

Low-Noise Multi-Value Bootstrapping via Log-Unit Lattice Search and LUT Shifting

Olivier Bernard , Nolan Carouge , Marc Joye  ,
Jean-Baptiste Orfila  and Samuel Tap 

Zama, Paris, France

Abstract

Programmable bootstrapping is a central procedure in the FHEW and TFHE families of fully homomorphic encryption schemes, but its computational cost remains a major performance bottleneck. Multi-value bootstrapping amortizes this cost by sharing a blind rotation among several functions evaluated on the same encrypted input. However, the standard choice of common factor for multi-value bootstrapping can produce unnecessarily large noise amplification for many collections of lookup tables, thereby reducing the practical benefits of sharing the blind rotation. The main contribution of this work is a log-unit lattice search for selecting common factors that substantially reduce this amplification. This search is complemented with LUT shifting, which generates alternative function-preserving test-polynomial representations and further enlarges the search space. Both optimizations are performed offline and leave the online procedure of multi-value bootstrapping essentially unchanged. The proposed construction is evaluated in TFHE at failure probability 2^{-128} . For example, on 256-bit homomorphic integer multiplication, it achieves a latency speed-up of $1.47\times$ and a throughput improvement of $1.59\times$ relative to a baseline that evaluates each output with a separate programmable bootstrapping.

Keywords: Fully homomorphic encryption (FHE), FHEW/TFHE, Programmable bootstrapping, Multi-value bootstrapping, LUT shifting, Log-unit lattice search

1 Introduction

Fully homomorphic encryption (FHE) enables arbitrary computations on encrypted data without revealing the inputs or intermediate results [RAD78, Gen10]. At the core of every FHE scheme lies the challenge of controlling the noise that accumulates with each homomorphic operation. Without a mechanism to refresh ciphertexts, the number of operations that can be performed is strictly bounded. Gentry’s bootstrapping technique [Gen09] resolves this by homomorphically evaluating the decryption circuit itself, restoring a ciphertext to a nominal noise level and thereby enabling unbounded computation. The efficiency of bootstrapping has since become a central bottleneck determining the practicality of FHE schemes.

Programmable bootstrapping Among the main families of FHE schemes, a distinguishing feature of FHEW- and TFHE-like constructions [DM15, CGGI20, BIP⁺22] is that their bootstrapping operation can be augmented with programmability, yielding what is known as *programmable bootstrapping* (PBS) [CJL⁺20, CJP21, Joy22]. Given a ciphertext encrypting a message m , programmable bootstrapping simultaneously refreshes the ciphertext and

Email: olivier.bernard@zama.ai (Olivier Bernard)

evaluates a univariate function f on m , producing a ciphertext with a nominal level of noise. The function f is encoded as a lookup table (LUT), or equivalently as a *test polynomial*, which is homomorphically applied to the input ciphertext through a *blind rotation*. It thus provides a particularly powerful primitive. Over a sufficiently small plaintext domain, an arbitrary function can be evaluated at essentially the cost of a bootstrapping.

Multi-value bootstrapping Introduced by Carpov, Izabachène, and Mollimard [CIM19] in the context of gate bootstrapping, *multi-value bootstrapping* (MVB) amortizes the cost of the blind rotation over several functions of the same encrypted input.

Let v_i be the test polynomial representing a function f_i . The central idea of MVB is to factor each test polynomial as

$$v_i = u_i \cdot v_0 ,$$

where v_0 is common to all functions. A single blind rotation is performed using v_0 . The resulting ciphertext is then multiplied by each function-dependent quotient u_i to recover the effect of a blind rotation with v_i . The dominant part of PBS is therefore shared, while each additional output requires only comparatively inexpensive polynomial operations. The MVB technique has consequently become a useful building block, including for tree-based and chained functional bootstrapping [GBA21, CBSZ23] and their applications [TCBS23, TCB⁺25, TBS26], circuit bootstrapping [WHS⁺25], and multivariate functional bootstrapping [BBR26, WDLL26].

This amortization, however, is not free. Multiplication by u_i amplifies the output-noise variance by a factor $\|u_i\|_2^2$ [CIM19, Theorem 3.3]. In the power-of-two cyclotomic setting, and for an even scaling factor Δ , the original construction fixes the common factor to

$$v_0(x) = \frac{\Delta}{2} (1 + x + \dots + x^{N-1}) . \quad (1)$$

With this choice, the coefficients of the resulting quotient u_i are essentially successive differences between the entries of the LUT for f_i , together with a boundary term involving its endpoints. Hence, the amplification of the noise variance, driven by $\|u_i\|_2^2$, may be moderate for regular functions but can become very large for irregular LUTs, such as cryptographic S-boxes, or whenever large and small output values are interleaved.

This additional noise has an immediate performance impact. Practical TFHE parameters are selected for sub-graphs called *atomic patterns* composed of leveled operations, key switching, and PBS, and must account for the noise accumulated throughout the pattern [CJP21, BBB⁺23]. Large quotient norms can therefore force a more expensive parameter set and partly or entirely erase the savings obtained by sharing the blind rotation. Consequently, reducing the number of blind rotations is not by itself sufficient. The common factor must also be selected with the resulting quotient norms in mind.

Two choices of common factor present themselves immediately, and both are read off directly from the closed-form expression for the test polynomial, independently of the function f being evaluated. As detailed in Section 2, the test polynomial v associated with a general function f , given in Definition 2, contains the factor $\Delta/(1-x)$, itself equal to $\frac{\Delta}{2} (1 + x + \dots + x^{N-1})$; this is precisely the choice of v_0 used by [CIM19] (i.e., Equation (1)). A second, simpler choice is the scalar $v_0 = \Delta$ alone. Since Δ appears explicitly as a factor in Definition 2 and the remaining part of v is already an integer polynomial, the quotient v/Δ is integral without further reduction. This scalar choice recovers the folklore approach (mentioned e.g., in [GBA21, opening paragraph of Section 2.3.2]) of performing a single blind rotation with a constant test polynomial and directly multiplying the resulting ciphertext by each v_i ; see e.g., [MKG24] for a related application. Both choices extend beyond Δ . In particular, the greatest common divisor $\hat{\Delta}$ of all coefficients of the v_i , a multiple of Δ , also leaves the quotients integral [GBA21, Page 235]. Combining both observations gives the common factor $\hat{\Delta}/(1-x)$, the natural counterpart of Equation (1) for a family of LUTs whose coefficients share a factor beyond Δ .

Publicly available uses of the MVB method rely on one of these factors. Going beyond them, selecting v_0 so as to minimize the coefficients-norms of the quotients u_i for a specific collection of LUTs, is a problem that, to the best of our knowledge, has not been addressed.

The low-noise common-factor problem Given test polynomials $v_1, \dots, v_g$, where each v_i encodes the output function f_i , we seek a common factor $\tilde{v}_0$ such that

$$v_i = u_i \cdot \tilde{v}_0 \quad \text{for every } i \in \{1, \dots, g\},$$

where every quotient u_i is *integral* and has *small* coefficients-norm. Specifically, the common factor should be chosen so as to minimize a workload-dependent objective on the quotient coefficients-norms, such as $\max_i \|u_i\|_2$ or an aggregate of these norms $\sum_i \|u_i\|_2$. This is a high-dimensional multiplicative optimization problem in a cyclotomic field, where the selected element must divide all test polynomials while simultaneously producing small quotients. As will be made apparent in Section 2.3, the larger these norms, the larger the parameters required for correctness, and the slower every bootstrapping.

Our new decomposition techniques

We introduce two complementary techniques (Section 3) that tackle exactly this problem.

1. *Log-unit lattice search*: We replace the fixed factor of [CIM19] (cf. Equation (1)) by a common factor $\tilde{v}_0$ adapted to the test polynomials at hand. Integrality of the quotients requires $\tilde{v}_0$ to divide every test polynomial. As the systematic common part of the latter belongs to a power of the totally ramified prime above 2, we search among the elements $\eta \cdot (1 - x)^k$, with η a unit of the cyclotomic field and k an integer. The exponent k contributes a single dimension and is enumerated over a small window, leaving the actual search to bear on the unit part only. In the power-of-two case, the cyclotomic units are freely generated by an explicit family of Kummer units, and a logarithmic embedding turns multiplication by units into addition in the associated log-unit lattice. Selecting a unit then becomes a closest-vector problem. From a reduced basis of that lattice, we compute vectors close to a target aggregated from the embedded test polynomials, and retain the candidate minimizing a workload-dependent objective on the quotient norms.
2. *LUT shifting*: We exploit the fact that a constant may be subtracted from a LUT before bootstrapping and restored afterwards by adding a trivial ciphertext. This transformation preserves the function ultimately evaluated, but modifies the associated test polynomial, enlarging the set of test polynomials that can be considered when selecting a common factor. Searching over suitable shifts therefore produces alternative LUTs for the same functions, whose corresponding quotients may have much smaller coefficients-norms.

Both optimizations are performed offline, for a fixed cyclotomic field and set of LUTs; the lattice reduction is carried out once for the cyclotomic field, and the target-dependent search once per family of LUTs. They leave the structure of the original MVB essentially unchanged. The online phase receives the optimized common factor, its quotients and the shifts, performs a single blind rotation followed by one multiplication per output as in [CIM19], and merely adds a trivial ciphertext to each output to undo the shift. LUT shifting and the log-unit lattice search can be applied independently, although their combination always gives the best results in our experiments.

Benefits and experimental results

When applied to a variety of real-world LUT families, our new decomposition techniques consistently and significantly reduce the maximum squared coefficients-norms of the

quotients, even when many LUTs are involved. For instance, [Examples 1 to 3](#) ([Section 3.2.2](#)) and [Tables 2 and 4](#) ([Section 4](#)) feature reduction factors ranging from $1.75\times$ to $6.59\times$, including a $5.95\times$ ratio for a family of 15 LUTs.

We also validate the approach experimentally in [Section 4](#) on two applications implemented in the `TFHE-rs` library [[Zam](#)], with Gaussian errors, a security level of 128 bits and a failure probability of at most 2^{-128} , in line with the IND-CPA^D model.

The homomorphic evaluation of PRINCEv2 [[BEK⁺20](#)], the successor of the lightweight block cipher PRINCE [[BCG⁺12](#)], serves as a first, drop-in test. Its efficient implementation [[BJ26](#)] is especially suited for multi-value bootstrapping, in particular for the S-box evaluations and the diffusion layers. As multi-value bootstrapping amplifies the noise, it may call for larger parameters, which in turn increase the latency of every individual PBS. With the original MVB, an optimized implementation requires parameters incurring an 11.5 % latency overhead per PBS. Combining the log-unit lattice search with LUT shifting divides the maximum noise variance that must be supported by the circuit by 1.75, so that this per-PBS overhead drops to only 3.5 %, without any other change to the implementation.

Homomorphic integer multiplication then serves as the main illustration ([Section 4.2](#)). The schoolbook multiplication of `TFHE-rs` is tuned for independent bootstrappings. In order for the algorithm to benefit from the reduced variances of the MVB outputs, we reorganize it using *noise-aware foldings* ([Section 4.2.2](#)), whose admissible foldings and schedule are selected offline by simulation. This reorganization is not by itself sufficient. Using the original MVB decomposition still forces a parameter set more than 50 % slower per PBS, so that it overtakes plain independent bootstrappings only at the largest widths. In contrast, combined with our new decompositions, the selected parameters increase the latency of an individual PBS by only 4.6 %, from 14.364 ms to 15.021 ms. In turn, this gives latency speed-ups of respectively $1.18\times$, $1.31\times$, and $1.47\times$, for 64-, 128-, and 256-bit multiplication. For example, at 256 bits, latency decreases from 3.516 s to 2.389 s. Throughput improves by up to $1.59\times$ over independent bootstrappings.

2 Definitions and Notation

Let q denote a ciphertext modulus, $p = 2t$ with $p < q$ a plaintext modulus and $\Delta = \lfloor \frac{q}{p} \rfloor$ the message encoding offset. Messages are taken in $\mathcal{M} := \mathbb{Z}/p\mathbb{Z}$, which we represent throughout by its centered representatives $\llbracket -t, t-1 \rrbracket$. We write $\lfloor y \rfloor$ for $\lfloor y + \frac{1}{2} \rfloor$. For $M \not\equiv 2 \pmod{4}$, $\mathcal{K}_M$ denotes the M -th cyclotomic field, defined as $\mathcal{K}_M = \mathbb{Q}[x]/\langle \Phi_M(x) \rangle$ where Φ_M is the M -th cyclotomic polynomial of degree $N = \varphi(M)$. The corresponding ring of cyclotomic integers is $\mathcal{R} = \mathbb{Z}[x]/\langle \Phi_M(x) \rangle$, and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. The coefficients-norm of $v(x) = \sum_i v_i x^i \in \mathcal{R}$ is defined as $\|v\|_2 = \sqrt{\sum_i v_i^2}$. For some support S , the uniform distribution over S is denoted $\mathcal{U}(S)$. The distribution $\mathcal{G}_{\sigma^2}$ refers to a centered discrete Gaussian distribution of variance σ^2 . Sampling an element d from a distribution $\mathcal{D}$ is written as $d \leftarrow \mathcal{D}$.

Throughout, and for concreteness, M , q and p (and thus also Δ and $N = M/2$) are all powers of two with $p < N < q$, which is the typical TFHE setting. In particular, $\Phi_M(x) = x^N + 1$ for the power-of-two conductor $M = 2N$.

2.1 Blind Rotation

We recall here the core mechanisms of the FHEW- and TFHE-like schemes that are needed in the sequel, and refer to [[Joy22](#), [BBB⁺25](#)] for extensive details. [Figure 1](#) depicts a complete programmable bootstrapping, as it occurs within an atomic pattern. Starting from a GLWE ciphertext, it chains a sample extraction, a key switch, a modulus switch down to $2N$, and finally the *blind rotation*, shaded in the figure, which is where the evaluation takes place, and is the costliest of the four. Its output is again a GLWE ciphertext in the

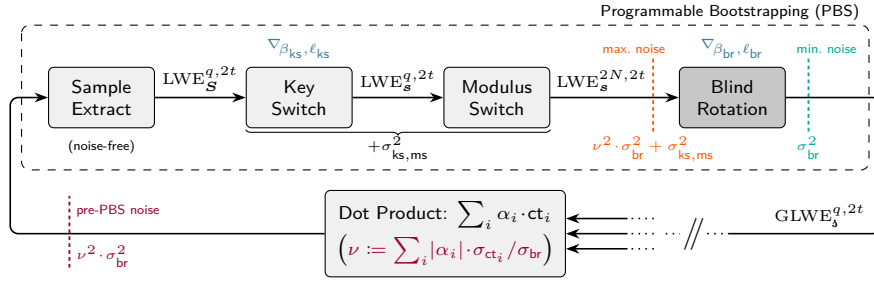


Figure 1: Atomic pattern and programmable bootstrapping (PBS). The noise variances shown are worst-case upper bounds, valid without any independence assumption on the ciphertexts.

input format, with its noise reset to a nominal level. Only the blind rotation is modified in this paper, so it is the only step we describe in detail. The noise annotations of Figure 1 and the leveled operations that close the pattern are discussed in Section 2.3.

Ciphertext format TFHE operates on both LWE (*Learning with Errors* [Reg09]) and GLWE (*generalized-LWE* [SSTX09, LPR10, LS15]) ciphertexts. As above, an LWE ciphertext carries a message $m \in \mathcal{M}$, encoded as $\Delta \cdot m \in \mathbb{Z}/q\mathbb{Z}$ prior to encryption. In the general case, a message $m \in \mathcal{R}_p$ is encoded coefficient-wise as $\Delta \cdot m \in \mathcal{R}_q$.

Definition 1 (GLWE and LWE ciphertexts). For a parameter $k \in \mathbb{N}$, let $\mathfrak{s} = (s_1, \dots, s_k) \in \mathcal{R}^k$ be the private key, where each s_i is sampled from a uniform binary, uniform ternary or Gaussian distribution over $\mathcal{R}$. Let also $\mathfrak{a} = (a_1, \dots, a_k) \leftarrow \mathcal{U}(\mathcal{R}_q)^k$ be the mask and let $e \leftarrow \mathcal{G}_{\sigma^2}(\mathcal{R})$ be the noise. Then, a GLWE encryption of $\Delta m \in \mathcal{R}_q$ under the private key $\mathfrak{s}$ is defined as

$$\text{GLWE}_{\mathfrak{s}}^{q,p}(\Delta m) \leftarrow (a_1, \dots, a_k, b = \sum_{i=1}^k a_i \cdot s_i + \Delta m + e) \in \mathcal{R}_q^{k+1}.$$

Similarly, an LWE encryption of dimension $n \in \mathbb{N}$ of $\Delta m \in \mathbb{Z}/q\mathbb{Z}$ under the scalar private key $s = (s_1, \dots, s_n)$ is defined as

$$\text{LWE}_s^{q,p}(\Delta m) \leftarrow (a_1, \dots, a_n, b = \sum_{i=1}^n a_i \cdot s_i + \Delta m + e) \in (\mathbb{Z}/q\mathbb{Z})^{n+1},$$

with $\mathfrak{a} := (a_1, \dots, a_n) \leftarrow \mathcal{U}(\mathbb{Z}/q\mathbb{Z})^n$ and $e \leftarrow \mathcal{G}_{\sigma^2}(\mathbb{Z})$.

Gadget decompositions A crucial ingredient for containing the noise during homomorphic operations is the *gadget decomposition* concept [BGV14]. A gadget decomposition $\nabla_{\beta,\ell}$ of quality β and level ℓ w.r.t. a gadget vector $\mathfrak{g} = (g_1, \dots, g_\ell)$ is a function $\nabla: \mathcal{R}_q \rightarrow \mathcal{R}^\ell$ such that for any $f \in \mathcal{R}_q$, the amplitude of all coefficients in ∇f is bounded by β and $\langle \nabla f, \mathfrak{g} \rangle \approx f$ [CGGI20, BJ25], up to a decomposition error controlled by β and ℓ . Both β and ℓ play an essential role when choosing TFHE parameters, see Section 2.3; distinct gadget decompositions may be used in distinct contexts, see e.g., Figure 1.

Test polynomial Functions of the message space are viewed as lookup tables (LUTs) whose values are repeated in the coefficients of so-called *test polynomials*. This is what effectively makes TFHE a *programmable* scheme. Test polynomials are the input of the blind rotation described next, which rotates them by an amount determined by the encrypted message, thereby moving the coefficient holding the corresponding LUT entry into the constant position, from where it is later extracted.

Definition 2 (Test polynomial). Let $f: \mathcal{M} \rightarrow \mathcal{M}$ be a negacyclic function, with $\mathcal{M}$ represented as $\llbracket -t, t-1 \rrbracket$ where $t = p/2$. The *test polynomial* associated with f in $\mathcal{K}_{2N}$ is defined as

$$v(x) := \underbrace{x^{-N/2t}}_{\text{centering}} \cdot \underbrace{\left(\frac{1-x^{N/t}}{1-x} \right)}_{\text{redundancy}} \cdot \underbrace{\sum_{m=0}^{t-1} \Delta \cdot f(m) \cdot x^{m \cdot N/t}}_{\text{LUT part}} \mod x^N + 1.$$

In particular, v depends on f only through its values on $\llbracket 0, t-1 \rrbracket$.

The role of the initial rotation by $x^{-N/2t}$ is to center the rounding-error correction around 0. The fractional factor accounts for the redundancy of the evaluations of f in the test polynomial; indeed $\frac{1-x^{N/t}}{1-x} = \sum_{i=0}^{N/t-1} x^i$, thereby repeating coefficients when multiplied by the sparse polynomial encoding f every N/t positions. Finally, multiplying by Δ encodes the output values in $\llbracket -\frac{q}{2}, \frac{q}{2} - 1 \rrbracket$.

Blind rotation At the heart of the programmable bootstrapping, the blind rotation is primarily meant to reset the noise of ciphertexts to a nominal level of variance σ_{br}^2 that is *independent* of the input noise. Let

$$\text{ct}_{\text{in}} = \left(\bar{a} = (\bar{a}_1, \dots, \bar{a}_n), \bar{b} = \sum_{i=1}^n \bar{a}_i s_i + \frac{N}{t}m + \bar{e} \right) \in \text{LWE}_{\mathbf{s}}^{2N, 2t} \left(\frac{N}{t}m \right)$$

be a valid ciphertext modulo $2N$, where $|\bar{e}| < \frac{N}{2t}$ and $\mathbf{s}$ is binary, as we describe the GINX variant [GINX16]. The blind rotation uses an accumulator initialized to a given test polynomial v as in Definition 2, and homomorphically evaluates $x^{-\bar{b} + \langle \bar{a}, \mathbf{s} \rangle} \cdot v(x)$ using a sequence of n homomorphic CMuxes [CGG120, Section 3.4]. Additional key material is required, known as bootstrapping keys $\text{bsk}[1 \dots n]$, which are a collection of GGSW [GSW13] encryptions of the s_i under the GLWE key $\mathbf{s}$. Algorithmically, $\text{BlindRotate}(\text{ct}_{\text{in}}, v, \text{bsk})$ is

```

acc ← (0, ..., 0, x- $\bar{b}$  · v)                                /* Trivial, noiseless GLWEsq, 2t encryption */
for i = 1 to n do
  acc ← acc + ((x $\bar{a}_i$  - 1) · acc) ⊗ bsk[i]                /* Homomorphic CMux branching on si */
return acc                                                    /* GLWEsq, 2t encryption of x-( $\frac{N}{t}m + \bar{e}$ ) · v(x) */

```

Writing $j := \frac{N}{t}m + \bar{e}$ for the resulting rotation amount, the constant coefficient of the encrypted polynomial is

$$\text{coeff}_0(x^{-j} \cdot v(x)) = \begin{cases} \Delta \cdot f(m) & \text{if } m \in \llbracket 0, t-1 \rrbracket, \\ -\Delta \cdot f(m+t) & \text{if } m \in \llbracket -t, -1 \rrbracket, \end{cases} \quad (2)$$

where $m = \lfloor tj/N \rfloor$ exactly, the rounding absorbing $\bar{e}$ thanks to $|\bar{e}| < \frac{N}{2t}$. Indeed, the redundancy factor repeats each value $\Delta f(m)$ over N/t consecutive coefficients, the centering factor places this block around the exponent mN/t , and the relation $x^N = -1$ negates the blocks wrapping past the degree. The blind rotation thus evaluates f on $\llbracket 0, t-1 \rrbracket$ and $-f(\cdot + t)$ on $\llbracket -t, -1 \rrbracket$, a *negacyclicity* imposed by the cyclotomic field $\mathcal{K}_{2N}$. An arbitrary function can nonetheless be programmed on the half $\llbracket 0, t-1 \rrbracket$ of $\mathcal{M}$ alone, the first case of Equation (2), at the cost of halving the usable message space. This is the usual setting of TFHE-rs, in which the most significant plaintext bit is reserved as a padding bit and kept at zero.

2.2 Multi-Value Bootstrapping

The multi-value bootstrapping (MVB) technique, outlined in [Section 1](#), rests on expressing each test polynomial as the product of a common polynomial and a function-dependent one, together with a fixed choice for the former. We now make both explicit. The common polynomial used for the mutualized blind rotation is defined in [\[CIM19\]](#) on the torus for binary plaintexts as $v_0(x) = \frac{1}{2} \cdot \sum_{i=0}^{N-1} x^i$. Using our notations, this corresponds to the polynomial

$$v_0(x) = \frac{\Delta}{1-x} \mod x^N + 1, \quad (3)$$

whose expanded form is [Equation \(1\)](#). Note that v_0 has integer coefficients exactly because the algebraic norm of $(1-x)$ is $2 \mid \Delta$. This particular factor is naturally read out of the formula in [Definition 2](#), and therefore divides every test polynomial.

Then, any test polynomial v can be factored as $v = u \cdot v_0$, where the quotient polynomial $u(x) := x^{-N/2t} \cdot \sum_{j=0}^{t-1} u_j \cdot x^{jN/t}$ is defined coefficient-wise by ([\[CIM19, Theorem 3.2\]](#))

$$\begin{cases} u_0 = f(0) + f(t-1), \\ u_j = f(j) - f(j-1) \quad \text{for } 1 \leq j \leq t-1. \end{cases} \quad (4)$$

Indeed, from [Definition 2](#) and [Equation \(3\)](#), $u(x) = x^{-N/2t} \cdot (1-x^{N/t}) \cdot \sum_{m=0}^{t-1} f(m) \cdot x^{m \cdot N/t}$, so the result simply follows from the fact that $x^N = -1$. In particular, one observes in this setting that the quotient polynomial u is always sparse and that, except for the constant coefficient u_0 , its coefficients are successive differences of the values of f , so that their sizes depend on the particular shape of f .

Consider now g distinct functions $f_1, \dots, f_g$, encoded by the test polynomials $v_1, \dots, v_g$. To evaluate all of them on the same input ciphertext, the MVB technique of [\[CIM19\]](#) runs the blind rotation only once on the common polynomial v_0 of [Equation \(3\)](#), and multiplies its encrypted output by each of the quotients $u_i := v_i/v_0$. However, the multiplication by u_i multiplies the output-noise variance by $\|u_i\|_2^2$ [\[CIM19, Theorem 3.3\]](#). Depending on the shape of f_i , this factor can be significant, reaching $t(2t)^2 = 4t^3$ [\[CIM19, Proposition 1\]](#) in the worst case, each coefficient of a quotient u_i being at most $2t$ in absolute value in the centered representation $\llbracket -t, t-1 \rrbracket$. The resulting procedure is summarized algorithmically as $\text{MVBlindRotate}^{v_0}(\text{ct}_{\text{in}}, \{v_1, \dots, v_g\}, \text{bsk})$:

```

ct0 ← BlindRotate(ctin, v0, bsk)                                /* A single shared blind rotation */
for i = 1 to g do
    ui ← vi/v0
    cti ← ui · ct0                                              /* Multiplies noise variance by \|ui\|22 */
return (ct1, ..., ctg)                                          /* GLWEsg,2t encryptions of x-N/t · vi(x) */

```

In practice, the quotients u_i may be provided directly as inputs, each of them being given by [Equation \(4\)](#) applied to f_i .

2.3 Atomic Patterns and Noise Budget

Graphs of FHE computations can be written as series of atomic patterns, namely small subgraphs combining leveled operations with a programmable bootstrapping. [Figure 1](#) depicts the atomic pattern considered in this paper. Its leveled part is a *dot product*: input ciphertexts $\text{ct}_i \in \text{GLWE}_s^{g,2t}(\Delta m_i)$ are combined as

$$\text{ct}_{\text{out}} = \sum_i \alpha_i \cdot \text{ct}_i \in \text{GLWE}_s^{g,2t}(\Delta \cdot \sum_i \alpha_i m_i)$$

for known integer weights α_i . As pictured by the interrupted arrows, these inputs may come out of a blind rotation, of this pattern or of another one in the graph, or from elsewhere,

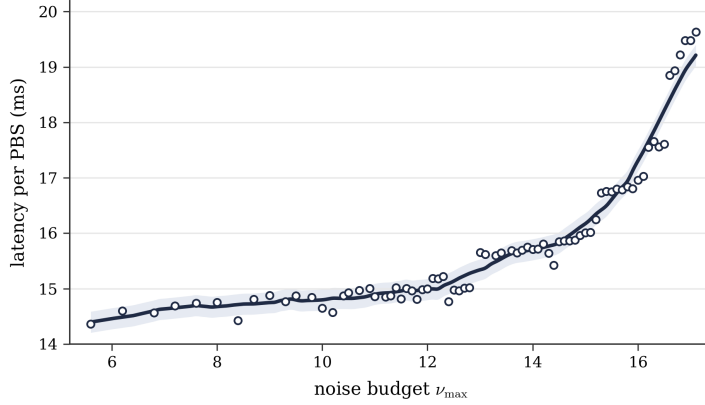


Figure 2: Measured per-PBS latency as a function of the noise budget $\nu_{\max}$.

e.g., fresh encryptions. The resulting ciphertext ct_{out} then traverses the PBS, whose blind rotation resets the noise to the nominal variance σ_{br}^2 , independently of the accumulated input noise. The noise annotations of Figure 1 are worst-case bounds. Writing σ_{ct_i} for the standard deviation of the noise of ct_i , the dot product outputs a ciphertext of standard deviation at most

$$\sum_i |\alpha_i| \cdot \sigma_{\text{ct}_i} =: \nu \cdot \sigma_{\text{br}}. \quad (5)$$

Standard deviations are summed rather than variances. Indeed, the inputs, sharing ancestors in the computation graph, are generally correlated, and the bound is attained under full correlation. The sample extraction adds no noise, while the key switch and modulus switch add $\sigma_{\text{ks,ms}}^2$, so the noise culminates at $\nu^2 \cdot \sigma_{\text{br}}^2 + \sigma_{\text{ks,ms}}^2$ right before the blind rotation, and is reset to σ_{br}^2 right after.

This maximal noise level is what correctness constrains. The blind rotation returns the correct value as long as the rounding of Equation (2) absorbs the accumulated error, which fails with probability at most p_{fail} provided

$$\nu^2 \cdot \sigma_{\text{br}}^2 + \sigma_{\text{ks,ms}}^2 \leq \left(\frac{\Delta/2}{r(p_{\text{fail}})} \right)^2, \quad (6)$$

where $r(p_{\text{fail}}) := \sqrt{2} \operatorname{erfc}^{-1}(p_{\text{fail}})$ is the $(1 - p_{\text{fail}})$ -quantile of the standard half-normal distribution; e.g., $r \approx 13.11$ for $p_{\text{fail}} = 2^{-128}$. The failure probability is not only a correctness matter. Under the IND-CPA^D security model [LM21], and its recent strengthening [BJSW25], decryption failures can give rise to concrete attacks when p_{fail} is insufficiently small [CCP⁺24, CSBB24]. We accordingly target $p_{\text{fail}} \leq 2^{-128}$ throughout.

The correctness constraint caps the noise a ciphertext may accumulate before being bootstrapped, and thereby the dot products an application may perform. Cryptographic parameters are accordingly generated for a prescribed bound $\nu_{\max}$ on the multiplication factor ν , so that an atomic pattern parameterized for $\nu_{\max}$ supports any dot product with $\sum_i |\alpha_i| \cdot \sigma_{\text{ct}_i} \leq \nu_{\max} \cdot \sigma_{\text{br}}$. Considering one atomic pattern whose output is looped back into its input then suffices to fix all the parameters, namely the dimensions n and $k \cdot N$, the noise levels, and the gadget parameters $(\beta_{\text{ks}}, \ell_{\text{ks}})$ and $(\beta_{\text{br}}, \ell_{\text{br}})$ of the two decompositions appearing in Figure 1. For each value of $\nu_{\max}$, these are selected by a noise-aware optimizer in the spirit of [BBB⁺23], minimizing the algorithmic cost of the PBS subject to the correctness constraint given in Equation (6) and to a security level of at least 128 bits. The evaluation keys, such as bsk , are then shared across all atomic patterns of the graph.

Supporting a larger $\nu_{\max}$ forces larger parameters and thus a slower bootstrapping. Figure 2 quantifies this dependence. For each value of the noise budget $\nu_{\max}$, a parameter

set is generated and its bootstrapping latency is measured. Representative parameter sets used in our experiments, together with exact figures, are listed in [Table 1](#).

This trade-off is central to this paper. The noise of a multi-value bootstrapping output has standard deviation $\sigma_{\text{ct}_i} = \|u_i\|_2 \cdot \sigma_{\text{br}}$, so that it enters a subsequent dot product with integer weight α_i but with its standard deviation scaled by $\|u_i\|_2$. This is why ν is measured relative to σ_{br} in [Equation \(5\)](#). Reducing the quotient norms lowers each σ_{ct_i} , hence the budget ν_{max} the parameters must accommodate.

3 Reducing the Noise Amplification

As mentioned in [Section 2.2](#), the multi-value bootstrapping multiplies the noise variance of each output by the squared coefficients-norm of the quotient of its test polynomial by the common factor. The common factor of [Equation \(3\)](#) being fixed once and for all, these norms are entirely determined by the functions being evaluated. This section revisits that choice. Our main technique, called *log-unit lattice search*, adapts the common factor to the given test polynomials and expresses its selection as a geometric problem. A second technique, called *LUT shifting*, adds a further degree of freedom by acting on the LUTs themselves. The two can be combined, and both are carried out offline, before any encrypted computation.

Remark 1. The techniques of this paper are expected to extend to general cyclotomics. Test polynomials can be adapted to that setting as described in [\[JW22, CKL26\]](#). Our method may then use the explicit circular units from [\[Ku92\]](#), together with the additional elements from [\[BLNR22\]](#) and the saturation techniques from [\[BFL24\]](#).

3.1 Log-Unit Lattice Search

Our first technique leverages so-called *logarithmic embeddings* in order to transform the multiplicative problem at hand into an additive representation that is amenable to classical lattice techniques.

Let $v_1, \dots, v_g \in \mathcal{R}$ be the considered set of test polynomials as given in [Definition 2](#). The goal is to find $\tilde{v}_0 \in \mathcal{R}$ dividing every v_i , i.e., such that all quotients $u_i := v_i / \tilde{v}_0$ belong to $\mathcal{R}$, while keeping a workload-dependent function of their coefficients-norms small, e.g., $\sum_{i=1}^g \|u_i\|_2$ or $\max\{\|u_i\|_2 \mid i \in \llbracket 1, g \rrbracket\}$. The choice of this function has no impact on the search itself; it only governs the final selection among the produced candidates.

3.1.1 Common factor search space

Let $\hat{\Delta}$ be the gcd of the coefficients of all v_i , which is a multiple of Δ . Without loss of generality, we directly search for a common factor $\hat{v}_0$ of the normalized polynomials $\hat{v}_i := v_i / \hat{\Delta}$, which have no non-trivial common integer factor, and set $\tilde{v}_0 = \hat{\Delta} \cdot \hat{v}_0$, which preserves all quotients. In particular, $\hat{v}_0$ tolerates denominators as long as they divide $\hat{\Delta}$.

Since each quotient $u_i = v_i / \tilde{v}_0 = \hat{v}_i / \hat{v}_0$ needs to be an integer polynomial, a natural target for $\hat{v}_0$ is the group of units $\mathcal{R}^\times$ of the cyclotomic field $\mathcal{K}_{2N}$, i.e., invertible elements of $\mathcal{R}$. This would however miss the usual divisor $v_0 = \frac{\hat{\Delta}}{1-x}$ from [\[CIM19, GBA21\]](#) since $\frac{1}{1-x}$ is not a unit. In all generality, the valuations of $\hat{v}_0$ at any prime ideal $\mathfrak{p}$ must not exceed the minimum of the valuations of the $\hat{v}_i$ at $\mathfrak{p}$. One (very expensive) possibility would then be to compute the prime ideal factorization of the gcd ideal $\mathfrak{d}$ of $\langle \hat{v}_1 \rangle, \dots, \langle \hat{v}_g \rangle$ (e.g., by computing the gcd of their algebraic norms and taking the minimum valuations at prime ideals dividing this gcd), and search for $\hat{v}_0$ amongst the so-called subgroup of $\mathfrak{d}$ -units of $\mathcal{K}_{2N}$, i.e., elements whose valuations are exactly 0 at all prime ideals not dividing $\mathfrak{d}$.

However, $\mathfrak{d}$ typically collapses to a power of the totally ramified prime ideal $\mathfrak{e} := \langle 1-x \rangle$ of norm 2 and we restrict the search accordingly. Indeed, the algebraic norm of the

redundancy factor $(1 - x^{N/t})/(1 - x)$ is always a power of 2, thus contributing only powers of ϵ . Meanwhile, the primes dividing the algebraic norms of the LUT parts of $\hat{v}_i$, once normalized by $\hat{\Delta}$, i.e., $\sum_{m=0}^{t-1} \frac{\Delta}{\hat{\Delta}} f_i(m) \cdot x^{m \cdot N/t}$, tend to behave as those of random elements. Since prime ideals of small odd norm are scarce in power-of-two cyclotomic fields, a shared odd prime is unlikely; as a matter of fact, odd primes very rarely showed up in our experiments. Note this restriction could at worst affect optimality.

In this specific case of ϵ -units in $\mathcal{K}_{2N}$, a simple description as a freely generated multiplicative group exists. The first ingredient is the so-called *circular units* subgroup;² see e.g., [Was97, Section 8.1] for a definition. In the power-of-two case, the subgroup of circular units $\mathcal{U}^\times$ is (freely, multiplicatively, modulo torsion) generated by the following $(N/2 - 1)$ Kummer units [Was97, Lemma 8.1]

$$\mathcal{U}^\times := \left\langle \eta_a := x^{(1-a)/2} \cdot \frac{1-x^a}{1-x} \mid a \in \llbracket 2, N \rrbracket, \gcd(a, 2) = 1 \right\rangle. \quad (7)$$

Assuming Weber’s conjecture,³ we have that $\mathcal{R}^\times = \mathcal{U}^\times$ according to [Sin78, Theorem p. 107] on the multiplicative index $[\mathcal{R}^\times : \mathcal{U}^\times]$. Further, by definition of ϵ , a fundamental element for the ϵ -part of the ϵ -units group is trivially given by $(1 - x)$. Hence, this completes the algebraic description of our search space for $\hat{v}_0$, i.e., as elements of the form

$$\hat{v}_0 = \tilde{v}_0 / \hat{\Delta} = \eta \cdot (1 - x)^k, \quad \eta \in \mathcal{U}^\times, \quad k \in \mathbb{Z}. \quad (8)$$

Notably, the original v_0 from [CIM19, GBA21] corresponds to $k = -1$, $\eta = 1$.

Remark 2. Units and ϵ -units technically contain the torsion subgroup generated by x . Since this torsion part has no impact on the size of the quotients in power-of-two conductor cyclotomic fields, all elements are equivalent *up to torsion* as far as we are concerned. Therefore, for conciseness, we almost systematically omit the “up to torsion” part.

Remark 3. Even if Weber conjecture were to be false, circular units have finite (multiplicative) index h_{2N}^+ in $\mathcal{R}^\times$. We may then obtain a complete description by saturation techniques, i.e., searching for a power-product of circular units that is an e -th power, for each factor $e \mid h_{2N}^+$, and extract its e -th root, see e.g., [BFL24].

3.1.2 Logarithmic embeddings

A logarithmic embedding denotes any injective homomorphism Log from the multiplicative group of $\mathcal{K}_{2N}$ (modulo torsion) to the additive group of some vector space, i.e., satisfying $\text{Log}(\alpha \cdot \beta) = \text{Log} \alpha + \text{Log} \beta$ for any $\alpha, \beta \in \mathcal{K}_{2N}^\times$. In other words, a logarithmic embedding transforms a multiplicative instance of the problem into an additive one.

There are plenty of classical choices for logarithmic embeddings, see e.g., [BLNR22, Section 2.3]. We only detail the simplest option here. Let $\text{Hom}(\mathcal{K}_{2N}, \mathbb{C}) / \langle \tau_{-1} \rangle$ be the group of $N/2$ complex embeddings of $\mathcal{K}_{2N}$ up to complex conjugation, i.e., all homomorphisms $\tau_a : x \mapsto \zeta^a$ for odd $a \in \llbracket 1, N \rrbracket$, where ζ is any (fixed) fundamental complex $2N$ -th root of unity, e.g., $\zeta = \exp(2i\pi/(2N))$. Hence, the following map qualifies as a logarithmic embedding:

$$\text{Log}_\infty \alpha := \left(\ln |\tau(\alpha)| \mid \tau \in \text{Hom}(\mathcal{K}_{2N}, \mathbb{C}) / \langle \tau_{-1} \rangle \right). \quad (9)$$

By Dirichlet’s unit theorem, the image $\text{Log}_\infty \mathcal{R}^\times$ of the units group under this logarithmic embedding is the so-called *log-unit lattice* which spans a hyperplane in $\mathbb{R}^{N/2}$, and is, for this choice of Log_∞ , orthogonal to the all-ones vector. From the fundamental elements η_a

²This is the historical terminology, see e.g., [Sin78, Kuč92], though [Was97] uses *cyclotomic units*.

³This conjecture states that the class number of the maximal real subfield of cyclotomic fields whose conductor is a power-of-two is 1, i.e., formally $h_{2^e}^+ = 1$ for any $e > 1$.

described in Equation (7), it is easy to obtain a basis $\mathcal{B}_{\mathcal{U}^\times} := (\text{Log}_\infty \eta_a \mid a \in \llbracket 2, N \rrbracket, a \text{ odd})$ of this lattice.

Additionally, the log-unit lattice offers a particularly nice geometry, as quantitatively proven in [CDPR16], and experimentally battle-tested e.g., in [BLNR22]. As a matter of fact, we were able, using `fp111 v5.5.0` [FPL23], to obtain a BKZ- β -reduced basis of the log-unit lattice $\text{Log}_\infty \mathcal{R}^\times$ for $\beta = N/2 - 1 = 1023$ (i.e., effectively an HKZ-reduced basis), in about 8 minutes. Finally, we stress that this pre-computation of a reduced basis of the log-unit lattice only needs to be performed *once* for a particular GLWE dimension N , independently of the input test polynomials v_i .

Remark 4. Under this definition of Log_∞ , the whole group of $\mathfrak{e}$ -units (i.e., not only $\mathcal{R}^\times$) is sent to a full-rank lattice in $\mathbb{R}^{N/2}$, since the description of $\mathfrak{e}$ -units involves exactly *one* additional fundamental element. A logarithmic embedding that smoothly generalizes beyond this single-prime case may alternatively append a valuation coordinate as, e.g.,

$$\text{Log}_{\infty, \mathfrak{e}} \alpha := \left(\{2 \ln |\tau(\alpha)| \mid \tau \in \text{Hom}(\mathcal{K}_{2N}, \mathbb{C}) / \langle \tau_{-1} \rangle\}, -v_{\mathfrak{e}}(\alpha) \cdot \ln 2 \right).$$

The weighting scheme chosen here guarantees orthogonality to the all-ones vector in $\mathbb{R}^{N/2+1}$. Identical geometric characteristics have been observed on the obtained lattice [BLNR22].

3.1.3 Finding the common factor

We first follow the classical blueprint in algorithmic number theory for reducing an element modulo units. In order to reduce $v \in \mathcal{R} \setminus \{0\}$ modulo $\mathcal{R}^\times$, one usually computes a close element $w \in \text{Log}_\infty \mathcal{R}^\times$ to (an orthogonal projection of) $v := \text{Log}_\infty v$, lifts w back to $\tilde{w} \in \mathcal{R}^\times$ such that $w = \text{Log}_\infty \tilde{w}$, and outputs $v/\tilde{w} \in \mathcal{R}$.

From the g logarithmic images $v_i := \text{Log}_\infty \hat{v}_i$, we compute an aggregated target $\tilde{v}$ in the logarithmic space,⁴ e.g., their barycenter, geometric or coordinate-wise median, or bounding-box midpoint.⁵ Then, we compute a close vector $\tilde{w}$ to the projection of $\tilde{v}$ into the log-unit lattice using Babai's Nearest Plane algorithm [Bab86]. As a byproduct, this also yields the combination vector $\mathbf{y} = (y_1, \dots, y_{N/2-1})$ such that $\tilde{w} = \mathbf{y} \cdot \mathcal{B}_{\mathcal{U}^\times}$, hence it is easy to compute a lift of $\tilde{w}$ to a candidate $\hat{v}_0 \in \mathcal{R}$ such that $\tilde{w} = \text{Log}_\infty \hat{v}_0$ as the product $\hat{v}_0 := \prod_{a \in \llbracket 2, N \rrbracket, a \text{ odd}} \eta_a^{y_a(a-1)/2}$. Since each of the $(\text{Log}_\infty \hat{v}_i - \text{Log}_\infty \hat{v}_0)$ are small by construction, we expect, heuristically, that the quotients $u_i = \hat{v}_i/\hat{v}_0$ have reduced coefficients.

The process may also be iterated by randomly shifting the aggregated target $\tilde{v}$ inside a ball of small radius and retaining the best found candidate.

From units to $\mathfrak{e}$ -units In the above description, we deliberately ignored $\mathfrak{e}$ -units that are not units, i.e., we did not consider candidates $\hat{v}_0$ with $k \neq 0$ in the above parametrization. A tempting solution consists in running the above framework inside the whole $\mathfrak{e}$ -unit lattice from Remark 4. However, it is then a bit cumbersome to ensure that all quotients have integer coefficients, i.e., that they have *non-negative* valuation at $\mathfrak{e}$ [BLNR22, Section 5.4, §*Reconstruction of a solution*]. This may lead to unsatisfactory results in practice.

In our case, the (multiplicative) extra dimension introduced by the $\mathfrak{e}$ -units is 1, hence it is easier to brute-force out the power of $(1-x)$ to include in $\hat{v}_0$ and perform the lattice search only on the units part as above. Indeed, it is always sufficient to consider powers of $(1-x)$ inside a window of size N , since we stripped away from the v_i the gcd of their coefficients and since $\langle 2 \rangle = \mathfrak{e}^N$. Further, ensuring that all quotients have integer

⁴Note that this aggregated target is not even required to belong to $\text{Log}_\infty \mathcal{K}_{2N}^\times$.

⁵Any reasonable notion of center is suitable, and led to equivalent selections in our experiments; the reported results use the barycenter.

Algorithm 1: Log-Unit Lattice Search

Input: $v_1, \dots, v_g \in \mathcal{R}$, score function ϕ on quotient norms
Data: Reduced basis $\mathcal{H}_{\mathcal{U}^\times} = \mathbf{U} \cdot \mathcal{B}_{\mathcal{U}^\times}$ of $\text{Log}_\infty \mathcal{R}^\times$
Output: Candidate common factor $\tilde{v}_0$, quotients $u_i := v_i/\tilde{v}_0$, global score

1. Normalize each $\hat{v}_i \leftarrow v_i/\hat{\Delta}$, where $\hat{\Delta} \leftarrow \gcd(\text{coeffs}(v_1), \dots, \text{coeffs}(v_g))$
2. Precompute the log-embeddings $\mathbf{v}_e \leftarrow \text{Log}_\infty(1-x)$, $\mathbf{v}_i \leftarrow \text{Log}_\infty \hat{v}_i$
/* Brute-force the valuation at $\mathfrak{e} = \langle 1-x \rangle$ of the common factor */
3. Compute $v_{\max} \leftarrow \min\{v_{\mathfrak{e}}(\hat{v}_i) \mid i \in \llbracket 1, g \rrbracket\}$ from the $\hat{v}_i$ algebraic norms
4. **for** $k = v_{\max}$ **down to** $v_{\max} - N + 1$ **do**
5. Set aggregated target to $\tilde{\mathbf{v}} \leftarrow \frac{1}{g} \sum_{i=1}^g \mathbf{v}_i - k \cdot \mathbf{v}_e$ /* e.g., ℓ_2 -norm barycenter */
6. Find a close vector as $\mathbf{w}, \mathbf{y}_H \leftarrow \text{Babai NP}(\tilde{\mathbf{v}}, \mathcal{H}_{\mathcal{U}^\times})$ s.t. $\mathbf{w} = \mathbf{y}_H \cdot \mathcal{H}_{\mathcal{U}^\times}$
7. Lift $\mathbf{w}$ to $\mathcal{R}$ as $\hat{v}_0^{(k)} \leftarrow (1-x)^k \cdot \prod_{a \in \llbracket 2, N \rrbracket, a \text{ odd}} \eta_a^{y_{(a-1)/2}}$ where $\mathbf{y} \leftarrow \mathbf{y}_H \cdot \mathbf{U}$
8. Select $\hat{v}_0$ as the $\hat{v}_0^{(k)}$ minimizing $\phi(\{\|\hat{v}_i/\hat{v}_0^{(k)}\|_2 \mid i \in \llbracket 1, g \rrbracket\})$ /* (on the fly) */
9. **return** $\tilde{v}_0 \leftarrow \hat{\Delta} \cdot \hat{v}_0$, quotients $u_i \leftarrow \hat{v}_i/\hat{v}_0$, score $\phi(\{\|u_i\|_2 \mid i \in \llbracket 1, g \rrbracket\})$

coefficients requires the valuation of $\hat{v}_0$ at $\mathfrak{e}$ not to exceed some $v_{\max}$, set to the minimum valuation over the $\hat{v}_i$ at $\mathfrak{e}$.

Hence, the loop ranges over $\llbracket v_{\max} - N + 1, v_{\max} \rrbracket$, without loss of generality, and selects the best candidate depending on the workload-dependent objective for the coefficients-norms of the quotients. This yields Algorithm 1.

3.2 LUT Shifting

Our second technique acts on the test polynomials themselves rather than on the common factor. It relies on the observation that a blind rotation may be programmed with a *shifted* version of the target function, the shift being undone afterwards at negligible cost. Since the function ultimately evaluated is unchanged, this yields a whole family of test polynomials representing the same function, any member of which may be used in the log-unit search of Section 3.1 for a multi-value decomposition.

3.2.1 Shifting a lookup table

Shifting a function generally breaks the negacyclicity of Equation (2). We therefore first treat the case in which messages lie in $\llbracket 0, t-1 \rrbracket$, the first case of Equation (2), as when the most significant plaintext bit is reserved and kept at zero. Fix $\delta \in \mathcal{M}$ and consider the shifted function

$$f^{(\delta)}: \llbracket 0, t-1 \rrbracket \longrightarrow \mathcal{M}, \quad m \longmapsto f(m) - \delta \pmod{2t}.$$

Its test polynomial $v^{(\delta)}$ is obtained by substituting $f^{(\delta)}$ for f in Definition 2, which by construction uses only the values on $\llbracket 0, t-1 \rrbracket$.

Programming $f(m) - \delta$ and restoring $f(m)$ Replacing v by $v^{(\delta)}$ programs $f^{(\delta)}$ in place of f . Hence, writing j for the rotation amount resulting from the modulus switch, the first case of Equation (2), i.e., for $m = \lfloor \frac{t}{N} \rfloor \in \llbracket 0, t-1 \rrbracket$, gives

$$\text{coeff}_0(x^{-j} v^{(\delta)}(x)) = \Delta \cdot f^{(\delta)}(m) = \Delta (f(m) - \delta).$$

The ciphertext feeding the sample extraction therefore encrypts $x^{-j} v^{(\delta)}$, whose constant coefficient is $\Delta(f(m) - \delta)$ rather than $\Delta f(m)$. The original value is recovered by adding

the trivial, noiseless GLWE encryption of the constant $(0, \dots, 0, \Delta \delta) \in \mathcal{R}_q^{k+1}$, which costs a single addition and introduces no noise. Only that coefficient needs fixing, as the sample extraction discards the others; the rest of the atomic pattern (cf. Figure 1) is unchanged.

On the choice of δ The value of δ is unconstrained, as the correction above restores f regardless of the choice. A given function is therefore represented by p distinct test polynomials, all of which evaluate it. Moreover, when several functions $f_1, \dots, f_g$ are evaluated by a single multi-value bootstrapping, each output carries its own correction. Consequently, the shifts $\delta_1, \dots, \delta_g$ may be chosen *independently*, and the family of admissible representations has size at most p^g .

Restoring shifts over the full domain As stated above, the negacyclicity of a given function f_i does not imply that of $f_i^{(\delta_i)}$. Thus, restoring $f_i(m)$ when m ranges over the whole domain $\mathcal{M}$ requires a value $\pm \Delta \delta_i$ whose sign is actually undetermined. This is remedied by homomorphically evaluating, as part of the same multi-value bootstrapping, both the shifted function $f_i^{(\delta_i)}$ along with the constant function δ_i (its negacyclic version). This still requires a single blind rotation, now with at most $2g$ multiplications, and summing the two outputs pairwise restores $f_i(m)$ over the whole of $\mathcal{M}$.

3.2.2 Combining with the log-unit lattice search

The freedom in the choice of the shifts enlarges the input of the common-factor search. Instead of running that search on the single family $v_1, \dots, v_g$, we run it on each shifted family $v_1^{(\delta_1)}, \dots, v_g^{(\delta_g)}$ and keep the best outcome. Concretely, the methodology of Section 3.1 is used unchanged, as the inner routine of the following procedure:

1. select a tuple $\delta := (\delta_1, \dots, \delta_g) \in \mathcal{M}^g$ and form the shifted test polynomials $v_i^{(\delta_i)}$ for $1 \leq i \leq g$ (together with the constant functions δ_i when working over the whole domain);
2. run the log-unit lattice search of Section 3.1 on this family, yielding a candidate common factor $\tilde{v}_0^{(\delta)}$ and the quotients $u_i^{(\delta_i)} = v_i^{(\delta_i)} / \tilde{v}_0^{(\delta)}$;
3. score the candidate by the workload-dependent objective, e.g., $\sum_i \|u_i^{(\delta_i)}\|_2$.

These three steps are repeated over shift tuples, and the best triple $((\delta_i)_i, \tilde{v}_0^{(\delta)}, (u_i^{(\delta_i)})_i)$ is retained.

Several strategies are available for selecting the shift tuples. When both g and p remain small, the tuples can for instance be enumerated exhaustively; as an example, the standard representation of TFHE-rs uses $p = 2^5$, giving at most 2^{5g} tuples. For larger families, a strategy can instead be derived from the empirical observation that the best shifts are generally those minimizing the algebraic norm of the shifted test polynomials.

Example 1. As an example, consider for $N = 2048$ the following four functions with $t = 16$, each represented as a table whose i -th entry is $f(i)$, given as a hexadecimal value:

$$\begin{aligned} f_1 &\rightarrow [0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7], & f_2 &\rightarrow [0, 1, 2, 3, 0, 1, 2, 3, 8, 9, \text{a}, \text{b}, 8, 9, \text{a}, \text{b}], \\ f_3 &\rightarrow [0, 1, 0, 1, 4, 5, 4, 5, 8, 9, 8, 9, \text{c}, \text{d}, \text{c}, \text{d}], & f_4 &\rightarrow [0, 0, 2, 2, 4, 4, 6, 6, 8, 8, \text{a}, \text{a}, \text{c}, \text{c}, \text{e}, \text{e}]. \end{aligned}$$

These actually correspond to [TBS26, Figure 6, first row of $\tilde{M}^{(0)}$] in their homomorphic implementation of the linear layer of the PRINCE blockcipher.

- The original common factor $v_0 = \frac{\Delta}{1-x}$ from [CIM19] yields quotients $u_1, \dots, u_4$ of respective squared coefficients-norms $(112, 176, 208, 224)$ as predicted by Equation (4).

- The log-unit search (without shifting) for the $\sum \|u_i\|_2$ score function outputs Δ times the norm- 2^{1919} element $\hat{v}_0 = \frac{2}{(1-x) \cdot (1-x^{128})}$; the respective squared coefficients-norms become $(57, 85, 99, 112)$. Optimizing for $\max \|u_i\|_2$ yields $\hat{v}_0 = \frac{x^{1536} + x^{1024} + x^{512} + 1}{1-x}$ instead, with quotients of squared coefficients-norms $(60, 100, 92, 104)$.
- The smallest quotients are however obtained when combining the log-unit lattice search with the LUT shifting technique. An exhaustive search settles on the best shifts $\delta = (3, 5, 6, 7)$ with $\hat{v}_0^{(\delta)} = \frac{2}{(1-x) \cdot (1-x^{256})}$; the squared coefficients-norms of the quotients become as small as $(34, 26, 26, 2)$.

As a result, the maximum variance of the MVB output when using $\hat{v}_0^{(3,5,6,7)}$ is divided by 6.59 compared to when using v_0 , which is a significant noise reduction.

Example 2. As a second example, we target less structured LUTs such as the PRINCE inverse S-Box layer with two bits output at a time. Concretely, consider the following functions:

$$f_1 \rightarrow [2, 1, 0, 2, 0, 1, 3, 0, 3, 3, 2, 2, 1, 1, 3, 0] , \quad f_2 \rightarrow [3, 2, 2, 1, 3, 3, 1, 1, 3, 2, 0, 2, 0, 1, 0, 0] .$$

The original common factor yields quotients of squared coefficients-norms $(52, 38)$. Meanwhile, the log-unit lattice search with shifting outputs $\hat{v}_0^{(2,1)} = \frac{x^{512} - x^{384} + x^{128} - 1}{1-x}$, for which the quotients have squared coefficients-norms $(18, 18)$, more than halved.

Example 3. In the context of the tree-based method [GBA21], on the order of t functions need to be applied in parallel for reconstructing an encrypted test polynomial. In order to see the potential of our method on such a large number of test polynomials, we consider the first stage of the homomorphic multiplication of two 4-bit plaintexts with $t = 16$. For the least significant bits, the following functions are evaluated simultaneously for $i \in \llbracket 1, 15 \rrbracket$:

$$f_i: m \in \mathbb{Z}/16\mathbb{Z} \mapsto (i \cdot m) \mod 16 .$$

The resp. quotients u_i by v_0 satisfy $\max \{\|u_i\|_2\} = \sqrt{1024} = 32$ and $\sum_i \|u_i\|_2 \approx 395.5$. On this particular example, we target shifts minimizing the algebraic norms and search for δ around this position coordinate-wise. For $\delta = (7, 7, 7, 6, 8, 7, 8, 4, 8, 7, 8, 6, 8, 7, 7)$, the common factor $\hat{v}_0^{(\delta)} = \frac{-x^{1152} + x^{1024} - x^{128} + 1}{1-x}$ yields a maximum quotient coefficients-norm at $\sqrt{172} \approx 13.11$ (i.e., individual variances of the multi-value output are divided by nearly 6), whereas $\sum_i \|u_i^{(\delta)}\|_2$ is reduced to ≈ 193.38 . An even better coefficients-norm sum is attained e.g., for $\hat{v}_0^{(\delta)} = \frac{-x^{1664} + x^{1536} - x^{1152} + x^{1024} - x^{640} + x^{512} - x^{128} + 1}{1-x}$, at ≈ 184.36 , though in that case the maximum norm is $\sqrt{262} \approx 16.19$.

4 Applications and Experimental Results

We validate our approach on two applications. As a warm-up, we consider the homomorphic evaluation of the PRINCEv2 block cipher. Our main illustration is homomorphic integer multiplication. Multi-value bootstrapping may require larger parameters, and is therefore not necessarily faster than a sequence of regular bootstrappings. This is the case for the schoolbook multiplication of TFHE-rs, and reorganizing it around our framework reverses the outcome.

Experimental setting All measurements were run on an AWS `hpc7a.96xlarge` instance using TFHE-rs v1.7.0 with a target failure probability $p_{\text{fail}} = 2^{-128}$ and at least 128 bits of

Table 1: Latency overheads (relative to the $\nu_{\max} = 5.6$ baseline) per PBS depending on $\nu_{\max}$, along with the associated parameters for $p_{\text{fail}} \leq 2^{-128}$. Common to all rows: $q = 2^{64}$.

ID	$\nu_{\max}$	LWE		GLWE			Gadgets $\nabla_{\beta,\ell}$				Measured	
		n	σ_{LWE}	k	N	σ_{GLWE}	β_{br}	ℓ_{br}	β_{ks}	ℓ_{ks}	Lat. (ms)	Ovh. %
$\mathcal{P}_{\mathcal{G},5.6}$	5.6	866	$2.05 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^3	5	14.364	—
$\mathcal{P}_{\mathcal{G},11.4}$	11.4	887	$1.42 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^3	5	14.869	3.5
$\mathcal{P}_{\mathcal{G},12.8}$	12.8	930	$6.78 \cdot 10^{-7}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^4	4	15.021	4.6
$\mathcal{P}_{\mathcal{G},14.8}$	14.8	899	$1.16 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^3	6	15.879	10.5
$\mathcal{P}_{\mathcal{G},15.0}$	15.0	904	$1.06 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^3	6	16.011	11.5
$\mathcal{P}_{\mathcal{G},16.8}$	16.8	863	$2.15 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{23}	1	2^1	17	19.224	33.8
$\mathcal{P}_{\mathcal{G},22.4}$	22.4	862	$2.19 \cdot 10^{-6}$	1	2048	$2.85 \cdot 10^{-15}$	2^{15}	2	2^3	5	22.228	54.7

security. The noise budget $\nu_{\max}$ of an atomic pattern is as defined in Section 2.3. Reported values of $\nu_{\max}$ are rounded up to one decimal place, so that the stated budget always covers the exact norm. We denote by $\mathcal{P}_{\mathcal{G},\nu_{\max}}$ the Gaussian parameter set generated for a given budget. The parameter sets used in our experiments, together with the measured latency of one PBS under each of them, are listed in Table 1.

4.1 Warming Up: PRINCEv2

Our baseline is the carefully optimized homomorphic circuit of [BJ26, Section 6], which relies on regular bootstrappings. We first derive a variant of their implementation using the original MVB decomposition [CIM19], wherever applicable and without making any other change. Then, we replace these original MVB decompositions by our new decompositions from Section 3 with LUT shifting. The associated squared coefficients-norms of the quotient polynomials in each homomorphic layer are given in Table 2 for both cases. In particular, Example 2 in Section 3.2.2 details our polynomials and shifts for the decomposition of the backward S-Box layer.

Table 2: Squared coefficients-norms of the function-specific polynomials for PRINCEv2 homomorphic layers as implemented in [BJ26]. In this implementation, forward and backward linear layers are identical, and forward XOR layers only involve distinct inputs.

Hom. layer	Score function	Original MVB		Ours (w/ shift)	
		$(\ u_i\ _2^2)$	Score	$(\ u_i\ _2^2)$	Score
Forward S-Boxes	$4 \cdot \max\ u_i\ _2$	(14, 8, 10, 12)	≈ 14.97	(8, 8, 8, 8)	≈ 11.31
Linear layers	$\sum\ u_i\ _2$	(12, 10, 14, 6)	≈ 12.82	(6, 5, 7, 3)	≈ 9.06
Backward S-Boxes	$1 + \max\ u_i\ _2$	(52, 38)	≈ 8.21	(18, 18)	≈ 5.25
Backward XORs	$4 \cdot \max\ u_i\ _2$	(12, 6)	≈ 13.85	(4, 4)	8

In both variants, a single parameter set serves the whole cipher, as in the original circuit. Therefore, the cost of the algorithm is driven by the maximum admissible noise $\nu_{\max}$ that the circuit must support. This is given by the maximum over the score functions of the coefficients-norms of the quotient polynomials of each respective layer. Both score values and score functions, determined by the subsequent dot products between the ciphertexts output by each layer, are detailed in Table 2. Thus, the original MVB is constrained by $\nu_{\max} = \sqrt{224} \approx 14.97$, incurring a +11.5% per-PBS overhead over the baseline (parameter set $\mathcal{P}_{\mathcal{G},15.0}$ in Table 1). In comparison, our decomposition with LUT shifting brings it down

Table 3: Latency of one 64-bit block transciphering using PRINCEv2 depending on the number of parallel threads T .

Decomposition	Latency (s)							
	$T =$	1	2	4	8	16	32	64
Classical ($\mathcal{P}_{\mathcal{G},5.6}$)		23.850	12.775	7.130	3.596	1.837	1.041	0.697
Original MVB ($\mathcal{P}_{\mathcal{G},15.0}$)		12.050	6.766	3.600	1.887	1.058	0.816	0.802
Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},11.4}$)		11.514	6.366	3.404	1.773	0.974	0.759	0.748

to $\sqrt{128} \approx 11.31$, for only +3.5 % ($\mathcal{P}_{\mathcal{G},11.4}$).

Table 3 reports the resulting benchmarks. Both multi-value methods remove many bootstrappings for only a modest overhead, and both beat the classical baseline up to 32 threads. Ours is consistently faster than the original MVB, by about 6.2 % on average, thanks to its lighter parameters. The gain over the classical baseline is largest with few threads, about $2\times$ from 1 to 16 threads. With more threads the classical approach catches up, as its independent bootstrappings can run concurrently.

This is the expected behavior for multi-value bootstrapping, which does not improve latency on a massively parallel architecture but shines on a limited number of threads.

4.2 Homomorphic Integer Multiplication

Integer encoding and constraints In TFHE-rs [Zam], an unsigned integer of X bits is decomposed in base $\sqrt{t}$, with $t = 2^4$, into $\lceil X/2 \rceil$ digits $\mu_i < \sqrt{t}$, each encrypted in its own ciphertext $\text{ct}_i \in \text{LWE}_{\mathbf{s}}^{q,2t}(\Delta \mu_i)$. The base is chosen so that the concatenation of two digits, i.e., $(x||y) := x \cdot \sqrt{t} + y$, lies in $\llbracket 0, t-1 \rrbracket$, letting a single bootstrapping evaluate a function of a pair of digits. A homomorphic integer **FheUIntX** is the tuple of these ciphertexts, indexed by columns i . Each digit μ_i occupies only the lower part $\llbracket 0, \sqrt{t}-1 \rrbracket$ of the message space, the rest absorbing the growth of leveled operations.

Two constraints govern how many ciphertexts can be summed before bootstrapping. The *arithmetic condition* keeps the sum in $\llbracket 0, t-1 \rrbracket$, the first case in Equation (2), which is the setting of TFHE-rs. With $t = 2^4$ and $\mu_i \leq \sqrt{t}-1 = 3$, up to five ciphertexts can be summed. The *noise condition*, established in Section 2.3, bounds by $\nu_{\max}$ the factor by which a ciphertext’s noise may grow before bootstrapping. Its impact on performance and the associated parameter set are given in Table 1. A set $\{\text{ct}_j\}_j$ of ciphertexts can thus be summed when

$$\underbrace{\#\{\text{ct}_j\}_j \leq \frac{t-1}{\sqrt{t}-1} = 5}_{\text{arithmetic condition}} \quad \text{and} \quad \underbrace{\sum_j \sigma_{\text{ct}_j} / \sigma_{\text{br}} \leq \nu_{\max}}_{\text{noise condition}} . \quad (10)$$

At the time of writing, in TFHE-rs, the noise condition is implicitly tied to the arithmetic condition. Indeed, the parameters used in the multiplication are generated for the smallest $\nu_{\max} \geq 5$ (see Equation (6)), so that the noise condition holds when summing any five ciphertexts with $\sigma_{\text{ct}} \leq \sigma_{\text{br}}$. Section 4.2.2 decouples the two conditions.

Schoolbook multiplication The TFHE-rs homomorphic multiplication follows the schoolbook method [BBB⁺25, Section 3.13.1] in three stages, mapping two **FheUIntX** to an **FheUIntX** of the same width. Every digit of one operand is multiplied by every digit of the other, each partial product landing in the column given by the sum of the two positions; the columns are then folded; and the remaining carries are finally propagated.

Partial products For every pair of columns j and k encrypting digits x and y , the product xy exceeds the two-bit range and is split into a low and a high part, each recovered by its own bootstrapping,

$$P_{ls}(x\|y) = xy \bmod 4, \quad P_{ms}(x\|y) = \lfloor xy/4 \rfloor, \quad (11)$$

the low part landing in column $j + k$ and the high part in column $j + k + 1$.

Column folding Each column then holds several ciphertexts, whose number must be reduced iteratively. In **TFHE-rs**, only the arithmetic condition of Equation (10) is enforced, since the parameters guarantee that the noise condition holds when summing any five ciphertexts. Thus, a column is folded five ciphertexts at a time, each sum being split into a low and a high part by two bootstrappings,

$$F_{ls}(x) = x \bmod 4, \quad F_{ms}(x) = \lfloor x/4 \rfloor, \quad (12)$$

the low part staying in the current column and the high part being carried to the next, until no more than five ciphertexts remain.

Carry propagation The resulting digits may still have a non-empty high part. A final pass propagates the remaining carries. This chain is inherently sequential and, on a parallel machine, becomes the latency bottleneck, which **TFHE-rs** resolves in a logarithmic number of steps with a Hillis–Steele-style parallel prefix [BBB⁺25, Section 3.5]. By design, this last stage is not eligible for multi-value bootstrapping, and it is left unmodified in the sequel.

4.2.1 Naive multi-value bootstrapping substitution

Best multi-value decompositions Both (P_{ls}, P_{ms}) and (F_{ls}, F_{ms}) are pairs of functions of the same input, so a single multi-value bootstrapping may evaluate each pair at once, halving the number of bootstrappings in the partial-product and folding stages.

For both pairs, we report in Table 4 the coefficients-norms of the quotients obtained respectively from the original MVB decomposition [CIM19], and from our decompositions in Section 3. The best decomposition for (P_{ls}, P_{ms}) is obtained using a shift of $(1, 0)$ and common factor $\hat{v}_0^{(1,0)} = \frac{x^{1664}+1}{1-x}$; likewise, the best decomposition for (F_{ls}, F_{ms}) is obtained for the common factor $\hat{v}_0^{(1,1)} = \frac{x^{1536}+x^{1024}+x^{512}+1}{1-x}$.

Table 4: Coefficients-norms, for each multi-value decomposition, of the function-specific polynomials u_f for functions $f \in (P_{ls}, P_{ms})$ (partial products) and $f \in (F_{ls}, F_{ms})$ (folds).

Decomposition	Partial product		Fold	
	$\ u_{P_{ms}}\ _2$	$\ u_{P_{ls}}\ _2$	$\ u_{F_{ms}}\ _2$	$\ u_{F_{ls}}\ _2$
Original MVB	$\sqrt{8} \approx 2.8$	$\sqrt{40} \approx 6.3$	$\sqrt{12} \approx 3.5$	$\sqrt{48} \approx 6.9$
Ours (w/o shift)	$\sqrt{11} \approx 3.3$	$\sqrt{20} \approx 4.5$	$\sqrt{5} \approx 2.2$	$\sqrt{12} \approx 3.5$
Ours (w/ shift)	$\sqrt{5} \approx 2.2$	$\sqrt{20} \approx 4.5$	$\sqrt{1} = 1.0$	$\sqrt{8} \approx 2.8$

Naive multi-value bootstrapping substitution We call *naive substitution* the algorithm obtained by this replacement alone, i.e., keeping the schedule, the folding rule and, as in classical multiplication, a single parameter set for the whole algorithm. Since any five ciphertexts of a column may then be summed, that parameter set must be sized for the noisiest output, i.e., for $\nu_{\max} = 5 \cdot \max_f \|u_f\|_2$ with the norms of Table 4. Even using our

best decomposition, this yields $\nu_{\max} = 5\sqrt{20} \approx 22.4$, whose associated parameter set $\mathcal{P}_{\mathcal{G}, 22.4}$ induces a 54.7% per-PBS overhead over the baseline as per Table 1.

The resulting number of sequential bootstrappings is expected to be roughly halved,⁶ e.g., from 1604 to 828 for a sequential **FheUint64** multiplication, excluding the carry-propagation stage (cf. Table 5, column $T = 1$, rows **TFHE-rs** and **TFHE-rs+MVB**). This saving shrinks as the number of threads grows, since the independent bootstrappings of the classical multiplication parallelize well while arithmetic dependencies leave threads idle, and each remaining bootstrapping costs 54.7% more. In practice, the timings given in Table 6 on 192 threads show that the naive substitution only brings a throughput gain from **FheUint64** onwards, and a latency speed-up only for **FheUint256**.

4.2.2 Noise-aware folding

Inherited from **TFHE-rs**, the naive substitution only checks the arithmetic condition, sizing the parameters so that *any* five outputs can be summed. This is wasteful once outputs carry different noise levels, as the budget is then dictated by the noisiest of them.

Our key idea is to fix a smaller $\nu_{\max}$ and decouple the noise condition from the arithmetic condition. Two observations make this workable. First, the noise condition can be checked exactly and in advance. Call the *type* of a ciphertext the function among $P_{ls}, P_{ms}, F_{ls}, F_{ms}$ that produced it: its normalized noise standard deviation σ_{ct}/σ_{br} is the coefficients-norm of the corresponding quotient in Table 4. Second, the schedule is data-independent: the type of each ciphertext, and hence the composition of every column at every stage, is fixed by the structure of the computation and not by the encrypted values. The whole folding schedule can therefore be computed offline, once and for all, and we simulate the multiplication to select the fastest one.

Folding rules A folding rule selects, among the sets of ciphertexts of a column satisfying Equation (10), which one to fold next. We consider three greedy options: (i) *cheapest* maximizes the number of ciphertexts folded, breaking ties by smallest total noise, (ii) *expensive* maximizes the number of ciphertexts of the noisiest type, then of the second noisiest, and so on, and (iii) *heaviest* picks the fold whose total noise comes closest to $\nu_{\max}$, breaking ties by the number of distinct types it contains.

Column scheduling On a machine with T threads, bootstrappings run in successive *ticks*, i.e., rounds of at most T parallel bootstrappings, and latency is driven by the number of ticks rather than by the total bootstrapping count. When several columns have folds ready, the column order decides which ones are scheduled first. It matters both because folding a column feeds carries into the next one and because a tick holds at most T bootstrappings. Again we consider three options: *forward* starts from the least significant column, *backward* from the most significant, and *balanced* advances all columns together, giving priority to the lowest-order ones when there are fewer threads than columns, so as to flush carries upward quickly. Combined with the three folding rules, this gives nine strategies.

Choosing the strategy We search offline, over the budget $\nu_{\max}$ and the nine strategies, for the schedule minimizing the tick count *weighted by the per-PBS overhead* of Table 1, over a range of thread counts. The search settles on the *heaviest* rule with the *balanced* order in all cases. The picture is a bit noisier for determining $\nu_{\max}$ depending on the **FheUintX** size, the number of threads T and the multi-value decomposition norms. However,

- for the original MVB decomposition, the best figures are obtained for $\nu_{\max} = 22.4$;

⁶Not exactly halved, since the classical multiplication with independent bootstrappings avoids computing the most significant digits beyond the integer size; for such digits, the multi-value bootstrapping is useless.

Table 5: Total ticks, i.e., rounds of at most T parallel bootstrappings (cf. Section 4.2.2), by thread count T , for the partial-product and column-folding stages of an **FheUint64** multiplication, for the two baselines and for noise-aware folding under three strategies.

Strat.	Decomposition	Total ticks						
		$T =$	1	8	12	64	96	192
heaviest balanced	TFHE-rs ($\mathcal{P}_{\mathcal{G},5.6}$)		1604	204	136	27	19	11
	TFHE-rs + MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		828	108	72	16	11	7
	Original MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		904	114	77	17	12	8
	Ours (w/o shift) ($\mathcal{P}_{\mathcal{G},14.8}$)		944	120	81	18	13	8
	Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},12.8}$)		888	112	76	17	12	7
expensive forward	Original MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		964	127	85	19	14	9
	Ours (w/o shift) ($\mathcal{P}_{\mathcal{G},14.8}$)		1002	131	88	20	14	8
	Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},12.8}$)		914	119	81	18	13	8
cheapest backward	Original MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		1043	136	93	26	18	10
	Ours (w/o shift) ($\mathcal{P}_{\mathcal{G},14.8}$)		1054	135	88	19	13	8
	Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},12.8}$)		1013	145	96	20	14	8

- for the log-unit search alone, setting $\nu_{\max} = 14.8$ is optimal in most configurations and, except in a very few cases, stays within 1.5 % of the best budget;
- for the log-unit search combined with LUT shifting, setting $\nu_{\max} = 12.8$ is generally the optimal choice and stays within 2.2 % of the best budget almost everywhere.

In the last two cases, the few outliers are concentrated on small sizes $\mathbf{x} \leq 32$ and small number of threads $T \leq 12$. For these combinations, the column heights are relatively small and the folds are not saturated, so that a smaller ν may (marginally) win purely on the per-PBS overhead acting on the same fold list.

A heuristic explanation is that the best $\nu_{\max}$ must generally cover self-sustaining folds that reabsorb their own emissions of $(\mathbf{F}_{\text{ms}}, \mathbf{F}_{\text{ls}})$ ciphertexts while maximizing their number of partial product entries $(\mathbf{P}_{\text{ms}}, \mathbf{P}_{\text{ls}})$. For our new decompositions, this corresponds to $(2\mathbf{P}_{\text{ls}} + \mathbf{F}_{\text{ms}} + \mathbf{F}_{\text{ls}})$ folds whose noise is resp. 12.77 and 14.64. For the original MVB, the key fold is $(2\mathbf{P}_{\text{ms}} + \mathbf{P}_{\text{ls}} + \mathbf{F}_{\text{ms}} + \mathbf{F}_{\text{ls}})$ whose noise is 22.37.

Table 5 reports the resulting tick counts for an **FheUint64** multiplication; other widths behave similarly. **TFHE-rs + MVB** folds five ciphertexts every time and thus needs the fewest ticks, but at $\mathcal{P}_{\mathcal{G},22.4}$ each tick costs 54.7 % more than at the baseline $\mathcal{P}_{\mathcal{G},5.6}$, which outweighs the saving. The original MVB, even with noise-aware folding, still requires $\nu_{\max} = 22.4$; its outputs are too noisy for the noise condition to buy anything. At 192 threads and 64 bits, this tick count does not compensate the per-PBS overhead, so it is not expected to beat the classical baseline there. It does so only at larger widths, as confirmed by the measurements of Section 4.2.3.

4.2.3 Experimental evaluation

We benchmark the three multi-value decompositions, each with the noise-aware folding of Section 4.2.2, for **FheUint** widths from 8 to 256 bits. In Table 6, they are labelled Original MVB, Ours (w/o shift) and Ours w/ shift. Two baselines are included. The first is the classical multiplication of **TFHE-rs**. The second is the naive substitution **TFHE-rs + MVB** of Section 4.2.1, which uses the same decomposition as Ours w/ shift but the original **TFHE-rs** folding. The table reports latency and throughput on 192 threads.

Table 6: Multiplication benchmarks for different decompositions, for FheUintX with $x \in \{8, 16, \dots, 256\}$, measured on AWS hpc7a.96xlarge using all 192 threads.

Decomposition	$x =$	Latency (s)					
		8	16	32	64	128	256
TFHE-rs ($\mathcal{P}_{\mathcal{G},5.6}$)		0.081	0.118	0.184	0.352	1.040	3.516
TFHE-rs + MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		0.152	0.212	0.280	0.458	1.050	3.072
Original MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		0.130	0.182	0.249	0.443	1.099	3.391
Ours (w/o shift) ($\mathcal{P}_{\mathcal{G},14.8}$)		0.097	0.137	0.191	0.342	0.876	2.672
Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},12.8}$)		0.094	0.125	0.179	0.299	0.794	2.389

Decomposition	$x =$	Throughput (elem/s)					
		8	16	32	64	128	256
TFHE-rs ($\mathcal{P}_{\mathcal{G},5.6}$)		285.510	66.947	17.924	4.491	1.156	0.290
TFHE-rs + MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		193.770	49.689	16.164	4.614	1.326	0.355
Original MVB ($\mathcal{P}_{\mathcal{G},22.4}$)		218.130	46.466	15.505	4.434	1.266	0.335
Ours (w/o shift) ($\mathcal{P}_{\mathcal{G},14.8}$)		290.340	75.552	21.081	5.517	1.585	0.426
Ours (w/ shift) ($\mathcal{P}_{\mathcal{G},12.8}$)		323.470	81.014	22.203	6.145	1.793	0.462

Even with noise-aware folding, the original MVB is slower than the classical baseline up to 64 bits, in both latency and throughput. It only overtakes at the largest widths, in throughput from 128 bits and in latency at 256 bits, where the reduced bootstrapping count outweighs the 54.7 % per-PBS overhead.

Our new decompositions turn this loss into a gain, benefiting from the noise-aware folding. With the log-unit search alone, throughput improves at every width and latency from 64 bits. Combining it with LUT shifting further improves throughput at every width, and improves the baseline latency from 32 bits, reaching $1.47\times$ in latency and $1.59\times$ in throughput on a FheUint256. Measured against the original MVB, the same combination is $1.38\times$ to $1.48\times$ faster in latency and $1.38\times$ to $1.74\times$ in throughput, at every width.

References

- [Bab86] László Babai. On Lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. doi:10.1007/BF02579403.
- [BBB⁺23] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Parameter optimization and larger precision for (T)FHE. *Journal of Cryptology*, 36(28), 2023. doi:10.1007/s00145-023-09463-5.
- [BBB⁺25] Mathieu Ballandras, Loris Bergerat, Charlotte Bonte, Carl Bootland, Benjamin R. Curtis, Jad Khatib, Jakub Klemsa, Thomas Montagu, Jean-Baptiste Orfila, and Samuel Tap. *TFHE-rs: A (Practical) Handbook*. Zama, 1st edition, February 2025. URL: <https://github.com/zama-ai/tfhe-rs>.
- [BBR26] Sonia Belaïd, Nicolas Bon, and Matthieu Rivain. Decomposition of large look-up tables for fast homomorphic evaluation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(3):247–278, 2026. doi:10.46586/tches.v2026.i3.247-278.

- [BCG⁺12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın. PRINCE – A low-latency block cipher for pervasive computing applications. In X. Wang and K. Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012. doi:[10.1007/978-3-642-34961-4_14](https://doi.org/10.1007/978-3-642-34961-4_14).
- [BEK⁺20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. PRINCEv2: More security for (almost) no overhead. In O. Dunkelman, M. J. Jacobson, Jr., and C. O’Flynn, editors, *Selected Areas in Cryptography (SAC 2020)*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, Cham, 2020. doi:[10.1007/978-3-030-81652-0_19](https://doi.org/10.1007/978-3-030-81652-0_19).
- [BFL24] Olivier Bernard, Pierre-Alain Fouque, and Andrea Lesavourey. Computing e -th roots in number fields. In *2024 Proceedings of the Symposium on Algorithmic Engineering and Experiments (ALENEX)*, pages 207–219. SIAM, 2024. doi:[10.1137/1.9781611977929.16](https://doi.org/10.1137/1.9781611977929.16).
- [BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3):13:1–13:36, 2014. doi:[10.1145/2633600](https://doi.org/10.1145/2633600).
- [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](https://doi.org/10.1007/978-3-031-22966-4_7).
- [BJ25] Olivier Bernard and Marc Joye. Approximate CRT-based gadget for fully homomorphic encryption. In C. Cid and N. Yanai, editors, *Advances in Information and Computer Security (IWSEC 2025)*, volume 16208 of *Lecture Notes in Computer Science*, pages 45–64. Springer, Singapore, 2025. doi:[10.1007/978-981-95-4674-9_3](https://doi.org/10.1007/978-981-95-4674-9_3).
- [BJ26] Olivier Bernard and Marc Joye. Hash function constructions from lightweight block ciphers for fully homomorphic encryption. Cryptology ePrint Archive, Report 2026/309, 2026. URL: <https://ia.cr/2026/309>.
- [BJSW25] Olivier Bernard, Marc Joye, Nigel Smart, and Michael Walter. Drifting towards better error probabilities in fully homomorphic encryption schemes. In S. Fehr and P.-A. Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part VIII*, volume 15608 of *Lecture Notes in Computer Science*, pages 181–211. Springer, Cham, 2025. doi:[10.1007/978-3-031-91101-9_7](https://doi.org/10.1007/978-3-031-91101-9_7).
- [BLNR22] Olivier Bernard, Andrea Lesavourey, Tuong-Huy Nguyen, and Adeline Roux-Langlois. Log- $\mathcal{S}$ -unit lattices using explicit Stickelberger generators to solve approx ideal-SVP. In S. Agrawal and D. Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 677–708. Springer, Cham, 2022. doi:[10.1007/978-3-031-22969-5_23](https://doi.org/10.1007/978-3-031-22969-5_23).
- [CBSZ23] Pierre-Emmanuel Clet, Aymen Boudguiga, Renaud Sirdey, and Martin Zuber. ComBo: a novel functional bootstrapping method for efficient evaluation of nonlinear functions in the encrypted domain. In N. El Mrabet, L. De Feo, and S. Duquesne, editors, *Progress in Cryptology – AFRICACRYPT 2023*, volume

- 14064 of *Lecture Notes in Computer Science*, pages 317–343. Springer, Cham, 2023. doi:[10.1007/978-3-031-37679-5_14](https://doi.org/10.1007/978-3-031-37679-5_14).
- [CCP⁺24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the IND-CPA^D security of exact FHE schemes. In E. Kirda and D. Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS’24*, pages 2505–2519. Association for Computing Machinery, 2024. doi:[10.1145/3658644.3690341](https://doi.org/10.1145/3658644.3690341).
- [CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In M. Fishlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585. Springer, Berlin, Heidelberg, 2016. doi:[10.1007/978-3-662-49896-5_20](https://doi.org/10.1007/978-3-662-49896-5_20).
- [CGGI20] Ilaria Chillotti, 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](https://doi.org/10.1007/s00145-019-09319-x).
- [CIM19] Sergiu Carpov, Malika Izabachène, and Victor Mollimard. New techniques for multi-value input homomorphic evaluation and applications. In M. Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 106–126. Springer, Cham, 2019. doi:[10.1007/978-3-030-12612-4_6](https://doi.org/10.1007/978-3-030-12612-4_6).
- [CJL⁺20] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE. In M. Brenner and T. Lepoint, editors, *WAHC’20: 8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 57–63. Leibniz Universität IT Services, 2020. doi:[10.25835/0072999](https://doi.org/10.25835/0072999).
- [CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In S. Dolev, O. Margalit, B. Pinkas, and A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning (CSCML 2021)*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Cham, 2021. doi:[10.1007/978-3-030-78086-9_1](https://doi.org/10.1007/978-3-030-78086-9_1).
- [CKL26] Philippe Chartier, Michel Koskas, and Mohammed Lemou. Exploring general cyclotomic rings in torus-based fully homomorphic encryption. *IACR Communications in Cryptology*, 3(1), 2026. doi:[10.62056/ab01jb0kr](https://doi.org/10.62056/ab01jb0kr).
- [CSBB24] Marina Checri, Renaud Sirdey, Aymen Boudguiga, and Jean-Paul Bultel. On the practical CPA^D security of “exact” and threshold FHE schemes and libraries. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Cham, 2024. doi:[10.1007/978-3-031-68382-4_1](https://doi.org/10.1007/978-3-031-68382-4_1).
- [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:[10.1007/978-3-662-46800-524](https://doi.org/10.1007/978-3-662-46800-524).
- [FPL23] FPLLL development team. fpLLL, a lattice reduction library, Version: 5.5.0. Available at <https://github.com/fplll/fplll>, 2023.

- [GBA21] Antonio Guimarães, Edson Borin, and Diego F. Aranha. Revisiting the functional bootstrap in TFHE. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):229–253, 2021. doi:10.46586/tches.v2021.i2.229–253.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, 2009. doi:10.1145/1536414.1536440.
- [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.
- [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.
- [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: 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–11. Association for Computing Machinery, New York, 2022. doi:10.1145/3560827.3563376.
- [Kuč92] Radan Kučera. On bases of the Stickelberger ideal and of the group of circular units of a cyclotomic field. *Journal of Number Theory*, 40(3):284–316, 1992. doi:10.1016/0022-314X(92)90003-8.
- [LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In A. Canteaut and F.-X. Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 648–677. Springer, Cham, 2021. doi:10.1007/978-3-030-77870-5_23.
- [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, Berlin, Heidelberg, 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.
- [MKG24] Johannes Mono, Kamil Klucznik, and Tim Güneysu. Improved circuit synthesis with multi-value bootstrapping for FHEW-like schemes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(4):633–656, 2024. doi:10.46586/tches.v2024.i4.633–656.

- [RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 165–179. 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.
- [Sin78] Warren Sinnott. On the Stickelberger ideal and the circular units of a cyclotomic field. *Annals of Mathematics*, 108(1):107–134, 1978. doi:10.2307/1970932.
- [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. Springer, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-10366-7_36.
- [TBS26] Daphné Trama, Aymen Boudguiga, and Renaud Sirdey. Running standard block ciphers beyond AES with TFHE: Experiments and lessons learnt. *IACR Communications in Cryptology*, 2(4), 2026. doi:10.62056/avom-4tw9.
- [TCB⁺25] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, Renaud Sirdey, and Nicolas Ye. Designing a general-purpose 8-bit (T)FHE processor abstraction. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(2):535–578, 2025. doi:10.46586/tches.v2025.i2.535-578.
- [TCBS23] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. A homomorphic AES evaluation in less than 30 seconds by means of TFHE. In M. Brenner, A. Costache, and K. Rohloff, editors, *WAHC’23: 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 79–90. Association for Computing Machinery, New York, 2023. doi:10.1145/3605759.3625260.
- [Was97] Lawrence C. Washington. *Introduction to Cyclotomic Fields*, volume 83 of *Graduate Texts in Mathematics*. Springer New York, 2 edition, 1997. doi:10.1007/978-1-4612-1934-7.
- [WDL26] Kunyu Wu, Kuiyan Duan, Dengfa Liu, and Hongbo Li. A new framework for efficient multivariate functional bootstrapping. Cryptology ePrint Archive, Report 2026/1401, 2026. URL: <https://ia.cr/2026/1401>.
- [WHS⁺25] Ruida Wang, Jincheol Ha, Xuan Shen, Xianhui Lu, Chunling Chen, Kunpeng Wang, and Jooyoung Lee. Refined TFHE leveled homomorphic evaluation and its application. In *CCS’ 25: ACM SIGSAC Conference on Computer and Communications Security*, pages 2309–2323. Association for Computing Machinery, New York, 2025. doi:10.1145/3719027.3744873.
- [Zam] Zama. TFHE-rs: A pure Rust implementation of the TFHE scheme for boolean and integer arithmetics over encrypted data. URL: <https://github.com/zama-ai/tfhe-rs>.