Descent-PBS: Programmable Bootstrapping beyond the Ring Dimension

Zhenyu Xiong[1], Mingsheng Wang[1], Han Wang[1] and Zhedong Wang[2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
xiongzhenyu@iie.ac.cn, wangmingsheng@iie.ac.cn, wanghan@iie.ac.cn
[2] Shanghai Jiao Tong University, Shanghai, China
wzdstill@sjtu.edu.cn

Abstract. Programmable bootstrapping (PBS) in FHEW/TFHE evaluates a lookup table (LUT) while refreshing a ciphertext, but a t-entry negacyclic table must fit, with redundancy, into the $2N$ signed rotation positions of a ring accumulator. Module bootstrappings, namely extended bootstrapping [LY23] (PKC'23), monomial-matrix bootstrapping [DLLM25] (TCC'25) and their successors [WBL+25, BORLT25], rotate a redundant table once in a module of rank τ, at a cost linear in both the rank and the redundancy. Meta-PBS [MHW+26] (CCS'26) removes the redundancy by iterating the rotation in the ring, but supports plaintext moduli only up to $2N$, and running it inside the module pays the rank in every rotation.

We present Descent-PBS, which pays the module rank of a large table only in its first rotation. We prove that for a module secret embedded from the subring, truncation commutes with the secret, so one public functional key switch with merged columns repacks all τ_0 components of a module accumulator into the ring and removes the per-component factor τ_0 of repacking. The bootstrapping then continues with Meta-PBS refinements in the ring, at $\tau_0 + K$ instead of $(K' + 1)\tau_0$ external products per key. For block one-hot secrets, the first-rotation exponents are multiples of τ_0, so every combined key is a scalar matrix: the τ_0 components share one key per block, and since all exponents are public in advance, one pass over the bootstrapping key builds the keys of all rotations of a negacyclic table. Ties-to-even gadget rounding removes a refinement bias that we trace to half-up rounding on the FFT grid.

In a TFHE-rs-based engine, Descent-PBS evaluates a 16-bit negacyclic LUT in 196 ms on one thread at 128-bit security and a model-predicted failure probability below 2^{-64}. With the same keys and failure budget, the descent alone makes it 1.44–2.40× faster than Meta-PBS run inside the module for dense secrets; for block one-hot secrets, the descent and the shared keys together make it 1.51–3.32× faster, and 1.37–3.57× against the same baseline re-planned for more conservative 131-bit parameters. General 13–16-bit LUTs take 179–368 ms, 3.8–6.6× faster than WoP-PBS in its authors' configuration on the same single-ciphertext interface, at about twice its memory. The failure probabilities are predictions of a model that we validate where failures are observable, and the 16-bit keys take 1,208 MB.

Keywords: Fully homomorphic encryption · TFHE · Programmable bootstrapping · Large lookup tables · Sparse secrets · Module-to-ring repacking

1 Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that allows one to compute on encrypted data without learning anything about it. Since Gentry's first construction [Gen09], the FHEW/TFHE line [DM15, CGGI16, CGGI20] has made

bootstrapping cheap enough to run after every gate, and its *programmable bootstrapping* (PBS) [CJP21, CLOT21] evaluates an arbitrary lookup table $f\colon \mathbb{Z}_t \to \mathbb{Z}_t$ while it refreshes an LWE ciphertext; it is the basic operation of TFHE-rs [Zam22]. However, the table lives in the test polynomial of $\mathcal{R}_N = \mathbb{Z}_q[X]/(X^N+1)$ with a redundancy ρ large enough to absorb the modulus-switching error, so one rotation needs $2N \geq \rho t$. For one LWE ciphertext in and one out, 13–16-bit tables exceed the $2N = 4096$ signed rotation positions of the standard ring, and a long line of work has evaluated them through digits, trees, circuits of smaller tables or circuit bootstrapping on several ciphertexts [LMP22, GBA21, BBB+23, WHS+25, BBR26], or by enlarging the ring [LY23, DLLM25, WBL+25, BORLT25].

In this work we concentrate on blind-rotation pipelines for a single ciphertext, without ciphertext products or circuit bootstrapping. Every such pipeline that we know of can be cast as a *schedule of ranks*: its i-th rotation runs in a free $\mathcal{R}_N$-module of rank τ_i, i.e. in $\mathcal{R}_M = \mathbb{Z}_q[Y]/(Y^M+1)$, $M = \tau_i N$, with all keys in $\mathcal{R}_N$ [LY23, WBL+25], and costs τ_i external products per key. With n_{k} keys per rotation (n for a dense secret, h for a block one-hot secret), it minimizes

$$\textstyle\sum_{i=0}^{K} \tau_i \cdot n_{\mathrm{k}} \cdot \mathsf{EP} \quad \text{s.t.} \quad 2N\tau_0 \geq r_0 t, \quad r_{i+1} \leq r_i \beta_i, \quad (2T_i + 1 + \epsilon_i \tau_i)\beta_i \leq N, \tag{1}$$

where r_i is the redundancy after rotation i and the last constraint says that each repacking fits into $\mathcal{R}_N$. Extended and monomial-matrix bootstrapping [LY23, DLLM25] use the schedule $(\rho\tau_0; -)$: one rotation of a redundant table in a large module. Meta-PBS [MHW+26] uses $(1;1)$: it removes the redundancy by dividing the input with remainder and refining the rotation in $\mathcal{R}_N$, but its first table must fit, $t \leq 2N$, and for higher precision its authors recommend decomposition-based methods [MHW+26, §6.2.5]. Running Meta-PBS in a module instead would pay the rank in every rotation, since each repacking would act on all components. This raises the following question:

Can a table beyond $2N$ entries be bootstrapped while paying the rank only once?

1.1 Our Contributions

We resolve the tension between the redundancy of one-shot module rotations and the $2N$ limit of iterative ones by a module-to-ring descent. Our main contributions are summarized below:

1. **Core construction: a module-to-ring descent.** For a subring secret, truncation commutes with the secret, $\mathrm{truncRepeat}_{\tau_0}(A\,\iota(S)) = \sum_k S[k]\,\mathrm{truncRepeat}_{\tau_0}(AY^{k\tau_0})$, so public functional key switching with columns merged at stride $\tau_0\beta$ repacks all τ_0 components into the ring; for a fixed merge width it removes the factor τ_0 of componentwise repacking, while the feasible width shrinks with τ_0 through the merge condition. The new element is not the key switch but the fact that one collision-free merged key switch ends the module computation after the first rotation while keeping the coefficient geometry that the Meta-PBS refinements need, so the schedule $(\tau_0; 1)$ costs $(\tau_0 + K)\, n_{\mathrm{k}}\, \mathsf{EP}$ instead of $(K'+1)\,\tau_0\, n_{\mathrm{k}}\, \mathsf{EP}$. We prove exact window and selection conditions from the index arithmetic and give a per-event failure model, with Gaussian-free Chernoff bounds and validation where failures are observable.

2. **A geometry–key co-design for block-sparse secrets.** The module member divides first by $2M$ and keeps the first window narrow; the parallel member divides by $2N$ and accepts a τ_0 times wider window, but its first-rotation exponents become divisible by τ_0, so every combined key is a scalar matrix and the τ_0 components share one key per block (algebraic). Since all digits are public before the first rotation, one fused pass over the bootstrapping key then builds the keys of all rotations (implementation); general

tables add one pass for the sign. The member that loses with dense keys thus wins with sparse keys, by 1.23–1.70×.

3. **Numerical validation and evaluation.** We trace the refinement bias that Ma et al. removed by randomization [MHW+26, App. H.1], in our engine, to half-up gadget rounding on the grid of double-precision FFT outputs and remove it by ties-to-even rounding. In one TFHE-rs-based engine, the descent alone is 1.44–2.40× faster than Meta-PBS inside the module with the same dense keys, the descent and the shared keys together 1.51–3.32× with one-hot keys, and 1.37–3.57× against a baseline re-planned for 131 bits, and general tables are 3.8–6.6× faster than WoP-PBS in its authors' configuration.

1.2 Technical Overview

We begin by recalling how PBS and Meta-PBS place a table in a test vector. Let $c = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ encrypt $m \in \mathbb{Z}_t$. After switching to $2N$, its phase is $\frac{2N}{t}m + e'$, where the perturbation e' is dominated by the modulus-switching rounding. PBS rotates a test vector in which every entry is repeated $r_0 = 2N/t$ times and succeeds when $|e'|$ stays below about half of this redundancy. Meta-PBS [MHW+26] removes this requirement by alternating two steps after a first rotation of a non-redundant table:

- **Step A: amplify the redundancy.** TR^* keeps a window $[\![-T, T]\!]$ around the constant term and repeats each kept coefficient β times,

$$\mathrm{truncRepeat}(P, [\![-T, T]\!], \beta) = \textstyle\sum_{u=-T}^{T} P[u]\, X^{u\beta} \sum_{v \in [\beta]^{\mathrm{sym}}} X^v.$$

- **Step B: refine the perturbation.** A division of the input with remainder yields digits $c_0, c_1, \ldots$; rotating by c_{i+1} moves the β copies onto the target and leaves a new perturbation at the scale of one modulus switch.

Both steps act on one ring element, so the first test vector must hold the whole table, $t \le 2N$. For $t = 2\tau_0 N$ the table needs τ_0 ring polynomials, and Descent-PBS changes only where they are rotated.

Step 1: rotate once in the module. The non-redundant table is a single element of $\mathcal{R}_M$, $M = \tau_0 N$, or equivalently τ_0 interleaved sub-tables of $\mathcal{R}_N$,

$$\mathrm{TV} = \textstyle\sum_{j<M} \frac{q}{t} f(j)\, Y^j \in \mathcal{R}_M, \qquad \mathrm{TV}_\nu = \sum_{p<N} \frac{q}{t} f(p\tau_0 + \nu)\, X^p \in \mathcal{R}_N,\ \nu < \tau_0.$$

With keys $\mathsf{RGSW}(s_i)$ over $\mathcal{R}_N$, one rotation costs τ_0 external products per key [LY23]. Proposition 1 shows that this rank cannot be avoided when the later steps only copy coefficients.

Step 2: descend to the ring by one key switch. With phase $B - A\,\iota(S)$, $S \in \mathcal{R}_N$ and $\iota(S) = \sum_k S[k] Y^{k\tau_0}$,

$$\mathrm{truncRepeat}\big(A\,\iota(S)\big) = \textstyle\sum_{k<N} S[k]\ \mathrm{truncRepeat}\big(AY^{k\tau_0}\big), \tag{2}$$

which is a public functional key switch [CGGI20], [MHW+26, §2.4] whose decompositions read all τ_0 components in the clear. We realise it with the column merging of Meta-PBS at stride $\tau_0\beta$ and call it M2R: for a fixed merge width it needs τ_0 times fewer decompositions than repacking each component separately, every output coefficient copies one input coefficient, and τ_0 enters only through the merge condition $(2T_0 + 1 + \epsilon_0\tau_0)\beta_0 \le N$. The closest related structure appears in the first attempt at Meta-PBSmanyLUT [MHW+26, §4.1.1]. It interleaves sub-tables from a subring $R_{q'}$ of $\mathcal{R}_N$ under a secret in $\mathcal{R}_N$, and the authors set it aside; an earlier ePrint version of [MHW+26] notes that the fast TR^* does not apply there, since $S \notin R_{q'}$. Descent-PBS uses the opposite placement: the table spans

the module and the secret lies in the subring, so truncation commutes with the secret and one key switch ends the module computation.

Step 3: refine in the ring. After M2R the accumulator holds β_0 copies of the target in $\mathcal{R}_N$, and Steps A and B of Meta-PBS finish with K rotations in $\mathcal{R}_N$, at total cost $(\tau_0 + K)\, n_{\mathrm{k}}$ EP instead of $(K' + 1)\, \tau_0\, n_{\mathrm{k}}$ EP for Meta-PBS inside the module.

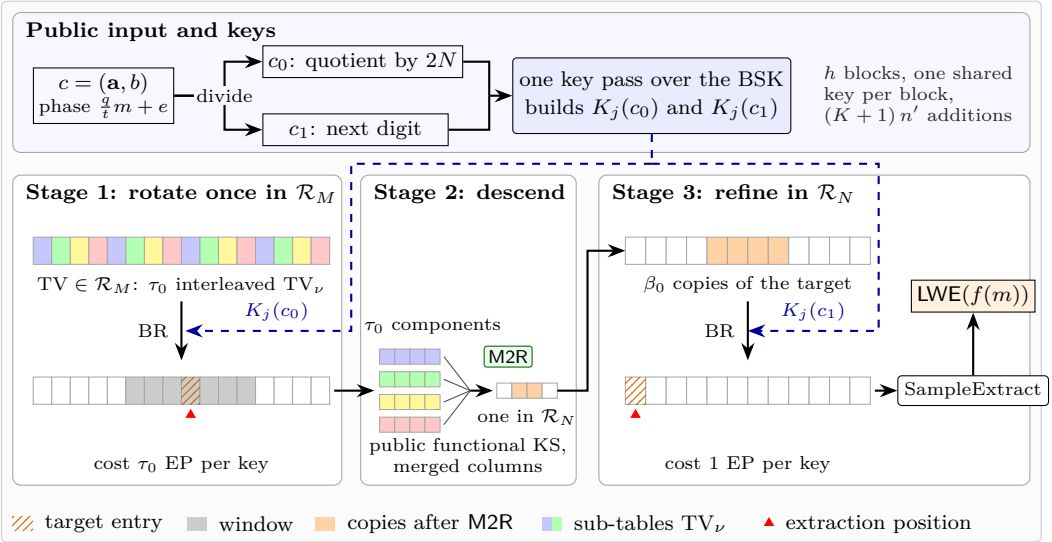


Figure 1: The workflow of DescPar-S with $t = 2\tau_0 N$, $\tau_0 = 4$, and one refinement.

An example. Figure 1 shows one bootstrapping of DescPar-S with $\tau_0 = 4$, that is, a negacyclic table of 2^{14} entries at $N = 2048$, with one refinement. The input is first key-switched to the block one-hot secret and divided with remainder, by $2N$ and then by $\tau_0\beta_0$, which yields the digit c_0 of the first rotation and the digit c_1 of the refinement. Both digits are known before any rotation, so a single pass over the bootstrapping key forms the combined keys $K_j(c_0)$ and $K_j(c_1)$ of all h blocks, using only shifted additions. The table is stored once, without redundancy, as τ_0 interleaved sub-tables that together form one element of $\mathcal{R}_M$. Because the first division is by $2N$, every exponent of c_0 is a multiple of τ_0, and all τ_0 components rotate under the same scalar key; the first rotation therefore costs $\tau_0 h$ external products in $\mathcal{R}_N$ and leaves the target inside a known window. M2R then reads the τ_0 components and returns a single accumulator in $\mathcal{R}_N$ that holds β_0 consecutive copies of the target, so that the rank is one from this point on. The refinement rotates by the small digit c_1 at a cost of h external products and moves one copy onto the constant term, which is extracted as LWE$(f(m))$.

For a 16-bit table, the same pipeline has $\tau_0 = 16$ components and $K = 3$ refinements, and costs $(16 + 3) \cdot 72 = 1368$ external products and one key pass. The dense module member costs $(16 + 3) \cdot 1210$, while Meta-PBS inside the module costs $16(K' + 1) \cdot 1210$, since it carries all 16 components through each of its $K' + 1$ rotations. The descent thus turns the dependence on the rank from multiplicative into additive.

1.3 Related Works

Our work is most closely related to programmable bootstrapping of tables beyond the ring dimension on a single LWE ciphertext.

Table 1: Single-ciphertext PBS for t-entry negacyclic tables as rank schedules. Cost in $n_{\mathrm{k}} \cdot \mathsf{EP}$; ρ is the redundancy of a one-shot table and $K \le 3$ in all our plans. †Baseline introduced here. ‡Below the frontier $\tau_0(\delta_0+1) \lesssim N/4$. §Structurally; at fixed q, N and noise, a feasible chain and input-noise budget are required.

route	method	schedule	cost	range
single rotation	TFHE PBS [CGGI20, CJP21]	ring $\rho t/2$	$\approx \rho\tau_0$	any t§
module, one shot	EBS [LY23] ≡ BootMMPM [DLLM25]; MBS [WBL+25]; SBS [BORLT25]	$(\rho\tau_0; -)$	$\le \rho\tau_0$	any t§
ring, iterative composition†	Meta-PBS [MHW+26]	$(1;1)$	$K+1$	$t \le 2N$
	Meta-Mod: Meta-PBS in the module	$(\tau_0;\tau_0)$	$(K+1)\tau_0$	any t§
Descent-PBS	**DescMod (ours)**	$(\tau_0;1)$	τ_0+K	any t§
	DescPar, DescPar-S (ours)	$(\tau_0$ par.$;1)$	τ_0+K	frontier‡

Module bootstrappings [LY23, DLLM25, WBL+25, BORLT25] rotate a ρ-redundant table once in a module of rank $\rho\tau_0$; Meta-PBS [MHW+26], the closest work, removes the redundancy by iteration in the ring but stops at $t \le 2N$; digit and tree methods [LMP22, GBA21, WHS+25, BBR26] and circuit bootstrapping [BBB+23] work on several ciphertexts. Block one-hot secrets [LMSS23, JLLS26] and mean compensation [dRDV26] reduce the external products and the rounding variance of a rotation.

Our work. Unlike module bootstrappings, which pay the rank and the redundancy in one rotation, and unlike Meta-PBS, which stays in the ring, we pay the module rank once and descend to the ring by a single key switch. Additionally, with block one-hot secrets, the components of the first rotation share one combined key per block and one pass over the bootstrapping key serves all rotations of a negacyclic table. Both techniques compose with mean compensation and are orthogonal to digit-decomposition methods.

Table 1 compares these approaches as rank schedules $(\tau_0;\tau_{\ge 1})$: one-shot module rotations grow linearly with the rank and the redundancy, Meta-PBS inside the module pays τ_0 in every step, and Descent-PBS pays it once, with one ciphertext in and out.

2 Preliminaries

2.1 Notations

We work in $\mathcal{R}_N = \mathbb{Z}_q[X]/(X^N+1)$ and in its rank-τ extension $\mathcal{R}_M$, $M = \tau N$. A table $f\colon \mathbb{Z}_t \to \mathbb{Z}_t$ is negacyclic if $f(x+t/2) = -f(x)$. Table 2 collects the symbols used throughout.

Table 2: Notation.

symbol	meaning	symbol	meaning
$[a]$, $[\![a,b]\!]$	$\{0,\dots,a-1\}$, $\mathbb{Z}\cap[a,b]$	q, N	2^{64}; ring dimension, a power of two
$[a]^{\mathrm{sym}}$	$[\![v_{\mathrm{lo}}(a), v_{\mathrm{lo}}(a)+a-1]\!]$	$v_{\mathrm{lo}}(a)$	$-\lfloor (a-1)/2 \rfloor$
$\mathcal{R}_N$	$\mathbb{Z}_q[X]/(X^N+1)$	$\mathcal{R}_M$	$\mathbb{Z}_q[Y]/(Y^M+1)$, $M=\tau N$
τ	module rank	ι	$\mathcal{R}_N \to \mathcal{R}_M$, $X \mapsto Y^\tau$
t	plaintext modulus, a power of two	f	table $\mathbb{Z}_t \to \mathbb{Z}_t$
τ_0	$\max(1, t/2N)$	r_0	$\max(1, 2N/t)$
z	input and output LWE secret	$\mathbf{s}$	block one-hot secret
h, w	number and width of blocks	n'	hw
β_i, T_i	base and half-width of repacking i	ϵ_i	merge width of repacking i
K	number of refinements	V_{in}	input noise variance
V_{br}	blind-rotation noise variance	V_{MS}	modulus-switch rounding variance
z_p	$\sqrt{2}\,\mathrm{erfc}^{-1}(p)$	EP	cost of one external product

2.2 LWE, RLWE and External Products

$\mathsf{LWE}_s(\mu) = (a, b)$ with $b - \langle a, s\rangle = \mu + e$; $\mathsf{RLWE}_S(P) = (A, B) \in \mathcal{R}_N^2$ with $B - AS = P + E$, $S \in \mathcal{R}_N$ binary. For a gadget (B_g, ℓ), $\mathsf{RGSW}(m)$ consists of 2ℓ RLWE encryptions of zero to which $m\, g_i$, with the gadget vector $g_i = q/B_g^i$, $i = 1, \ldots, \ell$, is added in the mask or the body; the ℓ body-side rows form $\mathsf{RLev}(m)$, the key-switching key. The external product $\mathsf{RGSW}(m) \boxdot C$ rounds both polynomials of C to $\ell \log_2 B_g$ bits, decomposes them into ℓ balanced digits, multiplies with the key in the Fourier domain and transforms back; it returns $\mathsf{RLWE}_S(mP)$ with additive noise

$$2\ell N \tfrac{B_g^2}{12} \sigma_{\text{key}}^2 + \sigma_{\text{rnd}}^2, \qquad \sigma_{\text{rnd}}^2 = \tfrac{q^2}{B_g^{2\ell}} \cdot \tfrac{N/2+1}{12}, \tag{3}$$

the second term being the rounding; for a combined key of a block of w keys σ_{key}^2 becomes $w\sigma_{\text{key}}^2$. We denote its cost by EP. A blind rotation of $\mathrm{TV} \in \mathcal{R}_N$ by an LWE ciphertext switched to $\mathbb{Z}_{2N}$ returns $\mathsf{RLWE}_S(\mathrm{TV} \cdot X^{-\psi(c)})$ with one external product per key, adding variance V_{br}.

2.3 Division with Remainder and Repacking

Division with remainder [MHW+26, Def. 3]. The digits of every rotation are integers computed in the clear from $c = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$, coefficient-wise. Let $\psi^*(\mathbf{a}, b) = b - \langle \mathbf{a}, \mathbf{s}\rangle \in \mathbb{Z}$ be the unreduced phase and $\lfloor\cdot\rceil$ rounding half up.

- *First division.* For a power of two $q_0 \mid q$ and the unit $u = q/q_0$,

$$\begin{aligned} c_{\text{rem}} &= c - u\lfloor c/u \rceil \in [-\tfrac{u}{2}, \tfrac{u}{2})^{n+1}, \quad c_{\text{quo}} = (c - c_{\text{rem}})/u \in \mathbb{Z}_{q_0}^{n+1}, \\ \psi^*(c) &= u\, \psi^*(c_{\text{quo}}) + \psi^*(c_{\text{rem}}). \end{aligned}$$

- *Later divisions.* An integer base $\beta \geq 2$ divides the unreduced remainder $r \in [-\frac{u}{2}, \frac{u}{2})^{n+1}$ with the same unit,

$$\begin{aligned} c'_{\text{quo}} &= \lfloor \beta r/u \rceil \in [\![-\lceil \beta/2 \rceil, \lceil \beta/2 \rceil]\!]^{n+1}, \quad r' = \beta r - u\, c'_{\text{quo}}, \\ \beta\, \psi^*(r) &= u\, \psi^*(c'_{\text{quo}}) + \psi^*(r'). \end{aligned}$$

  Since r is never reduced modulo q, no divisibility $\beta \mid q$ is needed.

- *Rounding noise.* Each division adds one rounding per nonzero secret coefficient and the body, of variance $V_{\text{MS}} = n/24 + 1/12$ for a dense binary secret and $V_{\text{MS}} = (h+1)/12$ for a block one-hot secret [MHW+26].

Repacking [MHW+26, Def. 2, §5.2]. For $P \in \mathcal{R}_N$, a window $[\![-T, T]\!]$ and $\beta \geq 2$,

$$\begin{aligned} \text{truncRepeat}(P, [\![-T, T]\!], \beta) &= \textstyle\sum_{u=-T}^{T} P[u]\, X^{u\beta} \sum_{v \in [\beta]^{\text{sym}}} X^v, \\ \text{truncRepeat}(B - AS) &= \text{truncRepeat}(B) - \textstyle\sum_{k<N} S[k]\ \text{truncRepeat}(AX^k), \end{aligned}$$

the second identity by $\mathbb{Z}_q$-linearity. Hence:

- a column-method key switch with the keys $\mathsf{RLev}(S[k] \sum_v X^v)$ evaluates truncRepeat homomorphically;
- the merged version $\mathsf{TR}^*(C, [\![-T, T]\!], \beta, \epsilon)$ groups $\epsilon + 1$ columns per decomposition and costs $\lceil N/(\epsilon+1) \rceil$ decompositions;
- it is exact on the useful rows $[\![-T, T]\!]\beta + [\beta]^{\text{sym}}$ if $(2T + 1 + \epsilon)\beta \leq N$.

2.4 Module Accumulators

Following Duan et al. [DLLM25], we identify an element of $\mathcal{R}_M$ with the vector of its τ components in $\mathcal{R}_N$,

$$P = \textstyle\sum_{j<\tau} \iota(P_j)\, Y^j \iff (P_0, \dots, P_{\tau-1}) \in \mathcal{R}_N^\tau, \qquad P[u] = P_{u \bmod \tau}\big[\lfloor u/\tau \rfloor\big].$$

If $S \in \mathcal{R}_N$ and $C = (A, B) \in \mathcal{R}_M^2$ satisfies $B - A\,\iota(S) = P + E$, then $B_j - A_j S = P_j + E_j$ for every $j < \tau$, i.e. each component is an RLWE_S ciphertext. Multiplication by Y^c acts on the components as the monic monomial permutation matrix

$$Y^c \cdot P \iff \Phi(c) \begin{pmatrix} P_0 \\ \vdots \\ P_{\tau-1} \end{pmatrix}, \qquad \Phi(c) = \begin{pmatrix} 0 & X^{a+1} I_b \\ X^a I_{\tau-b} & 0 \end{pmatrix},$$

where $c = a\tau + b$ with $a \in [2N]$ and $b \in [\tau]$, with $\Phi(a\tau) = X^a I_\tau$ and $\Phi(c)\Phi(c') = \Phi(c + c')$, so Φ represents $\mathbb{Z}_{2M}$. Consequently, a CMux over $\mathcal{R}_M$ with a key over $\mathcal{R}_N$ costs τ external products, and a module rotation costs $\tau \cdot n \cdot \mathsf{EP}$ and adds the noise V_{br} of one $\mathcal{R}_N$ rotation to each component [LY23, Prop. 2]. Extended bootstrapping [LY23], monomial-matrix bootstrapping [DLLM25], MBS [WBL+25] and SBS [BORLT25] rotate a ρ-redundant table in one shot with $\tau = \rho\tau_0$; a 12-bit table needs $\tau = 256$ at $N = 2048$.

2.5 Meta-PBS

Meta-PBS [MHW+26, Alg. 1] computes $\mathsf{HomDivRem}(c, 2N)$, rotates a test vector whose values are r_0-redundant, and repeats for $i < K$

$$C_i' \leftarrow \mathsf{TR}^*(C_i, [\![-T_i, T_i]\!], \beta_i, \epsilon_i)\, X^{\Delta_i}, \qquad (c_{\mathrm{quo},i+1}, c_{\mathrm{rem},i+1}) \leftarrow \mathsf{HomDivRem}(c_{\mathrm{rem},i}, \beta_i),$$
$$C_{i+1} \leftarrow \mathsf{BR}(c_{\mathrm{quo},i+1}, C_i'),$$

before it extracts $\mathsf{SampleExtract}(C_K, 0)$. Its correctness argument [MHW+26, Thm. 2] tracks after rotation i a perturbation e_i of the target position and a redundancy r_i. With $\alpha_i = \prod_{j<i} \beta_j$ and $\kappa_i = (2N\alpha_i/q)^2 V_{\mathrm{in}}$, each $|e_i|$ is bounded by

$$\delta_i = \left\lceil z_p \sqrt{V_{\mathrm{MS}} + \kappa_i} + \tfrac{1}{2} \right\rceil$$

except with probability p, and the algorithm is correct if

$$r_0 \le \frac{2N}{t}, \quad T_i \ge \delta_i + \left\lfloor \frac{r_i}{2} \right\rfloor, \quad (2T_i + 1 + \epsilon_i)\beta_i \le N, \quad r_{i+1} \le r_i \beta_i, \quad r_K \ge 2\delta_K + 1. \tag{4}$$

The first condition is the only one that couples t and N, and it is the limit $t \le 2N$; for higher precision the authors recommend decomposition-based methods [MHW+26, §6.2.5]. Running Meta-PBS inside the module, at rank τ_0 throughout, is our baseline Meta-Mod of cost $(K'+1)\,\tau_0\, n\,\mathsf{EP}$. Its refinements have small exponents; Ma et al. observed a biased output noise, larger for small β_i, and removed it by multiplying the accumulator by X^{r_i} between CMux gates with $\sum_i r_i = 0$, a randomized blind rotation used in all their benchmarks [MHW+26, App. H.1, §6.1].

2.6 Block One-Hot Secrets

A block one-hot secret $\mathbf{s} \in \{0,1\}^{n'}$ [LMSS23, JLLS26] has h blocks $B_1, \dots, B_h$ of $w = n'/h$ coordinates, each with exactly one nonzero entry $k_j^* \in B_j$. Since $s_k = 0$ for every other k, a blind rotation by the exponents $\mathbf{e}$ computes

$$X^{-b+\langle \mathbf{e}, \mathbf{s} \rangle} \cdot \mathrm{TV} = X^{-b} \cdot \mathrm{TV} \cdot \prod_{j \le h} X^{e_{k_j^*}}, \qquad K_j(\mathbf{e}) = \sum_{k \in B_j} X^{e_k} \cdot \mathsf{RGSW}(s_k) = \mathsf{RGSW}\big(X^{e_{k_j^*}}\big),$$

where the *combined key* K_j is formed from the w keys of a block by monomial shifts and additions. A rotation thus costs h external products instead of n plus n' shifted additions, and $V_{\mathrm{MS}} = (h+1)/12$ instead of $n/24 + 1/12$; the shifted additions read the whole bootstrapping key once per rotation, which Jain et al. note may become non-negligible [JLLS26, p. 7] and which dominates at $N = 2048$, the *naive sparse* baseline.

3 The Descent-PBS Framework

3.1 Our Construction of Descent-PBS

We pay the rank of a large table once, in the first rotation, and descend to $\mathcal{R}_N$ by a single key switch for all later rotations, after which the refinements of Meta-PBS run unchanged.
Motivation: beyond the ring dimension. As recalled in Section 2.5, Meta-PBS needs $t \le 2N$. The reason is that a rotation and a copy both act on coefficients by signed permutations: for $P \in \mathcal{R}_N$ and $x \in \mathbb{Z}_{2N}$,

$$(P \cdot X^{-x})[v] = (-1)^{\lfloor ((v+x) \bmod 2N)/N \rfloor} P[(v+x) \bmod N], \qquad v < N. \tag{5}$$

A rotation therefore moves coefficients and flips their signs, but never creates a value that was not already in the test polynomial. The same holds for the operations that Meta-PBS applies between rotations: TR^* keeps a window and repeats each kept coefficient, and the final extraction reads one coefficient. Whatever the number of rotations, the output is one of the values that the test polynomials of the first rotation contained, up to sign. A polynomial of $\mathcal{R}_N$ holds at most N such values, while a negacyclic table on $\mathbb{Z}_t$ may take $t/2$ different absolute values, so a single ring polynomial cannot carry a table with $t > 2N$, no matter how the later steps are arranged. The following proposition makes this precise.

Proposition 1 (The $2N$ barrier for copy-only pipelines)**.** *Consider a pipeline that (a) fixes its test polynomials* $\mathrm{TV}_1, \ldots, \mathrm{TV}_k \in \mathcal{R}_N$ *of the first rotation before the input arrives, (b) rotates* $\mathcal{R}_N$*-accumulators by LWE ciphertexts, (c) between rotations applies only maps with* $P'[v] \in \{0, \pm P[\phi(v)]\}$ *for some index map* ϕ*, i.e. collision-free selections, copies, sign changes and extractions, and (d) rotates only accumulators produced earlier. Let* $v_f = \big|\{|f(x)| \neq 0 : x \in [t/2]\}\big|$ *for centred values. Then*

(i) $\mathrm{Dec}(\mathrm{out}) \in \mathcal{V} = \{0\} \cup \{\pm \mathrm{TV}_j[u] : j \le k,\, u < N\}$, $|\mathcal{V}| \le 2kN+1$; *(ii)* $k \ge \lceil v_f/N \rceil$;

(iii) if $v_f = t/2$*, then* $k \ge \tau_0$ *and a CMux-based first rotation costs at least* $\tau_0\, n_{\mathrm{k}}\, \mathsf{EP}$.

Proof (sketch). By (5) and (c), every step maps each coefficient of its output to 0 or to $\pm$ one coefficient of its input, so by induction over the steps, using (d), every coefficient of every accumulator lies in $\mathcal{V}$, which is fixed by (a); this is (i). The nonzero elements of $\mathcal{V}$ have at most kN absolute values, and each of the v_f values $|f(x)|$ must be one of them, so $v_f \le kN$, which is (ii); $v_f = t/2 = \tau_0 N$ gives $k \ge \tau_0$. Each of the n_{k} CMuxes multiplies all k polynomials by an RGSW ciphertext, which gives (iii). The full proof is in the full version. □

For Meta-PBS $k = 1$, which is its limit $r_0 \le 2N/t$; more divisions cannot help, since they only select which coefficient is output. Programs that combine coefficients linearly are outside this copy-only model; for them the full version gives a rank–precision bound in an idealised noise model, as an auxiliary explanation only. A module of rank τ_0 supplies exactly $k = \tau_0$ polynomials of $\mathcal{R}_N$, the least that the proposition allows; extended and monomial-matrix bootstrapping use the larger rank $\rho\tau_0$, because their single rotation must also hold the redundancy. The bound concerns only the first rotation, however. It does not say that the later rotations must also run at rank τ_0, and this is the freedom that Descent-PBS uses.

Key observation: a subring secret lets the module descend. Repacking each component with its own TR^* keeps the accumulator in the module, so Meta-PBS inside the module, Meta-Mod, pays τ_0 in every step; with a secret in the subring $\mathcal{R}_N$, one key switch repacks all components into $\mathcal{R}_N$ (Figure 2). To see why, write a module element as $A = \sum_{\nu<\tau_0} A_\nu(X)\,Y^\nu$ with $A_\nu \in \mathcal{R}_N$ and $X = Y^{\tau_0}$. The embedded secret $\iota(S) = \sum_k S[k]Y^{k\tau_0}$ is the polynomial S evaluated at X, so multiplication by $\iota(S)$ multiplies every component by the same S and does not mix the components: $A\,\iota(S) = \sum_\nu (A_\nu S)(X)\,Y^\nu$. Every coefficient of $A\,\iota(S)$ is therefore a public linear form in the coefficients of S, and so is every coefficient that a repacking keeps, whichever component it comes from. A single public functional key switch with keys for the coefficients of S can thus evaluate the repacking of all τ_0 components at once and write the result into one polynomial of $\mathcal{R}_N$, provided that the repeated windows of the different components do not collide. This collision-freeness is the merge condition below. After this key switch the accumulator is an ordinary RLWE_S ciphertext, and every later step of Meta-PBS applies to it unchanged. For a secret that is not embedded from the subring, the product $A\,\iota(S)$ mixes the components and this argument does not apply.

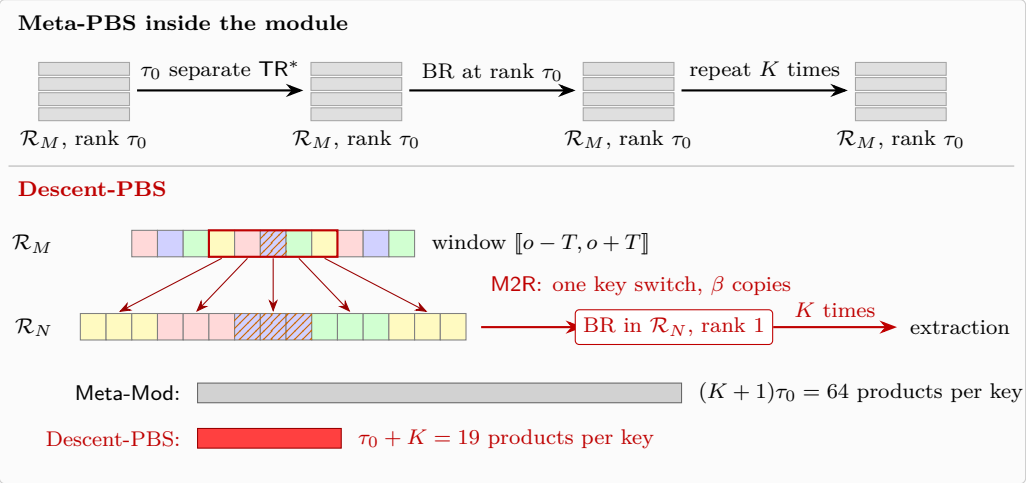


Figure 2: Meta-PBS inside the module and Descent-PBS, illustrated with $\tau_0 = 16$ and the same $K = 3$; colours in $\mathcal{R}_M$ mark the components. The tuned baselines of Section 5 use their own K'.

Let $\tau = \tau_0$ and $S \in \mathcal{R}_N$. For $P \in \mathcal{R}_M$, a centre o and a window $[\![-T, T]\!]$, $T < M/2$, the *centred* repacking $\mathrm{truncRepeat}_\tau(P, [\![o-T, o+T]\!], \beta) = \sum_{u=-T}^{T} P[o+u]\,X^{u\beta} \sum_{v\in[\beta]^{\mathrm{sym}}} X^v \in \mathcal{R}_N$ writes the coefficient at offset u from the centre to the positions $u\beta + [\beta]^{\mathrm{sym}}$; we write it for $o = 0$ below (a general centre is a monomial shift of the input). It acts on big-ring coefficients, and let $\mathrm{truncPad}_\tau$ be the same without the repetition. For $C = (A, B) \in \mathcal{R}_M^2$, a merge width ϵ and $k \in (\epsilon+1)\mathbb{Z} \cap [N]$ define $D_k = \mathrm{truncPad}_\tau(A\,Y^{k\tau}, [\![-T-\epsilon\tau, T]\!], \beta) \in \mathcal{R}_N$ and $\tilde{S}_k = \sum_{0\le s\le\epsilon} S[k+s]X^{s\tau\beta} \sum_{v\in[\beta]^{\mathrm{sym}}} X^v$ (with $S[k+s] = 0$ for $k+s \ge N$), and

$$\mathsf{M2R}(C) = \Big(\textstyle\sum_k \langle G^{-1}(-D_k), \mathsf{RLev}(\tilde{S}_k)\rangle_A, \\ \mathrm{truncRepeat}_\tau(B, [\![-T, T]\!], \beta) + \sum_k \langle G^{-1}(-D_k), \mathsf{RLev}(\tilde{S}_k)\rangle_B \Big).$$

Here G^{-1} is the gadget decomposition of the repacking keys and $\langle\cdot,\cdot\rangle_A, \langle\cdot,\cdot\rangle_B$ are the two components of the gadget inner product. A noiseless unit test of our implementation

checks every useful row for $\tau \in \{1, 2, 4\}$ with and without merging. The lower half of Figure 2 shows the map on the plaintext.

Lemma 1 (Module-to-ring repacking)**.** *Let $C = \mathsf{RLWE}_S^{(M)}(P; E)$ with an arbitrary error E, let $\beta \geq 2$, and let*

$$(2T + 1 + \epsilon\tau)\,\beta \leq N. \tag{6}$$

Then $\mathsf{M2R}(C) = \mathsf{RLWE}_S(P'; E')$ in $\mathcal{R}_N$, and:

(i) Phase. *For every $u \in [\![-T, T]\!]$ and $v \in [\beta]^{\mathrm{sym}}$,*

$$P'[u\beta + v] = P[u], \qquad E'[u\beta + v] = E[u] + E''[u\beta + v],$$

where E'' is the error of the column key switch.

(ii) Cost. $\mathsf{M2R}$ *uses $\lceil N/(\epsilon + 1)\rceil$ gadget decompositions and as many* RLev *keys, independently of τ for a fixed ϵ.*

(iii) Noise. *With the repacking gadget $(B_{\mathrm{tr}}, \ell_{\mathrm{tr}})$, key noise of variance σ_{k}^2 independent of the digits, and balanced digits with $\mathbb{E}[d^2] \leq (B_{\mathrm{tr}}^2 + 2)/12$, as for digits uniform on their balanced range,*

$$\mathrm{Var}(E'') \leq V_{\mathrm{round}} + \left\lceil \frac{N}{\epsilon + 1} \right\rceil (2T + 1 + \epsilon\tau)\,\ell_{\mathrm{tr}}\,\frac{B_{\mathrm{tr}}^2 + 2}{12}\,\sigma_{\mathrm{k}}^2,$$

plus, in an FFT engine, the floating-point term of [MHW+26, Lem. 6], whose rounding term is V_{round}.

Proof (sketch). Write $\iota(S) = \sum_k S[k]Y^{k\tau}$. By linearity, $\mathrm{truncRepeat}_\tau(A\,\iota(S)) = \sum_k S[k]\,\mathrm{truncRepeat}_\tau(AY^{k\tau})$, and column $k + s$ equals column k with the window shifted by $s\tau$ and the output multiplied by $X^{s\tau\beta}$. Merging $\epsilon + 1$ columns therefore reproduces every useful term once and places the surplus terms in two blocks of $\epsilon\tau$ rows beside the useful block; by (6) the useful block and one surplus block span at most N consecutive exponents, so no surplus term reaches a useful row, even after the negacyclic wrap. The key switch of $-D_k$ then subtracts $\mathrm{truncRepeat}_\tau(A\,\iota(S))$ on the useful rows, which leaves $\mathrm{truncRepeat}_\tau(P + E)$ plus the key-switching error: one input coefficient per useful row, which gives (i); (ii) counts the columns, and (iii) sums ℓ_{tr} digit polynomials with at most $2T + 1 + \epsilon\tau$ nonzero coefficients per column. The full proof is in the full version. □

Remarks. At $\tau = 1$, $\mathsf{M2R}$ is the TR^* of Meta-PBS. Since the feasible ϵ shrinks with τ by (6), our sparse plans still use 21, 38, 56 and 128 decompositions at $\tau_0 = 2, 4, 8, 16$, against τ_0 times as many componentwise. At $\epsilon = 0$, $\mathsf{M2R}$ is public functional key switching [CGGI20], [MHW+26, §2.4], realised with the column merging of Meta-PBS at stride $\tau\beta$; trace packing [GHPS12, CDKS21] uses automorphisms, and module accumulators [LY23, WBL+25] stay in the module.

Concrete example: $t = 2^{16}$**,** $N = 2048$**,** $\tau_0 = 16$**.** The 16-bit plan of $\mathsf{DescPar\text{-}S}$ has $\beta = (2, 19, 11)$, $T = (388, 25, 49)$, $\epsilon = (15, 56, 87)$. **Step 1:** the 16 sub-tables are rotated by the same first digit, at $16 \cdot 72$ external products. **Step 2:** $\mathsf{M2R}$ keeps $[\![-388, 388]\!]$ twice each, $(777 + 240) \cdot 2 = 2034 \leq 2048$, with 128 decompositions instead of $16 \cdot 128$. **Step 3:** three rotations of 72 products each follow, with centres $o = (0, 0, 9)$. **Step 4:** the total is $(16 + 3) \cdot 72 = 1368$ external products; $\mathsf{Meta\text{-}Mod}$ would pay $(K + 1)\,\tau_0\,h = 4608$ with the same K, and our tuned Meta-Mod-S, which needs only $K' = 1$, pays 2304.

Algorithm 1 Descent-PBS with a dense secret

Require: $\mathsf{LWE}_z(\frac{q}{t}m)$, z the coefficient vector of S; negacyclic f on $\mathbb{Z}_t$, $t = 2\tau_0 N$; keys $\mathsf{RGSW}(s_i)$ over $\mathcal{R}_N$; key switch $z \to \mathbf{s}$; M2R and TR* keys; plan $(\beta_i, T_i, \epsilon_i)_{i<K}$
Ensure: $\mathsf{LWE}_z(\frac{q}{t}f(m))$

// Step 1: digits
$c \leftarrow \mathsf{KS}_{z\to\mathbf{s}}(c)$ ▷ input key switch, counted in all timings
$q_0 \leftarrow 2M$, $m_0 \leftarrow \beta_0$ for DescMod; $q_0 \leftarrow 2N$, $m_0 \leftarrow \tau_0\beta_0$ for DescPar; $m_i \leftarrow \beta_i$ for $0 < i < K$
$(d^{(0)}, r^{(0)}) \leftarrow \mathsf{HomDivRem}(c, q_0)$; $(d^{(i+1)}, r^{(i+1)}) \leftarrow \mathsf{HomDivRem}(r^{(i)}, m_i)$ for $i < K$
// Step 2: first rotation at rank τ_0
if DescMod **then** $C \leftarrow \mathsf{BR}_{\mathcal{R}_M}(d^{(0)}, \mathrm{TV})$ with $\mathrm{TV} = \sum_{j<M} \lfloor \frac{q}{t} f(j) \rceil Y^j$
else $C_\nu \leftarrow \mathsf{BR}_{\mathcal{R}_N}(d^{(0)}, \mathrm{TV}_\nu)$ with $\mathrm{TV}_\nu = \sum_{p<N} \lfloor \frac{q}{t} f(p\tau_0 + \nu) \rceil X^p$, $\nu < \tau_0$; $C \leftarrow \sum_\nu \iota(C_\nu) Y^\nu$
end if
$\lambda \leftarrow \rho \leftarrow 0$
// Step 3: descent and refinements in $\mathcal{R}_N$
for $i = 0, \ldots, K-1$ **do**
  $o \leftarrow \lfloor (\lambda + \rho)/2 \rfloor$ ▷ centre on the copies
  $C \leftarrow \mathsf{R}_i(C, [\![o - T_i, o + T_i]\!], \beta_i, \epsilon_i)$ ▷ $\mathsf{R}_0 = \mathsf{M2R}$, $\mathsf{R}_i = \mathsf{TR}^*$
  $(\lambda, \rho) \leftarrow \big((\lambda - o)\beta_i + v_{\mathrm{lo}}(\beta_i),\ (\rho - o)\beta_i + v_{\mathrm{lo}}(\beta_i) + \beta_i - 1\big)$
  $C \leftarrow \mathsf{BR}_{\mathcal{R}_N}(d^{(i+1)}, C)$ ▷ n_{k} external products
end for
// Step 4: extraction
return $\mathsf{SampleExtract}(C, 0)$ ▷ an LWE ciphertext under z

3.2 The Module and Parallel Members

Algorithm 1 has two members, which differ only in Steps 1 and 2:

- **The module member DescMod** divides by $2M$ and rotates the non-redundant table $\mathrm{TV} \in \mathcal{R}_M$ by $d^{(0)}$ modulo $2M$, with keys over $\mathcal{R}_N$.
- **The parallel member DescPar** divides by $2N$ and rotates the τ_0 sub-tables $\mathrm{TV}_\nu \in \mathcal{R}_N$ by the same $d^{(0)}$ modulo $2N$; read as components, the outputs form one ciphertext of $\mathcal{R}_M$,
$$C = \textstyle\sum_{\nu<\tau_0} \iota(C_\nu)\, Y^\nu, \qquad C[u] = \frac{q}{t}\, f\big(\tau_0 \psi(d^{(0)}) + u\big),$$
and its keys are those of an $\mathcal{R}_N$ rotation.

Comparison with Meta-PBS. Steps 3 and 4 are those of Meta-PBS, except that the first repacking descends from $\mathcal{R}_M$ and the windows follow the tracked interval. Hence both members cost the same, and the gain over Meta-Mod grows with the rank:

$$\text{cost} = (\tau_0 + K)\, n_{\mathrm{k}}\, \mathsf{EP} \text{ for both members}, \quad \frac{\text{cost of Meta-Mod}}{\text{cost}} = \frac{(K'+1)\,\tau_0}{\tau_0 + K} \to K' + 1.$$

The first window. The division by $2N$ is τ_0 times coarser than the division by $2M$, so the first window of DescPar is about τ_0 times wider. Whether a window fits is a matter of geometry:

Proposition 2 (Frontier)**.** *For a prescribed integer half-width T_0, a first repacking with $\beta_0 \geq 2$ and $\epsilon_0 \geq 0$ exists iff $2(2T_0 + 1) \leq N$.*

With $\delta_0 = \lceil z_p \sqrt{V_{\mathrm{MS}}} \rceil$, the approximations

$$T_0^{\mathsf{DescMod}} \approx \delta_0, \qquad T_0^{\mathsf{DescPar}} \approx \tau_0(\delta_0 + 1)$$

explain the observed frontier, $\delta_0 < N/4$ for DescMod independently of τ_0 and roughly $\tau_0(\delta_0 + 1) < N/4$ for DescPar; they do not characterise a complete chain, which our plans check with Corollary 1. With a dense secret of dimension 1210 the planner finds DescPar plans up to 14 bits, 15 with mean compensation; with a one-hot secret, $h = 72$, the frontier passes every 16-bit table.

3.3 Correctness and Failure Probability

Correctness of both members follows from one statement about the index arithmetic of Algorithm 1. Let u be the unit of the first division, $\bar{t} = \max(t, 2N)$, $r_0 = \max(1, 2N/t)$ and e the input noise, and define

$$x_i = \frac{\psi^*(r^{(i)})}{u}, \qquad s_i = \begin{cases} \tau_0, & i = 0 \text{ for } \mathsf{DescPar}, \\ 1, & \text{otherwise}, \end{cases} \qquad \alpha_i = \prod_{j<i} \beta_j,$$

$$y_i = s_i x_i - \frac{\alpha_i}{\alpha_K} x_K, \qquad \varphi = \alpha_K \frac{\bar{t}e}{q} - x_K.$$

The centres o_i and the intervals $[\![\lambda_i, \rho_i]\!]$ are those of the algorithm, and indices are read modulo $2N$ with the negacyclic sign.

Theorem 1 (Correctness of descending chains)**.** *Let every repacking satisfy (6). Then:*

(a) Backward path. *Following the extracted coefficient backwards defines a formal offset w_i from o_i at repacking i, which is the offset of a true source coefficient iff $|w_i| \le T_i$, and*

$$w_i = y_i + \eta_i, \qquad \eta_i \in (\eta_i^-, \eta_i^+],$$

$$\eta_i^- = \frac{z_{i+1}^- - v_{\mathrm{lo}}(\beta_i)}{\beta_i} - 1, \qquad \eta_i^+ = \frac{z_{i+1}^+ - v_{\mathrm{lo}}(\beta_i)}{\beta_i}, \qquad z_i^\pm = \eta_i^\pm + o_i,$$

with $z_K = 0$ exactly and $z_K^\pm = 0$ in the recursion.

(b) Selection. *$\varphi \in \mathbb{Z}$, and if $|w_i| \le T_i$ for all $i < K$, the extraction reads a copy of the test-vector entry of m iff*

$$\varphi \in [\![\lambda_K - \lfloor r_0/2 \rfloor \alpha_K,\ \rho_K + (r_0 - 1 - \lfloor r_0/2 \rfloor)\alpha_K]\!].$$

(c) Scope. *These conditions are sufficient for the output to encrypt $\frac{q}{t} f(m)$ for every table; given $|w_i| \le T_i$ for all i, the condition of (b) is also necessary for tables whose reachable entries differ from $f(m)$, e.g. tables with distinct values.*

Proof (sketch). Division $i{+}1$ multiplies every unreduced remainder by m_i and splits it with the same unit, so by linearity of ψ^* the rotation amounts are exactly $d_{i+1} = s_i \beta_i x_i - x_{i+1}$. *Backwards*: let v_i be the position read before repacking i on the path to the extracted coefficient and $z_i = v_i - y_i$; rotation $i+1$ reads position $v_{i+1} + d_{i+1} = z_{i+1} + \beta_i y_i$ of the repacked accumulator, and the centred repacking maps offset w to $w\beta_i + v_{\mathrm{lo}}(\beta_i) + [\beta_i]$, so $w_i = \lfloor y_i + (z_{i+1} - v_{\mathrm{lo}}(\beta_i))/\beta_i \rfloor$ and $z_i = o_i + w_i - y_i$; induction from $z_K = 0$ gives (a), and by the merge condition the position is a useful row, with a unique source, iff $|w_i| \le T_i$. *Forwards*: after the first rotation, the target sits at $s_0 x_0 - \bar{t}e/q$, an integer, and if the copies of the target occupy $S_i = s_i x_i - \alpha_i \bar{t}e/q + [\![\lambda_i, \rho_i]\!]$, one repacking and one rotation map them to $S_{i+1} = x_{i+1} - \alpha_{i+1}\bar{t}e/q + [\![\lambda_{i+1}, \rho_{i+1}]\!]$, because $s_i x_i$ cancels against d_{i+1}. The path of (a) and the sets S_i follow the same position arithmetic, so the extracted coefficient is a copy of the target iff $0 \in S_K = -\varphi + [\![\lambda_K, \rho_K]\!]$, widened by α_K for each of the r_0 copies; this is (b), and (c) follows because any other position holds another entry. The full proof is in the full version. □

A bit-exact plaintext replay of our index arithmetic confirms the theorem on $4.2 \cdot 10^6$ inputs over all plans, $6.9 \cdot 10^5$ of them failing by inflated noise. Unlike in Meta-PBS, the input noise enters only (b), and the final interval is the exact set tracked by the centres.

Window variables. The variable y_i combines two remainders of the same input, but composing the divisions gives, per coordinate, y_i as an exact rounding of the earlier

remainder, whose variance for a uniform remainder is $s_i^2 v(A_i)$ with $A_i = s_i \alpha_K / \alpha_i$ and $v(A) = (1 - A^{-2})/12$ for odd A, $(1 + 2A^{-2})/12$ for even A (full version). The model further assumes independent coordinates, a Gaussian sum and input noise independent of the mask remainders; mean compensation enters through V_{MS}.

Corollary 1 (Model failure probability)**.** *Under these assumptions, let*

$$\sigma_{y_i}^2 = s_i^2 \cdot 12 V_{\mathrm{MS}} \cdot v(A_i), \qquad \sigma_\varphi^2 = \alpha_K^2 \kappa + V_{\mathrm{MS}}, \qquad \kappa = (\bar{t}/q)^2 V_{\mathrm{in}}.$$

Then the model probabilities of the failure events of Theorem 1 are at most

$$\hat{p}_{W_i} = Q\Big(\frac{T_i + 1 + \eta_i^-}{\sigma_{y_i}}\Big) + Q\Big(\frac{T_i + 1 - \eta_i^+}{\sigma_{y_i}}\Big), \qquad \hat{p}_{\mathrm{sel}} = Q\Big(\frac{\rho'_K + \frac{1}{2}}{\sigma_\varphi}\Big) + Q\Big(\frac{-\lambda'_K + \frac{1}{2}}{\sigma_\varphi}\Big),$$

for the interval $[\![\lambda'_K, \rho'_K]\!]$ of (b), and with $\hat{p}_{\mathrm{amp}}$ the model probability that the final noise exceeds $q/2t$,

$$\hat{p}_{\mathrm{dec}} \le \textstyle\sum_{i<K} \hat{p}_{W_i} + \hat{p}_{\mathrm{sel}} + \hat{p}_{\mathrm{amp}}.$$

The sum is a union bound over the events of Theorem 1 and needs no independence between them; the assumptions above enter only the marginal probabilities, so $\hat{p}_{\mathrm{dec}}$ is a model prediction and not a guarantee for the implementation.

Tail validation. Where failures are observable, at rates 2^{-16}–2^{-6} reached by inflating the noise, the replay of 10- to 16-bit plans matches $\hat{p}_{\mathrm{sel}}$ within 0.25 bits and stays 0.07–0.33 bits below $\hat{p}_{W_0}$, and 40,000 real end-to-end bootstrappings of DescPar-S with fresh keys fail 0.15–0.34 bits less often than predicted at their measured input margin; below about 2^{-16} the model is extrapolated. Replacing every window and selection tail by a Chernoff bound on the exact discrete distribution of the rounding terms, which keeps the independence assumptions but drops the Gaussian approximation, gives at most $2^{-72.1}$ per bootstrapping for DescPar-S and DescPar-G and $2^{-64.0}$ for all our members, dense ones included. With all standard deviations inflated by 1% and the worst of three keys minus a confidence correction, every selection event of every plan has model probability at most $2^{-68.0}$ and their sum at most $2^{-67.5}$; with $\hat{p}_{\mathrm{amp}} \le 2^{-68}$ the predicted failure probability stays below 2^{-66}. The full version lists every event.

4 Descent-PBS with Block-Sparse Keys

After Section 3, the online task of DescPar with a block one-hot secret is to perform $K+1$ rotations, the first at rank τ_0, each multiplying the accumulator by the combined keys of Section 2.6,

$$\mathrm{ACC} \leftarrow \mathrm{ACC} \cdot \prod_{j \le h} K_j\big(\mathbf{e}^{(i)}\big), \qquad K_j\big(\mathbf{e}^{(i)}\big) = \sum_{k \in B_j} X^{e_k^{(i)}} \cdot \mathsf{RGSW}(s_k), \qquad i = 0, \dots, K. \tag{7}$$

Formed per rotation, these keys read the whole bootstrapping key $K+1$ times, which dominates at $N = 2048$. We observe that the exponents of Descent-PBS have two properties that those of a generic rotation lack: the first-rotation exponents of DescPar are multiples of τ_0, and all exponents are public before the first rotation. Based on this observation, we propose DescPar-S, which uses $(\tau_0 + K)h$ external products and a single pass over the bootstrapping key. We first show that the first rotation shares one key per block (§4.1), then derive the one-pass evaluation with its correctness and cost (§4.2), extend it to general tables (§4.3), and finally remove a rounding bias that the small refinement exponents expose (§4.4).

4.1 Shared Scalar-Matrix Keys

Over $\mathcal{R}_M$, a block of keys combined for exponents $\mathbf{e} \in \mathbb{Z}_{2M}^{n'}$ is a GSW encryption of the matrix $\Phi(e_{k_j^*})$ of Section 2.4,

$$\mathcal{K}_j(\mathbf{e}) = \textstyle\sum_{k \in B_j} \Phi(e_k) \cdot \mathsf{RGSW}(s_k), \qquad \sum_k s_k \Phi(e_k) = \Phi(e_{k_j^*}).$$

In general $\Phi(e_{k_j^*})$ permutes the components, and our DescMod-S implementation forms and transforms $\mathcal{K}_j$ at length M, τ_0 times larger than a key over $\mathcal{R}_N$. The first division of DescPar, however, is by $2N$, so its exponents are $e_k = \tau_0 e'_k$ and $\Phi(\tau_0 e'_k) = X^{e'_k} I_{\tau_0}$.

Lemma 2 (Scalar matrix keys)**.** *If $\tau \mid e_k$ for all k, then*

$$\mathcal{K}_j(\mathbf{e}) = K_j(\mathbf{e}/\tau) \cdot I_\tau,$$

so the τ components of the first rotation of DescPar share one combined key per block, formed and transformed once over $\mathcal{R}_N$.

Consequently, the first rotation of DescPar-S costs $\tau_0 h$ external products EP and h key transforms of length N, against h products EP_M over $\mathcal{R}_M$ and h transforms of length M for DescMod-S. This is why, in our engine, the parallel member wins with sparse keys although the module member wins with dense ones.

4.2 Fusing the Key Formation into One Pass

Given the task (7), the evaluation is designed in three steps.

- **Step 1: Compute all digits first.** The divisions of Algorithm 1 act on the input alone.

  Lemma 3 (Public exponents)**.** *All exponent vectors $\mathbf{e}^{(0)}, \ldots, \mathbf{e}^{(K)}$ of DescPar and DescMod are computed by* HomDivRem *from the input, before the first rotation.*

- **Step 2: Form all keys in one pass.** Each key $\mathsf{RGSW}(s_k)$ is loaded once from main memory and shifted into the $K+1$ combined keys of its block,

  $$G_j^{(i)} \mathrel{+}= X^{e_k^{(i)}} \cdot \mathsf{RGSW}(s_k) \quad \text{for all } i \le K, \qquad k \in B_j,$$

  where a shifted addition is a two-segment vector addition with a sign flip; the blocks run in parallel.

- **Step 3: Share the first-rotation key.** By Lemma 2, the τ_0 components of the first rotation use the same $K_j^{(0)}$, and the refinements use the precomputed $K_j^{(i)}$.

Algorithm 2 carries out Steps 1–3, followed by the descent of Algorithm 1.
Consequently, the key pass is paid once instead of $K+1$ times, and the rotations are those of DescPar with the combined keys of [JLLS26]. Steps 1, 4 and 5 of Algorithm 2 are those of DescPar in Algorithm 1.

Theorem 2 (Correctness and cost of DescPar-S)**.** *Algorithm 2 outputs $\mathsf{LWE}_z(\frac{q}{t} f(m))$ whenever the events of Theorem 1 hold, with $V_{\mathrm{MS}} = (h+1)/12$ in Corollary 1. It uses*

$$\underbrace{\tau_0 h}_{\text{Step 3}} + \underbrace{Kh}_{\text{Step 4}} = (\tau_0 + K)\, h \text{ external products in } \mathcal{R}_N, \qquad \underbrace{(K+1)\, n'}_{\text{Step 2}} \text{ shifted additions,}$$

$(K+1)h$ key transforms, K repackings and one pass over the bootstrapping key.

Algorithm 2 DescPar-S: sparse Descent-PBS with one key pass

Require: $\mathsf{LWE}_z(\frac{q}{t}m)$, the output of a previous bootstrapping; negacyclic f on $\mathbb{Z}_t$ with $\tau_0 = \max(1, t/2N)$ and $r_0 = \max(1, 2N/t)$; keys $\mathsf{RGSW}(s_k)$, $k < n'$, in coefficient form; key switch $z \to \mathbf{s}$; $\mathsf{M2R}/\mathsf{TR}^*$ keys; plan $(\beta_i, T_i, \epsilon_i)_{i<K}$
Ensure: $\mathsf{LWE}_z(\frac{q}{t}f(m))$

// Step 0: test vectors, once per table
if $t > 2N$ **then** $\mathrm{TV}_\nu[p] \leftarrow \lfloor \frac{q}{t} f(p\tau_0 + \nu) \rceil$ for $\nu < \tau_0$, $p < N$ ▷ interleaved sub-tables, no redundancy
else $\mathrm{TV}_0[p] \leftarrow \lfloor \frac{q}{t} f(\lfloor (p + \lfloor r_0/2 \rfloor)/r_0 \rfloor) \rceil$ for $p < N$ ▷ $\tau_0 = 1$: r_0 centred copies
end if
// Step 1: digits
$\mathbf{c} \leftarrow \mathsf{KS}_{z\to\mathbf{s}}(\mathbf{c})$ ▷ to the block one-hot secret
$(\mathbf{e}^{(i)}, b^{(i)})_{i\le K} \leftarrow \mathsf{HomDivRem}(\mathbf{c};\, 2N,\ \tau_0\beta_0,\ \beta_1, \ldots, \beta_{K-1})$ ▷ Lemma 3
// Step 2: one key pass
for each block B_j, $j \le h$, **in parallel do**
  $G_j^{(i)} \leftarrow 0$ for $i \le K$
  for $k \in B_j$ **do** ▷ each key loaded once
    $G_j^{(i)} \leftarrow G_j^{(i)} + X^{e_k^{(i)}}\,\mathsf{RGSW}(s_k)$ for all $i \le K$ ▷ shifted additions of all 4ℓ polynomials
  end for
  $\widehat{K}_j^{(i)} \leftarrow \mathrm{FFT}(G_j^{(i)})$ for $i \le K$
end for
// Step 3: first rotation at rank τ_0 with shared keys
$\mathrm{ACC}_\nu \leftarrow X^{-b^{(0)}}(0, \mathrm{TV}_\nu)$ for $\nu < \tau_0$
for $j = 1, \ldots, h$ **do**
  $\mathrm{ACC}_\nu \leftarrow \widehat{K}_j^{(0)} \boxdot \mathrm{ACC}_\nu$ for $\nu < \tau_0$ **in parallel** ▷ Lemma 2: one key, τ_0 products
end for
$C \leftarrow \sum_\nu \iota(\mathrm{ACC}_\nu) Y^\nu \in \mathcal{R}_M^2$; $\lambda \leftarrow \rho \leftarrow 0$ ▷ one big-ring accumulator; copies of the target at $[\![\lambda, \rho]\!]$
// Step 4: descent and refinements in $\mathcal{R}_N$
Step 3 of Algorithm 1 on C, rotation $i+1$ by $\widehat{K}_j^{(i+1)} \boxdot C$, $j \le h$ ▷ precomputed keys
// Step 5: extraction
return $\mathsf{SampleExtract}(C, 0)$ ▷ selects $f(m)$ if the conditions of Theorem 1 hold

Proof (sketch). By Lemma 2, the τ_0 components of the first rotation in (7) share the key $K_j(\mathbf{e}^{(0)}/\tau_0)$ of each block, so this rotation costs $\tau_0 h$ external products over $\mathcal{R}_N$. By Lemma 3 all $\mathbf{e}^{(i)}$ are known before it, so Step 2 forms the $(K+1)h$ combined keys in one pass of $(K+1)n'$ shifted additions, and each of the K refinements costs h products. The rotations are those of DescPar with one key per block, so Theorem 1 applies with the modulus-switching variance of the $h+1$ rounded terms, $V_{\mathrm{MS}} = (h+1)/12$. The full proof is in the full version. □

4.3 Extending to General Tables

A general table $f\colon \mathbb{Z}_t \to \mathbb{Z}_t$ on an input without padding bit needs one more bit of the phase. As in Meta-WoPPBS [MHW+26, Alg. 3], we read the input over $\mathbb{Z}_{2q}$, so that its phase is $\frac{q}{t}m + e + \gamma q$ for an unknown bit γ, and the lifted division drives Algorithm 2 with $\tau_0' = 2\tau_0$ on two negacyclic tables over $\mathbb{Z}_{2t}$,

$$f^*(x) = f(x), \qquad g^*(x) = t/4 \qquad \text{for } x \in [0, t),$$

which return $(-1)^\gamma \frac{q}{t} f(m)$ and $(-1)^\gamma \frac{q}{4}$; a final rotation by the second cancels the sign of the first. Both tables share Steps 1 and 2 of Algorithm 2, and the sign table needs only one first accumulator.

Lemma 4 (One sign accumulator)**.** *The components* $\mathrm{TV}_r^g[p] = \frac{q}{t} g^*(p\tau_0' + r)$, $r < \tau_0'$, $p < N$, *are all equal to* $\frac{q}{t} \cdot \frac{t}{4}$, *since* $p\tau_0' + r < N\tau_0' = t$.

Algorithm 3 summarizes the resulting pipeline DescPar-G. Its sign cancellation uses $\beta_{\mathrm{cs}} = 61$, $\varepsilon_{\mathrm{cs}} = \lfloor N/\beta_{\mathrm{cs}} \rfloor - 1$ and a coarse key switch, $(2^4, 5)$ with noise 2^{-22} under the same one-hot

secret, and its single window event is checked like the others. Its rotation is driven by a ciphertext produced in Step 4, so its combined keys need a second, single-rotation pass over the bootstrapping key, which our timings include. Since our plans admit $\tau_0' \le 16$, i.e. general tables of 15 bits, a 16-bit table is split by the first level of the decomposition of Hwang et al. [HLMS25, Lem. 1] into a negacyclic table G_0, evaluated on $\mathbf{c}$ by Algorithm 2, and a table H_1 that depends only on $x \bmod t/2$, evaluated on $2\mathbf{c}$ by Algorithm 3 for $t/2$; both share one key pass, so one level suffices.

Proposition 3 (16-bit general tables)**.** *Let $\Delta = q/t$, $G_0(x) = \frac{1}{2}(F(x) - F(x + \frac{t}{2}))$ and $H_1(x) = \frac{1}{2}(F(x) + F(x + \frac{t}{2}))$. Encode the integer tables $2G_0$ and $2H_1$ at the unit $\Delta/2$. If both pipelines meet the conditions of Theorem 1 and the sign cancellation selects its window, their outputs encrypt $\frac{\Delta}{2} \cdot 2G_0(m) + E_G$ and $\frac{\Delta}{2} \cdot 2H_1(m) + E_H$ under z, and their sum encrypts*

$$\tfrac{\Delta}{2}\big(2G_0(m) + 2H_1(m)\big) + E_G + E_H = \Delta F(m) + E_G + E_H.$$

No parity condition on F is needed, since $2G_0$ and $2H_1$ are integers. On $2\mathbf{c}$ the phase, the noise and the unit all double, so the input margin of the H_1 pipeline is that of $\mathbf{c}$. The failure events are those of both pipelines and the window of the sign cancellation, 9 in total, all checked with Corollary 1 and listed in the full version; since the two pipelines share keys, E_G and E_H may be correlated, so $\hat{p}_{\text{amp}}$ uses the measured variance of the summed output.

Algorithm 3 DescPar-G: general LUTs with a shared key pass

Require: $\mathsf{LWE}_z(\frac{q}{t}m)$ without padding bit; $f\colon \mathbb{Z}_t \to \mathbb{Z}_t$; the keys of Algorithm 2; a coarse key switch $z \to \mathbf{s}$; plans for f^* and g^*
Ensure: $\mathsf{LWE}_z(\frac{q}{t}f(m))$

// **Step 1: lifted digits**
$\mathbf{c} \leftarrow \mathsf{KS}_{z\to\mathbf{s}}(\mathbf{c})$, read over $\mathbb{Z}_{2q}$; $(\mathbf{e}^{(i)}, b^{(i)})_{i\le K} \leftarrow \mathsf{HomDivRem}(\mathbf{c};\, 2N, \tau_0'\beta_0, \beta_1, \dots)$ ▷ $\tau_0' = 2\tau_0$
// **Step 2: one key pass shared by both tables**
$\widehat{K}_j^{(i)} \leftarrow \mathrm{FFT}\big(\sum_{k\in B_j} X^{e_k^{(i)}}\, \mathsf{RGSW}(s_k)\big)$ for $i \le K$, $j \le h$ ▷ Step 2 of Algorithm 2
// **Step 3: two descending pipelines**
$C_f \leftarrow$ Steps 3–4 of Algorithm 2 on f^* ▷ encrypts $(-1)^\gamma \frac{q}{t} f(m)$
$C_g \leftarrow$ Steps 3–4 of Algorithm 2 on g^* with one first accumulator ▷ Lemma 4; encrypts $(-1)^\gamma \frac{q}{4}$
// **Step 4: sign cancellation**
$c_g \leftarrow \mathsf{KS}_{z\to\mathbf{s}}\big(\mathsf{SampleExtract}(C_g, 0)\big)$ switched to $\mathbb{Z}_{2N}$ ▷ phase $N/2 + \gamma N$
$C \leftarrow \mathsf{BR}_{\mathcal{R}_N}\big(c_g,\ X^{N/2}\, \mathsf{TR}^*(C_f, [0,0], \beta_{\text{cs}}, \varepsilon_{\text{cs}})\big)$ ▷ multiplies by $(-1)^\gamma$
// **Step 5: extraction**
return $\mathsf{SampleExtract}(C, 0)$

4.4 Unbiased Gadget Rounding

The refinements in Step 4 rotate by small exponents, $|e| \le \lceil \beta_i/2 \rceil$. With half-up rounding before the gadget decomposition, as in TFHE-rs, they were up to four bits noisier in our engine than rotations with uniform exponents, like the bias that Ma et al. observed in Meta-PBS. We show that exact arithmetic has no such bias and remove it in double precision; for TFHE-go, whose library we did not modify, this is a consistent hypothesis.

Lemma 5 (Equivariance)**.** *If the rounding* rnd *to a multiple of $2^{64-\ell\log_2 B_g}$ and the gadget decomposition D are odd on the rounded coefficients of C, then for every monomial*

$$D(X^u C) = X^u D(C), \qquad (X^r K) \boxdot (X^u C) = X^{r+u}\,(K \boxdot C). \tag{8}$$

Hence public block offsets and the randomized blind rotation of [MHW+26] return the output of the plain rotation.

Since $q/2 = -q/2$ in $\mathbb{Z}_q$, no balanced decomposition is odd when its top digit is $\pm B_g/2$. Our ties-to-even decomposition breaks the other digit ties by the parity of the next digit and is odd on all remaining values, as a script confirms on random values and near all ties; on the excluded fraction of about $1/B_g$ of the coefficients, both sides of (8) decompose the same value and differ only in gadget error of the same distribution. In exact arithmetic, a zero-mean rounding error therefore gains no systematic part from small exponents. In double precision, the FFT outputs x are exact only on a grid, on which half-up rounding is not odd, and the constant error accumulates over the h steps as

$$c = \mathbb{E}\big[\mathrm{rnd}_{\uparrow}(x) - x\big] \approx +2^{\lfloor \log_2 |x| \rfloor - 53}, \qquad \underbrace{h\,|c|\,\|w\|/\sqrt{N}}_{\text{exponents near zero}} \quad \text{versus} \quad \underbrace{\sqrt{h}\,|c|\,\|w\|/\sqrt{N}}_{\text{uniform exponents}},$$

where w is the same vector at every step and c is computed for residues uniform on the grid of step $2^{\lfloor \log_2 |x| \rfloor - 52}$ modulo the rounding step, the ties all rounding up. Rounding *ties to even* sends, under the same uniformity, half of the ties down, so its mean error is zero; it removes the coherent bias at no measurable cost, whereas randomization only turns it into rotated copies. Measured on one rotation with $h = 72$, the bias costs 3.1–3.9 bits for the rounding-dominated gadgets, and ties to even come within 0.5 bits of exact arithmetic. Mean compensation [dRDV26] assumes $\mathbb{E}[u] = 0$, which ties to even restore; it lowers the dense rounding variance from $n/24 + 1/12$ to $n/48 + 1/12$ but the one-hot one only by $1 - 1/w$, so we apply it to the dense baselines. All our evaluators round ties to even.

5 Implementation and Evaluation

We evaluate Descent-PBS against Meta-PBS inside the module, against public implementations of prior large-table routes, and across its own design choices.

5.1 Experimental Setup

Platform. All experiments run on a shared Intel Xeon Gold 6230R host, 2×26 cores at 2.1 GHz, with Rust 1.91 and Go 1.25. Our engine reuses the FFT, key generation and LWE key switching of TFHE-rs [Zam22]; the key combination, the ties-to-even decomposition, the external products, M2R, TR* and the module rotations are ours. All runs of Tables 4 and 5 form one session of six rounds in random order, each with 8 bootstrappings in one worker thread pinned to one logical CPU; we report the median process CPU time over rounds, which includes the input key switch, the input-dependent key combination with its FFTs and the extraction, and excludes key generation and the FFT of the static keys. On an idle physical core with an idle sibling, the 16-bit DescPar-S takes 138 ms and DescPar-G 306 ms. Every speedup is the ratio of two medians shown in a table, and the rounds won count the per-round pairs. Every output is decrypted, and 61,212 bootstrappings ran without a wrong output, a functional check rather than a measurement of the tail.

Table 3: Security parameters, $q = 2^{64}$, one-hot dimension $n' = 2016$.

method	ref.	LWE secret	N	σ_{LWE}/q	σ_{GLWE}/q	λ_{LWE}, λ_{GLWE}	failure
DescPar-S, DescPar-G	ours	one-hot, $h = 72$	2048	2^{-28}	2^{-50}	129.9, 128.9	$< 2^{-66}$
DescPar-S, 131-bit	ours	one-hot, $h = 84$	2048	2^{-29}	2^{-49}	136.4, 131.4	$< 2^{-66}$
DescMod, Meta-Mod	ours	binary, $n = 1210$	2048	2^{-28}	2^{-50}	130.1, 128.9	$< 2^{-66}$
Meta-PBS	[MHW+26]	binary, $n = 1170$	2048	$2^{-27.44}$	$2^{-50.22}$	128.4, 128.3	$< 2^{-64}$
WoP-PBS, set #8	[BBB+23]	binary, $n = 549$	2×1024	$2^{-11.62}$	$2^{-51.49}$	127.0, 125.3	2^{-35}

Parameters and security. Table 3 lists the parameters of our evaluators and of the compared works. Security is estimated with the lattice estimator [APS15], commit `5ba00f5`,

in the classical MATZOV model; for the block one-hot secrets we also implemented a structured Meet-LWE [May21] and dual and primal hybrids that guess whole blocks, of which a structured dual meet-in-the-middle binds at 129.9 bits. Since this margin is thin, we also report DescPar-S with $h = 84$ blocks of $w = 24$ at 136.4 and 131.4 bits. The module secret $\iota(S)$ has the entropy of S, so GLWE security is that of $\mathcal{R}_N$. The binding estimate of the 128-bit set is the GLWE one, 128.9 bits. We claim IND-CPA under the circular security already assumed for bootstrapping and TR* keys; the M2R keys encrypt the same kind of messages, monomial combinations of coefficients of S, and we make no IND-CPA-D claim; the binding attack is derived in the full version. The dense evaluators use $n = 1210$: with our key-switching noise 2^{-28}, the $n = 1170$ of Meta-PBS would give only 125.9 bits, whereas with its own noise $2^{-27.44}$ it gives 128.4.

Parameter selection. Table 6 lists, for each width, the rotation gadget $(B_{\mathrm{br}}, \ell_{\mathrm{br}})$, the input key switch $(B_{\mathrm{ks}}, \ell_{\mathrm{ks}})$ and the number K of repackings; the 131-bit and mean-compensated ones are in the full version. The gadget is the cheapest one that meets the output target, and the plans $(\beta_i, T_i, \epsilon_i)$ are chosen by our planner for each secret, noise and input margin, measured on chains of 300 consecutive bootstrappings whose outputs are key-switched and fed back as inputs, with $\mu_{\mathrm{in}} = \log_2 \frac{q}{2t\sigma_{\mathrm{in}}}$ for the input standard deviation σ_{in}. With the worst of three key sets minus a one-sided 95% confidence correction and all standard deviations inflated by 1%, Corollary 1 bounds the model-predicted failure probability by $2^{-67.5}$ per negacyclic and $2^{-70.8}$ per general bootstrapping, and with an output margin of at least 3.39 bits every bootstrapping stays below 2^{-66}; random, identity, constant, threshold and alternating 16-bit tables keep margins of 3.81–4.65 bits. Plans, failure events and tolerances are in the full version.

Configurations. With the same keys (except at 131 bits), budget, key switch and rounding, we run Meta-PBS inside the module (Meta-Mod) with dense keys, with mean compensation [dRDV26], and with one-hot keys formed per rotation or in one pass (Meta-Mod-S); DescPar-S with keys formed per rotation; DescMod, DescMod-S, DescPar-S and DescPar-G.

5.2 End-to-End Results

Table 4 reports Descent-PBS on all widths and parameter sets, with time, output margin and persistent keys side by side.

Table 4: Performance of Descent-PBS, one thread: CPU time (ms), output margin (bits) and persistent keys (MB).

	128-bit, $h = 72$				131-bit, $h = 84$				dense, $n = 1210^*$	
	DescPar-S			DescPar-G	DescPar-S			DescPar-G	plain	MC
bits	ms	margin	keys	ms	ms	margin	keys	ms	ms	ms
8	36 (27%)	6.86	232	81	39	6.41	232	82	75	75
10	43 (24%)	4.86	270	98	64	10.16	400	159	184	139
12	88 (33%)	8.65	432	164	88	8.03	431	171	187	170
13	97 (31%)	7.66	466	179	95	6.83	465	177	221	208
14	111 (23%)	6.63	534	187	112	5.91	534	202	361	324
15	132 (30%)	5.61	704	269	135	4.87	704	327	544	529
16	196 (41%)	4.55	1,208	368	215	3.66	1,213	599	1,311	1,282

Percentages: spread of the six rounds, (max−min)/median. *Meta-PBS up to 12 bits, DescMod from 13 bits; MC: mean compensation.

Performance. DescPar-S evaluates negacyclic 13–16-bit tables in 97–196 ms and DescPar-G general ones in 179–368 ms on one thread. The 131-bit parameters cost 0.97–1.48× the time for negacyclic and 0.99–1.63× for general tables, through twelve more external

products per rotation and other plans; even so, DescPar-S at 131 bits is 1.49–3.04× faster than Meta-Mod-S at 128 bits.

Memory. The persistent keys grow from 466 to 1,208 MB, dominated at 16 bits by the uncompressed input key-switching key, which also sets the input margin. Each plan accepts inputs of deviation up to $q/(2t \cdot 2^{\mu_{\text{in}}})$, a key-switched bootstrapping output; at 16 bits this may grow by 16.5% (12.5% at 131 bits) before the prediction exceeds 2^{-64}. In the 16-bit DescPar-S, the key pass takes 43–45% of the time, the first rotation 20%, repackings and refinements 11%, key switch and extraction 24%: key processing, not the external-product count, dominates. **Sparse against dense.** With all steps of Section 4, DescPar-S is 2.05–6.68× faster than the dense evaluator of the same width and 1.92–6.53× faster than the mean-compensated one.

5.3 Comparison with Meta-PBS inside the Module

Each technique of Sections 3 and 4 is a separate step of the pipeline, so we isolate it by switching it off. Table 5 reports the configurations at 13–16 bits, where Meta-PBS alone no longer applies, against Meta-PBS inside the module with the same keys.

Table 5: CPU time (ms) at 13–16 bits; speedup over Meta-PBS inside the module in parentheses; rounds won: rounds, over the four widths, in which the row is faster than its baseline.

method	$t = 2^{13}$	$t = 2^{14}$	$t = 2^{15}$	$t = 2^{16}$	rounds won
dense secret, $n = 1210$					
Meta-Mod	319	573	1,136	3,142	baseline
DescMod (ours)	221 (1.44×)	361 (1.59×)	544 (2.09×)	1,311 (2.40×)	24/24
dense secret with mean compensation					
Meta-Mod	296	522	1,534	2,478	baseline
DescMod (ours)	208 (1.42×)	324 (1.61×)	529 (2.90×)	1,282 (1.93×)	24/24
block one-hot secret, $h = 72$					
Meta-Mod-S	147	167	307	652	baseline
Meta-Mod-S, one key pass	143 (1.03×)	181 (0.93×)	335 (0.92×)	660 (0.99×)	10/24
DescPar-S, keys per rotation	115 (1.27×)	128 (1.30×)	151 (2.04×)	241 (2.71×)	24/24
DescMod-S (ours)	120 (1.22×)	139 (1.21×)	195 (1.58×)	333 (1.96×)	24/24
DescPar-S (ours)	97 (1.51×)	111 (1.51×)	132 (2.32×)	196 (3.32×)	24/24
DescPar-S, 131-bit (ours)	95 (1.55×)	112 (1.49×)	135 (2.27×)	215 (3.04×)	24/24

Descending. With dense keys, DescMod is 1.44–2.40× faster than Meta-Mod, and 1.42–2.90× when both use mean compensation; the gap widens with the width, as $(K' + 1)\tau_0/(\tau_0 + K) \to K' + 1$.

Shared keys. With one-hot keys, DescPar-S is 1.51–3.32× faster than Meta-Mod-S, in two measurable steps: the descent alone, DescMod-S against Meta-Mod-S, gives 1.21–1.96×, and the shared scalar keys, DescPar-S against DescMod-S, whose first-rotation keys are big-ring keys, a further 1.23–1.70×. At 16 bits the external-product counts differ only by $2304/1368 \approx 1.68$, so key processing accounts for the rest; a single key pass alone does not help Meta-Mod-S, whose keys are τ_0 times larger. Against Meta-Mod-S re-planned for the same 131-bit parameters, DescPar-S at 131 bits is 1.37–3.57× faster in an isolated session (full version).

Single key pass. Forming the keys per rotation costs DescPar-S a factor 1.12–1.31. Re-measured on an idle physical core, every ratio of Table 5 stays within 17% of this session, CPU and wall time agree, every row wins all 24 rounds, and this factor is 1.26–1.38 (full version).

Table 6: Plaintext moduli 2^8–2^{16} on one thread: parameters, CPU time, speedup over EBS, output margin and persistent keys.

t, $(B_{\mathrm{ks}}, \ell_{\mathrm{ks}})$	method	rank	$(B_{\mathrm{br}}, \ell_{\mathrm{br}})$	K	time (ms)	speedup	margin (bits)	keys (MB)
2^8	EBS	2^4	$(2^{23}, 1)$	0	422	1.0×	5.93	139
$(2^6, 3)$	Meta-PBS	1	$(2^{23}, 1)$	1	75	5.6×	5.35	142
	DescPar-S	1	$(2^{23}, 1)$	1	36	11.6×	6.86	232
2^{10}	EBS	2^6	$(2^{17}, 2)$	0	2,843	1.0×	10.08	238
$(2^5, 4)$	Meta-PBS	1	$(2^{17}, 2)$	3	184	15.4×	9.17	241
	DescPar-S	1	$(2^{23}, 1)$	1	43	65.7×	4.86	270
2^{12}	EBS	2^8	$(2^{17}, 2)$	0	11,953	1.0×	8.08	258
$(2^5, 5)$	Meta-PBS	1	$(2^{17}, 2)$	3	187	64.0×	7.09	261
	DescPar-S	1	$(2^{17}, 2)$	2	88	136×	8.65	432
2^{13}	EBS	2^9	$(2^{17}, 2)$	0	25,719	1.0×	7.08	278
$(2^4, 6)$	DescMod	$2^1 \to 1$	$(2^{17}, 2)$	2	221	116×	6.49	305
	DescPar-S	$2^1 \to 1$	$(2^{17}, 2)$	2	97	264×	7.66	466
2^{14}	EBS	2^{10}	$(2^{17}, 2)$	0	53,041	1.0×	6.08	317
$(2^3, 8)$	DescMod	$2^2 \to 1$	$(2^{17}, 2)$	3	361	147×	5.19	322
	DescPar-S	$2^2 \to 1$	$(2^{17}, 2)$	2	111	479×	6.63	534
2^{15}	EBS	2^{11}	$(2^{17}, 2)$	0	107,504	1.0×	5.08	417
$(2^2, 13)$	DescMod	$2^3 \to 1$	$(2^{17}, 2)$	3	544	198×	4.11	424
	DescPar-S	$2^3 \to 1$	$(2^{17}, 2)$	2	132	813×	5.61	704
2^{16}	EBS	2^{12}	$(2^{15}, 3)$	0	253,975	1.0×	5.93	793
$(2^1, 28)$	DescMod	$2^4 \to 1$	$(2^{15}, 3)$	3	1,311	194×	5.03	806
	DescPar-S	$2^4 \to 1$	$(2^{17}, 2)$	3	196	1,294×	4.55	1,208

Dense secrets: binary, $n = 1210$; DescPar-S: one-hot, $h = 72$. Repacking gadget $(2^{11}, 3)$. EBS: blind rotation only, without key switch and extraction, so its speedups are lower bounds; three rounds.

5.4 Comparison with Prior Work

We compare against the public implementations of the prior routes to large tables: Meta-PBS [MHW+26] in TFHE-go, commit `5df5940`;[1] BootMMPM [DLLM25], commit `541bdb9f7`, which returns wrong outputs because of an inconsistent sign convention, so that we measure its rotation in our engine, with a minimal reproduction and the patch in the full version;[2] WoP-PBS [BBB+23] and the LUT decomposition [BBR26] from the artifact of [BBR26], commit `088029c`;[3] and Refined LHE [WHS+25].[4]

Comparison across plaintext moduli. Table 6 scans $t = 2^8$–2^{16} against extended bootstrapping (EBS) [LY23], the one-shot rotation of module bootstrappings, which we run as the corrected BootMMPM rotation of our engine with the dense secret, the rotation gadget of our dense evaluator and, at each width, the least rank $\tau = t/2^4$ that keeps the failure probability below 2^{-64} at the input margins of our dense chains. Paying $\rho\tau_0 = \tau$ external products per key in one rotation, EBS grows linearly with t, to 254 s at 16 bits. Meta-PBS is 5.6–64.0× faster up to $t = 2N$; beyond, DescMod is 116–198× faster with the same secret and similar keys, and DescPar-S, which adds the sparse secret, 264–1,294× with 1.5–1.7× the keys; only the DescMod rows isolate the schedule.

Comparison with Meta-PBS at $t \le 2N$. Against the public code of Meta-PBS with its own parameters ($n = 1170$), interleaved on one core in three rounds, our engine with the same parameters and plans is 1.3–1.8× faster, which estimates the engine's share, and DescPar-S at our parameters 3.2–5.9×, the rest coming from $h = 72$ instead of $n = 1170$ external products per rotation (per-width times in the full version).

[1] https://github.com/msh086/META-FBS
[2] https://github.com/GoneKuei/tfhe-rs-mpm
[3] https://github.com/CryptoExperts/artifact-HLUT-TCHES
[4] https://github.com/KAIST-CryptLab/refined-tfhe-lhe

Comparison with decomposed inputs. Table 7 runs the external routes for general 13–16-bit tables on the same host. WoP-PBS with one block, run in its authors' configuration (set #8, failure 2^{-35}, 125.3 bits in our estimate) rather than re-tuned to our budget, has our interface and is 3.8–6.6× slower than DescPar-G, which needs about twice its peak memory (peak RSS, which unlike the persistent keys of Table 4 includes working buffers). The LUT decomposition of [BBR26] needs only 150 MB but is 32–98× slower on its native digits; its paper and artifact evaluate up to 14 bits. Refined LHE, measured in a separate session, takes 1.8–3.2× the time of DescPar-G, an indicative ratio. At 131 bits DescPar-G remains 2.4–6.5× faster than WoP-PBS, and when f is negacyclic, DescPar-S needs no sign cancellation and is 7.2–11.8× faster.

Table 7: Arbitrary 13–16-bit LUTs $f\colon \mathbb{Z}_t \to \mathbb{Z}_t$ against external routes, one thread: time (ms), our speedup over WoP-PBS in parentheses, and peak memory (MB). One-LWE routes take one ciphertext without padding bit and return one; digit routes take the input already split into digits. DescPar-S applies when f is negacyclic.

method	ref.	input	$t = 2^{13}$	$t = 2^{14}$	$t = 2^{15}$	$t = 2^{16}$	RSS
WoP-PBS, one block	[BBB+23]	one LWE	1,148	1,231	1,311	1,408	823
LUT decomposition, 2^{-40}	[BBR26]	digits	5,767	6,159	–	–	150
LUT decomposition, 2^{-128}	[BBR26]	digits	17,435	18,203	–	–	150
Refined LHE*	[WHS+25]	digits	566	554	662	658	n/m
DescPar-G, 128 bits	ours	one LWE	179 (6.4×)	187 (6.6×)	269 (4.9×)	368 (3.8×)	1,675
DescPar-G, 131 bits	ours	one LWE	177 (6.5×)	202 (6.1×)	327 (4.0×)	599 (2.4×)	1,997
DescPar-S, negacyclic f	ours	one LWE	97 (11.8×)	111 (11.1×)	132 (9.9×)	196 (7.2×)	1,447

*Separate session, indicative. –: not evaluated in the paper or artifact. n/m: not measured.

Relationship to other approaches. Meta-PBS remains the method of choice for $t \le 2N$, where Descent-PBS reduces to it. Meta-PBS in a larger ring $N' = \tau_0 N$, which we did not implement, pays $(K'+1)\,n_{\mathrm{k}}$ products of length $\tau_0 N$, each at least τ_0 times one of length N, i.e. at least the cost of Meta-Mod. BFV-based functional bootstrapping [ZLW26] handles one LWE ciphertext outside our external-product model; we did not measure it. Decomposition-based methods remain preferable for digit inputs, multivariate functions or tight memory.

6 Conclusion

We presented Descent-PBS, a single-ciphertext PBS beyond the ring dimension that pays the module rank once: under a subring secret, one public functional key switch repacks the module accumulator into the ring, where the refinements of Meta-PBS run unchanged, and block one-hot secrets add shared keys and one key pass. At 128 bits it evaluates 16-bit tables in 196 ms, negacyclic, and 368 ms, general, on one thread. The 1,208 MB of keys at 16 bits and the model-predicted failure probabilities are the main limitations; seeded keys and hardware implementations remain future work.

References

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

[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(3):28, 2023.

[BBR26] Sonia Belaïd, Nicolas Bon, and Matthieu Rivain. Decomposition of large look-up tables for fast homomorphic evaluation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2026(3):247–278, 2026.

[BORLT25] Loris Bergerat, Jean-Baptiste Orfila, Adeline Roux-Langlois, and Samuel Tap. Accelerating TFHE with sorted bootstrapping techniques. ASIACRYPT 2025; Cryptology ePrint Archive, Paper 2025/2214, 2025.

[CDKS21] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. In *Applied Cryptography and Network Security, ACNS 2021*, volume 12726 of *LNCS*, pages 460–479. Springer, 2021.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 3–33. Springer, 2016.

[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.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *CSCML 2021*, volume 12716 of *LNCS*, pages 1–19. Springer, 2021.

[CLOT21] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In *ASIACRYPT 2021*, volume 13092 of *LNCS*, pages 670–699. Springer, 2021.

[DLLM25] Kuiyuan Duan, Hongbo Li, Dengfa Liu, and Guangsheng Ma. Large-plaintext functional bootstrapping in FHE with small bootstrapping keys. In *TCC 2025*, volume 16268 of *LNCS*, pages 253–279. Springer, 2025.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 617–640. Springer, 2015.

[dRDV26] Thomas de Ruijter, Jan-Pieter D'Anvers, and Ingrid Verbauwhede. Don't be mean: Reducing approximation noise in TFHE through mean compensation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2026(1):82–104, 2026.

[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.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, pages 169–178. ACM, 2009.

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN 2012*, volume 7485 of *LNCS*, pages 19–37. Springer, 2012.

[HLMS25] Intak Hwang, Shinwon Lee, Seonhong Min, and Yongsoo Song. Efficient full domain functional bootstrapping from recursive LUT decomposition. In *Selected Areas in Cryptography – SAC 2025*, 2025. Cryptology ePrint Archive, Paper 2025/1255.

[JLLS26] Aayush Jain, Huijia Lin, Zeyu Liu, and Sagnik Saha. New techniques for fast and shallow FHE bootstrapping and beyond. CRYPTO 2026; Cryptology ePrint Archive, Paper 2026/1730, 2026.

[LMP22] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In *ASIACRYPT 2022*, volume 13792 of *LNCS*, pages 130–160. Springer, 2022.

[LMSS23] Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Faster TFHE bootstrapping with block binary keys. In *ASIA CCS 2023*, pages 2–13. ACM, 2023.

[LY23] Kang Hoon Lee and Ji Won Yoon. Discretization error reduction for high precision torus fully homomorphic encryption. In *PKC 2023*, volume 13941 of *LNCS*, pages 33–62. Springer, 2023.

[May21] Alexander May. How to meet ternary LWE keys. In *CRYPTO 2021*, volume 12826 of *LNCS*, pages 701–731. Springer, 2021.

[MHW+26] Shihe Ma, Tairong Huang, Anyu Wang, Changtong Xu, Tao Wei, and Xiaoyun Wang. Meta-PBS: Compact high-precision programmable bootstrapping. ACM CCS 2026; Cryptology ePrint Archive, Paper 2025/2284, revised 2026-08-12, 2026.

[WBL+25] Ruida Wang, Jikang Bai, Yijian Liu, Xinxuan Zhang, Xianhui Lu, Lutan Zhao, Kunpeng Wang, and Rui Hou. Bootstrapping over free R-module. Cryptology ePrint Archive, Paper 2025/1753, 2025.

[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 2025*, pages 2309–2323. ACM, 2025.

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

[ZLW26] Xiaopeng Zheng, Hongbo Li, and Dingkang Wang. Functional bootstrapping for a single LWE ciphertext with $\tilde{O}(1)$ polynomial multiplications. ASIACRYPT 2026; Cryptology ePrint Archive, Paper 2026/975, 2026.