Bootstrapping (T)FHE Ciphertexts via Automorphisms: Closing the Gap Between Binary and Gaussian Keys

Olivier Bernard and Marc Joye

Zama, Paris, France
firstname.lastname@zama.ai

Abstract. The GINX method in TFHE enables low-latency ciphertext bootstrapping with relatively small bootstrapping keys but is limited to binary or ternary key distributions. In contrast, the AP method supports arbitrary key distributions, however, at the cost of significantly large bootstrapping keys. Building on AP, automorphism-based methods, introduced in LMK$^+$ (EUROCRYPT 2023), achieve smaller key sizes. Each automorphism application nevertheless necessitates a key switch, introducing additional computational overhead and noise accumulation.
This paper advances automorphism-based methods in two important ways. First, it proposes a novel traversal blind rotation algorithm that optimizes the number of key switches for a given key material. Second, it introduces a new external product that is automorphism-parametrized and seamlessly applies an automorphism to one of the input ciphertexts. Together, these techniques substantially reduce the number of key switches, leading to faster bootstrapping and improved noise control.
In a typical setting, by utilizing additional key material, the LLW$^+$ approach (TCHES 2024) reduces key switches by 17% compared to LMK$^+$. Our combined techniques achieve a 46% reduction using similar key material and can eliminate an arbitrary large number (e.g., more than 99%) of key switches with only a moderate ($9\times$) increase in key material size.
As an independent contribution, this paper introduces a comprehensive theoretical framework for analyzing the expected number of automorphism key switches, whose predictions perfectly align with the results of extensive numerical experiments, demonstrating its practical relevance.

Keywords: Fully homomorphic encryption (FHE)· Ciphertext bootstrapping· Blind rotation· Automorphisms· Implementation

1 Introduction

Fully homomorphic encryption (FHE) schemes [RAD78, Gen10] enable the evaluation of any circuit over encrypted data, ensuring that the data remains end-to-end encrypted without requiring decryption for processing. Since their inception in 2009, substantial research has focused on enhancing the practical efficiency of FHE, which remains a critical challenge for its widespread deployment.

Current FHE implementations require noisy ciphertexts for their security. However, as homomorphic operations are performed, noise accumulates, and beyond a certain threshold, ciphertexts become undecryptable. This issue is resolved via a bootstrapping technique that refreshes ciphertexts by reducing the noise to an acceptable level. Ciphertext bootstrapping is carried out through a homomorphic evaluation of the decryption algorithm [Gen10]. On input a (highly) noisy ciphertext, the output is a ciphertext encrypting the same message, but with a reduced level of noise. This is a public operation. The most efficient instantiations of ciphertext bootstrapping makes use of an astute technique known as blind rotation [AP14, DM15, GINX16, CGGI20].

Using a polynomial representation, the operation of blind rotation consists in the homomorphic evaluation of $x^{\sum_{i=1}^{n} a_i s_i} \cdot \mathcal{w}$ for some polynomial $\mathcal{w} := \mathcal{w}(x)$, where vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ is public and vector $\boldsymbol{s} = (s_1, \ldots, s_n)$ is secret. The blind rotation can be used as a convenient way to implement a homomorphic look-up table with polynomials; see e.g., [Joy22]. When coupled with a bootstrapping operation, the homomorphic evaluation of a look-up table is also know as programmable bootstrapping [CJP21]. Since any univariate function (over a small domain) can always be expressed as a look-up table, an encryption scheme allowing homomorphic addition and homomorphic table look-up turns out to be fully homomorphic. Note that the regular bootstrapping (whose primary goal is to reduce the noise) corresponds to a programmable bootstrapping with the identity map for function f.

Several approaches are known for the blind rotation, with different trade-offs between vector coefficients, evaluation key material, and number of operations:

1. GINX algorithm [GINX16, CGGI20] comparatively features a small evaluation key material associated to secret vector $\boldsymbol{s}$; however requiring $\boldsymbol{s}$ to be *binary*, which may be restrictive.
2. AP algorithm [AP14, DM15], in its basic version, requires a huge amount of evaluation key material associated to secret vector $\boldsymbol{s}$ (exponential in the coefficients space of public vectors $\boldsymbol{a}$) but with basically no restriction on the distribution of $\boldsymbol{s}$.
3. Automorphism-based algorithms [BDF18, LMK+23, WWL+24], which ultimately are based on AP, decrease the global amount of evaluation key material thanks to homomorphic evaluations of automorphisms; although, each of these automorphism evaluations requires a key switch, inducing *in fine* a significant increase in the number of operations—and thus in the overall computational cost, as well as a significant noise growth.

Previous attempts to enhance GINX-type algorithms for using other distributions of the secret vector coefficients conclude to limit keys to small distributions (typically, binary distribution or ternary distribution) in order to avoid a blow-up of evaluation key material, thereby reducing their applicability; see e.g., [JP22].

Certain adaptations and extensions of the AP algorithm to various encryption schemes can be found in [XZD+23, MKMS24, LLW+24]. Again, all those improved, adapted or extended methods have in common to rely on the presence of automorphisms that can be homomorphically evaluated.

Our techniques and results The external product of ciphertexts is a fundamental operation in fully homomorphic encryption. It enables the multiplication of two ciphertexts that are not necessarily in the same format. Specifically, given a ciphertext $c \leftarrow \mathrm{Enc}_{\mathsf{sk}}(\mu)$ encrypting a plaintext μ under the scheme Enc, and another (extended) ciphertext $\bar{c} \leftarrow \mathrm{Enc}^{\circledast}_{\mathsf{sk}}(\bar{\mu})$ encrypting a plaintext $\bar{\mu}$ under an associated scheme $\mathrm{Enc}^{\circledast}$, the external product operation, denoted $\circledast$, produces a new ciphertext $c' = c \circledast \bar{c}$, which encrypts the product of the plaintexts, $\mu' = \mu \cdot \bar{\mu}$. One key application of the external product lies in enabling blind rotation for (programmable) bootstrapping. It also plays a critical role in advanced techniques such as batched bootstrapping [MKMS24] and circuit bootstrapping [WWL+24]. To optimize blind rotation, automorphism-based variants of the AP type focus on reducing the computational overhead introduced by the key switches incurred from the homomorphic evaluation of automorphisms.

We observe that in all of the automorphism-based methods, the homomorphic evaluation of automorphisms, including the associated key switch, is most of the time combined with an external product. Specifically, these methods require the encryption of $\psi(\mu) \cdot \bar{\mu}$ from the encryptions of μ and of $\bar{\mu}$, for some automorphism ψ. This process is typically carried out in three sequential steps, beginning with the ciphertexts $c \leftarrow \mathrm{Enc}_{\mathsf{sk}}(\mu)$ and $\bar{c} \leftarrow \mathrm{Enc}^{\circledast}_{\mathsf{sk}}(\bar{\mu})$. First, the automorphism ψ is applied to c, resulting in $c_1 \leftarrow \psi(c) \in \mathrm{Enc}_{\psi(\mathsf{sk})}(\psi(\mu))$. Next, a key switch operation is performed to transform c_1 into c_2, such that $c_2 \leftarrow \mathrm{KeySwitch}_{\psi(\mathsf{sk})\to\mathsf{sk}}(c_1)$, producing $\mathrm{Enc}_{\mathsf{sk}}(\psi(\mu))$. Finally, the transformed ciphertext c_2 is combined with $\bar{c}$ through an external product operation, yielding $c_3 \leftarrow c_2 \circledast \bar{c}$, which encrypts the product $\psi(\mu) \cdot \bar{\mu}$, i.e., $c_3 \leftarrow \mathrm{Enc}_{\mathsf{sk}}(\psi(\mu) \cdot \bar{\mu})$.

The main contribution of this paper is the introduction of the *Automorphism-Parametrized External Product*, a novel operation for FHE. This operation integrates three key steps—automorphism evaluation, its associated key switch, and an external product—at the computational cost of a *single* external product. Specifically, we define this new operator as follows:

$$\mathrm{Enc}_{\mathsf{sk}}(\mu) \circledast_{\psi} \mathrm{Enc}^{\circledast,\psi}_{\mathsf{sk}}(\bar{\mu}) \leftarrow \mathrm{Enc}_{\mathsf{sk}}(\psi(\mu) \cdot \bar{\mu}) \ ,$$

where $\mathrm{Enc}_{\mathsf{sk}}(\mu) \circledast_{\psi} \mathrm{Enc}^{\circledast,\psi}_{\mathsf{sk}}(\bar{\mu})$ is computed as $\psi(\mathrm{Enc}_{\mathsf{sk}}(\mu)) \circledast \mathrm{Enc}^{\circledast,\psi}_{\mathsf{sk}}(\bar{\mu})$. Here, $\mathrm{Enc}^{\circledast,\psi}_{\mathsf{sk}}(\bar{\mu})$ is a newly introduced ciphertext format, which we term *automorphism-extended* ciphertext. As will become evident in Section 4, this format naturally arises from the associated encryption scheme $\mathrm{Enc}^{\circledast}$; in particular, both formats coincide when the secret key is $\psi(\mathsf{sk})$ instead of sk. Notably, our new operator eliminates the need for a key switch. This presents a significant practical advantage, as key switches are known to introduce substantial computational overhead and add extra noise to the resulting ciphertext. It is also important to note that although the format of the second ciphertext, $\mathrm{Enc}^{\circledast,\psi}_{\mathsf{sk}}(\bar{\mu})$, has been modified, our new external product still outputs a regular $\mathrm{Enc}_{\mathsf{sk}}$ ciphertext as before. This ensures seamless integration in most applications that rely on the product in the exponent based on automorphisms, including blind rotation. Indeed, in such cases the second ciphertext is typically a global input (e.g., a bootstrapping key in the context of blind rotation).

Another contribution of this paper is an improved algorithm for blind rotation in FHE, termed the *Traversal Windowed Hörner-like Method* and detailed in Algorithm 3.2. Building on Algorithm 7 of [LMK+23], which has been reformulated in Algorithm 3.1 for clarity and easier adaptation, our new approach addresses gaps between two non-empty sets of mask values by incorporating sign changes directly into automorphism evaluations. This integration reduces the average gap size between automorphism applications, enhancing efficiency, particularly for smaller window sizes. Compared to the original method, our traversal algorithm achieves consistent reductions in the number of automorphism key switches, with typical gains of up to 8% depending on the parameters, though converging to comparable efficiency for larger windows. Detailed algorithms highlight the work-flow and key-switching optimizations.

When combined with our automorphism-parametrized external product, the proposed traversal method yields the so-called $\mathcal{S}$*-Parametrized Method*, formalized in Algorithm 4.1. Depending on a set $\mathcal{S}$ of admissible automorphisms that can be leveraged using the new parametrized external product, it provides flexible trade-offs between key size, which is linearly linked to $\sharp\mathcal{S}$, and performance and noise growth, which both improve when $\sharp\mathcal{S}$ grows, i.e., when the number of required key switches is lowered. To illustrate the benefits of these techniques, we experimentally evaluate its impact on several parameter sets:

- parameter sets from [LMK+23, LLW+24], designed for Gaussian keys and boolean messages, enabling direct comparisons with previous works;
- a parameter set specified in the `TFHE-rs` library [Zam22], primarily tailored for binary keys and 4-bit payloads, reflecting practical application scenarios.

With only a moderate increase in bootstrapping key sizes, as in [LLW+24], we reduce the number of key switches by approximately 49.1% (resp. 46.4%) compared to [LMK+23], significantly surpassing an adaptation of [LLW+24], which would achieve only an 18.5% (resp. 17.0%) reduction. By allowing a slightly larger increase in bootstrapping key sizes—2.5 times that of [LMK+23] (compared to 2 times in [LLW+24])—we achieve a 59.4% (resp. 55.5%) reduction in key switches relatively to [LLW+24] and 66.9% (resp. 63.1%) compared to [LMK+23].

Notably, the additional keys can be pre-selected according to the mask components, minimizing the impact on memory bandwidth. When the keys are just 9 times larger than those in [LMK+23], the number of key switches drops to as few as 2 or 3 on average, far outperforming AP bootstrapping which for comparable performance requires key that are more than 2 orders of magnitude larger.

In summary, our new techniques not only offer substantial reductions in key-switching overhead but also provide flexible trade-offs to meet varying performance and resource requirements. Although primarily presented for the TFHE cryptosystem, our methods and techniques are adaptable to other FHE schemes, such as FHEW [DM15] and FINAL [BIP+22], offering versatile and practical improvements for bootstrapping ciphertexts in FHE.

Finally, as a last contribution, we provide a thorough analysis of the complexity and noise growth of automorphism-based blind rotation algorithms, focusing on the impact of key switches and external products. We develop a theoretical framework that reduces the problem to studying the distribution of gaps in random divisions of an interval. Our new automorphism-parametrized methods demonstrate fewer key switches and improved efficiency compared to existing approaches. Numerical experiments validate these improvements, highlighting significant computational and practical benefits for bootstrapping algorithms.

Outline of the paper The rest of this paper is organized as follows. Section 2 reviews relevant background and introduces essential notations. Section 3 reformulates the [LMK+23] blind rotation algorithm using automorphisms and proposes an improved variant. Section 4 introduces a parametrized external product alongside a modified GLWE$^{\circledast}$-like ciphertext format and explores its application to the enhanced blind rotation algorithm. Finally, Section 5 presents the study framework and numerical experiments.

2 Definitions and Notations

Let q, $t < q$, and k be positive integers, and let $\Delta = \left\lfloor \frac{q}{t} \right\rfloor$. For the m-th cyclotomic polynomial Φ_m, define the cyclotomic field of conductor m, for $m \not\equiv 2 \bmod 4$, as $\mathcal{K} = \mathbb{Q}[x]/\langle \Phi_m(x) \rangle$. The degree of Φ_m is $N = \varphi(m)$, where φ is Euler's totient function. Common values for m include m a power of two, a prime $p > 2$, or of the form p^k, $4p^k$, $2^a 3^b$ [JW22]. Let also $\mathcal{R} = \mathbb{Z}[x]/\langle \Phi_m(x) \rangle$ be the ring of integers of $\mathcal{K}$, and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. The Galois group of $\mathcal{K}/\mathbb{Q}$ is isomorphic to $(\mathbb{Z}/m\mathbb{Z})^\times$ and consists of automorphisms τ_u defined by $\tau_u(x) = x^u$, for $u \in (\mathbb{Z}/m\mathbb{Z})^\times$; the identity automorphism τ_1 is also denoted as id.

GLWE ciphertexts GLWE stands for *generalized-LWE* and LWE refers to *Learning with Errors* [Reg09]. GLWE-type encryption appears for example in [SSTX09, LPR10, LS15]. Cleartext messages in a GLWE encryption scheme are polynomials in $\mathcal{R}$ with coefficients in $[\![0, t-1]\!]$. Prior to encryption, a cleartext message m is first encoded as a plaintext $\mu = \Delta \cdot m \in \mathcal{R}_q$. The GLWE encryption of $\mu \in \mathcal{R}_q$ under private key $\mathfrak{s} = (s_1, \ldots, s_k) \in \mathcal{R}^k$ is given by

$$\mathrm{GLWE}_{\mathfrak{s}}(\mu) \leftarrow \left(a_1, \ldots, a_k, b = \textstyle\sum_{j=1}^{k} a_j \cdot s_j + \mu + e\right) \quad \in \mathcal{R}_q^{k+1}$$

where $a_1, \ldots, a_k$ are polynomials randomly drawn in $\mathcal{R}_q$ and $e \in \mathcal{R}$ is some random noise polynomial with small coefficients. Vector $(a_1, \ldots, a_k)$ is called the mask and b is called the body. The error e present in $\mathrm{GLWE}_{\mathfrak{s}}(\mu)$ is denoted by $\mathrm{Err}(\mathrm{GLWE}_{\mathfrak{s}}(\mu))$.

Gadget-GLWE ciphertexts and extended-GLWE ciphertexts GGSW encryption generalizes the GSW encryption [GSW13]. Following the presentation of [MP21], the simplest way to view GGSW ciphertexts is through gadget decomposition of GLWE ciphertexts. Gadgets decompose elements as vectors of small pieces whose inner product with the gadget vector reconstructs (an approximation of) the original elements. Applied to a polynomial $f \in \mathcal{R}_q$, the gadget decomposition of f with respect to gadget vector $\boldsymbol{g} = (g_1, \ldots, g_\ell) \in \mathcal{R}_q^\ell$ is given by a vector $\nabla f \in \mathcal{R}^\ell$ such that $\langle \nabla f, \boldsymbol{g} \rangle \approx f$; see e.g., [CGGI20, BJ24]. When there is equality, the gadget decomposition is termed exact.

The corresponding *gadget-GLWE ciphertext* (indicated with a ∇ superscript) of a plaintext $\bar{\mu} \in \mathcal{R}_q$ under private key $\mathfrak{s} = (\mathfrak{s}_1, \ldots, \mathfrak{s}_k) \in \mathcal{R}^k$ is defined as

$$\mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(\bar{\mu}) \leftarrow \big(\mathrm{GLWE}_{\mathfrak{s}}(g_1 \cdot \bar{\mu}), \ldots, \mathrm{GLWE}_{\mathfrak{s}}(g_\ell \cdot \bar{\mu})\big) \ .$$

This leveled encryption gives rise to an *extended-GLWE ciphertext*

$$\mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu}) \leftarrow \big(\mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(-\mathfrak{s}_1 \cdot \bar{\mu}), \ldots, \mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(-\mathfrak{s}_k \cdot \bar{\mu}), \mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(\bar{\mu})\big) \ ,$$

whose definition coincides with the definition of a GGSW ciphertext.

External product A GLWE ciphertext $\mathrm{GLWE}_{\mathfrak{s}}(\mu)$ can be combined with an extended GLWE ciphertext $\mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu})$ to yield another GLWE ciphertext through *external product*. Specifically, their external product, denoted by $\circledast$, is defined as

$$\mathrm{GLWE}_{\mathfrak{s}}(\mu) \circledast \mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu}) := \\ \big\langle \nabla b, \mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(\bar{\mu}) \big\rangle + \textstyle\sum_{j=1}^{k} \big\langle \nabla a_j, \mathrm{GLWE}_{\mathfrak{s}}^{\nabla}(-\mathfrak{s}_j \cdot \bar{\mu}) \big\rangle \ .$$

In certain cases (e.g., [BCG+23, Theorem 3]), different gadget decomposition levels are used for the mask and the body of the ciphertext, denoted respectively by ℓ_1 and ℓ_2. This is indicated by writing the individual decompositions as ∇_{ℓ_1} and ∇_{ℓ_2}, with the overall decomposition given by $\nabla = \nabla_{\ell_1, \ell_2}$.

It can be verified that $\mathrm{GLWE}_{\mathfrak{s}}(\mu) \circledast \mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu}) \leftarrow \mathrm{GLWE}_{\mathfrak{s}}(\mu \cdot \bar{\mu})$, provided that (i) the gadget decomposition is sufficiently exact and (ii) $e \cdot \bar{\mu}$ is sufficiently small, where $e = \mathrm{Err}\big(\mathrm{GLWE}_{\mathfrak{s}}(\mu)\big)$ [CGGI20, Theorem 3.13 and Corollary 3.14].

Proposition 2.1. *Assume m is a power of two. Let $\nabla = \nabla_{\ell_1, \ell_2}$ be a gadget decomposition of quality $\beta_\nabla = (\beta_1, \beta_2)$ and precision $\varepsilon_\nabla = (\varepsilon_1, \varepsilon_2)$, whose output values are uniform and centered around 0. Let e_{in} and $\bar{\boldsymbol{e}}$ represent the error associated with valid samples $\mathrm{GLWE}_{\mathfrak{s}}(\mu)$ and $\mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu})$, respectively. Then, $\mathrm{GLWE}_{\mathfrak{s}}(\mu) \circledast \mathrm{GLWE}_{\mathfrak{s}}^{\circledast}(\bar{\mu})$ is a sample of $\mathrm{GLWE}_{\mathfrak{s}}(\mu \cdot \bar{\mu})$ with an error $\mathscr{E}$ of variance*

$$\sigma_{\circledast}^2 \leq \|\bar{\mu}\|_2^2 \cdot \sigma_{\mathrm{in}}^2 + N\Big(\ell_2 \tfrac{\beta_2^2}{12} + k\ell_1 \tfrac{\beta_1^2}{12}\Big) \cdot \sigma_\nabla^2 + \|\bar{\mu}\|_2^2 \Big(\tfrac{\varepsilon_2^2}{12} + kN \tfrac{\varepsilon_1^2}{12} \cdot \mathbb{E}[\mathfrak{s}_{j,i}^2]\Big) \ .$$

Proof. This is a special case of the generalized result presented in Proposition 4.3, obtained by taking $\psi = \mathrm{id}$ and $C_\infty = 1$. In particular, the exact expression of the error term $\mathscr{E} = \mathscr{B} - \langle \boldsymbol{\mathscr{A}}, \mathfrak{s} \rangle - \mu \cdot \bar{\mu}$ is given by

$$\mathscr{E} = \bar{\mu} \cdot e_{\mathrm{in}} + \Big(\big\langle \nabla_{\ell_2} b, \bar{\boldsymbol{e}}_0 \big\rangle + \sum_{1 \leq j \leq k} \big\langle \nabla_{\ell_1} a_j, \bar{\boldsymbol{e}}_j \big\rangle\Big) \\ + \bar{\mu} \cdot \Big(e_{\nabla_{\ell_2}}(b) - \sum_{1 \leq j \leq k} \mathfrak{s}_j \cdot e_{\nabla_{\ell_1}}(a_j)\Big) \ ,$$

where $e_{\nabla_{\ell_u}}(w) := \langle \nabla_{\ell_u} w, \boldsymbol{g}_u \rangle - w$ for $u \in \{1, 2\}$ and any $w \in \mathcal{R}_q$. $\square$

AP blind rotation An important application of the external product is the evaluation of an inner product in the exponent, or the related task of performing a *blind rotation*. Given an LWE ciphertext $\tilde{\boldsymbol{c}} = \big(\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}\big) \in (\mathbb{Z}/m\mathbb{Z})^{n+1}$ under a private key $\boldsymbol{s} = \big(s_1, \ldots, s_n\big)$ and a so-called test polynomial $\mathscr{v} \in \mathcal{R}_q$, the blind rotation consists in evaluating $x^{-\tilde{b}+\sum_{i=1}^{n} \tilde{a}_i s_i} \cdot \mathscr{v}(x)$ homomorphically. Additional key material known as bootstrapping keys is made available for the computation, namely the encryption of the key digits $s_1, \ldots, s_n$. In its generic version, the AP blind rotation requires a set of $n(m-1)$ bootstrapping keys,

$$\mathsf{bsk}^{\mathrm{AP}} := \Big\{ \mathsf{bsk}^{\mathrm{AP}}[i,u] \leftarrow \mathrm{GLWE}^{\circledast}_{\mathfrak{s}}(x^{u s_i}) \;\Big|\; i \in [\![1,n]\!] \text{ and } u \in [\![1,m-1]\!] \Big\} \;.$$

Let $\mathscr{q}_0 = x^{-\tilde{b}} \cdot \mathscr{v}$. The AP method makes use of an accumulator $\mathsf{ACC} \in \mathcal{R}_q^{k+1}$ that successively contains a GLWE encryption of $\mathscr{q}_i \leftarrow \mathscr{q}_{i-1} \cdot x^{\tilde{a}_i s_i} = x^{-\tilde{b}+\sum_{j=1}^{i} \tilde{a}_j s_j} \cdot \mathscr{v}$, for $i \in [\![1,n]\!]$. At the end of the iteration, the accumulator indeed contains a GLWE encryption of $x^{-\tilde{b}+\sum_{i=1}^{n} \tilde{a}_i s_i} \cdot \mathscr{v}$. In an algorithmic form, this writes as:

> $\mathsf{ACC} \leftarrow (0, \ldots, 0, x^{-\tilde{b}} \cdot \mathscr{v})$
> **for** $i = 1$ **to** n **do**
>   **if** $\tilde{a}_i \neq 0$ **then** $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AP}}[i, \tilde{a}_i]$
> **return** ACC

Note that the accumulator is initialized with $(0, \ldots, 0, x^{-\tilde{b}} \cdot \mathscr{v})$ which is a trivial GLWE encryption of $\mathscr{q}_0 = x^{-\tilde{b}} \cdot \mathscr{v}$, i.e., with an all-zero mask.

Automorphism-based methods The use of automorphisms aims at reducing the size of the additional key material in the blind rotation while containing the computational overhead. Without loss of generality, automorphism-based methods assume that each mask component $\tilde{a}_i$ of the input LWE ciphertext, $i \in [\![1,n]\!]$, is either 0 or belongs to $(\mathbb{Z}/m\mathbb{Z})^{\times}$, so that when non-zero, each indeed corresponds to an automorphism $\tau_{\tilde{a}_i} : x \mapsto x^{\tilde{a}_i}$. Different strategies from [BDF18, MKMS24, LMK+23, WWL+24] are detailed in Appendix A to reduce to this setting.

Given an automorphism $\tau_u : x \mapsto x^u$ for some unit $u \in (\mathbb{Z}/m\mathbb{Z})^{\times}$ and a GLWE ciphertext $\mathrm{GLWE}_{\mathfrak{s}}(\mu) \leftarrow \big(\mathscr{a}_1, \ldots, \mathscr{a}_k, \mathscr{b}\big) \in \mathcal{R}_q^{k+1}$ of plaintext $\mu := \mu(x)$ under private key $\mathfrak{s} = \big(\mathscr{s}_1, \ldots, \mathscr{s}_k\big)$, automorphism-based methods observe that

$$\tau_u\big(\mathrm{GLWE}_{\mathfrak{s}}(\mu)\big) \leftarrow \big(\tau_u(\mathscr{a}_1), \ldots, \tau_u(\mathscr{a}_k), \tau_u(\mathscr{b})\big) \in \mathrm{GLWE}_{\tau_u(\mathfrak{s})}(\tau_u(\mu))$$

is a GLWE ciphertext of plaintext $\tau_u(\mu) := \mu(x^u)$ under key $\tau_u(\mathfrak{s}) = (\tau_u(\mathscr{s}_1), \ldots, \tau_u(\mathscr{s}_k))$, provided that $\tau_u(\mathscr{e})$ with $\mathscr{e} = \mathrm{Err}\big(\mathrm{GLWE}_{\mathfrak{s}}(\mu)\big)$ stays sufficiently small. Now let $\mathsf{ak}_u := \mathsf{ksk}_{\tau_u(\mathfrak{s}) \to \mathfrak{s}}$ be an *automorphism key*, i.e., a key-switching key that converts a ciphertext under key $\tau_u(\mathfrak{s})$ back to a ciphertext under key $\mathfrak{s}$. Specifically, ak_u is defined as $\mathsf{ak}_u \leftarrow \big(\mathrm{GLWE}^{\nabla}_{\mathfrak{s}}(-\tau_u(\mathscr{s}_1)), \ldots, \mathrm{GLWE}^{\nabla}_{\mathfrak{s}}(-\tau_u(\mathscr{s}_k))\big)$. Then, a subsequent key-switching $\mathrm{KS}_{\mathsf{ak}_u}$ on $\tau_u\big(\mathrm{GLWE}_{\mathfrak{s}}(\mu)\big)$ with ak_u yields the ciphertext $\mathrm{GLWE}_{\mathfrak{s}}(\tau_u(\mu))$. The composition of these two operations, application of τ_u and key-switching $\mathrm{KS}_{\mathsf{ak}_u}$, will be denoted by $\mathrm{HomAut}_u(\cdot, \mathsf{ak}_u)$:

$$\mathrm{HomAut}_u\big(\mathrm{GLWE}_{\mathfrak{s}}(\mu), \mathsf{ak}_u\big) \leftarrow \mathrm{KS}_{\mathsf{ak}_u} \circ \tau_u(\mathrm{GLWE}_{\mathfrak{s}}(\mu)) \in \mathrm{GLWE}_{\mathfrak{s}}(\tau_u(\mu)) \;.$$

Correctness supposes that the resulting noise keeps below a certain threshold.

Proposition 2.2. *Assume m is a power of two. Let $\nabla_{\ell_{\mathrm{ks}}}$ be a gadget decomposition of quality β_{ks} and precision $\varepsilon_{\mathrm{ks}}$, whose output values are uniform and centered around 0. Let e_{in} and $\bar{\boldsymbol{e}}_{\mathrm{ks},j}$, where $j \in [\![1,k]\!]$, represent the error associated with valid samples $\mathrm{GLWE}_{\mathfrak{s}}(\mu)$ and $\mathrm{GLWE}^{\nabla}_{\mathfrak{s}}(-\tau_u(\mathfrak{s}_j))$, respectively. Then, $\mathrm{HomAut}_u(\mathrm{GLWE}_{\mathfrak{s}}(\mu), \mathsf{ak}_u)$ is a GLWE sample of $\mathrm{GLWE}_{\mathfrak{s}}(\tau_u(\mu))$ with error $\mathscr{E}$ of variance*

$$\sigma^2_{\mathrm{aut}} \leq \sigma^2_{\mathrm{in}} + N\left(k\ell_{\mathrm{ks}}\frac{\beta^2_{\mathrm{ks}}}{12}\right)\cdot\sigma^2_{\mathrm{ks}} + kN\left(\mathbb{E}[\mathfrak{s}^2]\cdot\frac{\varepsilon^2_{\mathrm{ks}}}{12}\right) .$$

Proof. The exact expression of the error term $\mathscr{E} = \mathscr{B} - \langle \boldsymbol{\mathscr{A}}, \boldsymbol{\mathfrak{s}}\rangle - \tau_u(\mu)$ is given by

$$\mathscr{E} = \tau_u(e_{\mathrm{in}}) + \sum_{1\leq j\leq k}\left\langle \nabla_{\ell_{\mathrm{ks}}}\tau_u(a_j), \bar{\boldsymbol{e}}_{\mathrm{ks},j}\right\rangle - \sum_{1\leq j\leq k}\tau_u(\mathfrak{s}_j)\cdot e_{\nabla_{\ell_{\mathrm{ks}}}}\big(\tau_u(a_j)\big) ,$$

where $e_{\nabla_{\ell_{\mathrm{ks}}}}(w) := \left\langle \nabla_{\ell_{\mathrm{ks}}} w, \boldsymbol{g}_{\mathrm{ks}}\right\rangle - w$ for any $w \in \mathcal{R}_q$. □

As described above, a loop iteration of the AP blind rotation consists in computing a GLWE ciphertext $\boldsymbol{c}^{(i)}$ of $q_i(x) = q_{i-1}(x)\cdot x^{\tilde{a}_i s_i}$ from a GLWE ciphertext $\boldsymbol{c}^{(i-1)}$ of $q_{i-1}(x)$. Using automorphisms as in [BDF18, Algorithms 5 and 6],[1] this can be achieved in three consecutive steps as

$$\begin{aligned}
&(1)\ \boldsymbol{c}^{(i)} \leftarrow \mathrm{HomAut}_{1/\tilde{a}_i}(\boldsymbol{c}^{(i-1)}, \mathsf{ak}_{1/\tilde{a}_i}) && \in \mathrm{GLWE}_{\mathfrak{s}}\big(q_{i-1}(x^{1/\tilde{a}_i})\big)\\
&(2)\ \boldsymbol{c}^{(i)} \leftarrow \boldsymbol{c}^{(i)} \circledast \mathrm{GLWE}^{\circledast}_{\mathfrak{s}}(x^{s_i}) && \in \mathrm{GLWE}_{\mathfrak{s}}\big(q_{i-1}(x^{1/\tilde{a}_i})\cdot x^{s_i}\big)\\
&(3)\ \boldsymbol{c}^{(i)} \leftarrow \mathrm{HomAut}_{\tilde{a}_i}(\boldsymbol{c}^{(i)}, \mathsf{ak}_{\tilde{a}_i}) && \in \mathrm{GLWE}_{\mathfrak{s}}\big(q_{i-1}(x)\cdot x^{\tilde{a}_i s_i}\big)
\end{aligned}$$

where the inverses are taken modulo m. Define $J_0 = \{i \in [\![1,n]\!] \mid \tilde{a}_i \neq 0\}$. The resulting blind rotation algorithm (depicted in Figure 2.1(a)) requires the following key material:

$$\mathsf{bsk}^{\mathrm{AUT}} := \left\{\mathsf{bsk}^{\mathrm{AUT}}[i] \leftarrow \mathrm{GLWE}^{\circledast}_{\mathfrak{s}}(x^{s_i}) \;\middle|\; i \in [\![1,n]\!]\right\} \tag{2.1}$$

and

$$\mathsf{ak}^{\mathrm{AUT}} := \left\{\mathsf{ak}^{\mathrm{AUT}}[u] \leftarrow \mathsf{ksk}_{\tau_u(\mathfrak{s})\to\mathfrak{s}} \;\middle|\; u \in (\mathbb{Z}/m\mathbb{Z})^{\times}\setminus\{1\}\right\} . \tag{2.2}$$

The number of homomorphic automorphism evaluations must not be overlooked as each involves a key switch. As noted e.g., in [XZD+23, Algorithm 1] or [MKMS24, Algorithm 2, inner loop], the two automorphisms of each loop iteration can be combined together, halving the number of required key switches. This trick is applied and detailed in Figure 2.1(b) (telescoping method). We stress that when $u = 1$, HomAut_u is the identity map and so is always skipped. Hence, by re-arranging J_0 the $\tilde{a}_i$'s can be regrouped by values, so that if the set $\{\tilde{a}_i \mid i \in [\![1,n]\!]\}\setminus\{0,1\}$ has cardinality α, at most $\alpha < \varphi(m)$ homomorphic automorphism evaluations and key switches are performed.

[1] Originally in the circulant setting, however conceptually it is exactly the same.

$\mathsf{ACC} \leftarrow (0, \ldots, 0, x^{-\tilde{b}} \cdot v)$
for $i \in J_0$ **do**
  $u \leftarrow 1/\tilde{a}_i \bmod m$
  $\mathsf{ACC} \leftarrow \mathrm{HomAut}_u(\mathsf{ACC}, \mathsf{ak}^{\mathrm{AUT}}[u])$
  $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AUT}}[i]$
  $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{\tilde{a}_i}(\mathsf{ACC}, \mathsf{ak}^{\mathrm{AUT}}[\tilde{a}_i])$
return ACC

(a) Basic method.

$\tilde{a}_{\mathrm{old}} \leftarrow 1, \quad \mathsf{ACC} \leftarrow (0, \ldots, 0, x^{-\tilde{b}} \cdot v)$
for $i \in J_0$ **do**
  $u \leftarrow \tilde{a}_{\mathrm{old}}/\tilde{a}_i \bmod m, \quad \tilde{a}_{\mathrm{old}} \leftarrow \tilde{a}_i$
  $\mathsf{ACC} \leftarrow \mathrm{HomAut}_u(\mathsf{ACC}, \mathsf{ak}^{\mathrm{AUT}}[u])$
  $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AUT}}[i]$
$u \leftarrow \tilde{a}_{\mathrm{old}}$
$\mathsf{ACC} \leftarrow \mathrm{HomAut}_u(\mathsf{ACC}, \mathsf{ak}^{\mathrm{AUT}}[u])$
return ACC

(b) Telescoping method.

Fig. 2.1: Automorphism-based methods.

Table 2.2: Generic vs. automorphism-based methods.

Blind rotation (BR)	Key material ($\sharp$GLWE$^{\triangledown}$)	$\sharp$Key switches
Generic AP method	$n(k+1)(m-1)$	n/a
Telescoping method	$n(k+1) + k(\varphi(m) - 1)$	$\min\{n, \varphi(m)\}$

Compared to the generic, non automorphism-based AP blind rotation presented earlier, the telescoping method already behaves much nicer; see Table 2.2. The key material drops from $n(m-1)$ $\mathsf{GLWE}^{\circledast}$ ciphertexts (i.e., $n(k+1)(m-1)$ $\mathsf{GLWE}^{\triangledown}$ ciphertexts) to n $\mathsf{GLWE}^{\circledast}$ ciphertexts plus $k(\varphi(m)-1)$ $\mathsf{GLWE}^{\triangledown}$ ciphertexts. Computation-wise, at most $\min\{n, \varphi(m)\}$ extra key switches are required; the number of external products nevertheless remains equal to n.

3 Enhanced Blind Rotation Algorithms

In this section, we present a Hörner-like method for the blind rotation, which is a minor variation of [LMK+23, Algorithm 7]. Our reformulation primarily offers the advantage of providing a simpler basis for analyzing the results of Section 4. We also propose a new method derived from it—the *traversal windowed Hörner-like method*—which consistently outperforms [LMK+23]. These methods aim to reduce both the size of the keys and the number of automorphism applications.

3.1 Windowed Hörner-like Method

The idea of re-arranging the mask components $\tilde{a}_i \in (\mathbb{Z}/m\mathbb{Z})^{\times}$ has been extended in [LMK+23, Section 3], where the $\tilde{a}_i$'s are not only regrouped by values, but also ordered according to the group structure of $(\mathbb{Z}/m\mathbb{Z})^{\times}$. The immediate consequence is a reduction of the number of automorphism keys down to the number of cyclic components of $(\mathbb{Z}/m\mathbb{Z})^{\times}$.

As an illustration, following [LMK+23], we focus on the power-of-two conductor case.[2] Let $m = 2N$ with $N = 2^\nu$, $\nu \geq 2$, so that $(\mathbb{Z}/m\mathbb{Z})^\times = \langle -1 \rangle \times \langle g \rangle$ using e.g., $g = 5$. Then, every element $\tilde{a}_i \in (\mathbb{Z}/m\mathbb{Z})^\times$ can be written as

$$\tilde{a}_i = (-1)^{\epsilon_i} \cdot g^{t_i} \mod m, \qquad \epsilon_i \in \{0, 1\},\ 0 \leq t_i < 2^{\nu-1} .$$

The high-level core idea of [LMK+23, Section 3.1] can be alternatively expressed as adapting the ordering of the mask components so as to ensure that the quotient $\tilde{a}_{\text{old}}/\tilde{a}_i$ in the telescoping method (Figure 2.1(b)) is always a small power of g. In order to formalize this, it is useful to introduce the sets

$$I_t^+ \coloneqq \left\{ i \in [\![1, n]\!] \;\middle|\; \tilde{a}_i = g^t \mod 2N \right\} \tag{3.1}$$

and

$$I_t^- \coloneqq \left\{ i \in [\![1, n]\!] \;\middle|\; \tilde{a}_i = -g^t \mod 2N \right\} . \tag{3.2}$$

Then the computation of $\langle \tilde{\boldsymbol{a}}, \boldsymbol{s} \rangle = \sum_{i=1}^n \tilde{a}_i s_i$ can be reordered as

$$\sum_{i=1}^{n} (-1)^{\epsilon_i} g^{t_i} \cdot s_i = \sum_{t=0}^{N/2-1} g^t \cdot \left(\textstyle\sum_{i \in I_t^+} s_i \right) - \sum_{t=0}^{N/2-1} g^t \cdot \left(\textstyle\sum_{i \in I_t^-} s_i \right) , \tag{3.3}$$

which is naturally computed in a Hörner-like fashion as

$$\begin{aligned} &\textstyle\sum_{i \in I_0^+} s_i + g \Bigg(\sum_{i \in I_1^+} s_i + \cdots + g \Bigg(\sum_{i \in I_{N/2-1}^+} s_i \\ &\qquad - g \Bigg(\sum_{i \in I_0^-} s_i + g \Big(\sum_{i \in I_1^-} s_i + \cdots + g \big(\sum_{i \in I_{N/2-1}^-} s_i \big) \Big) \Bigg) \Bigg) \Bigg) . \end{aligned} \tag{3.4}$$

This results in Algorithm 3.1, which is an almost (see Remark 3.1) equivalent rewriting of [LMK+23, Algorithm 7], reorganized so as to follow the work-flow of Figure 2.1(b). Thus, starting from $\tilde{a}_{\text{old}} = -1 = -g^{N/2} \mod 2N$, it iterates on non-empty sets $I_t^\pm$ so that for $i \in I_t^\pm$, $\tilde{a}_{\text{old}}/\tilde{a}_i$ is always the smallest possible power of g. Gaps, when some of the $I_t^\pm$ are empty, are filled using a jumping strategy defined by a window size w and automorphism keys

$$\begin{aligned} \mathsf{ak}^{\text{HORN}} \coloneqq \left\{ \mathsf{ak}^{\text{HORN}}[0] \leftarrow \mathsf{ksk}_{\tau_{-g}(\mathfrak{s}) \to \mathfrak{s}} \right\} \cup& \\ \left\{ \mathsf{ak}^{\text{HORN}}[u] \leftarrow \mathsf{ksk}_{\tau_{g^u}(\mathfrak{s}) \to \mathfrak{s}} \;\middle|\; u \in [\![1, w]\!] \right\}& . \end{aligned} \tag{3.5}$$

Using $w = 1$ simply corresponds to repeated applications of τ_g, one per empty set $I_t^\pm$; a larger value for w decreases the number of calls to HomAut() at the expense of larger key material. Practical experiments from [LMK+23, Figure 3] suggest a rather small optimal window value of $w = 10$.

[2] Although only the power-of-two conductor case is treated in [LMK+23], their core result readily applies to e.g., the simpler case of prime-conductor cyclotomic fields.

Algorithm 3.1: Blind rotation w/automorphisms — Windowed Hörner

Input: $\tilde{\boldsymbol{c}} \leftarrow (\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$, $\tilde{a}_i \in (\mathbb{Z}/2N\mathbb{Z})^\times \cup \{0\}$; $\boldsymbol{v} \in \mathcal{R}_q$
Data: $\mathsf{bsk}^{\mathrm{AUT}}$ and $\mathsf{ak}^{\mathrm{HORN}}$ as defined in Equations (2.1) and (3.5) for a window size $w \geq 1$
Output: $\boldsymbol{c} \leftarrow \mathrm{GLWE}_{\boldsymbol{s}}(x^{-\tilde{\mu}} \cdot \boldsymbol{v}) \in \mathcal{R}_q^{k+1}$ with $\tilde{\mu} = \tilde{b} - \sum_{i=1}^{n} \tilde{a}_i s_i$

1. $t_{\mathrm{old}} \leftarrow 0, \quad \mathsf{ACC} \leftarrow \big(0, \ldots, 0, x^{\tilde{b}} \cdot \boldsymbol{v}(x^{-1})\big)$ /* i.e., $\tilde{a}_{\mathrm{old}} = -g^{N/2} = -1$ */
2. **for** $t = N/2 - 1$ **down to** 0 **such that** $I_t^- \neq \emptyset$ **do**
3. $\quad \delta \leftarrow (t_{\mathrm{old}} - t) \bmod N/2, \quad t_{\mathrm{old}} \leftarrow t$
   /* Homomorphically apply τ_{g^δ} using jumps of size at most w */
4. $\quad q_\delta \leftarrow \lfloor \delta / w \rfloor, \quad r_\delta \leftarrow \delta \bmod w$
5. $\quad$ **for** q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^w}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[w]\big)$
6. $\quad$ **if** $r_\delta \neq 0$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{r_\delta}}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[r_\delta]\big)$
   /* Compute all external products for I_t^- */
7. $\quad$ **for** $i \in I_t^-$ **do**
8. $\quad\quad \mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AUT}}[i]$

/* Apply τ_{-g} as in second line of Equation (3.4), see Remark 3.1 */
9. $t_{\mathrm{old}} \leftarrow \big(t_{\mathrm{old}} - 1\big) \bmod N/2, \quad \mathsf{ACC} \leftarrow \mathrm{HomAut}_{-g}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[0]\big)$

/* Same loop as the first loop, but for (non-empty) sets I_t^+ */
10. **for** $t = N/2 - 1$ **down to** 0 **such that** $I_t^+ \neq \emptyset$ **do**
11. $\quad \delta \leftarrow (t_{\mathrm{old}} - t) \bmod N/2, \quad t_{\mathrm{old}} \leftarrow t$
12. $\quad q_\delta \leftarrow \lfloor \delta / w \rfloor, \quad r_\delta \leftarrow \delta \bmod w$
13. $\quad$ **for** q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^w}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[w]\big)$
14. $\quad$ **if** $r_\delta \neq 0$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{r_\delta}}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[r_\delta]\big)$
15. $\quad$ **for** $i \in I_t^+$ **do**
16. $\quad\quad \mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AUT}}[i]$
17. $\delta \leftarrow t_{\mathrm{old}}, \quad q_\delta \leftarrow \lfloor \delta / w \rfloor, \quad r_\delta \leftarrow \delta \bmod w$
18. **for** q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^w}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[w]\big)$
19. **if** $r_\delta \neq 0$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{r_\delta}}\big(\mathsf{ACC}, \mathsf{ak}^{\mathrm{HORN}}[r_\delta]\big)$
20. **return** ACC

Remark 3.1. In [LMK⁺23], a "flush" homomorphically applying τ_{g^δ} is required *before* moving to the second loop, as shown by the condition "*[...] or $\ell = 1$*" in [LMK⁺23, Algorithm 3, Line 7]. Our rewriting and proof (Appendix B) makes visible that τ_{-g} can be applied at any time after handling the last non-empty set I_t^-, whereas filling the gap with τ_{g^δ} can be deferred to the second loop, simply adjusting t_{old} as in Line 9. This often saves one HomAut() evaluation.

3.2 A New Traversal Windowed Hörner-like Method

Since n is typically small w.r.t. $2N$, many sets I_t^+ or I_t^- are empty, which implies that $I_t \coloneqq I_t^- \cup I_t^+$ is often equal to either I_t^+ or I_t^-. This suggests a better strategy to enumerate $(\mathbb{Z}/2N\mathbb{Z})^\times$, where closing the gap between two non-empty sets I_t's

Algorithm 3.2: BR w/automorphisms — Traversal Windowed Hörner

Input: $\tilde{\mathbf{c}} \leftarrow (\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$, $\tilde{a}_i \in (\mathbb{Z}/2N\mathbb{Z})^{\times} \cup \{0\}$; $\mathfrak{v} \in \mathcal{R}_q$
Data: $\mathsf{bsk}^{\mathrm{AUT}}$ and $\mathsf{ak}^{\mathrm{TRAV}}$ as defined in Equations (2.1) and (3.6) for a window size $w = 2w'$
Output: $\mathfrak{c} \leftarrow \mathrm{GLWE}_{\mathfrak{s}}(x^{-\tilde{\mu}} \cdot \mathfrak{v}) \in \mathcal{R}_q^{k+1}$ with $\tilde{\mu} = \tilde{b} - \sum_{i=1}^{n} \tilde{a}_i s_i$

$\epsilon_{\mathrm{old}} \leftarrow +1$, $t_{\mathrm{old}} \leftarrow N/2$, $\mathsf{ACC} \leftarrow \left(0, \ldots, 0, x^{-\tilde{b}} \cdot \mathfrak{v}(x)\right)$
for $t = N/2 - 1$ **down to** 0 **do**
  /* First consider same sign as ϵ_{old}, then flip if $I_t^{-\epsilon_{\mathrm{old}}}$ is not empty */
  for $\epsilon \in \left\{\epsilon_{\mathrm{old}}, -\epsilon_{\mathrm{old}}\right\}$ **such that** $I_t^{\epsilon} \neq \emptyset$ **do**
    /* Compute $\sigma \cdot g^{\delta} = \epsilon_{\mathrm{old}} \cdot g^{t_{\mathrm{old}}}/(\epsilon \cdot g^{t})$, update tracking values */
    $\delta \leftarrow t_{\mathrm{old}} - t$, $t_{\mathrm{old}} \leftarrow t$, $\sigma \leftarrow \epsilon_{\mathrm{old}}/\epsilon$, $\epsilon_{\mathrm{old}} \leftarrow \epsilon$
    /* Apply τ_u for $u = \sigma \cdot g^{\delta}$, see Remark 3.2 */
    if $\delta = 0$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{-1}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[0]\right)$
    else
      Write $\delta = q_\delta \cdot w' + r_\delta$ with $r_\delta \in [\![1, w']\!]$ and $q_\delta \geq 0$
      for q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{w'}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[w']\right)$
      $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{\sigma \cdot g^{r_\delta}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[\sigma \cdot r_\delta]\right)$
    /* Compute all external products for I_t^{ϵ} */
    for $i \in I_t^{\epsilon}$ **do**
      $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}^{\mathrm{AUT}}[i]$
/* Finally, apply τ_u for $u = \epsilon_{\mathrm{old}} \cdot g^{t_{\mathrm{old}}}$ */
if $t_{\mathrm{old}} = 0$ **and** $\epsilon_{\mathrm{old}} = -1$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{-1}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[0]\right)$
else if $t_{\mathrm{old}} \neq 0$ **then**
  Write $\delta = t_{\mathrm{old}} = q_\delta \cdot w' + r_\delta$ with $r_\delta \in [\![1, w']\!]$ and $q_\delta \geq 0$
  for q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{w'}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[w']\right)$
  $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{\epsilon_{\mathrm{old}} \cdot g^{r_\delta}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathrm{TRAV}}[\epsilon_{\mathrm{old}} \cdot r_\delta]\right)$
return ACC

can be directly combined with a sign change. The main effect is to reduce $\mathbb{E}\left[\delta\right]$, the average size of the gaps. As will be demonstrated, this enhances efficiency, especially for smaller window sizes.

Doing so yields Algorithm 3.2, which exchanges the loops of Algorithm 3.1 by including the sign change, whenever needed, directly inside a unique loop on t. The required automorphism keys are defined as

$$\mathsf{ak}^{\mathrm{TRAV}} := \left\{\mathsf{ak}^{\mathrm{TRAV}}[0] \leftarrow \mathsf{ksk}_{\tau_{-1}(\mathfrak{s}) \to \mathfrak{s}}\right\} \cup \left\{\mathsf{ak}^{\mathrm{TRAV}}[\pm u] \leftarrow \mathsf{ksk}_{\tau_{\pm g^u}(\mathfrak{s}) \to \mathfrak{s}} \;\middle|\; u \in [\![1, w']\!]\right\} . \tag{3.6}$$

Remark 3.2. We chose to always combine a possible sign change with the application of $\tau_{g^{r_\delta}}$. When $\delta \neq 0$, it implies to modify the definition of q_δ to $\left\lfloor \frac{\delta - 1}{w'} \right\rfloor$ and $r_\delta = \delta - q_\delta \cdot w'$ to ensure $r_\delta \in [\![1, w']\!]$ so that $\mathsf{ak}^{\mathrm{TRAV}}[\sigma \cdot r_\delta]$ exists. Further, the case $\delta = 0$ can only occur when $\epsilon = -\epsilon_{\mathrm{old}}$, i.e., $\sigma = -1$ inside the loop.

Proposition 3.3. *Algorithm 3.2 is correct.*

Proof. For $t \in [\![0, N/2]\!]$, define recursively $\tilde{q}_t \in \mathcal{R}_q$ by $\tilde{q}_{N/2} = v \cdot x^{-\tilde{b}}$ and

$$\tilde{q}_t = \tilde{q}_{t+1} \cdot x^{g^t \cdot \left(\sum_{i \in I_t^+} s_i - \sum_{i \in I_t^-} s_i\right)} \ .$$

We inductively show that after each iteration, ACC contains a GLWE encryption of $\tau_{\epsilon_{\text{old}} \cdot g^{-t_{\text{old}}}}(\tilde{q}_t)$.

First, ACC is initialized to a (trivial) GLWE encryption of $x^{-\tilde{b}} \cdot v(x)$, which is indeed equal to $\tau_{+g^{N/2}}(\tilde{q}_{N/2})$. Inside the loop, the induction hypothesis is preserved as long as both I_t^+ and I_t^- are empty, since in this case $\tilde{q}_t = \tilde{q}_{t_{\text{old}}}$. Assume now, for $N/2 > t \geq 0$, that ACC contains a GLWE encryption of $\tau_{\epsilon_{\text{old}} \cdot g^{-t_{\text{old}}}}(\tilde{q}_{t+1})$, and exactly one I_t^ϵ is non-empty for $\epsilon \in \{\epsilon_{\text{old}}, -\epsilon_{\text{old}}\}$. Then, $\tau_{\sigma \cdot g^\delta}$ $(\delta \geq 1)$ is homomorphically applied to ACC to obtain a GLWE encryption of $\tau_{\epsilon g^{-t}}(\tilde{q}_{t+1})$, and the following external products finally yield as expected a GLWE encryption of

$$\tau_{\epsilon g^{-t}}(\tilde{q}_{t+1}) \cdot x^{\sum_{i \in I_t^\epsilon} s_i} = \tau_{\epsilon g^{-t}}\left(\tilde{q}_{t+1} \cdot x^{g^t \sum_{i \in I_t^\epsilon} \epsilon s_i}\right) = \tau_{\epsilon g^{-t}}(\tilde{q}_t) \ .$$

When both $I_t^\pm$ are non-empty, the previous reasoning is first applied on $I_t^{\epsilon_{\text{old}}}$; then for $\epsilon = -\epsilon_{\text{old}}$, we have $\delta = 0$ so τ_{-1} is applied to ACC and external products corresponding to indices in I_t^ϵ finally yield a GLWE encryption of

$$\begin{aligned} &\tau_{-\epsilon_{\text{old}} \cdot g^{-t}}\left(\tilde{q}_{t+1} \cdot x^{\epsilon_{\text{old}} \cdot g^t \sum_{i \in I_t^{\epsilon_{\text{old}}}} s_i}\right) \cdot x^{\sum_{i \in I_t^\epsilon} s_i} \\ &\qquad = \tau_{\epsilon g^{-t}}\left(\tilde{q}_{t+1} \cdot x^{g^t \left(\sum_{i \in I_t^\epsilon} \epsilon s_i + \sum_{i \in I_t^{-\epsilon}} (-\epsilon) s_i\right)}\right) = \tau_{\epsilon g^{-t}}(\tilde{q}_t) \ , \end{aligned}$$

which again is the induction hypothesis with Line 4 adjustments on ϵ_{old} and t_{old}.

Hence, after the loop, ACC contains a GLWE encryption of $\tau_{\epsilon_{\text{old}} \cdot g^{-t_{\text{old}}}}(\tilde{q}_0)$, implying the result since the last lines of the algorithm apply $\tau_{\epsilon_{\text{old}} \cdot g^{t_{\text{old}}}}$. □

For completeness, we provide in Tables 3.1(a) and 3.1(b) a numerical comparison of the expected number of automorphism key switches in Algorithm 3.1 (similar to [LMK+23, Algorithm 7]) vs. Algorithm 3.2 under two different parameter sets. The first set is taken from [LMK+23, Table 2] and the second set is `PARAM_MESSAGE_2_CARRY_2_KS_PBS_GAUSSIAN_2M64`[3] from `TFHE-rs` [Zam22], with parameters $(n, N, k) = (458, 1024, 1)$ and $(n, N, k) = (834, 2048, 1)$, respectively. For an easier comparison, we assume $w = 2w'$ so that in both cases the key material includes $w + 1$ automorphism keys. These tables show that, although both methods roughly converge to the same optimum, the new traversal method always shows superior performance due to its ability to combine smaller jumps on average with sign changes; in particular, for small values of $w = 2w'$.

[3] Git commit `400ce27beb5bea8fdc68826ad437099ec62680d0`, Sept. 25, 2024.

Table 3.1: Measured expected number of automorphism key switches for Algorithms 3.1 and 3.2 under two different parameters sets.

(a) For $n = 458$ and $N = 1024$.

$w = 2w'$	Alg. 3.1	Alg. 3.2	Ratio (%)
$w = 2$	625	578	92.5
$w = 4$	445	431	96.7
$w = 6$	396	390	98.4
$w = 8$	380	377	99.3
$w = 10$	373	371	99.5
$w = 12$	372	370	99.6
$w = 14$	372	370	99.7

(b) For $n = 834$ and $N = 2048$.

$w = 2w'$	Alg. 3.1	Alg. 3.2	Ratio (%)
$w = 2$	1232	1139	92.5
$w = 4$	857	827	96.5
$w = 6$	755	743	98.4
$w = 8$	721	715	99.2
$w = 10$	702	698	99.4
$w = 12$	698	696	99.7
$w = 14$	692	691	99.8

4 Automorphism-Parametrized Techniques

We propose a generalized external product that seamlessly incorporates a homomorphic automorphism evaluation on the input ciphertext. Specifically, by using a *modified* $\mathsf{GLWE}^{\circledast}$-like ciphertext, we show how the key switch associated with the homomorphic automorphism evaluation can be absorbed within the external product. In essence, our new operator enables in a single step and at the cost of a single external product, the execution of a homomorphic automorphism evaluation—including its key switch—followed by an external product.

4.1 Automorphism-Parametrized External Product

We generalize the extended-GLWE ($\mathsf{GLWE}^{\circledast}$, aka. GGSW) ciphertexts, which are used for computing external products, to embed information about the image of the key under a given automorphism ψ; i.e., $\psi = \tau_u$ for a fixed $u \in (\mathbb{Z}/m\mathbb{Z})^{\times}$.

Definition 4.1. *An* Automorphism-Extended-GLWE *ciphertext relatively to automorphism ψ and to gadget decomposition $\nabla = \nabla_{\ell_1,\ell_2}$ of a plaintext $\bar{\mathfrak{p}} \in \mathcal{R}_q$ under key $\mathfrak{s} \in \mathcal{R}_q^k$ is denoted by $\mathrm{GLWE}_{\mathfrak{s}}^{\circledast,\psi}(\bar{\mathfrak{p}})$ and defined as*

$$\left\{\mathrm{GLWE}_{\mathfrak{s}}^{\nabla_{\ell_1}}\left(-\psi(\mathfrak{s}_1)\cdot\bar{\mathfrak{p}}\right),\ldots,\mathrm{GLWE}_{\mathfrak{s}}^{\nabla_{\ell_1}}\left(-\psi(\mathfrak{s}_k)\cdot\bar{\mathfrak{p}}\right),\mathrm{GLWE}_{\mathfrak{s}}^{\nabla_{\ell_2}}\left(\bar{\mathfrak{p}}\right)\right\} \ .$$

In particular, for a given decomposition ∇, it holds that $\mathrm{GLWE}_{\mathfrak{s}}^{\circledast,\mathrm{id}} = \mathrm{GLWE}_{\mathfrak{s}}^{\circledast}$.

Such ciphertexts $\mathrm{GLWE}^{\circledast,\psi}$ enable the combination of a homomorphic evaluation of ψ on a GLWE input with an external product, *without* requiring an intermediate key switch, as demonstrated below.

Definition 4.2 (Automorphism-parametrized external product). *The* automorphism-parametrized external product*, relatively to the automorphism ψ and gadget decomposition $\nabla = \nabla_{\ell_1,\ell_2}$, is denoted by $\circledast_\psi$ and defined as*

$$\circledast_\psi : \mathrm{GLWE}_{\mathfrak{s}}\left(\mathfrak{p}\right) \times \mathrm{GLWE}_{\mathfrak{s}}^{\circledast,\psi}\left(\bar{\mathfrak{p}}\right) \longrightarrow \mathrm{GLWE}_{\mathfrak{s}}\left(\psi(\mathfrak{p})\cdot\bar{\mathfrak{p}}\right) \ ,$$

where, for $\mathrm{GLWE}_{\mathfrak{s}}(\mu) = (a_1, \ldots, a_k, b)$*, the result is computed as*

$$\left\langle \nabla_{\ell_2}\psi(b), \mathrm{GLWE}^{\nabla_{\ell_2}}_{\mathfrak{s}}(\bar{\mu}) \right\rangle + \sum_{j=1}^{k} \left\langle \nabla_{\ell_1}\psi(a_j), \mathrm{GLWE}^{\nabla_{\ell_1}}_{\mathfrak{s}}(-\psi(\mathfrak{s}_k)\cdot\bar{\mu}) \right\rangle \ .$$

In particular, for a given decomposition ∇*,* $\circledast_{\mathrm{id}}$ *coincides with* $\circledast$*.*

The noise associated to this new operation is given by Proposition 4.3, which essentially highlights a gain due to the removal of the key switch. It depends on a constant C_∞ which is set to 1 when m is a power of two. In the general case where m is not a power of two, $C_\infty > 1$ corresponds to the expansion factor of $\Phi_m(x)$, e.g., $C_\infty = 2$ when m is a prime $p > 2$ [MKMS24] or $m = 3^a$ [JW22], and $C_\infty = 4$ when $m = 2^b 3^a$ for $b \geq 2$ [JW22]. Notably, for any automorphism ψ and all $w \in \mathcal{R}_q$, it holds that $\left\|\psi(w)\right\|_\infty \leq C_\infty \cdot \left\|w\right\|_\infty$.

Proposition 4.3. *Let* $\nabla = \nabla_{\ell_1,\ell_2}$ *be a gadget decomposition of quality* $\beta_\nabla = (\beta_1, \beta_2)$ *and precision* $\varepsilon_\nabla = (\varepsilon_1, \varepsilon_2)$*, whose output values are uniform and centered around* 0*. Let* e_{in} *and* $\bar{\boldsymbol{e}}$ *represent the error associated with valid samples* $\mathrm{GLWE}_{\mathfrak{s}}(\mu)$ *and* $\mathrm{GLWE}^{\circledast,\psi}_{\mathfrak{s}}(\bar{\mu})$*, respectively. Then* $\mathrm{GLWE}_{\mathfrak{s}}(\mu) \circledast_\psi \mathrm{GLWE}^{\circledast,\psi}_{\mathfrak{s}}(\bar{\mu})$ *is a sample of* $\mathrm{GLWE}_{\mathfrak{s}}(\psi(\mu)\cdot\bar{\mu})$ *with an error* $\mathscr{E}$ *of variance*

$$\sigma^2_{\circledast_\psi} \leq C_\infty \cdot \left(\left\|\bar{\mu}\right\|_2^2 \cdot \sigma^2_{\mathrm{in}} + N\left(\ell_2 \tfrac{\beta_2^2}{12} + k\ell_1 \tfrac{\beta_1^2}{12}\right)\cdot\sigma^2_\nabla + \left\|\bar{\mu}\right\|_2^2 \left(\tfrac{\varepsilon_2^2}{12} + kN\tfrac{\varepsilon_1^2}{12}\cdot\mathbb{E}[\mathfrak{s}^2_{j,i}]\right)\right) \ .$$

Proof. Let $\mathbf{L}_0 \in \mathcal{R}_q^{k\times\ell_2}$ s.t. $\mathrm{GLWE}^{\nabla_{\ell_2}}_{\mathfrak{s}}(\bar{\mu}) = (\mathbf{L}_0, \boldsymbol{b}_0)$, i.e., the ℓ_2 columns of $\mathbf{L}_0$ are the respective masks of each $\mathrm{GLWE}_{\mathfrak{s}}(g_{2,i}\bar{\mu})$, and $\boldsymbol{b}_0 = \mathfrak{s}\cdot\mathbf{L}_0 + \bar{\mu}\cdot\boldsymbol{g}_2 + \bar{\boldsymbol{e}}_0$. Similarly, let for $j \in [\![1,k]\!]$, $\mathbf{L}_j \in \mathcal{R}_q^{k\times\ell_1}$ s.t. $\mathrm{GLWE}^{\nabla_{\ell_1}}_{\mathfrak{s}}(-\psi(\mathfrak{s}_j)\cdot\bar{\mu}) = (\mathbf{L}_j, \boldsymbol{b}_j)$, where $\boldsymbol{b}_j = \mathfrak{s}\cdot\mathbf{L}_j - \psi(\mathfrak{s}_j)\cdot\bar{\mu}\cdot\boldsymbol{g}_1 + \bar{\boldsymbol{e}}_j$. For $\mathrm{GLWE}_{\mathfrak{s}}(\mu) = (\boldsymbol{a}, b)$, the resulting ciphertext of the above-defined operation is $(\boldsymbol{\mathscr{A}}, \mathscr{B})$ with

$$\begin{cases} \boldsymbol{\mathscr{A}} = \nabla_{\ell_2}\psi(b)\cdot\mathbf{L}_0^\intercal + \sum_{1\leq j\leq k}\nabla_{\ell_1}\psi(a_j)\cdot\mathbf{L}_j^\intercal & \in \mathcal{R}_q^k \\ \mathscr{B} = \left\langle\nabla_{\ell_2}\psi(b), \boldsymbol{b}_0\right\rangle + \sum_{1\leq j\leq k}\left\langle\nabla_{\ell_1}\psi(a_j), \boldsymbol{b}_j\right\rangle & \in \mathcal{R}_q \end{cases} .$$

Rearranging terms after expanding the definition of the $\boldsymbol{b}_j$'s yields

$$\begin{aligned}\mathscr{B} = \left\langle\boldsymbol{\mathscr{A}}, \mathfrak{s}\right\rangle + \bar{\mu}\cdot\Big(\left\langle\nabla_{\ell_2}\psi(b), \boldsymbol{g}_2\right\rangle - \textstyle\sum_{1\leq j\leq k}\psi(\mathfrak{s}_j)\cdot\left\langle\nabla_{\ell_1}\psi(a_j), \boldsymbol{g}_1\right\rangle\Big) \\ + \Big(\left\langle\nabla_{\ell_2}\psi(b), \bar{\boldsymbol{e}}_0\right\rangle + \textstyle\sum_{1\leq j\leq k}\left\langle\nabla_{\ell_1}\psi(a_j), \bar{\boldsymbol{e}}_j\right\rangle\Big) ,\end{aligned}$$

hence the exact expression of the error term $\mathscr{E} = \mathscr{B} - \left\langle\boldsymbol{\mathscr{A}}, \mathfrak{s}\right\rangle - \psi(\mu)\cdot\bar{\mu}$ is given by

$$\begin{aligned}\mathscr{E} = \bar{\mu}\cdot\psi(e_{\mathrm{in}}) + \Big(\left\langle\nabla_{\ell_2}\psi(b), \bar{\boldsymbol{e}}_0\right\rangle + \textstyle\sum_{1\leq j\leq k}\left\langle\nabla_{\ell_1}\psi(a_j), \bar{\boldsymbol{e}}_j\right\rangle\Big) \\ + \bar{\mu}\cdot\Big(e_{\nabla_{\ell_2}}(\psi(b)) - \textstyle\sum_{1\leq j\leq k}\psi(\mathfrak{s}_j)\cdot e_{\nabla_{\ell_1}}(\psi(a_j))\Big)\end{aligned}$$

where $e_{\nabla_{\ell_u}}(w) := \left\langle \nabla_{\ell_u} w, \boldsymbol{g}_u \right\rangle - w$ for $u \in \{1, 2\}$ and any $w \in \mathcal{R}_q$.

Polynomials output by the decomposition are supposed uniform and centered around 0, independently of ψ, and uncorrelated with error polynomials $\bar{\boldsymbol{e}}_j$. Thus, the variance of e.g., $\left\langle \nabla_{\ell_1} \psi(a_j), \bar{\boldsymbol{e}}_j \right\rangle$ is upper-bounded by $\ell_1 \cdot C_\infty N \cdot \frac{\beta_1^2}{12} \sigma_\nabla^2$. Similarly, the variance of the coefficients of $\bar{\mathfrak{p}} \cdot \psi(e_{\text{in}})$ is given by $\left\|\bar{\mathfrak{p}}\right\|_2^2 \cdot C_\infty \sigma_{\text{in}}^2$. By hypothesis, the coefficients of $e_{\nabla_{\ell_u}}(w)$ have variance upper-bounded by $\frac{\varepsilon_u^2}{12}$. Specifically, for the terms of the form $\bar{\mathfrak{p}} \cdot \psi(\mathfrak{s}_j) \cdot e_{\nabla_{\ell_1}}(\psi(a_j))$, the final variance is $\left\|\bar{\mathfrak{p}}\right\|_2^2 \cdot C_\infty \left(\sigma_{\mathfrak{s}_j}^2 + \mathbb{E}[\mathfrak{s}_j]^2\right) \cdot N \frac{\varepsilon_1^2}{12}$. Bringing all these components together yields the final result. $\square$

4.2 Reducing Automorphism Key Switches in Blind Rotation

We now demonstrate how this new automorphism-parametrized external product can be utilized to reduce the number of key switches in the Traversal Windowed Hörner method.

In Algorithm 3.2, the transition from $I_{t_{\text{old}}}^{\epsilon_{\text{old}}}$ to I_t^{ϵ} involves a homomorphic automorphism evaluation, which includes at least one automorphism key switch. Roughly speaking, the new operation combines this homomorphic automorphism evaluation with the first external product in I_t^{ϵ}. In effect, this narrows the distance between two consecutive sets by eliminating the associated key switch.

For each gap addressed in this manner, e.g., corresponding to an automorphism $\psi = \tau_{\pm g^\delta}$, additional $\mathrm{GLWE}^{\circledast,\psi}$-ciphertexts are required. This creates a trade-off between the size of the keys (i.e., the size of the set of admissible gaps) and reduced performance and increased noise growth (i.e., more key switches). However, as shown in Section 5, around 63% of the gaps in Algorithm 3.2 are covered by $\{\tau_{-1}, \tau_{\pm g}\}$ and around 84% by $\{\tau_{-1}, \tau_{\pm g}, \tau_{\pm g^2}\}$.

Formal description Let $\mathcal{S}$ denote the set of admissible automorphisms, and assume that $\mathrm{id} \in \mathcal{S}$.[4] For convenience, a dual set $\mathcal{S}_* := \left\{(\delta, \epsilon) \mid \epsilon \in \{\pm 1\}, \tau_{\epsilon \cdot g^\delta} \in \mathcal{S}\right\}$ is defined, which encodes $\mathcal{S}$ in $(\mathbb{Z}/2N\mathbb{Z})^\times$. In particular, by the assumption on $\mathcal{S}$, it follows that $(0, 1) \in \mathcal{S}_*$. The associated keys are defined by

$$\mathsf{bsk}^{\mathcal{S}\text{-AUT}} := \left\{ \mathsf{bsk}_\psi^{\mathcal{S}\text{-AUT}}[i] \leftarrow \mathrm{GLWE}_{\mathfrak{s}}^{\circledast,\psi}(x^{s_i}) \;\middle|\; i \in [\![1, n]\!], \psi \in \mathcal{S} \right\}, \tag{4.1}$$

and

$$\mathsf{ak}^{\mathcal{S}\text{-AUT}} := \left\{ \mathsf{ak}^{\mathcal{S}\text{-AUT}}[0] \leftarrow \mathsf{ksk}_{\tau_{-1}(\mathfrak{s}) \to \mathfrak{s}} \right\} \cup \left\{ \mathsf{ak}^{\mathcal{S}\text{-AUT}}[\pm u] \leftarrow \mathsf{ksk}_{\tau_{\pm g^u}(\mathfrak{s}) \to \mathfrak{s}} \;\middle|\; u \in [\![1, w'']\!] \right\}. \tag{4.2}$$

For a fixed $i \in [\![1, n]\!]$, it is important to note that $\mathsf{bsk}_\psi^{\mathcal{S}\text{-AUT}}[i]$, $\psi \in \mathcal{S}$, *all* share the common term $\mathrm{GLWE}_{\mathfrak{s}}^{\nabla_{\ell_2}}(x^{s_i})$. Although this term is repeated for notational clarity, it implies that the size of each $\mathsf{bsk}^{\mathcal{S}\text{-AUT}}[i]$ is $(\sharp\mathcal{S} \cdot k + 1)$ Gadget-GLWE ciphertexts, rather than $\sharp\mathcal{S} \cdot (k+1)$ Gadget-GLWE ciphertexts.

[4] While not strictly necessary, including id in $\mathcal{S}$ always yields similar or better tradeoffs and simplifies both the presentation and the description of automorphism keys.

Algorithm 4.1: Blind Rotation: $\mathcal{S}$-parametrized method

Input: $\tilde{\mathfrak{c}} \leftarrow (\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$, $\tilde{a}_i \in (\mathbb{Z}/2N\mathbb{Z})^\times \cup \{0\}$; $\mathfrak{v} \in \mathcal{R}_q$
Data: $\mathsf{bsk}^{\mathcal{S}\text{-AUT}}$ and $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ as defined in Equations (4.1) and (4.2) for a window size w'' and a set $\mathcal{S}$ of admissible automorphisms
Output: $\mathfrak{c} \leftarrow \mathrm{GLWE}_{\mathfrak{s}}(x^{-\tilde{\mu}} \cdot \mathfrak{v}) \in \mathcal{R}_q^{k+1}$ with $\tilde{\mu} = \tilde{b} - \sum_{i=1}^{n} \tilde{a}_i s_i$

$\epsilon_{\text{old}} \leftarrow +1$, $\quad t_{\text{old}} \leftarrow N/2$, $\quad \mathsf{ACC} \leftarrow \left(0, \ldots, 0, x^{-\tilde{b}} \cdot \mathfrak{v}(x)\right)$ /* see Remark 4.4 */
for $t = N/2 - 1$ **down to** 0 **such that** $I_t^+ \cup I_t^- \neq \emptyset$ **do**
    $\epsilon_{\text{first}} \leftarrow \epsilon_{\text{old}}$ **if** $(t_{\text{old}} - t, +1) \in \mathcal{S}_*$ **else** $-\epsilon_{\text{old}}$ /* see Remark 4.5 */
    for $\epsilon \in \left\{\epsilon_{\text{first}}, -\epsilon_{\text{first}}\right\}$ **such that** $I_t^\epsilon \neq \emptyset$ **do**
        /* Compute $\sigma \cdot g^\delta = \epsilon_{\text{old}} \cdot g^{t_{\text{old}}}/(\epsilon \cdot g^t)$, update tracking values */
        $\delta \leftarrow t_{\text{old}} - t$, $\quad t_{\text{old}} \leftarrow t$, $\quad \sigma \leftarrow \epsilon_{\text{old}}/\epsilon$, $\quad \epsilon_{\text{old}} \leftarrow \epsilon$
        /* Jumping strategy: find $(\delta_*, \epsilon_*) \in \mathcal{S}_*$ alphabetically closest to (δ, σ) */
        $\delta_* \leftarrow \max\left\{(\delta_*, \cdot) \in \mathcal{S}_* \,\middle|\, \delta_* \leq \delta\right\}$, $\quad \epsilon_* \leftarrow \sigma$ **if** $(\delta_*, \sigma) \in \mathcal{S}_*$ **else** $-\sigma$
        /* Jumping strategy: apply τ_v for $v = \sigma \cdot g^\delta/(\epsilon_* \cdot g^{\delta_*})$ */
        if $(\delta - \delta_*, \sigma/\epsilon_*) = (0, -1)$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{-1}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[0]\right)$
        else if $(\delta - \delta_*) \neq 0$ **then**
            Write $\delta - \delta_* = q_\delta \cdot w'' + r_\delta$ with $r_\delta \in [\![1, w'']\!]$ and $q_\delta \geq 0$
            for q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{w''}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[w'']\right)$
            $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{\sigma/\epsilon_* \cdot g^{r_\delta}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[\sigma/\epsilon_* \cdot r_\delta]\right)$
        /* Jumping strategy: first external product parametrized by $\psi = \tau_{\epsilon_* \cdot g^{\delta_*}}$ */
        $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast_\psi \mathsf{bsk}_\psi^{\mathcal{S}\text{-AUT}}\left[I_t^\epsilon[0]\right]$
        /* Compute all remaining external products for I_t^ϵ */
        for $i \in I_t^\epsilon \setminus \{I_t^\epsilon[0]\}$ **do**
            $\mathsf{ACC} \leftarrow \mathsf{ACC} \circledast \mathsf{bsk}_{\text{id}}^{\mathcal{S}\text{-AUT}}[i]$
/* Finally, apply τ_u for $u = \epsilon_{\text{old}} \cdot g^{t_{\text{old}}}$ (see Remark 4.6) */
if $\epsilon_{\text{old}} = -1$ **then** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{-1}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[0]\right)$
if $t_{\text{old}} \neq 0$ **then**
    Write $t_{\text{old}} = q_\delta \cdot w'' + r_\delta$ with $r_\delta \in [\![1, w'']\!]$ and $q_\delta \geq 0$
    for q_δ times **do** $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{w''}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[w'']\right)$
    $\mathsf{ACC} \leftarrow \mathrm{HomAut}_{g^{r_\delta}}\left(\mathsf{ACC}, \mathsf{ak}^{\mathcal{S}\text{-AUT}}[r_\delta]\right)$
return ACC

The new jumping strategy is formalized in Algorithm 4.1. For a gap $u = \sigma \cdot g^\delta$, the approach involves finding the closest pair $(\delta_*, \epsilon_*) \in \mathcal{S}_*$, going forward, and thereafter decomposing the automorphism τ_u as $\psi \circ \tau_v$, where $\psi = \tau_{\epsilon_* \cdot g^{\delta_*}} \in \mathcal{S}$ and $v = u \cdot (\epsilon_* \cdot g^{\delta_*})^{-1}$. When the component τ_v is non-trivial, it is homomorphically applied to ACC with the windowed method as in Algorithm 3.2, which requires at least one automorphism key switch, whereas $\psi \in \mathcal{S}$ is applied as part of the new automorphism-parametrized external product $\circledast_\psi$.

Remark 4.4. As with any other automorphism-based blind rotation, the first automorphism evaluation on ACC is entirely free. Therefore, in practice, knowing the first (t_0, ϵ_0) in the loop s.t. $I_{t_0}^{\epsilon_0} \neq \emptyset$, we can modify the initialization step by directly setting $\epsilon_{\text{old}} = \epsilon_0$, $t_{\text{old}} = t_0$ and $\mathsf{ACC} = \left(0, \ldots, 0, x^{-u\tilde{b}} \cdot \mathfrak{v}(x^u)\right)$ for $u = \epsilon_0 \cdot g^{-t_0}$.

Remark 4.5. In the specific case where both $I_t^{\pm}$ are non-empty, $\delta_* = t_{\text{old}} - t$, *and* the set $\mathcal{S}$ is not symmetric, i.e., it contains τ_u for some $\pm u = g^{t_{\text{old}} - t}$ but does not necessarily include τ_{-u}, it is preferable to first consider the sign ϵ_{first} that will lead to the automorphism in $\mathcal{S}$, which is not always ϵ_{old}, as done in Line 3.

Remark 4.6. It is also worth noting that, when $\mathcal{S} \setminus \left\{\tau_{\pm 1}\right\}$ is symmetric, then during the main loop only non-negative indices of $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ are necessary. Consequently, we modified the final steps of Algorithm 4.1, in particular Line 15, in order to ensure they also only require $\mathsf{ak}^{\mathcal{S}\text{-AUT}}[0] \cup \mathsf{ak}^{\mathcal{S}\text{-AUT}}[1 \ldots w'']$. Thus, in many cases, $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ can be made twice shorter than indicated in Equation (4.2).

Proposition 4.7. *Algorithm 4.1 is correct.*

Proof. The iteration invariant is the same as in Algorithm 3.2, i.e., after iteration t, ACC contains a GLWE encryption of $\tau_{\epsilon_{\text{old}} \cdot g^{-t_{\text{old}}}}\left(\tilde{\mathfrak{q}}_t\right)$ under key $\mathfrak{s}$. The output is $\text{HomAut}_u(\mathsf{ACC})$ for $u = \epsilon_{\text{old}} \cdot g_{\text{old}}^t$, which yields $\text{GLWE}_{\mathfrak{s}}\left(\mathfrak{v} \cdot x^{\langle \boldsymbol{a}, \boldsymbol{s} \rangle}\right)$ by induction. Note that the modified inner loop initialization (see Remark 4.5) only modifies adaptively its order, but has no impact on correctness. $\square$

Noise analysis The noise growth of Algorithm 4.1 is directly linked to the number of external products (whether automorphism-parametrized or not), which is always n, and the number $\kappa(w'')$ of remaining automorphism key switches, which gets smaller as $\sharp\mathcal{S}$ grows.

Proposition 4.8. *Let $\kappa(w'')$ be the number of automorphism key switches required by Algorithm 4.1. Then, using the same notations and gadget decompositions hypotheses as in Propositions 4.3 and 2.2, the error term of the output of Algorithm 4.1 has variance*

$$\sigma^2_{\mathcal{S}\text{-AUT}} \leq n \cdot \mathfrak{s}^2_{\circledast_{\mathcal{S}}} + \kappa(w'') \cdot \mathfrak{s}^2_{\text{aut}} \ ,$$

$$\textit{where} \begin{cases} \mathfrak{s}^2_{\circledast_{\mathcal{S}}} \leq C_\infty\left(N\left(\ell_2 \frac{\beta_2^2}{12} + k\ell_1 \frac{\beta_1^2}{12}\right) \cdot \sigma^2_{\nabla} + \left(\frac{\varepsilon_2^2}{12} + kN \frac{\varepsilon_1^2}{12} \cdot \mathbb{E}[\mathfrak{s}^2_{j,i}]\right)\right) \\ \mathfrak{s}^2_{\text{aut}} \leq C_\infty\left(N\left(k\ell_{\text{ks}} \frac{\beta^2_{\text{ks}}}{12}\right) \cdot \sigma^2_{\text{ks}} + kN\left(\mathbb{E}[\mathfrak{s}^2_{j,i}] \cdot \frac{\varepsilon^2_{\text{ks}}}{12}\right)\right) \ . \end{cases}$$

Proof. We first prove the result in the (unlikely) case $\kappa(w'') = 0$, i.e., when the algorithm consists of a series of (automorphism-parametrized or not) external products $\circledast_{\psi_t}$, $t \in [\![1, n]\!]$,where $\psi_t = \tau_{u_t} \in \mathcal{S}$ and $\psi_n \circ \cdots \circ \psi_1 = \text{id}$. Without any loss of generality, we assume that the LWE key indexes have been reordered so that the t-th operation $\circledast_{\psi_t}$ involves $\mathsf{bsk}_{\psi_t}^{\mathcal{S}\text{-AUT}}[t] = \text{GLWE}_{\mathfrak{s}}^{\circledast, \psi_t}\left(x^{s_t}\right)$.

We proceed by expressing directly the final error term.[5] Let $\mathscr{E}_t$ be the error term of $\mathsf{ACC} = \big(a_1, \ldots, a_k, b\big)$ after $\circledast_{\psi_t}$. Using the same notations as in the proof of Proposition 4.3, $\mathscr{E}_t = x^{s_t} \cdot \psi_t\big(\mathscr{E}_{t-1}\big) + \big(E_{\circledast}^{(t)} + x^{s_t} \cdot E_{\nabla,\psi_t}^{(t)}\big)$, where

$$\begin{cases} E_{\circledast}^{(t)} = \big\langle \nabla_{\ell_2} \psi_t(b), \bar{\boldsymbol{e}}_0^{(t)} \big\rangle + \sum_{1 \le j \le k} \big\langle \nabla_{\ell_1} \psi_t(a_j), \bar{\boldsymbol{e}}_j^{(t,\psi_t)} \big\rangle \\ E_{\nabla,\psi_t}^{(t)} = e_{\nabla_{\ell_2}}\big(\psi_t(b)\big) - \sum_{1 \le j \le k} \psi_t\big(s_j\big) \cdot e_{\nabla_{\ell_1}}\big(\psi_t(a_j)\big) \ , \end{cases}$$

with $e_{\nabla_{\ell_u}}(w) \coloneqq \big\langle \nabla_{\ell_u} w, \boldsymbol{g}_u \big\rangle - w$ for $u \in \{1,2\}$ and any $w \in \mathcal{R}_q$. A simple induction then yields, from $\mathscr{E}_0 = 0$ and $\mathscr{E}_1 = E_{\circledast}^{(1)}$, i.e., $E_{\nabla,\psi_1}^{(1)} = 0$,

$$\mathscr{E}_n = \sum_{t=1}^{n} \Big(\prod_{a=t+1}^{n} x^{s_a u_{a+1} \cdots u_n} \Big) \cdot \tau_{u_n \cdots u_{t+1}} \Big(E_{\circledast}^{(t)} + x^{s_t} \cdot E_{\nabla,\psi_t}^{(t)} \Big) \ .$$

We continue by looking at the case of one single automorphism key switch. For a given $t \in [\![1, n]\!]$, assume $\psi_t \notin \mathcal{S}$ can be decomposed as $\psi_t^* \circ \tau_{v_t}$ where $\psi_t^* \in \mathcal{S}$ and $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ contains a key corresponding to $\tau_{v_t} \neq \mathrm{id}$. From Proposition 2.2, the noise term after $\mathrm{HomAut}_{\tau_{v_t}}$ is then given by $\tau_{v_t}\big(\mathscr{E}_{t-1}\big) + \big(E_{\mathrm{ks}}^{(t)} + E_{\nabla_{\mathrm{ks}},\tau_{v_t}}^{(t)}\big)$, where

$$\begin{cases} E_{\mathrm{ks}}^{(t)} = \sum_{1 \le j \le k} \big\langle \nabla_{\ell_{\mathrm{ks}}} \tau_{v_t}(a_j), \bar{\boldsymbol{e}}_{\mathrm{ks},j}^{(v_t)} \big\rangle \\ E_{\nabla_{\mathrm{ks}},\tau_{v_t}}^{(t)} = -\sum_{1 \le j \le k} \tau_{v_t}\big(s_j\big) \cdot e_{\nabla_{\ell_{\mathrm{ks}}}}\big(\tau_{v_t}(a_j)\big) \ , \end{cases}$$

with $e_{\nabla_{\ell_{\mathrm{ks}}}}(w) \coloneqq \big\langle \nabla_{\ell_{\mathrm{ks}}} w, \boldsymbol{g}_{\mathrm{ks}} \big\rangle - w$ for any $w \in \mathcal{R}_q$. After applying the $\circledast_{\psi_t^*}$ operation, the formula for obtaining $\mathscr{E}_t$ from $\mathscr{E}_{t-1}$ becomes (note the ψ_t^* stars)

$$\mathscr{E}_t = x^{s_t} \cdot \psi_t\big(\mathscr{E}_{t-1}\big) + \big(E_{\circledast}^{(t)} + x^{s_t} \cdot E_{\nabla,\psi_t^*}^{(t)}\big) + x^{s_t} \cdot \psi_t^*\big(E_{\mathrm{ks}}^{(t)} + E_{\nabla_{\mathrm{ks}},\tau_{v_t}}^{(t)}\big) \ .$$

In the general case, let $\mathcal{A} = \big\{ t \in [\![1, n]\!] \ \big| \ \psi_t \notin \mathcal{S} \big\}$. Then, for any fixed $t \in \mathcal{A}$, Algorithm 4.1 decomposes ψ_t as $\psi_t^* \circ \tau_{v_t} \circ \tau_{g^{w''}}^{q_t}$ where $\psi_t^* \in \mathcal{S}$, $\tau_{v_t} \neq \mathrm{id}$ ($\Leftrightarrow v_t \neq 1$), and $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ contains a key corresponding to τ_{v_t}. Adapting the previous discussion to the case $q_t > 0$, it is easy to verify that for such $t \in \mathcal{A}$ we have

$$\mathscr{E}_t = x^{s_t} \cdot \psi_t\big(\mathscr{E}_{t-1}\big) + E_{\circledast}^{(t)} + x^{s_t} \cdot E_{\nabla,\psi_t^*}^{(t)} + x^{s_t} \cdot \psi_t^*\big(\mathcal{E}_{\mathrm{ks}}^{(t)}\big) \ ,$$

where $\mathcal{E}_{\mathrm{ks},v_t,q_t}^{(t)} \coloneqq \big(E_{\mathrm{ks}}^{(t)} + E_{\nabla_{\mathrm{ks}},\tau_{v_t}}^{(t)}\big) + \sum_{a=1}^{q_t} \tau_{v_t} \tau_{g^{w''}}^{a-1}\big(E_{\mathrm{ks}}^{(t,a)} + E_{\nabla_{\mathrm{ks}},\tau_{g^{w''}}}^{(t,a)}\big)$ captures the errors from the required automorphism key switches when $t \in \mathcal{A}$. By abuse of notation, we let $\psi_t^* = \psi_t$ also when $t \notin \mathcal{A}$, i.e., when $\psi_t \in \mathcal{S}$. This allows us to prove by induction that the final error term of ACC is given by

$$\mathscr{E}_n = \sum_{t=1}^{n} \Big(\prod_{a=t+1}^{n} x^{s_a u_{a+1} \cdots u_n} \Big) \cdot \tau_{u_{t+1} \cdots u_n} \Big(E_{\circledast}^{(t)} + x^{s_t} \cdot E_{\nabla,\psi_t^*}^{(t)} + \mathbb{1}_{t \in \mathcal{A}} \cdot x^{s_t} \cdot \psi_t^*\big(\mathcal{E}_{\mathrm{ks},v_t,q_t}^{(t)}\big) \Big) \ .$$

[5] In the general case where $C_\infty > 1$, applying n times Proposition 4.3 would result in an artificially large factor C_∞^n in the upper bound. Indeed, as opposed to the claim of [MKMS24, Page 11], applying an automorphism alone does affect the error by a factor C_∞.

It remains to bound the variance of $\mathscr{E}_n$ from this closed form.

First of all, the $E_{\circledast}^{(t)}$'s (resp. $E_{\mathrm{ks}}^{(t,a)}$) can be considered as independent random samples of a random variable $E_{\circledast}$ (resp. E_{ks}), since the output distribution of the gadget decomposition is independent of the automorphism ψ_t^*, and the decomposed elements are combined with the bootstrapping keys errors. The same argument applies for the decomposition error terms $e_{\nabla_{\ell_u}}(w)$ for any $w \in \mathcal{R}_q$ and $u \in \{1, 2, \mathrm{ks}\}$, however the images of the GLWE key $\psi(s_j)$ are not independent when t varies. In order to deal with this, we rewrite each $\psi(s_j) \cdot e_{\nabla_{\ell_u}}(w)$ as $\psi\left(s_j \cdot \psi^{-1}\left(e_{\nabla_{\ell_u}}(w)\right)\right)$. The second key point is to notice that *all* subsequent automorphism applications (resp. multiplications by some power of x) are actually permutations (resp. rotations) of the error coefficients modulo $x^m - 1$, so that their variance is only multiplied once by C_∞ when reducing modulo Φ_m.

Thus, it is sufficient to bound the variance of $\langle \nabla_u w, \bar{e} \rangle$ and $s_j \cdot \psi\left(e_{\nabla_{\ell_u}}(w)\right)$, for any $u \in \{1, 2, \mathrm{ks}\}$ and $w \in \mathcal{R}_q$. The result now follows from arguments similar to those in the proof of Proposition 4.3. $\square$

Comparison with [LLW+24] In [LLW+24, Section 4], the authors describe an improved automorphism-based blind rotation, originally for NTRU and identified as "merging the symmetric sets." Their method doubles the size of the keys, using both $\mathrm{GLWE}_{s}^{\circledast}(x^{\pm s_i})$ for each $i \in [\![1, n]\!]$. Algorithmically, this corresponds to the traversal method where the loop on $\epsilon \in \{\pm\epsilon_{\mathrm{old}}\}$ is replaced by a choice of operand in the external product conditioned on ϵ. This removes all homomorphic applications of complex conjugation in the traversal method, which account for approximately 18% of the total number of the key switches when using an optimal window size. By contrast, the new $\mathcal{S}$-parametrized method with $\mathcal{S} = \{\mathrm{id}, \tau_{-1}\}$ also allows removing all complex conjugations, thus using less keys to achieve the same performance level. For instance, when $k = 1$, it needs $(2k + 1) \cdot n = 3n$ Gadget-GLWE ciphertexts, whereas following the approach of [LLW+24] would require $2(k + 1) \cdot n = 4n$ Gadget-GLWE ciphertexts. More notably, setting $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}\}$ allows removing about 46% of the key switches from [LMK+23] for about the same key size as what is needed in [LLW+24] to remove only 18% of those. This highlights that our automorphism-parametrized external product provides a more effective solution than simply generalizing the approach in [LLW+24].

Furthermore, aiming for a method similar to the $\mathcal{S}$-parametrized approach with $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}\}$, generalizing the approach of [LLW+24] would require not only the keys $\mathrm{GLWE}^{\circledast}(x^{\pm s_i})$ as in [LLW+24], but also at least additional keys $\mathrm{GLWE}^{\circledast}(x^{\pm g \cdot s_i})$. For $k = 1$, this already amounts to $8n$ Gadget-GLWE ciphertexts vs. $5n$ for the $\mathcal{S}$-parametrized method. However, even with this allowed, it still does not reach the performance of the $\mathcal{S}$-parametrized method: since no automorphism is applied during the external products, the generalized method cannot handle more than two *consecutive* gaps of type $\pm g$ without key switch. On average in this setting, our measurements show that this generalization brings only 60% of the performance gains provided by the $\mathcal{S}$-parametrized method.

5 Analysis and Experiments

The complexity and noise analysis of the algorithms of this paper primarily reduce to evaluating the number of automorphism key switches performed, in addition to the n external products (whether automorphism-parametrized or not).

In previous works, this has been achieved using a rather loose worst-case upper bound (derived from) [LMK+23, Section 4.1], and with Monte Carlo simulations as in [WWL+24, Section 4.2] or [LLW+24, Figure 2]. In this work, we propose a theoretical framework for assessing the performance of our new automorphism-parametrized blind rotation, as well as prior automorphism-based algorithms. This framework is thoroughly validated through numerical experiments.

5.1 On Random Divisions of an Interval

We propose to reduce the problem of evaluating the number of automorphism key switches to analyzing the distribution of gaps in a *random cut* (with repetitions) of $[\![0, B]\!]$,[6] approximated by the continuous case $\left[0, B\right]$.

Consider n uniformly random variables $X_1, \ldots, X_n$ sampled from $\left[0, B\right]$, and denote their ordered values by $X_{(i)}$, such that $0 \leq X_{(1)} \leq \cdots \leq X_{(n)} \leq B$. For convenience, define $X_{(0)} = 0$ and $X_{(n+1)} = B$, capturing the starting and final points of the blind rotation loop. A *random cut* of $\left[0, B\right]$ is given by the $(n+1)$ random gaps $\Delta_i = X_{(i+1)} - X_{(i)}$, for $i \in [\![0, n]\!]$, constrained by $\sum_{i=0}^{n} \Delta_i = B$.

Average maximum distance In this context (see e.g., [DN03, Section 6.4]), the joint probability density function of $\Delta_{i_1}, \ldots, \Delta_{i_r}$, for any $r \in [\![1, n]\!]$ and choice of the i_j's, is known to be, for $\sum d_{i_j} \leq B$,

$$f\left(d_{i_1}, \ldots, d_{i_r}\right) = \frac{n!}{B^r (n-r)!} \cdot \left(1 - \frac{d_{i_1} + d_{i_2} + \cdots + d_{i_r}}{B}\right)^{n-r} .$$

This yields, by integrating r-times on $0 \leq \sum_{j=1}^{r} c_j \leq B$ [DN03, Equation 6.4.3],

$$\Pr\left[\Delta_{i_1} \geq c_1, \ldots, \Delta_{i_r} \geq c_r\right] = \left(1 - \frac{\sum_{j=1}^{r} c_j}{B}\right)^n .$$

The probability of the maximum to be greater than $c \leq B$ is given by the *union* of all events $\Delta_i \geq c$. By the inclusion/exclusion principle, this writes

$$\Pr\Big[\max_{0 \leq i \leq n}\left\{\Delta_i\right\} \geq c\Big] = \sum_{\substack{1 \leq u \leq n+1 \\ \text{s.t. } uc \leq B}} (-1)^{u-1} \cdot \binom{n+1}{u} \left(1 - \frac{uc}{B}\right)^n . \tag{5.1}$$

[6] For instance, $B = N$ in Algorithm 3.1 and $B = N/2$ in Algorithms 3.2 and 4.1.

The expectation is obtained by integrating this over all possible values of c, i.e.,

$$\begin{aligned}\mathbb{E}\Big[\max_{0\le i\le n}\{\Delta_i\}\Big] &= \sum_{u=1}^{n+1}(-1)^{u-1}\cdot\binom{n+1}{u}\int_0^{B/u}\Big(1-\frac{ux}{B}\Big)^n \mathrm{d}x \\ &= \frac{B}{n+1}\cdot\sum_{u=1}^{n+1}\binom{n+1}{u}\frac{(-1)^{u-1}}{u} = \frac{B}{n+1}\cdot\sum_{u=1}^{n+1}\frac{1}{u} \ . \end{aligned} \tag{5.2}$$

The last equality may be proven by induction. Therefore, the average maximum gap can be approximated by $\frac{B}{n+1}\big(\ln(n+1)+\gamma\big)$, where $\gamma \approx 0.577$ is the Euler–Mascheroni constant, perfectly matching our experiments in the discrete case.

Average number of gaps of a given size Furthermore, we also need a more precise estimation of the number N_t of gaps of size $t \in [\![0,B]\!]$. This can be obtained from the continuous case as follows.

Let (U_t) for $t \in [\![0,B]\!]$ be a partition of $[0,B]$ enclosing integers with some offset $\omega = \omega_{n,B} \in \,]0,1[$,[7] i.e., $U_0 \coloneqq [0,\omega]$, $U_t \coloneqq [t-1+\omega, t+\omega]$ for $t \in [\![1,B-1]\!]$ and finally $U_B \coloneqq [B-1+\omega, B]$. In addition, let $\tilde{N}_t = \sharp\{i \in [\![0,n]\!] \mid \Delta_i \in U_t\}$; we heuristically assume that $\tilde{N}_t$ follows the same distribution as its discrete counterpart N_t. We rely on the following result, scaled from $[0,1]$ to $[0,B]$.

Proposition 5.1 ([Dar53, Equation 4.2]). *Let* $W = \sum_{j=0}^{n} h(\Delta_j)$ *for any integrable function* h *on* $[0,B]$. *Then*

$$\mathbb{E}\big[W\big] = (n+1)\cdot\int_0^B n\Big(1-\frac{r}{B}\Big)^{n-1} h(r)\frac{\mathrm{d}r}{B} \ .$$

Applying Proposition 5.1 for $h_t(r) = \mathbb{1}_{U_t}(r)$, noting that $\tilde{N}_t = \sum_{j=0}^{n}\mathbb{1}_{U_t}(\Delta_j)$, we derive the following heuristic approximation of $\mathbb{E}\big[N_t\big]$, where $U_t = [u_t, v_t]$:

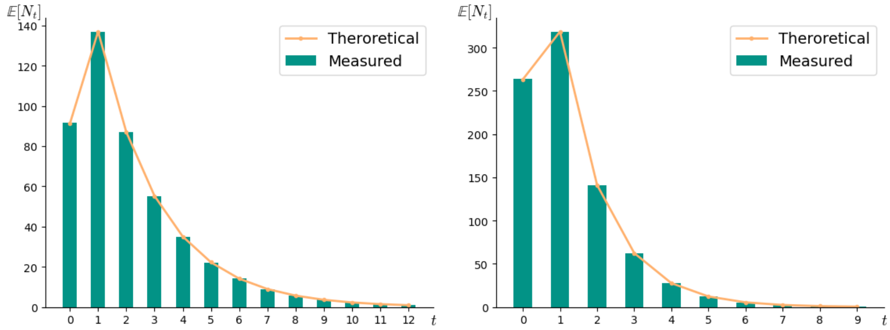
$$\mathbb{E}\big[N_t\big] \approx \mathbb{E}\big[\tilde{N}_t\big] = (n+1)\cdot\Big[\big(1-\tfrac{u_t}{B}\big)^n - \big(1-\tfrac{v_t}{B}\big)^n\Big] \ . \tag{5.3}$$

This approximation is close to $(n+1)\cdot\big(\mathrm{e}^{-n\cdot u_t/B} - \mathrm{e}^{-n\cdot v_t/B}\big)$, which is generally easier to work with in practice due to its simpler exponential form.

It remains to determine the appropriate offset ω. To do so, we calibrate it so that the above-computed heuristic $\mathbb{E}\big[\tilde{N}_0\big] = (n+1)-(n+1)\big(1-\frac{\omega}{B}\big)^n$ matches the formally proven $\mathbb{E}\big[N_0\big]$. In the discrete case, the average number of distinct values for sampling n integers amongst B values is given by $B - B(1-\frac{1}{B})^n$. This is equivalent to having $\mathbb{E}\big[N_0\big] = n - B + B(1-\frac{1}{B})^n$ collisions. Hence, we get

$$\omega = B - B\bigg(\frac{1+B-B(1-1/B)^n}{n+1}\bigg)^{1/n} \ . \tag{5.4}$$

[7] Intuitively, ω is expected to be close to $\frac{1}{2}$. See also Remark 5.2.



(a) For $n = 465$ and $B = 1024$.　　(b) For $n = 834$ and $B = 1024$.

Fig. 5.1: Expectation for N_t, averaged over 10^4 samples of n (discrete logarithms) values modulo B vs. theoretical expectations obtained from Eqn. (5.3). Displayed values stop after the first t s.t. $\mathbb{E}\left[N_t\right] < 1$.

Remark 5.2. Using instead the approximations $\mathbb{E}\left[\tilde{N}_0\right] \approx (n+1) \cdot \left(1 - \mathrm{e}^{-n\omega/B}\right)$ and $\mathbb{E}\left[N_0\right] \approx n - B\left(1 - \mathrm{e}^{-n/B}\right)$, we can obtain a simpler expression for ω. Specifically, we get $(n+1) \cdot \mathrm{e}^{-n/B \cdot \omega} \approx 1 + B(1 - \mathrm{e}^{-n/B})$, so that $\omega \approx -\frac{1}{x} \ln\left(\frac{1-\mathrm{e}^{-x}}{x}\right)$ for $x = \frac{n}{B}$. Assuming $x = o(1)$, its Taylor expansion writes

$$\omega \approx \tfrac{1}{2} - \tfrac{1}{24} \cdot \tfrac{n}{B} + \tfrac{1}{2880} \cdot \left(\tfrac{n}{B}\right)^3 - \dots ,$$

which provides an indication of how close ω is to $\frac{1}{2}$.

Numerical validation We experimentally measured $\mathbb{E}\left[N_t\right]$ for two sets of parameters (n, B): the ratio $\frac{n}{B}$ in Figure 5.1(a) is relevant for analyzing Algorithm 3.1, whereas the ratio $\frac{n}{B}$ in Figure 5.1(b) pertains to the analysis of Algorithms 3.2 and 4.1. In both cases, the observed results closely match the theoretical predictions, exhibiting the same shape. In particular, the results clearly show the exponential decay in the number of large gaps, which explains why a small set of automorphism keys suffices for automorphism-based blind rotations.

In addition, we observe significantly more gaps of size 1 compared to collisions. In practice, it is therefore more effective to handle gaps of size 1, e.g., with the $\mathcal{S}$-parametrized method using $\mathcal{S} = \left\{\mathrm{id}, \tau_g\right\}$, rather than gaining (roughly half of) the gaps of size 0 using complex conjugation.

5.2 Theoretical Analysis of Automorphism-based Methods

We now apply the above discussion to analyze the average number of automorphism key switches required by Algorithms 3.1, 3.2 and 4.1. This readily provides average-case estimations of their computational complexity and noise growth.

Let $\kappa \coloneqq \kappa(w)$ denote the random variable representing the number of key switches for a given window size w. Although κ also implicitly depends on n

and $B = N$ or $\frac{N}{2}$, we omit these parameters since n and N are fixed across all methods. Using a maximal window size $w = B$ gives a lower bound on the average number of automorphism key switches required by the method, denoted by κ_∞.

In this section, we provide explicit formulas closely approximating $\mathbb{E}\left[\kappa(w)\right]$ and $\mathbb{E}\left[\kappa_\infty\right]$ for all algorithms. We also discuss which choice of the window value w is optimal. In general, each algorithm is associated with a cost function $h(w,t)$, which represents the number of automorphism key switches required for a gap of size $t \in [\![0, B]\!]$ using a window parameter w. Consequently, $\kappa(w)$ is simply computed as $\sum_{t=0}^{B} h(w,t) \cdot N_t$, where N_t denotes the number of gaps of size t.

Average number of key switches for Windowed-Hörner methods The easiest case is that of the Windowed-Hörner method, as the sets $I_t^{\pm}$ can be modeled directly[8] as sampling n random values in $[\![0, N]\!]$ with the cost function $h(w,t) = \left\lceil \frac{t}{w} \right\rceil$.

Proposition 5.3. *The expected number of automorphism key switches for a window of size w in Algorithm 3.1 using* $\mathsf{ak}^{\text{HORN}}$ *satisfies*

$$\mathbb{E}\left[\kappa^{\text{HORN}}(w)\right] \approx \left(1 + N\left(1 - \mathrm{e}^{-n/N}\right)\right) \cdot \frac{1}{1 - \mathrm{e}^{-n/N \cdot w}} \quad .$$

In particular, the best possible average value is $\mathbb{E}\left[\kappa_\infty^{\text{HORN}}\right] \approx 1 + N(1 - \mathrm{e}^{-n/N})$.

Proof. For a gap of size t, exactly $\left\lceil \frac{t}{w} \right\rceil$ automorphism key switches must be performed, thus $\mathbb{E}\left[\kappa^{\text{HORN}}(w)\right]$ is given by $\sum_{t=0}^{N} \left\lceil \frac{t}{w} \right\rceil \cdot \mathbb{E}\left[N_t\right]$, i.e., by Equation (5.3),

$$\mathbb{E}\left[\kappa^{\text{HORN}}(w)\right] \approx (n+1) \cdot \sum_{t=1}^{N} \left\lceil \frac{t}{w} \right\rceil \cdot \left(\left(1 - \tfrac{u_t}{N}\right)^n - \left(1 - \tfrac{v_t}{N}\right)^n \right) \quad .$$

Grouping the terms by values of $\left\lceil \frac{t}{w} \right\rceil$, and after canceling successive terms using that $u_t = v_{t-1}$ for $t \in [\![1, N]\!]$ (the last term with v_N being 0), eventually yields

$$\begin{aligned} \mathbb{E}\left[\kappa^{\text{HORN}}(w)\right] &\approx (n+1) \cdot \sum_{k=0}^{\lceil N/w \rceil - 1} \left(1 - \tfrac{v_{kw}}{N}\right)^n \\ &\approx (n+1)\mathrm{e}^{-n/N \cdot \omega} \cdot \frac{1 - \mathrm{e}^{-n/N \cdot w \cdot \lceil N/w \rceil}}{1 - \mathrm{e}^{-n/N \cdot w}} \quad . \end{aligned}$$

The last expression is obtained by plugging $v_{kw} = kw + \omega$ and approximating each $\left(1 - \frac{v_{kw}}{N}\right)^n$ by $\mathrm{e}^{-n/N \cdot v_{kw}} = \mathrm{e}^{-n/N \cdot \omega} \mathrm{e}^{-n/N \cdot wk}$. The numerator is bounded by $\left(1 - \mathrm{e}^{-n}\right)$ and $\left(1 - \mathrm{e}^{-n(1+w/N)}\right)$, both of which are astronomically close to 1. Finally, the offset ω is precisely defined so that $(n+1) \cdot \mathrm{e}^{-\omega n/N}$ equals the expected number of distinct values plus one, i.e., $\mathbb{E}\left[\kappa_\infty^{\text{HORN}}\right] \approx 1 + N(1 - \mathrm{e}^{-n/N})$. □

For the Traversal Windowed-Hörner method, the sets $I_t = I_t^+ \cup I_t^-$ can be modeled as sampling n random values in $[\![0, \frac{N}{2}]\!]$, ignoring the signs. Since gap jumps always combine with possible sign changes, all gaps of size $t > 0$ in this

[8] Formally, using the bijection $g^a \mapsto a \in [\![0, \frac{N}{2} - 1]\!]$ and $-g^a \mapsto a + \frac{N}{2} \in [\![\frac{N}{2}, N - 1]\!]$.

model can be handled with exactly $h(w', t) = \lceil \frac{t}{w'} \rceil$ automorphism key switches, regardless of which I^ϵ is non-empty or processed first.

However, when both I_t^+ and I_t^- are non empty, we must account for additional sign changes. Luckily, the expected number of occurrences of this event is exactly given by the difference between the expected number of distinct values for n samples amongst N values vs. $N/2$, i.e., $N\big(1 - \mathrm{e}^{-n/N}\big) - \frac{N}{2}\big(1 - \mathrm{e}^{-2n/N}\big)$. This leads to the following proposition.

Proposition 5.4. *The expected number of automorphism key switches for a window of size w' in Algorithm 3.2 using $\mathsf{ak}^{\text{TRAV}}$ satisfies*

$$\mathbb{E}\left[\kappa^{\text{TRAV}}(w')\right] \approx \frac{N}{2} \cdot \big(1 - \mathrm{e}^{-n/N}\big)^2 + \left(1 + \frac{N}{2}\big(1 - \mathrm{e}^{-2n/N}\big)\right) \cdot \frac{1}{1 - \mathrm{e}^{-n/N \cdot 2w'}} \ .$$

In particular, the best possible average value is $\mathbb{E}\left[\kappa_\infty^{\text{TRAV}}\right] \approx 1 + N(1 - \mathrm{e}^{-n/N})$.

As shown by Propositions 5.3 and 5.4, both methods converge to the same optimum. However, we can theoretically quantify the improvement brought by the Traversal method for a (fixed) equivalent amount of automorphism keys, i.e., for $w = 2w'$. In that case, the difference simplifies to

$$\mathbb{E}\left[\kappa^{\text{HORN}}(w)\right] - \mathbb{E}\left[\kappa^{\text{TRAV}}(w')\right] \approx \frac{N}{2}\big(1 - \mathrm{e}^{-n/N}\big)^2 \cdot \left(\frac{1}{1 - \mathrm{e}^{-n/N \cdot w}} - 1\right) ,$$

which is strictly positive and decreases towards 0, as expected. This allows the corresponding ratio to be expressed directly as

$$\begin{aligned} \frac{\mathbb{E}\left[\kappa^{\text{TRAV}}(w')\right]}{\mathbb{E}\left[\kappa^{\text{HORN}}(w)\right]} &\approx 1 - \frac{1}{2} \cdot \frac{N(1 - \mathrm{e}^{-n/N})^2}{1 + N(1 - \mathrm{e}^{-n/N})} \cdot \mathrm{e}^{-n/N \cdot w} \\ &\approx 1 - \frac{1}{2} \cdot \big(1 - \mathrm{e}^{-n/N}\big) \cdot \mathrm{e}^{-n/N \cdot w} \ , \end{aligned}$$

which perfectly aligns with the experimental results presented in Table 3.1.

Average number of key switches for the automorphism-parametrized method As for the automorphism-parametrized method, the number of automorphism key switches depend on the specific set $\mathcal{S}$ of automorphisms utilized.

We first consider the simplest case where $\mathcal{S}$ contains all automorphisms τ_{g^k} up to $k = K \geq 0$, and is *symmetric*, i.e., $\tau_a \in \mathcal{S} \Rightarrow \tau_{-a} \in \mathcal{S}$. This basically means that all gap jumps of size at most K, including those involving possible sign changes, can be handled using our new parametrized external product.

Proposition 5.5. *Suppose $\mathcal{S} = \left\{\tau_{\pm 1}, \tau_{\pm g}, \ldots, \tau_{\pm g^K}\right\}$ for $K \geq 0$. The expected number of automorphism key switches for a window of size w'' in Algorithm 4.1 using $\mathsf{ak}^{\mathcal{S}\text{-AUT}}[0] \cup \mathsf{ak}^{\mathcal{S}\text{-AUT}}[1 \ldots w'']$ satisfies*

$$\mathbb{E}\left[\kappa^{\mathcal{S}\text{-AUT}}(w'')\right] \approx \frac{1}{2} + \left(1 + \frac{N}{2}\big(1 - \mathrm{e}^{-2n/N}\big)\right) \cdot \mathrm{e}^{-K \cdot 2n/N} \cdot \frac{1}{1 - \mathrm{e}^{-n/N \cdot 2w''}} \ .$$

This tends towards $\mathbb{E}\left[\kappa_\infty^{\mathcal{S}\text{-AUT}}\right] \approx \frac{1}{2} + \big(1 + \frac{N}{2}(1 - \mathrm{e}^{-2n/N})\big) \cdot \mathrm{e}^{-K \cdot 2n/N}$ with w''.

In other words, Proposition 5.5 shows that increasing K by 1 essentially reduces the number of automorphism key switches by a factor of $\mathrm{e}^{2n/N}$, which is approximately 2.26 for $n = 834$ and $N = 2048$ (resp. 2.48 for $n = 465$ and $N = 1024$). This matches the measurements presented in Table 5.4.

Remark 5.6. The case $K = 0$, corresponding to $\mathcal{S} = \{\mathrm{id}, \tau_{-1}\}$, is *computationally* equivalent to (though requiring *less* keys than) the proposal in [LLW+24]. Indeed, both settings eliminate precisely the homomorphic complex conjugations.

Proof (of Proposition 5.5). The modeling is the same as for the Traversal method with $B = \frac{N}{2}$. Due to the shape of $\mathcal{S}$, all gaps of size $t \leq K$, including a possible sign change, incur no cost. For gaps of size $t > K$, $\left\lceil \frac{t-K}{w''} \right\rceil$ automorphism key switches are required (followed e.g., by a new external product parametrized by $\tau_{\pm g^K}$). Thus, the expected number of automorphism key switches is

$$\mathbb{E}[\kappa^{\mathcal{S}\text{-AUT}}(w'')] \approx (n+1) \cdot \sum_{t=K+1}^{N/2} \left\lceil \frac{t-K}{w''} \right\rceil \cdot \left(\left(1 - \frac{u_t}{N/2}\right)^n - \left(1 - \frac{v_t}{N/2}\right)^n \right) .$$

Using the same reasoning as before, this simplifies to the right-hand side of the result. Finally, since only the "positive" part of $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ is used, applying a sign change using $\mathsf{ak}^{\mathcal{S}\text{-AUT}}[0]$ (Algorithm 4.1, Line 15) is required half of the time. $\square$

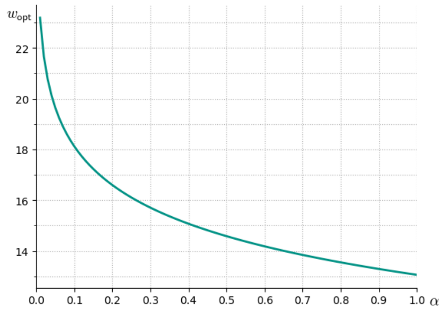
We also explore various tradeoffs where $\mathcal{S}$ is not symmetrical. In particular, if $\mathcal{S}$ only lacks τ_{-1} to be symmetrical, the situation is similar to Proposition 5.4 for the Traversal method: automorphism key switches using $\mathsf{ak}^{\mathcal{S}\text{-AUT}}[0]$ are only required when both $I_t^{\pm}$ are non-empty, so $\frac{N}{2} \cdot \left(1 - \mathrm{e}^{-n/N}\right)^2$ is simply added.

The situation is trickier in general. We focus on the case $\mathcal{S} = \{\mathrm{id}, \tau_g\}$, as other cases appear to be of limited interest. Notably, this is the first scenario where both the "negative" and "positive" parts of $\mathsf{ak}^{\mathcal{S}\text{-AUT}}$ are required. As before, we must add the complex conjugation count, i.e., $\frac{N}{2} \cdot \left(1 - \mathrm{e}^{-n/N}\right)^2$. However, we also must account for size-1 gaps that necessitate a sign change. Empirically, we observed that $\theta \approx 60\%$ of these size-1 gaps incur an additional sign change cost.

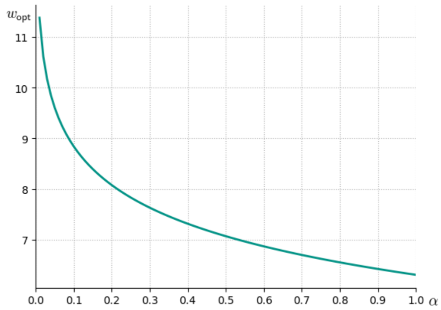
On the optimal window size In previous works, the optimal window size is guesstimated as the point where some experimental graph sufficiently flattens. For instance, [LMK+23, Figure 3] suggests using $w = 10$, while in an equivalent setting [LLW+24, Figure 2(b)] instead suggests using $w = 20$.

We propose a more robust definition of this optimal window size. Intuitively, the optimal window size should be set such that no greater gaps occur on average. More formally, we define w_{opt} as the smallest w s.t. $\mathbb{E}[\kappa(w)] \leq \mathbb{E}[\kappa_\infty] + \alpha$, where $0 < \alpha \leq 1$ is fixed. Thus, let w_{opt} (resp. w'_{opt}, w''_{opt}) denote the corresponding optimal window value w.r.t. Algorithm 3.1 (resp. Algorithm 3.2, resp. Algorithm 4.1 for a symmetric set of automorphisms $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq K\}$). Solving directly the condition for $w_{\mathsf{opt}}, w'_{\mathsf{opt}}$ using Propositions 5.3 and 5.4 yields the equivalent expression

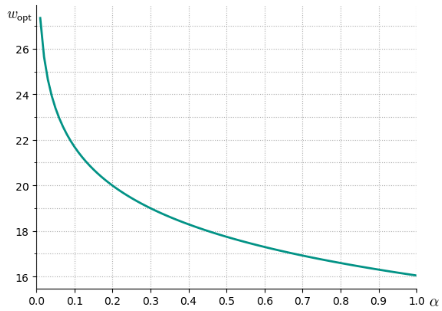
$$w_{\mathsf{opt}}^{(\prime)} = \left\lceil \frac{B}{n} \cdot \ln\left(1 + \frac{1 + B\left(1 - \mathrm{e}^{-n/B}\right)}{\alpha}\right) \right\rceil , \tag{5.5}$$



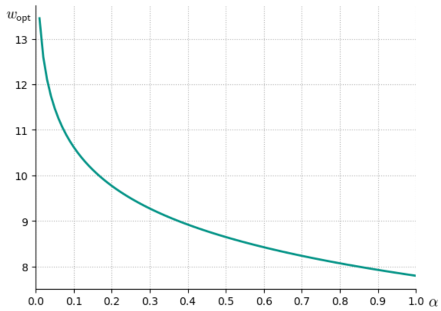
(a) For $n = 465$ and $B = N = 1024$.



(b) For $n = 465$ and $B = N/2 = 512$.



(c) For $n = 834$ and $B = N = 2048$.



(d) For $n = 834$ and $B = N/2 = 1024$.

Fig. 5.2: Optimal window size w_{opt}, depending on the average distance α to κ_∞. Using $B = N$ corresponds to Algorithm 3.1 and [LMK⁺23], whereas $B = N/2$ captures Algorithm 3.2 (w'_{opt}), Algorithm 4.1 for $\mathcal{S} = \{\tau_{\pm 1}\}$ and [LLW⁺24].

where $B = N$ for w_{opt} and $B = N/2$ for w'_{opt}. Likewise, using Proposition 5.5, we obtain that for a given $K \geq 0$, w''_{opt} verifies

$$w''_{\mathsf{opt}} = \left\lceil \frac{N}{2n} \ln\left(1 + \frac{1 + N/2(1 - \mathrm{e}^{-2n/N})}{\alpha} \cdot \mathrm{e}^{-K \cdot 2n/N}\right) \right\rceil \approx w'_{\mathsf{opt}} - K \ . \tag{5.6}$$

Curves for $w^{(\prime)}_{\mathsf{opt}}$ (ommitting the ceiling) are given in Figure 5.2. We remark that $w''_{\mathsf{opt}} = w'_{\mathsf{opt}}$ when $K = 0$, so that the figures with $B = N/2$ (w'_{opt}) also encompass optimal window sizes for Algorithm 4.1 (w''_{opt}) with $\mathcal{S} = \{\tau_{\pm 1}\}$ as well as [LLW⁺24] (see Remark 5.6). For example, for $n = 465$ and $N = 1024$, Figure 5.2(a) shows that choosing $w_{\mathsf{opt}} = 20$ ensures $\alpha \geq 0.043$, whereas Figure 5.2(b) indicates that setting $w'_{\mathsf{opt}} = 8$ as in [LLW⁺24] only yields $\alpha \geq 0.215$. Conversely, targeting $\alpha = 0.25$ yields resp. $w_{\mathsf{opt}} = \lceil 16.1 \rceil$ and $w'_{\mathsf{opt}} = \lceil 7.8 \rceil$, according to Equation (5.5). This also supports the intuition that $w_{\mathsf{opt}} \approx 2w'_{\mathsf{opt}}$, since the average maximum gap is roughly halved from $\frac{N/2}{n/2}\ln(n/2)$ to $\frac{N/2}{n}\ln n$.

Table 5.3: Comparison of automorphism-based blind rotations w.r.t. evaluation key material and best average number of automorphism key switches. For concision, $A_{n/N}$ denotes $\mathrm{e}^{-n/N}$, and experimental observations suggest $\theta \approx 60\%$.

	Keys size (♯GLWE$^{\nabla}$) bsk	ak	Avg. number of aut. key switches ($\mathbb{E}[\kappa_\infty]$, i.e., using $w_{\mathsf{opt}}^{(\prime,\prime\prime)}$)
Telescoping (Fig. 2.1(b))	$(k+1)n$	kN	$\min\{n, N\}$
Wind Hörner (Alg. 3.1)	$(k+1)n$	$k(w+1)$	$N \cdot (1 - A_{n/N})$
Traversal (Alg. 3.2)	$(k+1)n$	$k(2w'+1)$	$N \cdot (1 - A_{n/N})$
[LLW+24, Alg. 2]	$2(k+1)n$	kw'	$\frac{N}{2} \cdot (1 - A_{n/N}^2)$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}\}$	$(2k+1)n$	$k(w'+1)$	$\frac{N}{2} \cdot (1 - A_{n/N}^2)$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_g\}$	$(2k+1)n$	$k(2w'-1)$	$N \cdot (1 - A_{n/N}) - \frac{N}{2} \cdot \theta \big(1 - A_{n/N}^2\big)^2$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}\}$	$(3k+1)n$	kw'	$\frac{N}{2} \cdot \big(A_{n/N}^2 - A_{n/N}^4 + (1 - A_{n/N})^2\big)$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}\}$	$(4k+1)n$	kw'	$\frac{N}{2} \cdot (1 - A_{n/N}^2) \cdot A_{n/N}^2$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}, \tau_{\pm g^2}\}$	$(6k+1)n$	$k(w'-1)$	$\frac{N}{2} \cdot (1 - A_{n/N}^2) \cdot A_{n/N}^4$
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid k < K\}$	$(2Kk+1)n$	$k(w'-K+2)$	$\frac{N}{2} \cdot (1 - A_{n/N}^2) \cdot A_{n/N}^{2(K-1)}$

Remark 5.7. Using the approximation $1 + B(1 - \mathrm{e}^{-n/B}) \approx (n+1) \cdot \mathrm{e}^{-n\omega_{n,B}/B}$ (see Remark 5.2), and assuming $\omega_{n,B} < \frac{1}{2}$ and $n < B$, we easily obtain much simpler—though not fully accurate—expressions:

$$w_{\mathsf{opt}} \approx \left\lceil \tfrac{N}{n} \cdot \ln \tfrac{n+1}{\alpha} \right\rceil , \qquad w'_{\mathsf{opt}} \approx \left\lceil \tfrac{N}{2n} \cdot \ln \tfrac{n+1}{\alpha} \right\rceil ,$$
$$w''_{\mathsf{opt}} \approx \left\lceil \tfrac{N}{2n} \cdot \ln \tfrac{n+1}{\alpha} \right\rceil - K \ .$$

As expected, these values are close to the average maximum gap when $\alpha = 1$. This yields a direct explanation of why small window sizes suffice in practice and clearly shows that the optimal window grows logarithmically as α approaches 0.

Theoretical summary The performance of automorphism-based blind rotation algorithms is summarized in Table 5.3 according to Propositions 5.3, 5.4 and 5.5. As throughout this paper, the results are given for null or invertible mask components. We also consider in the ak column that $w' = w/2 = w'' + K$ (see Remark 5.7) in order to ease comparisons. The number of (automorphism-parametrized) external products is always exactly n and is therefore omitted.

5.3 Numerical Measurements

To demonstrate the impact of our techniques on automorphism-based blind rotations, we conducted experiments to measure the average number of (extended and classical) external products and automorphism key switches required by each of our new variants, comparing their performance against [LMK+23, LLW+24].

Table 5.4: Experimentally measured number of regular ($\circledast$), automorphism-parametrized ($\circledast_{\mathcal{S}}$) external products, and automorphism key switches (KS) for automorphism-based blind rotations, averaged over 10^4 samples. The optimal window values $w_{\mathsf{opt}}^{(\prime,\prime\prime)}$ are computed for $\alpha = 0.25$.

(a) Parameters $n = 465$ and $N = 1024$, $k = 1$.

Method	$w_{\mathsf{opt}}^{(\prime,\prime\prime)}$	Keys ($\sharp$GLWE$^\nabla$)	$\sharp(\circledast)$	$\sharp(\circledast_{\mathcal{S}})$	$\sharp$(KS)
Windowed-Hörner (Alg. 3.1, [LMK+23])	17	$2n + 18$	465	0	375.8
Traversal Windowed-Hörner (Alg. 3.2)	± 8	$2n + 17$	465	0	375.0
[LLW+24, Alg. 2]	8	$4n + 8$	465	0	306.1
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}\}$	8	$3n + 9$	210	255	306.6
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_g\}$	± 7	$3n + 15$	306	159	263.1
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}\}$	7	$4n + 8$	159	306	192.5
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_g\}$	± 7	$4n + 15$	91	374	195.4
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}\}$	7	$5n + 8$	91	374	124.3
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}, \tau_{\pm g^2}\}$	6	$6n + 7$	159	306	118.8
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}, \tau_{\pm g^2}\}$	6	$7n + 7$	91	374	50.6
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 3\}$	5	$9n + 6$	91	374	20.9
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 4\}$	4	$11n + 5$	91	374	9.0
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 5\}$	3	$13n + 4$	91	374	4.3
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 6\}$	3	$15n + 4$	91	374	1.9
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 7\}$	2	$17n + 3$	91	374	1.5

(b) Parameters $n = 834$ and $N = 2048$, $k = 1$.

Method	$w_{\mathsf{opt}}^{(\prime,\prime\prime)}$	Keys ($\sharp$GLWE$^\nabla$)	$\sharp(\circledast)$	$\sharp(\circledast_{\mathcal{S}})$	$\sharp$(KS)
Windowed-Hörner (Alg. 3.1, [LMK+23])	20	$2n + 21$	834	0	688.5
Traversal Windowed-Hörner (Alg. 3.2)	± 10	$2n + 21$	834	0	686.4
[LLW+24, Alg. 2]	10	$4n + 10$	834	0	571.4
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}\}$	10	$3n + 11$	376	458	571.9
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_g\}$	± 9	$3n + 19$	264	570	495.5
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}\}$	9	$4n + 10$	264	570	368.7
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_g\}$	± 9	$4n + 10$	149	685	380.8
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}\}$	9	$5n + 10$	149	685	254.0
$\mathcal{S}$-Aut, $\mathcal{S} = \{\mathrm{id}, \tau_{\pm g}, \tau_{\pm g^2}\}$	8	$6n + 9$	263	571	227.2
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm 1}, \tau_{\pm g}, \tau_{\pm g^2}\}$	8	$7n + 9$	149	685	112.5
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 3\}$	7	$9n + 8$	149	685	50.1
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 4\}$	6	$11n + 7$	149	685	23.0
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 5\}$	5	$13n + 6$	149	685	10.9
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 6\}$	4	$15n + 5$	149	685	5.4
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 7\}$	3	$17n + 4$	149	685	3.1
$\mathcal{S}$-Aut, $\mathcal{S} = \{\tau_{\pm g^k} \mid 0 \leq k \leq 8\}$	2	$19n + 3$	149	685	1.9

The results are summarized in Table 5.4 for the two different parameter sets, corresponding to those in Section 3—with $n = 465$ for the first set, as in [LLW+24], instead of $n = 458$. The optimal window values are computed from Equations (5.5) and (5.6) under the rather stringent[9] requirement that the distance between $\kappa\left(w_{\mathsf{opt}}^{(\prime,\prime\prime)}\right)$ and the corresponding κ_∞ does not exceed $\alpha = 0.25$ on average. The number of automorphism key switches and operation counts are then averaged over 10^4 random mask components in $(\mathbb{Z}/2N\mathbb{Z})^\times$. We stress that although there are separate counts for $\circledast$ and $\circledast_{\mathcal{S}}$ to highlight the use of the new parametrized external product, both operations are strictly equivalent regarding their computational cost.

The results indicate that, with the same increase of the bootstrapping keys as in [LLW+24], Algorithm 4.1 with $\mathcal{S} = \left\{\mathrm{id}, \tau_{\pm g}\right\}$ already requires 37.1% (resp. 35.5%) fewer key switches compared to [LLW+24] and 49.1% (resp. 46.4%) fewer compared to [LMK+23]). More strikingly, with keys that are 25% smaller than in [LLW+24], Algorithm 4.1 using the asymmetric set $\mathcal{S} = \left\{\mathrm{id}, \tau_g\right\}$ already outperforms [LLW+24] by over 14.0% (resp. 13.3%). In particular, this clearly demonstrates an advantage over using $\mathcal{S} = \left\{\tau_{\pm 1}\right\}$, as predicted from Figure 5.1.

Further trade-offs are also possible. With only a moderate increase of the size of $\mathsf{bsk}^{\mathcal{S}\text{-AUT}}$, i.e., multiplied by 2.5 compared to [LMK+23] instead of 2 as in [LLW+24], Algorithm 4.1 with $\mathcal{S} = \left\{\tau_{\pm 1}, \tau_{\pm g}\right\}$ achieves a reduction of the number of key switches of more than 59.4% (resp. 55.5%) relative to [LLW+24] and 66.9% (resp. 63.1%) relative to [LMK+23]). Moreover, the last rows of the tables illustrate the capability of our new $\mathcal{S}$-parametrized method to approach the computational efficiency of AP bootstrapping while maintaining reasonably sized keys. Indeed, considering keys only up to 9 times larger w.r.t. [LMK+23], the average number of key switches can be squeezed down to only 2 or 3, whereas AP keys are more than 2 orders of magnitude larger for comparable performance. As the convergence is quite fast, the set $\mathcal{S} = \left\{\tau_{\pm 1}, \tau_{\pm g}, \tau_{\pm g^2}\right\}$ yields an appealing middle ground: with a $3.5\times$ increase in $\mathsf{bsk}^{\mathcal{S}\text{-AUT}}$ compared to [LMK+23], it eliminates 86.5% (resp. 83.6%) of the key switches of the Traversal Windowed-Hörner method.

As a final remark, we emphasize that while the proposed variants imply increasing the size of the bootstrapping keys compared to [GINX16] or [LMK+23], only $(k+1)$ GLWE^∇ ciphertexts are ever needed for computing each external product, whether parametrized or not. Those can easily be prefetched as soon as the (mod-switched) mask components are known and sorted, therefore the bandwidth requirements for our methods remain unchanged.

[9] E.g., the choice $w'' = 8$ made in [LLW+24] could correspond to any $\alpha \in \left[0.22, 0.53\right]$.

References

AP14. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology — CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014. doi:10.1007/978-3-662-44371-2_17.

BCG+23. Mariya Georgieva Belorgey, Sergiu Carpov, Nicolas Gama, Sandra Guasch, and Dimitar Jetchev. Revisiting key decomposition techniques for FHE: Simpler, faster and more generic. Cryptology ePrint Archive, 2023. URL: https://ia.cr/2023/771.

BDF18. Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. In A. Joux, A. Nitaj, and T. Rachidi, editors, *Progress in Cryptology — AFRICACRYPT 2018*, volume 10831 of *Lecture Notes in Computer Science*, pages 217–251. Springer, Cham, 2018. doi:10.1007/978-3-319-89339-6_13.

BIP+22. Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In S. Agrawal and D. Lin, editors, *Advances in Cryptology — ASIACRYPT 2022, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 188–215. Springer, Cham, 2022. doi:10.1007/978-3-031-22966-4_7.

BJ24. Olivier Bernard and Marc Joye. Approximate CRT-based gadget decomposition and application to TFHE blind rotation. Cryptology ePrint Archive, Report 2024/909, 2024. URL: https://ia.cr/2024/909.

BJSW24. Olivier Bernard, Marc Joye, Nigel Smart, and Michael Walter. Drifting towards better error probabilities in fully homomorphic encryption schemes. Cryptology ePrint Archive, Report 2024/1718, 2024. URL: https://ia.cr/2024/1718.

CGGI20. Ilaria Chilotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020. doi:10.1007/s00145-019-09319-x.

CJP21. Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In S. Dolev et al., editors, *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021. doi:10.1007/978-3-030-78086-9_1.

Dar53. Donald A. Darling. On a class of problems related to the random division of an interval. *The Annals of Mathematical Statistics*, 24(2):239–253, 1953. doi:10.1214/aoms/1177729030.

DM15. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology — EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, Berlin, Heidelberg, 2015. doi:10.1007/978-3-662-46800-5_24.

DN03. Herbert A. David and Haikady N. Nagaraja. *Order Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 3rd edition, 2003. doi:10.1002/0471722162.

Gen10. Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010. doi:10.1145/1666420.1666444.

GINX16. Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology — EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 528–558. Springer, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-49896-5_19.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology — CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-40041-4_5.

HKLS24. Deokhwa Hong, Young-Sik Kim, Yongwoo Lee, and Eunyoung Seo. A new fine tuning method for FHEW/TFHE bootstrapping with IND-CPA$^{\mathrm{D}}$ security. Cryptology ePrint Archive, 2024. URL: https://ia.cr/2024/1052.

Joy22. Marc Joye. SoK: Fully homomorphic encryption over the [discretized] torus. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):661–692, 2022. doi:10.46586/tches.v2022.i4.661-692.

JP22. Marc Joye and Pascal Paillier. Blind rotation in Fully Homomorphic Encryption with extended keys. In J.Katz S. Dolev and A. Meisels, editors, *Cyber Security, Cryptology, and Machine Learning: 6th International Symposium (CSCML 2022)*, volume 13301 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Cham, 2022. doi:10.1007/978-3-031-07689-3_1.

JW22. Marc Joye and Michael Walter. Liberating TFHE: Programmable bootstrapping with general quotient polynomials. In M. Brenner, A. Costache, and K. Rohloff, editors, *WAHC'22: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–11. Association for Computing Machinery, New York, 2022. doi:10.1145/3560827.3563376.

LLW^{+}24. Zhihao Li, Xianhui Lu, Zhiwei Wang, Ruida Wang, Ying Liu, Yinhang Zheng, Lutan Zhao, Kunpeng Wang, and Rui Hou. Faster NTRU-based bootstrapping in less than 4 ms. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(3):418–451, 2024. doi:10.46586/tches.v2024.i3.418-451.

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 application to threshold homomorphic encryption. In C. Hazay and M. Stam, editors, *Advances in Cryptology — EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 227–256. Springer, Cham, 2023. doi:10.1007/978-3-031-30620-4_8.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. doi:10.1007/978-3-642-13190-5_1.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75:565–599, 2015. doi:10.1007/s10623-014-9938-4.

MKMS24. Gabrielle De Micheli, Duhyeong Kim, Daniele Micciancio, and Adam Suhl. Faster amortized FHEW bootstrapping using ring automorphisms. In Q. Tang and V. Teague, editors, *Public-Key Cryptography — PKC 2024, Part IV*, volume 14604 of *Lecture Notes in Computer Science*, pages 322–353. Springer, Cham, 2024. doi:10.1007/978-3-031-57728-4_11.

MP21. Daniele Micciancio and Yuriy Polyakov. Bootstrapping in FHEW-like cryptosystems. In M. Brenner, R. Player, and K. Rohloff, editors, *9th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2021)*, pages 17–28. ACM Press, 2021. doi:10.1145/3474366.3486924.

RAD78. Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In R. A. DeMillo et al., editors, *Foundations of Secure Computation*, pages 169–179. Academic Press, 1978. Available at https://people.csail.mit.edu/rivest/pubs.html#RAD78.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. doi:10.1145/1568318.1568324.

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In M. Matsui, editor, *Advances in Cryptology — ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Sprinber, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-10366-7_36.

WWL+24. Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. In M. Joye and G. Leander, editors, *Advances in Cryptology — EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 342–372. Springer, Cham, 2024. doi:10.1007/978-3-031-58723-8_12.

XZD+23. Binwu Xiang, Jiang Zhang, Yi Deng, Yiran Dai, and Dengguo Feng. Fast blind rotation for bootstrapping FHEs. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology — CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 3–36. Springer, Cham, 2023. doi:10.1007/978-3-031-38551-3_1.

Zam22. Zama. TFHE-rs: A pure Rust implementation of the TFHE scheme for boolean and integer arithmetics over encrypted data, 2022. URL: https://github.com/zama-ai/tfhe-rs.

A Dealing With Arbitrary Mask Components

In the general case, the modulus switching, as part of the (programmable) bootstrapping, does not directly yield mask components $\tilde{a}_i \in (\mathbb{Z}/m\mathbb{Z})^\times$ corresponding to automorphisms, or $\tilde{a}_i = 0$. Several techniques have been proposed to circumvent this:

1. changing the base ring to a circulant ring [BDF18] or a prime conductor cyclotomic ring of integers [MKMS24];
2. rewriting $\sum_i \tilde{a}_i s_i$ as $\sum_i \tilde{a}_i^* s_i^*$, where $\tilde{a}_i^* \in (\mathbb{Z}/m\mathbb{Z})^\times \cup \{0\}$, plus a correcting step implementing a trade-off between noise growth/performance and key material [LMK+23, Sections 3.2.1 and 3.2.2];

3. adapting the modulus switch to round to either 0 or directly to invertible elements [LMK+23, Section 3.3], or further, directly to elements of a cyclic subgroup thereof [WWL+24].

The first approach eludes the problem altogether. We therefore focus below on the two latter approaches, viewed as *pre-conditioning* algorithms for the LWE input ciphertext and keys before entering the blind rotation. They both suppose $m = 2N$ is a power of two so that every odd integer has an inverse modulo m.

A.1 Rewriting the inner product

Two reorganizations of the inner product are given in [LMK+23, Sections 3.2.1 and 3.2.2].

The first method [LMK+23, Algorithm 4] simply consists in setting the new mask components as $\tilde{a}_i^* = \tilde{a}_i - 1$ if $\tilde{a}_i \neq 0$ is even, and $\tilde{a}_i$ otherwise; then, additional external products, one per even value $\tilde{a}_i$, need to be applied to the accumulator.[10] This incurs on average a significant computational overhead of about $n/2$ external products, which can be strictly limited to $n/2 + 1$ using an extra bootstrapping key $\mathsf{bsk}_{\text{nsum}}^{\text{AUT}}$ encrypting $x^{-\sum_i s_i}$ and allowing to flip the parity of all mask components at once.

The second method [LMK+23, Algorithm 5] approximately doubles the size of the bootstrapping keys by including

$$\mathsf{bsk}^{\text{AUT}*} := \left\{ \text{GLWE}_{\mathfrak{s}}^{\circledast}(x^{s_i + s_{i+1}}) \;\middle|\; i \in [\![1, n-1]\!] \right\} .$$

Assuming without loss of generality that $\tilde{a}_1$ is odd, then iteratively when $\tilde{a}_{i+1}$ is even, $\tilde{a}_{i+1}^*$ is set to $\left(\tilde{a}_{i+1} - \tilde{a}_i^*\right)$ and the bootstrapping key for position i is modified to $\mathsf{bsk}^{\text{AUT}*}[i]$. Finally, Algorithm 3.1 (or Algorithm 3.2) is called with these adjusted mask components and bootstrapping keys.

A.2 Rounding to multiplicative (sub-)groups

A different strategy, described in [LMK+23, Section 3.3], involves modifying the modulus switching so that its outputs directly lie in $(\mathbb{Z}/2N\mathbb{Z})^\times \cup \{0\}$. This rounding to odd values (or to zero) can be expressed as[11]

$$\lfloor x \rceil_{\text{odd}} := \begin{cases} 0 & \text{if } |x| \leq \frac{1}{2} , \\ 1 + 2 \cdot \left\lfloor \frac{x}{2} \right\rfloor & \text{otherwise} . \end{cases}$$

[10] In [LMK+23, Algorithm 4], this is a post-processing step due to the initial value of the accumulator to $\tau_{-g}\left(x^{-\tilde{b}} \cdot v(x)\right)$. In the setting of Algorithm 3.2 though, this can be readily moved in the "pre-conditioning" phase.

[11] Rounding small values to 0 is a free optimization using that null (more generally, small, see e.g., [HKLS24]) mask components can be skipped entirely in the blind rotation loop.

Hence, the modulus switching of an LWE ciphertext $(a_1, \ldots, a_n, b) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ with the above rounding operator outputs $(\tilde{a}_1, \ldots, \tilde{a}_n, \tilde{b})$ with

$$\tilde{a}_i = \left\lfloor \tfrac{2N}{q} a_i \right\rceil_{\mathrm{odd}} \text{ for } i \in [\![1, n]\!], \quad \text{and} \quad \tilde{b} = \left\lfloor \tfrac{2N}{q} b \right\rceil \ .$$

This technique has been further extended in a recent work by Wang *et al.* [WWL+24] where the modulus switching gives elements of some predetermined cyclic subgroup of $(\mathbb{Z}/2N\mathbb{Z})^\times \simeq \langle \pm 1 \rangle \times \langle g \rangle$. Formally, for $\theta \in [\![1, \log_2 N]\!]$ with $N \geq 4$, let G_θ be the following cyclic subgroup

$$G_\theta \coloneqq \left\langle g^{2^{\theta-1}} \right\rangle = \left\{ u \in (\mathbb{Z}/2N\mathbb{Z})^\times \mid u \equiv 1 \mod 2^{\theta+1} \right\} \simeq \mathbb{Z}/\tfrac{N}{2^\theta}\mathbb{Z} \ .$$

The corresponding rounding to the nearest element in $G_\theta \cup \{0\}$ is defined as

$$\lfloor x \rceil_\theta \coloneqq \begin{cases} 0 & \text{if } x \in \left[\frac{1}{2} - 2^\theta, \frac{1}{2}\right] , \\ 1 + 2^{\theta+1} \cdot \left\lfloor \frac{x-1}{2^{\theta+1}} \right\rceil & \text{otherwise} \ . \end{cases} \tag{A.1}$$

Note $\lfloor x \rceil_{\mathrm{odd}} = \lfloor x \rceil_0$ corresponds to rounding to the (however non-cyclic) entire group $G_0 \coloneqq (\mathbb{Z}/2N\mathbb{Z})^\times$. The modulus switching then outputs

$$\tilde{a}_i = \left\lfloor \tfrac{2N}{q} a_i \right\rceil_\theta \text{ for } i \in [\![1, n]\!], \quad \text{and} \quad \tilde{b} = \left\lfloor \tfrac{2N}{q} b \right\rceil \ .$$

Since G_θ is a sparser cyclic subgroup of G_0 of cardinal $\frac{N}{2^\theta}$, the mask values after modulus switching are concentrated on a smaller set. As a result, the number of automorphism key switches is reduced, as are the gaps between distinct values in terms of jumps by $g^{2^{\theta-1}}$. However, and crucially, the modified modulus switching procedure induces an increased drift, leading to a higher overall failure probability (see e.g., [WWL+24, Table 4]). This shall be accounted for when designing parameters and makes it a lot harder to get a fair comparison in terms of performance or key size between works such as [LMK+23] and [WWL+24]. We refer the reader to [BJSW24] for a detailed discussion on the drift.

B Correctness of Algorithm 3.1

For completeness, we provide a proof of the correctness of Algorithm 3.1 below. While the algorithm essentially builds on [LMK+23], the proof differs noticeably from [LMK+23, Section 3.1].

Proposition B.1. *Algorithm 3.1 is correct.*

Proof. Let $\mathfrak{q}_i = x^{-\tilde{b} + \sum_{j=1}^{i} \tilde{a}_j s_j} \cdot \mathfrak{v}$. For $t \in [\![0, N/2]\!]$, consider the polynomials $\tilde{\mathfrak{q}}_t^\pm \in \mathcal{R}_q$ recursively defined by $\tilde{\mathfrak{q}}_{N/2}^- = \mathfrak{q}_0 = \mathfrak{v} \cdot x^{-\tilde{b}}$, $\tilde{\mathfrak{q}}_{N/2}^+ = \tilde{\mathfrak{q}}_0^-$, and for all $N/2 > t \geq 0$,

$$\tilde{\mathfrak{q}}_t^\pm = \tilde{\mathfrak{q}}_{t+1}^\pm \cdot x^{\pm g^t \cdot \sum_{i \in I_t^\pm} s_i} ,$$

where all $\pm$ choices are identical. By Equations (3.1), (3.2) and (3.3), it follows that $\tilde{q}_0^+ = q_n$. We now inductively show that after each iteration of the respective loops on $I^\pm$, ACC contains a GLWE encryption of $\tau_{\pm g^{-t_{\text{old}}}}\left(\tilde{q}_t^\pm\right)$.[12]

First, ACC is initialized to a (trivial) GLWE encryption of $\tau_{-1}\left(q_0\right)$, which is indeed equal to $\tau_{-g^{-N/2}}\left(\tilde{q}_{N/2}^-\right)$. Inside the loop, the induction hypothesis is preserved as long as $I_t^\pm$ is empty, since in this case $\tilde{q}_t^\pm = \tilde{q}_{t_{\text{old}}}^\pm$. Assume now that $I_t^\pm$ is non-empty for some $N/2 > t \geq 0$ and ACC contains a GLWE encryption of $\tau_{\pm g^{-t_{\text{old}}}}\left(\tilde{q}_{t+1}^\pm\right)$. Then, τ_{g^δ} is homomorphically applied to ACC to obtain a GLWE encryption of $\tau_{\pm g^{-t}}\left(\tilde{q}_{t+1}^\pm\right)$, and the following external products finally yield as expected a GLWE encryption of

$$\tau_{\pm g^{-t}}\left(\tilde{q}_{t+1}^\pm\right) \cdot x^{\sum_{i \in I_t^\pm} s_i} = \tau_{\pm g^{-t}}\left(\tilde{q}_{t+1}^\pm \cdot x^{\pm g^t \sum_{i \in I_t^\pm} s_i}\right) = \tau_{\pm g^{-t}}\left(\tilde{q}_t^\pm\right) \ .$$

After the first loop, ACC contains an encryption of $\tau_{-g^{-t_{\text{old}}}}\left(\tilde{q}_0^-\right)$, which after evaluating τ_{-g} on Line 9 becomes an encryption of $\tau_{g^{-t_{\text{old}}+1}}\left(\tilde{q}_0^-\right)$. Decreasing t_{old} by 1, this initializes correctly the second loop. Likewise, after the second loop ACC contains a GLWE encryption of $\tau_{g^{-t_{\text{old}}}}\left(\tilde{q}_0^+\right)$, implying the result since the last lines of the algorithm apply $\tau_{g^{t_{\text{old}}}}$. $\square$

[12] Actually, t_{old} encodes $\log_g \pm\tilde{a}_{\text{old}}$ where $\tilde{a}_{\text{old}}$ is as in Figure 2.1(b), with the masks $\tilde{a}_i$'s reordered according to $I_t^\pm$.