique similar to ours, adding an RLWE public key encryption of zero. Using the optimization we mentioned earlier, this requires an additional $N+1$ Gaussian samples. On the other hand, Kluczniak [Klu22] uses an LWE public key encryption of zero, which requires an additional $h+1$ Gaussian samples, where h is the length of the LWE public key.

We summarize the number of Gaussian samples required in Table 1. In short, our method reduces the required samples by up to a factor of $O(dn)$.

Ours	[BI22]	[Klu22]
$3N+3$	$2dnN+N+1$	$2dnN+h+1$

Table 1. Number of Gaussian samples required for sanitization.

5 Experiments

In this section, we present the experimental results of our sanitization algorithm from our proof-of-concept implementation and compare them with those of previous works.

5.1 Parameters Selection

Our parameters for the base TFHE scheme are given in Table 2. We set $p = 4$ and $q = 2^{64}$, which are common parameters for instantiating binary TFHE. We choose the LWE parameters n, α and the RLWE parameters N, β to achieve 128-bit security, as verified by the lattice estimator [APS15]. The gadget decomposition parameters B, d and B', d' were chosen to optimize performance while ensuring correct decryption.

n	N	p	q	α	β	B	d	B'	d'
612	2048	2^2	2^{64}	$2^{50.40}$	$2^{12.65}$	2^{11}	3	2^3	4

Table 2. Base parameters for TFHE.

Now we discuss the selection of Gaussian parameters $\sigma_r, \tau_r, \sigma_\ell, \tau_\ell$. In a nutshell, we select them to minimize the output error variance while satisfying the conditions of Lemma 4 and Lemma 7. Note that the conditions and the error terms for the mask rerandomization and randomized linear evaluation are not related; thus, we can choose them independently to minimize the final error. Moreover, we remark that the selection of parameters σ_r, τ_r and σ_ℓ, τ_ℓ can be done by solving the same minimization problem, as given below:

$$\begin{aligned} &\text{minimize } S\sigma^2 + \tau^2 \\ &\text{subject to } \frac{1}{\sigma^2} + \frac{T}{\tau^2} \leq \frac{1}{\eta^2}. \end{aligned}$$

By using Lagrange multipliers, one can easily check that the solution of the above problem is

$$\sigma = \eta \cdot \sqrt{\frac{\sqrt{S} + \sqrt{T}}{\sqrt{S}}},\ \tau = \eta \cdot \sqrt{T + \sqrt{ST}}.$$

In Lemma 4, we can take $\eta = \max(\eta_\varepsilon(\mathbb{Z}^{2N})\sqrt{2/\pi}, \rho)$ where $\mathsf{RLWE}^{1,N,q}_{\rho/\sqrt{2},\rho/\sqrt{2}}$ is hard, $S = N(\beta^2 + 1/4)$, and T as a bound of $\|\left[\mathbf{E}_{\mathsf{pk}}\ \mathbf{T}\right]\|_2^2$. A small issue is that since e_{pk} is sampled from a Gaussian distribution, its bound is infinite. Therefore, we use a probabilistic Gaussian bound of 12σ for a random variable sampled from a Gaussian distribution with parameter σ. This bound is satisfied with an overwhelming probability of approximately

$$1 - \operatorname{erfc}\left(\frac{12\sigma}{\sigma\sqrt{2}}\right) = 1 - \operatorname{erfc}(6\sqrt{2}) \geq 1 - 2^{-107}.$$

Using this bound, we set T as

$$\|\left[\mathbf{E}_{\mathsf{pk}}\ \mathbf{T}\right]\|_2^2 \leq \|\mathbf{E}_{\mathsf{pk}}\|_2^2 + \|\mathbf{T}\|_2^2 \leq N^2(12^2\beta^2 + 1) := T.$$

Similarly, for σ_ℓ, τ_ℓ, we can take $\eta = \eta_\varepsilon(p\mathbb{Z}^N)/\sqrt{2\pi}$, $S = NV_{\texttt{BlindRotate}}$ and $T = 12^2N^2V_{\texttt{BlindRotate}}$. For smoothing parameters, we use $\eta_\varepsilon(\mathbb{Z}^{2N}) \approx \eta_\varepsilon(\mathbb{Z}^N) \approx 6$ for $\varepsilon = 2^{-128}$. For the security parameter for Lemma 4, we use $\rho \approx 66.82$, which was also validated by lattice-estimator to support 128-bit security. We provide the selected Gaussian parameters in Table 3.

σ_r	τ_r	σ_ℓ	τ_ℓ
$2^{10.61}$	$2^{33.29}$	$2^{7.80}$	$2^{57.18}$

Table 3. Gaussian parameters for `Sanitize`.

Finally, we show that our sanitization algorithm returns a correct result with a high probability, given these parameters. We first measure the noise from each operation given the parameters. By substituting the parameters into the noise estimation in Lemmas 8 to 10 and Lemma 4, we obtain $V_{\texttt{KeySwitch}} = 2^{116.27}$, $V_{\texttt{ModSwitch}} = 2^{5.67}$, $V_{\texttt{BlindRotate}} = 2^{78.67}$ and $V_{\texttt{EncZero}} = 2^{66.60}$. Hence, in Theorem 1, the standard deviation of the distribution of error e is given by ≈ 70.52. Note that for correct bootstrapping, $\frac{p}{2N}e$ should not exceed $1/2$. Therefore, assuming that the distribution of e is close to Gaussian following the analysis from prior works [CGGI16], we obtain that the failure probability is bounded by $\approx 2^{-41.23}$.

On the other hand, the sanitization output may fail as well, due to the large noise growth. Substituting the parameters to the formula in Lemma 11 gives the bound $2^{57.18}$ for the output standard deviation of sanitization noise. Therefore, the decryption failure probability for the sanitization is bounded by $\approx 2^{-147.32}$.

5.2 Implementation Details

In this subsection, we describe some of the details for our implementation of `Sanitize`.

Gaussian Sampler. In our sanitization algorithm, we require two types of Gaussian samplers: the discrete Gaussian sampler and the rounded Gaussian sampler. For discrete Gaussian sampling, we need samples from two distributions: $\mathcal{D}_{\mathbb{Z}^N,\sigma_r}$ and $\mathcal{D}_{X^b+p\mathbb{Z}^N,\sigma_\ell}$ for some b. Given that σ_r and σ_ℓ are reasonably small in our parameter settings (see Table 3), we utilize the inversion sampler with precomputed cumulative distribution tables. For sampling from $\mathcal{D}_{X^b+p\mathbb{Z}^N,\sigma_\ell}$, we note that the negacyclic matrix of X^b is essentially a permutation matrix. Therefore, the following equivalence holds, where $\mathbf{X}_b$ is the negacyclic matrix of X^b:

$$\mathcal{D}_{X^b+p\mathbb{Z}^N,\sigma_\ell} \equiv \mathbf{X}_b\mathcal{D}_{1+p\mathbb{Z}^N,\sigma_\ell} \equiv \mathbf{X}_b\left((1,0,\ldots,0) + p\left(\mathcal{D}_{\mathbb{Z},\frac{1}{p},\frac{1}{p}\sigma_\ell} \times \prod_{i=1}^{N-1}\mathcal{D}_{\mathbb{Z},\frac{1}{p}\sigma_\ell}\right)\right).$$

Thus, we can sample from $\mathcal{D}_{X^b+p\mathbb{Z}^N,\sigma_\ell}$ using only two Gaussian samplers, each sampling from $\mathcal{D}_{\mathbb{Z},\sigma_\ell/p}$ and $\mathcal{D}_{\mathbb{Z},1/p,\sigma_\ell/p}$. For rounded Gaussian sampling, we use the Ziggurat method by Marsaglia and Tsang [MT00].

Exact Polynomial Multiplication. In most TFHE implementations, the Fast Fourier Transform (FFT) is used for fast polynomial multiplication. However, since FFT is computed over floating-point numbers, it can only provide an approximate result, especially when the coefficients of the polynomials are large. In the original TFHE scheme, this error was tolerable because all Ring-LWE operations are inherently noisy. However, when implementing randomized algorithms, this can be problematic, as the additional FFT error is not simulatable. Therefore, to securely implement the previous TFHE sanitization methods with randomizations [BI22,Klu22], much slower exact polynomial multiplication algorithms, such as the Number Theoretic Transform (NTT) or Karatsuba multiplication must be used, or parameters must be heavily restricted.

On the other hand, our sanitization algorithm uses the blind rotation of the original TFHE bootstrapping as a black box, freeing us from such restrictions. In our method, exact polynomial multiplication is only required in the `ReRand` and `RandLinEval` steps.

To perform exact polynomial multiplication, we use the split-FFT trick [Hwa23,WHS+24]. We first note that when computing $f \cdot g$ in the `ReRand` and `RandLinEval` steps, the coefficients of one polynomial, say f, are sampled from a Gaussian distribution with parameter σ_r or σ_ℓ, which is small. Then, to reduce FFT error, we decompose the other polynomial g as

$$g = g_0 + g_1 B_F + \cdots + g_{d_F-1} B_F^{d_F-1}$$

where $\|g_i\|_\infty \leq B_F$, and compute

$$f \cdot g = (f \cdot g_0) + (f \cdot g_1) B_F + \cdots + (f \cdot g_{d_F-1}) B_F^{d_F-1}.$$

In our work, we set $B_F = 2^{22}$ and $d_F = 3$ so that $B_F^{d_F} = 2^{66} \geq q = 2^{64}$. According to the estimation in [WHS+24], the variance of the FFT error is bounded by $2^{-108.6} N^2 \|f\|_\infty^2 B_F^2$. This translates to a failure probability of less than $2^{-2373.48}$ in our parameter settings.

5.3 Benchmark Results

We present the benchmark results of our and previous sanitization algorithms in Table 4, along with the base TFHE bootstrapping performance. We implemented our sanitization algorithm in Go, based on the TFHE-go library [Hwa23], which provides a fast AVX2-accelerated implementation of the TFHE scheme. All benchmark results of our implementation were measured on a server machine equipped with an Intel Xeon Platinum 8268 CPU. Our code is available at `https://github.com/SNUCP/tfhe-sanitization`.

We also implement the soak-spin-repeat sanitization [DS16] based on our parameters given in Table 2. For decryption failure rate $\approx 2^{-40}$, we can add error

with a standard deviation of approximately $2^{58.15}$ in each cycle. Since the standard deviation of the output error after a single bootstrapping is $\sqrt{V_{\texttt{BlindRotate}}} \approx 2^{39.34}$ in our parameters, the statistical distance between ciphertexts decreases by a factor of $2^{39.34}/2^{58.15} = 2^{-18.81}$ each cycle. Therefore, to achieve 80 bits of statistical security, we need $80/18.81 \approx 5$ cycles. Note that this increases the bootstrapping failure rate to $2^{-21.64}$. To target a lower failure rate, more cycles might be necessary, which would further degrade performance.

Finally, for the sanitization algorithms from [BI22,Klu22], we directly take the performance numbers of their algorithms and the base TFHE bootstrapping from the respective papers. Kluczniak [Klu22] provides two implementations, one based on NTT and the other on FFT. In the table, we denote them as (Int) and (Double) respectively. Note that [BI22] only achieves 100 bits of computational security, while [Klu22] achieves 128 bits. However, both target 80 bits of statistical security.

	Ours	[DS16]	[BI22]	[Klu22] (Int)	[Klu22] (Double)
Sanitization	35.88ms	173.00ms	7500ms	1360ms	1330ms
Bootstrapping	34.70ms		420ms	140ms	270ms

Table 4. Performance of sanitization algorithms.

As we can clearly see, our sanitization method offers significant speedups compared to previous works. Specifically, it is approximately 4.82, 209.03, and 37.07 times faster than [DS16,BI22,Klu22], respectively. When compared to the original TFHE bootstrapping with the same parameters, our method is only 3.4% slower, with a 1.18ms difference in performance. In contrast, [DS16], [BI22], and [Klu22] are 4.99, 17.85, and 4.92 times slower than the original TFHE bootstrapping, respectively.

References

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

ASP14. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 297–314. Springer, 2014.

BCL+23. Loris Bergerat, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, Adeline Roux-Langlois, and Samuel Tap. New secret keys for enhanced performance in (T)FHE. Cryptology ePrint Archive, Report 2023/979, 2023.

BdMW16. Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz,

editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany.

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.

BI22. Florian Bourse and Malika Izabachène. Plug-and-play sanitization for TFHE. Cryptology ePrint Archive, Paper 2022/1438, 2022. `https://eprint.iacr.org/2022/1438`.

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

dCKK+24. Leo de Castro, Duhyeong Kim, Miran Kim, Keewoo Lee, Seonhong Min, and Yongsoo Song. More efficient lattice-based OLE from circuit-private linear HE with polynomial overhead. Cryptology ePrint Archive, Paper 2024/1534, 2024.

DD22. Nico Döttling and Jesko Dujmovic. Maliciously circuit-private FHE from information-theoretic principles. Cryptology ePrint Archive, Report 2022/495, 2022.

DM15. Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 617–640. Springer, 2015.

DS16. Léo Ducas and Damien Stehlé. Sanitization of fhe ciphertexts. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35*, pages 294–310. Springer, 2016.

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

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

GHV10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Heidelberg, Germany.

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

Hwa23. Intak Hwang. TFHE-go. Online: `https://github.com/sp301415/tfhe-go`, 2023.

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In *Annual International Cryptology Conference*, pages 549–580. Springer, 2023.

Klu22. Kamil Kluczniak. Circuit privacy for FHEW/TFHE-style fully homomorphic encryption in practice. Cryptology ePrint Archive, Paper 2022/1459, 2022. `https://eprint.iacr.org/2022/1459`.

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 *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–256. Springer, 2023.

LMSS23. Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Faster TFHE bootstrapping with block binary keys. In Joseph K. Liu, Yang Xiang, Surya Nepal, and Gene Tsudik, editors, *ASIACCS 23: 18th ACM Symposium on Information, Computer and Communications Security*, pages 2–13, Melbourne, VIC, Australia, July 10–14, 2023. ACM Press.

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.

LY23. KangHoon Lee and Ji Won Yoon. Discretization error reduction for high precision torus fully homomorphic encryption. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 33–62, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

MKMS22. Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. Efficient lattice-based inner-product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 163–193. Springer, 2022.

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

MT00. George Marsaglia and Wai Wan Tsang. The ziggurat method for generating random variables. *Journal of statistical software*, 5:1–7, 2000.

OPP14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Heidelberg, Germany.

Pei10. Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *Annual Cryptology Conference*, pages 80–97. Springer, 2010.

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

WHS+24. Ruida Wang, Jincheol Ha, Xuan Shen, Xianhui Lu, Chunling Chen, Kunpeng Wang, and Jooyoung Lee. Fhew-like leveled homomorphic evaluation: Refined workflow and polished building blocks. Cryptology ePrint Archive, Report 2024/1318, 2024.