Concrete Quantum Circuit Implementations and Resource Estimates for the Quantum 3 and 4-Sieves

MohamadAli Khajeian

School of Engineering Sciences, College of Engineering, University of Tehran

khajeian@ut.ac.ir

November 15, 2026

Abstract

Chailloux and Loyer (2023) introduced quantum 3- and 4-sieve algorithms for the Shortest Vector Problem (SVP) with asymptotic analyses, but concrete circuit implementations remain unexplored. We present the first complete reversible circuits for both algorithms, designing inner-product predicates with signed symmetric squaring across three architectures (low-qubit, balanced, and low-depth). We validate these circuits using exact sparse state-vector simulations that match an independent mathematical model to a precision of 10^{-13}. Crucially, we show that the original amplitude-amplification oracle fails when inner Grover searches are approximate and correct it to test the full configuration. Applying our arithmetic techniques to the published NV 2-sieve oracle reduces its T-count by $9.4-14\times$ and T-depth by over three orders of magnitude. Finally, we provide exact per-call logical resource estimates for the configuration predicates and the NV oracle at cryptographic dimensions, detailing the architectural trade-offs for the core predicates used in quantum tuple sieving.

Contents

1 Introduction **3**

2 Preliminaries **4**
2.1 Lattices, sieving and configurations . . . 4
2.2 Gates, logical AND and cost metrics . . . 4
2.3 Quantum search and QRACM . . . 4

3 Reconstruction of the Chailloux–Loyer **5**
3.1 k-random product codes and tuple filters . . . 5
3.2 Framework and quantum subroutines . . . 5
3.3 Numerical reproduction . . . 5
3.4 QRACM-aware asymptotics . . . 6

4 Reproduction and validation of Kim et al. **6**
4.1 Methodology . . . 6
4.2 Results of the reproduction . . . 6
4.3 Reconstruction of the NV oracle . . . 7
4.4 Discrepancies . . . 7

5 Concrete quantum architectures **8**
5.1 What must be computed . . . 8
5.2 Partial products with folded constants . . . 8
5.3 Carry-save reduction and sign extraction . . . 9
5.4 Three predicate architectures . . . 9
5.5 QRACM . . . 10
5.6 Grover iterations, $\mathcal{A}$ and amplitude amplification . . . 10
5.7 An optimised NV oracle . . . 11

6 Implementation and experimental methodology **11**
6.1 Framework . . . 11
6.2 Validation . . . 12

7 Logical resource estimates **13**
7.1 Oracle level (generated circuits) 13

8 Limitations **13**

9 Conclusion **14**

1 Introduction

The security of ML-KEM, ML-DSA and of the KpqC finalists SMAUG-T and HAETAE is assessed through the cost of solving the Shortest Vector Problem (SVP) in a projected sublattice of dimension β (the *core-SVP* methodology). The fastest practical SVP algorithms are lattice sieves, whose main drawback is memory: a 2-sieve needs $2^{0.2075d+o(d)}$ vectors. *Tuple* (or k-) sieves trade time for memory by combining $k > 2$ vectors at once [17, 18, 19, 20]. Chailloux and Loyer [1] recently introduced a filtering framework tailored to k-sieving—k-Random Product Codes whose codewords sum to zero—and showed that a quantum 3-sieve reaches time $2^{0.3098d}$ in the minimal memory $2^{0.1887d}$, and a quantum 4-sieve gives new time–memory trade-offs down to memory $2^{0.1724d}$. These are asymptotic statements in the QRACM model; no concrete circuit exists.

At the other end of the spectrum, Kim et al. [2] gave a concrete, Clifford+T-level implementation of the Grover-assisted Nguyen–Vidick (NV) 2-sieve oracle, together with core-SVP and surface-code estimates for twelve standardised parameter sets. Their arithmetic (Draper carry-lookahead adders [4], a tree-based quantum carry-save adder [3] and logical-AND gates [6]) is the most detailed published architecture for a quantum sieve oracle.

We searched the literature for concrete circuits of tuple sieves. Prior concrete work covers 2-sieves only: Albrecht et al. [13] (search subroutines of NV/bgj1-type sieves), Doriguello et al. [12] (NV/Gauss sieves with QRAM and QEC), Prokop et al. [14] (an SVP solution-test oracle), Cho et al. [15] and Kim et al. [2] (NV oracle). Quantum tuple sieving has been studied asymptotically [20, 1, 16]; the recent algorithm of Engelberts et al. [16] improves the 3-sieve exponent to 0.2846 with two-level amplitude amplification but gives no circuits. To the best of our knowledge the circuits and estimates below are the first for any quantum k-sieve with $k > 2$.

This work turns the quantum 3-sieve (Algorithm 6) and 4-sieve (Algorithm 7) of [1] into reversible circuits that are *generated, simulated and counted*, and derives per-call logical resource estimates at cryptographic dimensions. Before building on [2] we reproduce it, and before building on [1] we re-derive it.

Our Contributions. We label every result as **[R]** (reproduction of prior work), **[N]** (new, implemented and validated).

1. **[R] Reproduction of Kim et al.** We re-implement their iteration model, QROM cost model and surface-code procedure and recover their Tables 7–10. The arithmetic-only rows agree to 0.01 bits; the T-count, T-depth and width of the QROM rows agree exactly. We also reconstruct their NV oracle circuit and identify five discrepancies (Section 4), notably that their oracle cost omits the uncomputation of the accumulator (a factor ≈ 2), that the reported widths grow linearly in d although ancillae are claimed to be reused, and that the squaring does not exploit symmetry.

2. **[R] Reconstruction of Chailloux–Loyer.** We re-derive the quantum 3-/4-sieve analysis, reproduce Propositions 13, 15, 17 and Tables 4–5 to four decimals, and correct five typographical errors (a sign in Lemma 1, two exponents, the closed form of Proposition 16, and the angle of Proposition 18).

3. **[N] Configuration-test circuits for two quantum vectors.** Unlike the NV oracle, both tuple members are in superposition. We design an inner-product predicate $[\langle \vec{x}_i | \vec{x}_j \rangle \leq \tau]$ using (i) Baugh–Wooley signed partial products or (ii) a polarisation identity with signed symmetric squaring that needs $s + \binom{s+1}{2}$ instead of s^2 ANDs per coordinate; all constants are folded into the threshold, every Toffoli is a 4-T temporary AND, and the sign is extracted with a log-depth parallel-prefix tree. Three architectures—low-qubit (LQ), balanced (BAL), low-depth (LD)—span T-depth 84 to $7.8 \cdot 10^4$ and width $1.2 \cdot 10^4$ to $1.0 \cdot 10^5$ at $d = 375$.

4. **[N] Complete quantum 3-sieve and 4-sieve circuits.** QROM (unary iteration, measurement-based unlookup, select–swap), Grover iterations, the nested operator $\mathcal{A}$ and amplitude amplification are generated as circuits and validated by exact sparse state-vector simulation against an independent model (agreement to 10^{-13}). We show that the amplitude-amplification oracle of Algorithms 6/7, which tests only the last pair, fails when the inner Grover searches are approximate and must test the full configuration (success probability $0.015 \to 0.999$ on a toy instance).

5. **[N] An optimised NV oracle.** Applying the same techniques to the NV oracle of [2] lowers its T-count by 9.4–14× and its T-depth by $1.2 \cdot 10^3$–$3.4 \cdot 10^3$× (with uncomputation included), at a higher width ($T_d \cdot M$ improves 63–316×); a low-qubit variant is better than the published oracle in every metric (4.6–6.4× fewer T gates).

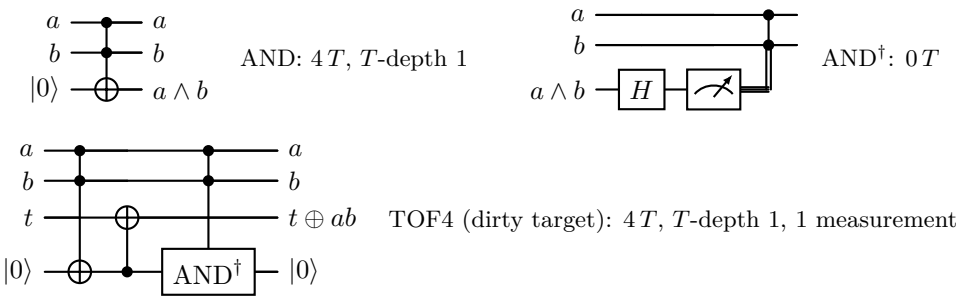

Figure 1: Primitive gates as implemented: temporary logical AND, its measurement-based inverse, and the dirty-target Toffoli TOF4 used inside carry-lookahead adders and select–swap networks.

2 Preliminaries

2.1 Lattices, sieving and configurations

A lattice $\mathcal{L}(B) = \{\sum_i z_i \vec{b}_i : z_i \in \mathbb{Z}\}$ is generated by a basis $B = (\vec{b}_1, \ldots, \vec{b}_d)$; SVP asks for a shortest non-zero $\vec{v} \in \mathcal{L}(B)$. Under the usual heuristic, sieve vectors behave like i.i.d. uniform points of the unit sphere $\mathcal{S}^{d-1}$ after normalisation. A k-sieve step takes a list L of vectors of norm $\leq R$ and outputs $|L|$ vectors of norm $\leq \gamma R$ of the form $\vec{x}_1 + \cdots + \vec{x}_k$. Since $\|\sum_i \vec{x}_i\|^2 = \sum_i \|\vec{x}_i\|^2 + 2\sum_{i<j}\langle \vec{x}_i | \vec{x}_j\rangle$, reducibility is implied by pairwise constraints $\langle \vec{x}_i | \vec{x}_j\rangle \leq C_{ij}$, where $C \in \mathbb{R}^{k\times k}$ (unit diagonal) is a *configuration*; with $\gamma = 1$ the configuration must satisfy the reduction constraint

$$k + 2\sum_{i<j} C_{ij} \leq 1. \tag{1}$$

For uniform points, $\Pr[\text{config } C] = \det(C)^{d/2+o(d)}$ [18], hence the list size $|L| = \det(C)^{-d/(2(k-1))}$ (balanced $C_{ij} = -1/k$ gives $2^{0.1887d}$ for $k = 3$ and $2^{0.1724d}$ for $k = 4$). For $I \subseteq [k]$ we write $C[I]$ for the principal submatrix, and the conditional list $L_j(\vec{x}_1, \ldots, \vec{x}_i) = \{\vec{x}_j \in L_j : \langle \vec{x}_\iota | \vec{x}_j\rangle \leq C_{\iota j}\ \forall \iota \leq i\}$ has expected size $|L_j|\,(\det C[1..i, j]/\det C[1..i])^{d/2}$ [20]. The normalised volume of a spherical cap of angle α is $V(\alpha) = \mathrm{poly}(d)\sin^d\alpha$ [21].

2.2 Gates, logical AND and cost metrics

We work in the Clifford+T model with the gate set $\{X, \mathrm{CNOT}, \mathrm{CZ}, H, S, T\}$, measurements and classically controlled Cliffords. The *temporary logical AND* of Jaques et al. [6] (after Gidney [5]) computes $a \wedge b$ into a fresh qubit with 4 T gates, 11 Cliffords, T-depth 1, depth 8 and one transient ancilla; its inverse AND† measures the target in the X basis and applies a classically controlled CZ (no T gate). A Toffoli on a *dirty* target is realised as AND–CNOT–AND† through a fresh ancilla ("TOF4", 4 T, T-depth 1), never with the 7-T circuit. Figure 1 shows the three primitives exactly as implemented. We report T-count ($\#T$), T-depth (T_d), full depth (FD), Clifford/total gate count (G) and width (M, logical qubits including transient ancillae); depths are computed by as-soon-as-possible scheduling of the decomposed gate costs on per-qubit timelines. Trade-off metrics $T_d{\cdot}M$, FD$\cdot M$, $G{\cdot}$FD follow [2, 6].

2.3 Quantum search and QRACM

Grover's algorithm finds one of t marked items among N with $\lfloor \frac{\pi}{4\theta} \rfloor$ iterations, $\sin^2\theta = t/N$; amplitude amplification (AA) boosts an algorithm $\mathcal{A}$ with success probability p to ≈ 1 with $\lfloor \pi/(4\arcsin\sqrt{p}) \rfloor$ iterates $\mathcal{A}S_0\mathcal{A}^\dagger S_\chi$ [23]. Quantum sieves access classical lists in superposition through QRACM, $|i\rangle|0\rangle \mapsto |i\rangle|L[i]\rangle$. In the circuit model this is realised by a QROM: unary iteration [7] costs $4(N-2)$ T gates and T-depth $N-2$ with $O(\log N)$ ancillae; select–swap [8] trades $\lambda\ell$ qubits for $4N/\lambda + 4\ell(\lambda-1)$ T gates; bucket-brigade QRAM [11, 12] has reaction depth $O(\log N)$ but $\Theta(N)$ qubits; Mukhopadhyay's depth-efficient QRAM [10] is used by [2]. A lookup is erased either by repeating it or, much more cheaply, by *measurement-based unlookup*: measure the data register in the X basis and correct the phases $(-1)^{\vec{m}\cdot L[i]}$ with an $O(\sqrt{N})$-T circuit [9].

3 Reconstruction of the Chailloux–Loyer

3.1 k-random product codes and tuple filters

Let $d = mb$ with $m = O(\mathrm{polylog}\, d)$. A *$k$-RPC* is $\mathcal{C} = Q \cdot (\mathcal{C}_1 \times \cdots \times \mathcal{C}_m)$ with Q a random rotation and each block code $\mathcal{C}_i \subset \sqrt{1/m}\,\mathcal{S}^{b-1}$ made of B/k groups $(A_1^j, \ldots, A_k^j)$ with $\langle A_i^j | A_{i'}^j \rangle = -\frac{1}{(k-1)m}$ and $\sum_i A_i^j = \vec{0}$; the product structure keeps list decoding efficient [21], and the codewords of a full codeword group satisfy $\langle A_i | A_{i'} \rangle = -1/(k-1)$, $\sum_i A_i = \vec{0}$. Each list vector is decoded to its nearest codeword A_i^j (angle $\approx \alpha$) and stored in list L_i^j, together with its residual vector $\vec{y} = (\vec{x} - \cos\alpha\, A_i^j)/\sin\alpha$. There are $\mathrm{NbFilters} = 1/V(\alpha)$ tuple-filters, each holding k lists of size $|L_1| = |L|V(\alpha)$.

Residual configuration (Lemma 1, corrected). With $\langle \vec{x}_i | \vec{x}_j \rangle = \cos^2\alpha \langle A_i | A_j \rangle + \sin^2\alpha \langle \vec{y}_i | \vec{y}_j \rangle$ and $\langle A_i | A_j \rangle = -1/(k-1)$, a configuration C on $\vec{x}$ corresponds to

$$C'_{ij}(\alpha) = \frac{1}{\sin^2\alpha}\Big(C_{ij} + \frac{\cos^2\alpha}{k-1}\Big), \tag{2}$$

without the leading minus sign printed in [1, Lemma 1] (with it, C' would be positive; the paper's own value $C' \approx -0.31$ for $C = -1/3$, $\alpha = 1.2343$ is obtained from (2)).

3.2 Framework and quantum subroutines

Algorithm 3 of [1] repeats NbRep times: sample a k-RPC, prefilter L, and solve the residual configuration problem in each filter with FAS (FindAllSolutionsWithinFilter). With

$$\mathrm{NbRep} = \max\Big\{1, \frac{\det(C)^{d/2}}{V(\alpha)^{k-1}\det(C')^{d/2}}\Big\}, \qquad T = \mathrm{NbRep}\big(|L| + V(\alpha)^{-1}\, T(\mathrm{FAS})\big). \tag{3}$$

Quantum 3-sieve (Algorithm 6). Prepare $|\psi_{L_1}\rangle$; Grover on register 2 for $\vec{y}_2 \in L_2(\vec{y}_1)$, $T_2 = (1 - C'^2_{12})^{-d/4}$ iterations (the printed $(1-C_{12})^{-d/4}$ lacks the square); Grover on register 3 for $L_3(\vec{y}_1)$; amplitude amplification of the resulting operator $\mathcal{A}$ with $T_{AA} = \sqrt{|L_1|/|\mathrm{Sol}|}\,\sqrt{|L_2(\vec{y}_1)||L_3(\vec{y}_1)|}$; repeat for every solution. Hence

$$T(\mathrm{FAS}_3) = \sqrt{|L_1|^3|\mathrm{Sol}|}\,\big[(1 - C'^2_{12})^{d/4} + (1 - C'^2_{13})^{d/4}\big], \tag{4}$$

with exponent $d/4$ (the simplified form in [1, Prop. 14] prints $d/2$, which does not reproduce the paper's 0.1210).

Quantum 4-sieve (Algorithm 7). Grover on register 2 ($L_2(\vec{y}_1)$), on register 3 ($L_3(\vec{y}_1, \vec{y}_2)$, two constraints), on register 4 ($L_4(\vec{y}_1, \vec{y}_2)$, two constraints), then AA on the last constraint. Writing $D_I = \det C'[I]$, Lemma 9 of [1] gives

$$T(\mathrm{FAS}_4) = |L_1|^2\sqrt{|\mathrm{Sol}|}\Big[\Big(\tfrac{D_{123}}{D_{12}}\Big)^{d/4}\Big(\tfrac{D_{124}}{D_{12}}\Big)^{d/4} + D_{12}^{d/4}\Big(\tfrac{D_{124}}{D_{12}}\Big)^{d/4} + D_{12}^{d/4}\Big(\tfrac{D_{123}}{D_{12}}\Big)^{d/4}\Big], \tag{5}$$

which replaces the closed form of [1, Prop. 16] (the printed form yields 0.302 instead of the paper's own 0.1722).

3.3 Numerical reproduction

Table 1 recomputes every number of Propositions 13–18 and Table 2 re-optimises (C, α) under the memory constraint and (1) (Nelder–Mead, 12 restarts). Propositions 15 and 17 and Table 4 are reproduced to ± 0.0004. For Proposition 18 the printed configuration with the printed $\alpha = 1.313$ gives memory 0.1833 and time 0.3172; our optimiser recovers the paper's time 0.3120 at memory 0.1813 with essentially the printed configuration and $\alpha = 1.2951$, whose $1/V(\alpha) = 2^{0.0555d}$ matches the paper's stated 0.0554. The angle 1.313 is thus a typo (copied from Proposition 17). At $m = 0.1724$ the 4-sieve curve is very steep: an unbalanced configuration within $3 \cdot 10^{-5}$ of the memory bound already gives 0.3260 instead of 0.3276.

Table 1: Reproduction of the propositions of [1] (exponents per d).

	NbRep		$1/V(\alpha)$		T(FAS)		T	
	paper	ours	paper	ours	paper	ours	paper	ours
Prop. 15 (3-sieve, m=0.1887)	0.1055	0.1055	0.0832	0.0833	0.121	0.1210	0.3098	0.3098
Prop. 17 (4-sieve, m=0.1724)	0.1069	0.1069	0.0484	0.0484	0.1722	0.1722	0.3276	0.3276
Prop. 18 (4-sieve, printed α=1.313)	0.1254	0.1097	0.0554	0.0485	0.1312	0.1591	0.312	0.3172

Table 2: Quantum time–memory trade-offs: [1, Tables 4–5] vs. our re-optimisation.

k	m	t (paper)	t (ours)	α (ours)	configuration C (ours)
3	0.1887	0.3098	0.3097	1.2343	(-0.333, -0.333, -0.333)
3	0.1890	0.3073	0.3073	1.2342	(-0.344, -0.344, -0.312)
3	0.1900	0.3056	0.3060	1.2336	(-0.355, -0.358, -0.287)
3	0.1907	0.3053	0.3053	1.2333	(-0.362, -0.362, -0.276)
4	0.1724	0.3276	0.3260	1.3050	(-0.260, -0.247, -0.246, -0.255)
4	0.1750	0.3153	0.3159	1.3035	(-0.325, -0.236, -0.238, -0.227)
4	0.1800	0.3127	0.3129	1.2957	(-0.373, -0.244, -0.214, -0.210)
4	0.1813	0.3120	0.3120	1.2951	(-0.385, -0.228, -0.227, -0.204)

3.4 QRACM-aware asymptotics

The analysis of [1] charges one time unit per QRACM access. If instead each access to a list of size $|L_1|$ costs $|L_1|^\beta$ ($\beta = 1$: gate cost of any circuit QROM; $\beta = 1/2$: select–swap with $\sqrt{|L_1|}$ qubits; $\beta = 0$: the QRACM assumption), every Grover iteration inside FAS is multiplied by $|L_1|^\beta$, and the optimal filter angle shrinks the per-filter lists. Table 3 and Figure 3 show the re-optimised exponents. For $\beta \geq 1/2$ both quantum tuple sieves are slower than the classical 3-sieve of [1] (0.338) and than the classical 2-sieve (0.292): the quantum advantage of these low-memory sieves exists only under the QRACM assumption. This mirrors the conclusion of [12] for 2-sieves.

4 Reproduction and validation of Kim et al.

4.1 Methodology

We transcribed Tables 3–10 of [2] and re-implemented, independently of their code (which is not public), (i) the NV iteration model of [2, Sec. IV-B]: $|C| = 2^{0.2352d+0.102\log_2 d+2.45}$, $|L| = d|C|$, $G_{iter} = \lceil 3.1\sqrt{|C|}\rceil$, $L_{iter} = |L|$, $S_{iter} = d$, $T_{iter} = S_{iter}L_{iter}G_{iter}$; (ii) the spherical-LSF variant ($|C_{\mathrm{LSF}}| = |C|\, t(\alpha)C_d(\alpha)^2$, $t = \ln(10^3)/W_d(\alpha, \alpha, \pi/3)$, α minimising filtering plus search cost); (iii) the QROM formulas of [2, Sec. III-A]; (iv) their aggregation "oracle $\times T_{iter}$, QROM cost doubled for uncomputation". We then reconstructed their oracle circuit from the description of [2, Sec. III] in our framework.

4.2 Results of the reproduction

Iteration model and aggregation [R]. With their oracle numbers, our pipeline reproduces every arithmetic-only entry of Table 7 to ≤ 0.01 bits (Table 4); the printed T_{iter} column is our value rounded *up* (e.g. $156.02 \to 157$), while their totals use the unrounded value. The QROM address lengths a agree for all 12 schemes. For LSF our α-optimisation yields $|C_{\mathrm{LSF}}|$ smaller by 2.3–3.2 bits and T_{iter}^{LSF} smaller by 1.4–2.0 bits; the paper does not fully specify the objective used for α, and variants of the polynomial factors bracket its values.

QROM rows [R]. $\#T$, T_d and M of both QROM models agree exactly with our implementation of the formulas (low-D: with the split $a = a_1 + a_2$ minimising $N_1 + (\ell + 1)N_2$, e.g. $a_1 = 52$ for ML-KEM-512). The gate counts and full depths are not derivable from the stated formulas, but we found that they are exactly explained by the classical-data CNOTs of the lookups: the low-D gate count equals $T_{iter} \cdot 2N\ell$ to 0.00 bits for all plain-NV schemes, and the low-Q gate count equals $(1.5\text{–}2.5) \cdot T_{iter} \cdot 2N\ell$; the low-Q full depth corresponds to ≈ 36 layers per unary-iteration step.

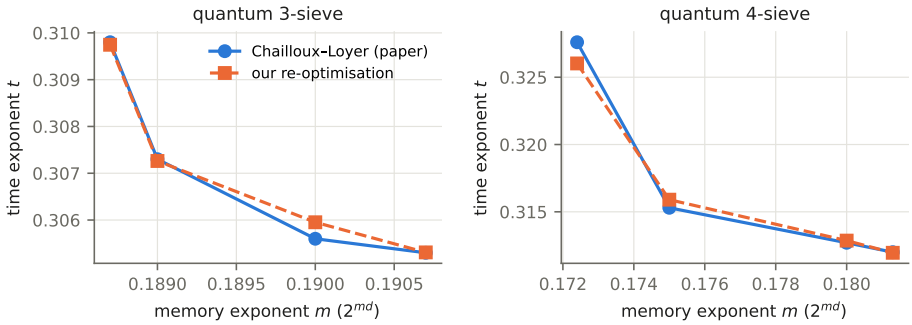

Figure 2: Reproduced quantum trade-off curves of the 3-sieve and 4-sieve.

Table 3: QRACM-aware optimum (memory m fixed, (C, α) re-optimised).

k	m	β	t	α	$\log_2 \lvert L_1 \rvert / d$
3	0.1888	0.0	0.3085	1.2343	0.1055
4	0.1725	0.0	0.3247	1.3140	0.1244
4	0.1813	0.0	0.3120	1.2951	0.1258
3	0.1888	0.5	0.3611	1.2343	0.1055
4	0.1725	0.5	0.3853	1.2995	0.1188
4	0.1813	0.5	0.3744	1.2965	0.1233
3	0.1888	1.0	0.4093	1.1761	0.0734
4	0.1725	1.0	0.4432	1.2995	0.1187
4	0.1813	1.0	0.4356	1.2974	0.1220

4.3 Reconstruction of the NV oracle

Figure 4 shows our reconstruction of one coordinate block of [2, Fig. 6]: in-place Draper CLA adding the classical v_i to the pre-negated centre coordinate, conditional two's complement controlled by the sign l, copy, $s \times s$ partial products (CNOT on the diagonal, AND elsewhere), Dadda carry-save tree, out-of-place CLA, in-place CLA accumulation into Res, then the reverse of everything but Res; after d blocks, one comparator. The circuit is validated on random inputs. Our implementation of Draper's in-place CLA reproduces the Toffoli count $10n - 3w(n) - 3w(n-1) - 3\lfloor \log n \rfloor - 3\lfloor \log(n-1) \rfloor - 7$ of [4] exactly for all tested n.

4.4 Discrepancies

1. **Missing uncomputation.** The oracle totals of [2, Table 5] are exactly the sums of the subtraction and squaring stages plus one comparator (T_d: $60000 + 60750 + 14 = 120764$ for ML-KEM-512). The accumulator Res (and with it the decision qubit's entanglement) is never uncomputed, yet a Grover phase oracle must return all work qubits to $|0\rangle$. The in-place additions into Res cannot be undone by measurement, so the correct oracle costs $\approx 2\times$ in $\#T$, T_d and FD (our reconstruction: 1.48 $\rightarrow$ 2.96 M T for ML-KEM-512). The QROM cost is doubled for uncomputation in [2] but the arithmetic is not.

2. **Width growing with d.** The squaring widths of [2, Table 4] grow linearly with d (ML-KEM: $3421, 4902, 6603$ for $d = 375, 586, 829$, i.e. $+7.0$ qubits per coordinate; ML-DSA $+8.0$; HAETAE $+6.0$), although ancillae are stated to be reused across coordinates. A correctly reused per-coordinate workspace is independent of d (ours: ≈ 540 work qubits for $s = 14$). We conjecture unreleased qubits in the original implementation.

3. **T-count of the arithmetic.** Our reconstruction uses 1.48 M T gates for one pass at ML-KEM-512 versus 4.48 M reported; with 7-T Toffolis in the dirty-target positions our T-depth per coordinate (138–204 for $s = 12$–25) is of the same order as theirs (152–184) but the T-count remains $\approx 2.2\times$ lower. We could not identify a reading of the description that reproduces their per-coordinate counts.

4. **Squaring ignores symmetry.** The $s \times s$ partial-product matrix uses $s^2 - s$ ANDs where $\binom{s}{2}$ suffice ($x_i x_j = x_j x_i$, Section 5.2).

5. **Unlookup cost.** Uncomputing a QROM read by repeating it doubles its cost; measurement-based unlookup costs $O(\sqrt{N})$ [9], i.e. the QROM-inclusive costs are overestimated by almost $2\times$ (validated in Section 6.2).

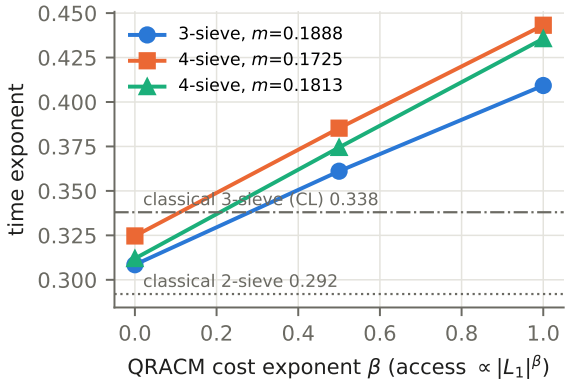

Figure 3: Time exponent of the quantum 3-/4-sieve when a QRACM access costs $|L_1|^\beta$.

Table 4: Maximum absolute deviation (in $\log_2$) between our recomputation of [2, Tables 7–8] (from their Table 5 oracle and our pipeline) and the published values, over the 12 parameter sets.

sieve	QROM	$\#T$	M	T_d	$\#G$	FD
plain	None	0.00	0.00	0.00	0.00	0.00
plain	lowQ	0.00	0.00	0.00	13.44	5.23
plain	lowD	0.06	0.06	0.00	122.06	105.00
LSF	None	1.10	0.00	1.10	1.10	1.09
LSF	lowQ	1.09	0.01	1.09	14.53	6.31
LSF	lowD	1.09	0.04	1.10	69.58	52.09

5 Concrete quantum architectures

5.1 What must be computed

Inside a tuple filter the quantum algorithm searches index registers $|i_1\rangle, \ldots, |i_k\rangle$ ($a = \lceil \log_2 |L_1| \rceil$ qubits each); a vector is brought into a data register only through QRACM. Every oracle evaluates *configuration predicates*

$$\mathcal{P}_{ij}(i_i, i_j) = \left[\langle \vec{x}_i | \vec{x}_j \rangle \le \tau_{ij}\right], \qquad \tau_{ij} = \lfloor C_{ij} R^2 \rfloor,$$

on two vectors that are *both in superposition* (the first-register vector is a superposition over L_1 throughout $\mathcal{A}$). This is the essential difference from the NV oracle of [2, 15], where the list vector $\vec{v}$ is classical. The 3-sieve needs the three pairwise products $\langle \vec{x}_1 | \vec{x}_2 \rangle$ (Grover on i_2), $\langle \vec{x}_1 | \vec{x}_3 \rangle$ (Grover on i_3) and $\langle \vec{x}_2 | \vec{x}_3 \rangle$ (AA); the 4-sieve needs all six, with two predicates per Grover oracle on i_3 and i_4.

We check the configuration on the lattice vectors $\vec{x}_i$ themselves rather than on the residual vectors $\vec{y}_i$: by Lemma 1 the two are equivalent (under the heuristic $A_i \perp \vec{y}_j$), the check on $\vec{x}$ is the one that matters for reduction, and it needs no real-valued residual arithmetic. Coordinates are s-bit two's-complement integers (equivalently fixed-point numbers with a common scale), with $s = s_0 + 2$, $s_0 = \lceil \log_2 q \rceil$, as in [2]; the accumulator width is $W = 2s + \lceil \log_2 d \rceil$ (one guard bit beyond the worst case). All arithmetic is exact modulo 2^W; For comparability with [2], we use the coefficient width. Because every tuple measured at the end of Algorithm 6/7 is verified classically, an approximate predicate only perturbs the marked set; it never yields a wrong output.

5.2 Partial products with folded constants

Signed product (Baugh–Wooley). For s-bit two's complement a, b,

$$ab = \sum_{i,j<s-1} a_i b_j 2^{i+j} + a_{s-1} b_{s-1} 2^{2s-2} + \sum_{j<s-1} \left(\overline{a_{s-1} b_j} + \overline{a_j b_{s-1}}\right) 2^{s-1+j} - \left(2^{2s-1} - 2^s\right),$$

since $-x = \bar{x} - 1$ for a bit x. All s^2 partial products are ANDs into fresh qubits, the negated ones followed by an X; the constant is summed classically over all coordinates and folded into the threshold. No conditional negation (Kim et al.'s "2's comp" stage) is needed.

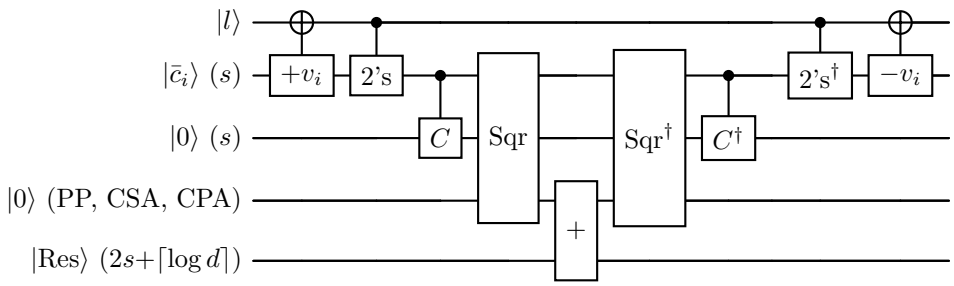

Figure 4: One coordinate block of the NV oracle of [2] as reconstructed. Repeated for $i = 1..d$, then a single comparator $\mathrm{Res} \le \gamma^2 R^2$.

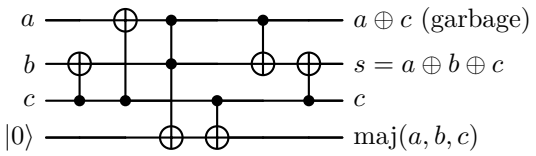

Figure 5: Quantum 3:2 compressor: one AND, five CNOTs; the uncompute is $\mathrm{AND}^\dagger$ (no T).

Signed symmetric square. For $x = u - x_{n-1}2^{n-1}$ with $u = \sum_{i<n-1} x_i 2^i$,

$$x^2 = \sum_{i<n-1} x_i 4^i + \sum_{i<j<n-1} x_i x_j 2^{i+j+1} + x_{n-1}4^{n-1} + \sum_{i<n-1} \overline{x_{n-1}x_i}\, 2^{n+i} - \sum_{i<n-1} 2^{n+i}.$$

This needs $\binom{n}{2}$ ANDs; the diagonal bits are the input qubits themselves (they enter the tree in place and are restored by the reverse). Kim et al.'s squaring uses $n^2 - n$ ANDs plus a copy register (Figure 4), i.e. twice as many partial products.

"Diff" predicate. Storing $N_i = \|\vec{x}_i\|^2$ next to each vector in QRACM ($2W$ extra data bits), we use

$$\langle \vec{x}_1 | \vec{x}_2 \rangle \le \tau \iff \|\vec{x}_2 - \vec{x}_1\|^2 - N_1 - N_2 + 2\tau \ge 0.$$

Per coordinate: one $(s{+}1)$-bit in-place ripple subtraction (s ANDs) and a signed square of $s+1$ bits ($\binom{s+1}{2}$ ANDs): 119 ANDs instead of 196 for $s = 14$. The norms enter the tree as complemented rows ($\bar{N} = -N - 1$, constants folded).

5.3 Carry-save reduction and sign extraction

All partial-product bits, the complemented rows and one constant row are reduced by a Dadda tree of quantum 3:2 compressors (Figure 5): one AND into a fresh qubit for the majority (carry) and five CNOTs for the sum, so each full adder costs 4 T and every AND has a clean target. Carries out of column $W - 1$ are not computed (two's complement modulo 2^W), so the top-column compressors need no AND. The two remaining rows R, R' are never added: only the sign of $R + R'$ is needed. It is extracted by a parallel-prefix carry tree (Figure 6): leaves $g_i = r_i \wedge r'_i$, $p_i = r_i \oplus r'_i$ (in place), internal nodes $g = g_{\mathrm{hi}} \oplus (g_{\mathrm{lo}} \wedge p_{\mathrm{hi}})$ (the two terms are mutually exclusive, so XOR equals OR and no OR gate is needed) and $p = p_{\mathrm{lo}} \wedge p_{\mathrm{hi}}$ only where required; $\mathrm{MSB} = r_{W-1} \oplus r'_{W-1} \oplus c_{W-1}$. This uses $\le 3W$ ANDs in T-depth $\lceil \log_2 W \rceil + 1$.

Lemma (measurement-based uncomputation). *If every AND of a compute block targets a fresh qubit and no input of an AND is modified after the AND until the block is reversed, the exact reverse of the block contains no T gate except for the inverses of in-place adders.*

5.4 Three predicate architectures

LD (low depth, global tree). All d coordinates are processed in parallel (independent ancillae, enforced by a *parallel region* in the allocator), one Dadda tree reduces all $\approx d\binom{s+1}{2}$ bits, then the sign tree (Figure 7). The reverse costs only the d subtractions: $\#T \approx 4d(2s + \binom{s+1}{2} + n_{FA}/d)$, $T_d = O(\log(ds))$, width $\Theta(ds^2)$.

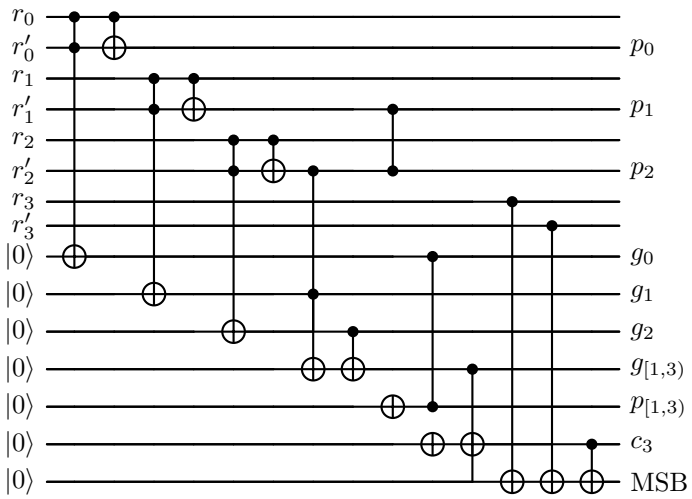

Figure 6: Sign extraction for $W = 4$. Every AND targets a fresh qubit; the circuit is uncomputed by AND† after the phase kick.

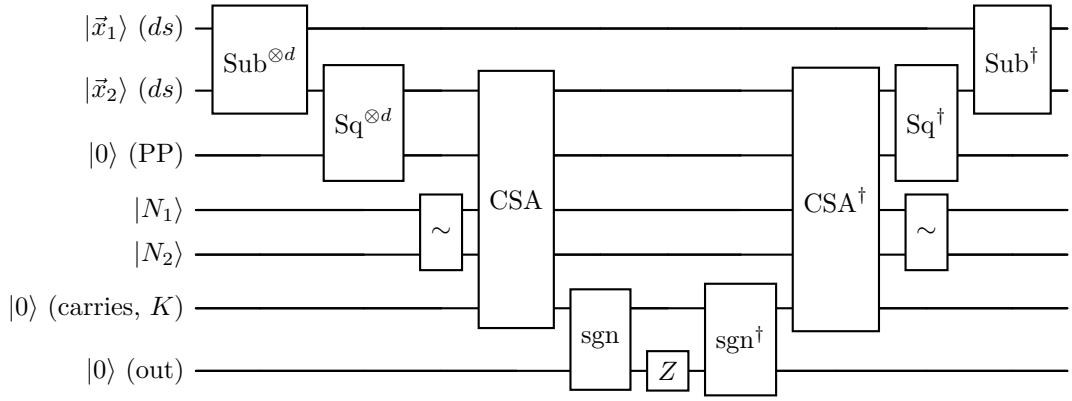

Figure 7: LD configuration predicate: coordinate-parallel subtraction and signed squaring, one global carry-save tree including $\bar{N}_1, \bar{N}_2$ and the folded constant K, sign extraction, phase, exact reverse.

LQ (low qubit, batch 1). One coordinate at a time: partial products, local tree, in-place ripple additions of the two rows into Res [5], local uncompute (Figure 8); finally the sign of Res $+ K$ against a classical constant (only the internal nodes of the prefix tree need ANDs). Width $\approx$ inputs $+O(s^2)$, $\#T \approx 2\times$ compute.

BAL (balanced, batch p). As LQ with $p = 16$ coordinates per tree and Draper in-place CLAs [4] for the accumulation (all dirty-target Toffolis as TOF4).

For several predicates in one oracle (4-sieve) the output bits are combined by a multi-controlled Z (log-depth AND tree) before the reverse.

5.5 QRACM

List L_i^j of a tuple filter is a classical table of $|L_1|$ words of $\ell = ds$ bits ($+2W$ for the norms in the diff method). We implement (Figure 9) the unary-iteration QROM [7] (validated T-count $4(N-2)$), the measurement-based unlookup (validated on superposed addresses: the state returns exactly to the uniform superposition for every measurement outcome) with T-count $4[2(2^{\lfloor a/2 \rfloor} - 2) + 2^{\lceil a/2 \rceil} - 2]$, and select–swap with $\lambda = 2^k$ copies (validated T-count $4(N/\lambda - 2) + 4\ell(\lambda - 1)$). The bucket-brigade QRAM is costed with [12, Lemma 9]. We pair them with the arithmetic as: LQ + unary QROM, BAL + select–swap (with λ minimising $\#T$), LD + bucket brigade.

5.6 Grover iterations, $\mathcal{A}$ and amplitude amplification

A Grover iteration on i_j loads $\vec{x}_j$, applies the phase oracle of the relevant predicates (with $\vec{x}_1$ [and $\vec{x}_2$] resident), unloads, and diffuses (Figure 10). The 3-sieve operator $\mathcal{A}$ and the AA iterate are shown in Figure 11; the

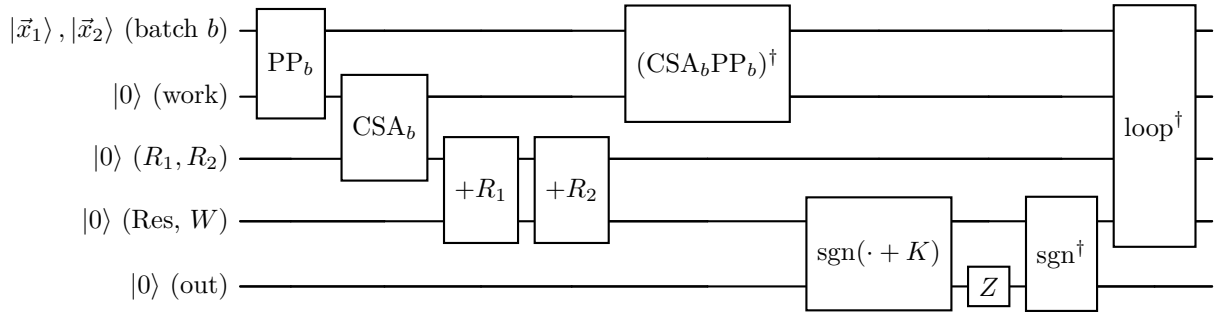

Figure 8: LQ/BAL predicate: per batch of p coordinates, partial products and a local tree whose two output rows are added in place into Res before the batch is uncomputed; sign of Res $+ K$; phase; reverse of the loop.

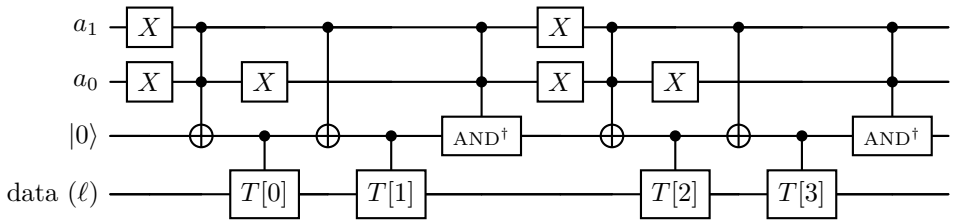

Figure 9: Unary-iteration QROM for $a = 2$ ($T[i]$: CNOT pattern writing word i).

4-sieve operator in Figure 12, where $\vec{x}_2$ is loaded once after the Grover search on i_2. Iteration counts are $k_j = \lfloor \pi/(4\theta_j) \rfloor$ with $\sin^2 \theta_j$ the expected marked fraction.

Full-configuration AA oracle (correction). Algorithms 6/7 [1] amplify only the *last* constraint, relying on the inner Grover searches to have produced exact superpositions over $L_2(\vec{y}_1)$, $L_3(\cdot)$, ... In any concrete instance the inner searches are approximate (the marked fraction varies with $\vec{y}_1$ and Grover overshoots/undershoots), so tuples violating an earlier constraint retain amplitude and *are amplified as well*. Our S_χ therefore tests the full configuration (3 resp. 6 predicates). Since S_χ is applied t_{AA} times versus $(2t_{AA} + 1) \sum_j k_j$ Grover iterations inside $\mathcal{A}$, the overhead is negligible (Section 6.2 quantifies the effect).

5.7 An optimised NV oracle

For comparability we apply the same techniques to the NV oracle: the pre-negated centre coordinate is sign-extended to $s + 1$ bits, the classical v_i is added in place with a ripple adder (s ANDs), the signed result is squared directly with $\binom{s+1}{2}$ ANDs (no conditional negation, no copy), all constants are folded, and one global tree plus the sign tree produce $[\|\vec{v} - \vec{c}\|^2 \le \gamma^2 R^2]$; the phase oracle includes the full reverse.

6 Implementation and experimental methodology

6.1 Framework

We implemented a small reversible-circuit framework (Python/NumPy) rather than ProjectQ or Qiskit, because we need (a) millions of gates per oracle at $d \approx 10^3$, (b) explicit AND/AND† semantics with precondition checks, and (c) exact simulation of circuits with hundreds of qubits but few superposed registers. A `Circuit` emits gates to pluggable back-ends:

- **Counter**: gate counts, $\#T$, Clifford count, measurements, T-depth and full depth by ASAP scheduling of the decomposed gate costs on per-qubit timelines, peak width plus the peak of concurrently needed transient AND ancillae. Ancillae are reused by an availability-aware allocator (the free qubit that became idle earliest), and a parallel region defers reuse when independent sub-circuits should run concurrently.
- **BitSim**: bit-sliced classical simulation (each qubit holds a bit-vector over up to thousands of inputs). It asserts that every AND target is $|0\rangle$, that every AND† target equals the AND of its controls (i.e. the measurement-based uncompute is valid), and that every released ancilla is $|0\rangle$.

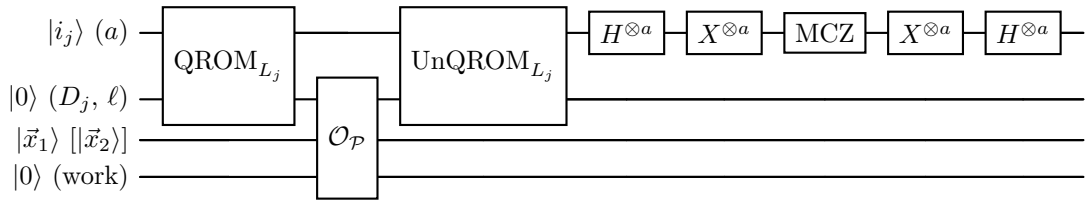

Figure 10: One Grover iteration on index register i_j.

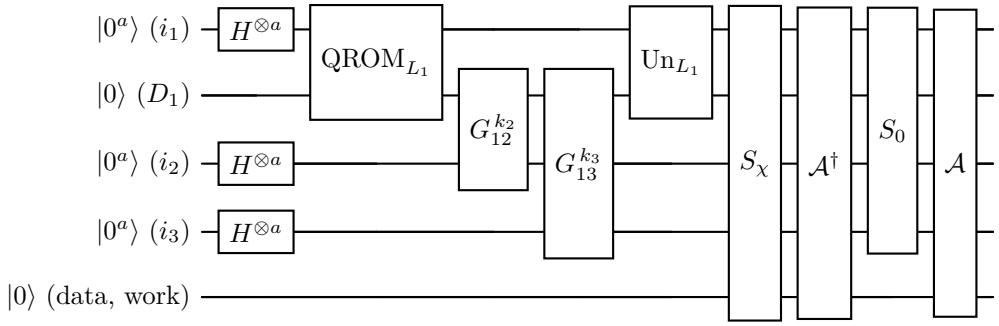

Figure 11: Quantum 3-sieve (Algorithm 6) as implemented: $\mathcal{A}$ followed by one AA iterate $\mathcal{A}S_0\mathcal{A}^\dagger S_\chi$ (repeated t_{AA} times). G_{1j} = Grover iteration of Figure 10; boxes may span wires they do not act on.

- **SparseSim**: exact sparse state-vector simulation over basis rows (64-bit words) with complex amplitudes; H splits and re-merges rows, X-basis measurements collapse. It optionally executes $\mathrm{AND}^\dagger$ literally as measurement plus classically controlled CZ.
- **Tape**: records a block so that its exact inverse is emitted automatically ($\mathrm{AND}\leftrightarrow\mathrm{AND}^\dagger$), which is how every compute/uncompute pair in this paper is produced.

6.2 Validation

Arithmetic. Gidney ripple and Draper in-place adders (with/without carry-out, $n \le 25$), Draper out-of-place adders, Dadda trees (up to 40 operands), Baugh–Wooley products and signed squares (exhaustive for $s \le 6$ resp. 8), and both sign extractors are compared with integer arithmetic; inverses restore all inputs and free ancillae clean. The Draper in-place adder matches the published Toffoli formula exactly.

Oracles. Kim-style and optimised NV oracles and all predicate variants (mult/diff × tree/batch × ripple/CLA) agree with exact integer evaluation on 600–1000 random inputs per configuration, with full uncomputation.

QRACM. Unary QROM and select–swap (all $k \le a$) reproduce every table entry and their T-counts equal the closed forms; the measurement-based unlookup returns a superposed address register exactly to the uniform state for every sampled outcome.

Complete sieve circuits. For random toy instances (lists of $N = 4$–16 vectors, $d \le 3$, $s \le 4$) the full circuit of Algorithm 6/7—QROM, predicates, nested Grover, AA—is simulated exactly and the distribution over $(i_1, \ldots, i_k)$ is compared with an independent index-space model built from truth tables (Table 5). The maximum deviation is $\le 1.1 \cdot 10^{-13}$ and no ancilla is left dirty in any branch (a separate run executes every $\mathrm{AND}^\dagger$ as a real X-measurement with classically controlled CZ). With the full-configuration oracle, AA reaches $p_{AA} \ge 0.974$ in all cases; with the literal oracle of Algorithms 6/7 ("last") the same instances only reach $p_{AA} = 0.124$ (3-sieve) and 0.015 (4-sieve), and the classical FindAllSolutions loop needs 19 resp. 212 runs instead of 4 resp. 3. The third row exposes a second effect that the asymptotic analysis hides: approximate inner searches leave the solutions with *unequal* amplitudes, so collecting all of them is a skewed coupon-collector problem (5946 runs for 4 solutions although $p_{AA} = 0.974$). At cryptographic sizes $|\mathrm{Sol}_f| \approx 1$ per filter, so this does not affect our per-call estimates.

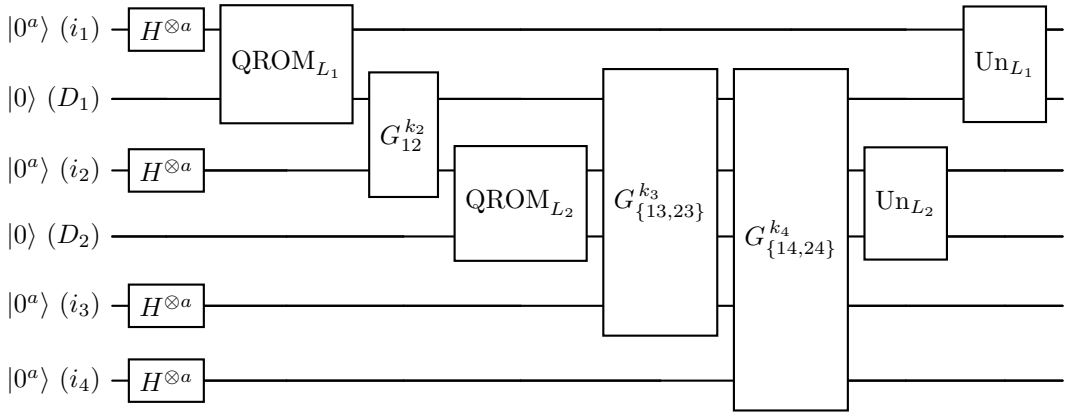

Figure 12: Operator $\mathcal{A}$ of the quantum 4-sieve (Algorithm 7) as implemented; $G_{\{13,23\}}$ tests two predicates per oracle call.

Table 5: Exact simulation of complete 3-/4-sieve circuits on toy instances. k_j: inner Grover iterations; max dev.: largest difference to the index-space model; runs: AA runs until every solution was sampled and verified classically.

k	N	d	s	arch	S_χ	|Sol|	k_j	t_{AA}	$p_{\mathcal{A}}$	p_{AA}	max dev.	qubits	$\#T$	runs
3	8	2	3	tree	full	2	(1, 1)	5	0.0220	0.9957	2.11×10^{-15}	189	9172	4
3	8	2	3	tree	last	2	(1, 1)	5	0.0220	0.1239	2.36×10^{-16}	79	7112	19
3	8	3	4	batch:1	full	4	(1, 1)	2	0.1154	0.9740	6.11×10^{-16}	155	17140	5946
3	8	2	4	batch:2	full	1	(1, 1)	11	0.0044	0.9980	5.55×10^{-15}	168	89804	1
3	16	2	3	tree	full	2	(2, 2)	35	0.0005	1.0000	1.49×10^{-14}	194	125920	4
4	4	2	3	tree	full	2	(1, 2, 1)	4	0.0312	0.9992	3.61×10^{-15}	370	17268	3
4	4	2	3	tree	last	2	(1, 2, 1)	4	0.0312	0.0153	3.96×10^{-16}	144	13396	212
4	8	2	3	tree	full	2	(1, 2, 2)	284	0.0000	1.0000	1.08×10^{-13}	352	1470784	3

7 Logical resource estimates

7.1 Oracle level (generated circuits)

Table 6 compares, for every parameter set, the NV oracle of [2] (as reported, one pass), our reconstruction (one pass, R1, and with the missing uncomputation, RF) and our optimised NV phase oracle in the low-qubit (LQ) and low-depth (LD) architectures, all including uncomputation. Table 7 gives the configuration predicate $[\langle\vec{x}_1|\vec{x}_2\rangle \le \tau]$ (diff method, compute+phase+reverse) for the three architectures. All numbers are counts of complete generated circuits at the stated (d, s); generation takes 1–10 s per circuit.

NV oracle. Across the 12 parameter sets, our low-depth NV phase oracle (LD) uses 9.4–14.1× fewer T gates than the one-pass oracle reported in [2] and $1.2\cdot10^3$–$3.4\cdot10^3\times$ smaller T-depth (76–130 instead of $1.2\cdot10^5$–$3.5\cdot10^5$), although it includes the full uncomputation; its width is larger ($8\cdot10^4$–$6\cdot10^5$ qubits), and its $T_d\cdot M$ product is 63–316× smaller. The low-qubit variant (LQ) dominates the published oracle in every metric: 4.6–6.4× fewer T gates, lower T-depth and 0.62–0.77× the width. The improvement comes from (i) direct signed squaring (no conditional negation, no copy, $\binom{s+1}{2}$ instead of s^2-s ANDs), (ii) a global carry-save tree whose reverse is measurement-based, and (iii) sign extraction instead of a full comparator.

Configuration predicate. For two superposed vectors the predicate costs 0.32–2.4 M (LD) to 0.71–4.9 M (LQ) T gates per call. LQ keeps the width within a few hundred qubits of the two input vectors ($1.1\cdot10^4$–$4.3\cdot10^4$ in total) at T-depth $7.4\cdot10^4$–$2.8\cdot10^5$; BAL cuts the T-depth by 5.4× for 30–50% more qubits; LD reaches T-depth 76–130 and full depth ≈ 1.2–$2.0\cdot10^3$ with $8\cdot10^4$–$6\cdot10^5$ qubits and half the T-count of LQ (its reverse is almost free).

8 Limitations

- **Finite-dimension models.** This chapter focuses on per-oracle circuit costs. Algorithm-level estimates require a finite-dimension iteration model to calculate list sizes and marked fractions at cryptographic dimensions, which is left for future work.

Table 6: NV oracle per call. Kim: [2, Tables 3–5] (one pass, no uncomputation of Res); R1/RF: our reconstruction one pass / with uncomputation; LQ/LD: our optimised phase oracle (Section 5.7).

	$\#T$ (10^6)					T-depth				qubits			
scheme	Kim	R1	RF	LQ	LD	Kim	R1	LQ	LD	Kim	R1	LQ	LD
KEM-512	4.48	1.48	2.96	0.82	0.38	120764	46850	76113	84	8659	5789	5927	94445
KEM-768	7.04	2.32	4.64	1.29	0.59	191050	73227	121290	85	13094	8746	9096	147521
KEM-1024	9.96	3.28	6.56	1.82	0.84	270268	103601	171591	86	18197	12148	12741	208652
DSA-44	10.82	3.79	7.58	2.35	1.14	152098	66432	131967	128	15081	11338	11025	285008
DSA-65	16.16	5.65	11.30	3.51	1.71	226596	99019	198970	129	21455	16167	16048	424536
DSA-87	22.53	7.88	15.75	4.90	2.38	315762	137982	277279	130	29078	21942	22054	591556
SMAUG-128	4.46	1.29	2.58	0.70	0.32	126294	46174	72971	76	8915	5338	5570	78844
SMAUG-192	6.76	2.22	4.44	1.20	0.55	190050	75841	119688	81	13114	8500	8912	136186
SMAUG-256	9.42	3.09	6.19	1.68	0.76	264894	105726	166830	82	18569	11659	12314	189786
HAETAE-120	7.31	2.48	4.95	1.43	0.68	152382	61609	107412	100	11436	8533	8583	169159
HAETAE-180	13.88	4.70	9.39	2.72	1.29	288374	116616	206534	102	20606	15413	15843	320050
HAETAE-260	16.93	5.73	11.45	3.31	1.57	351742	142248	251924	102	24878	18617	19225	390426

Table 7: Configuration predicate for two superposed vectors (one call: compute, phase, reverse). M includes both input vectors and the norm registers.

		LQ (batch 1, ripple)				BAL (batch 16, CLA)				LD (global tree)			
scheme	(d, s)	$\#T/10^6$	T_d	FD	M	$\#T/10^6$	T_d	FD	M	$\#T/10^6$	T_d	FD	M
KEM-512	(375,14)	0.83	77766	1116049	11630	0.80	14215	166547	15256	0.38	84	1277	100190
KEM-768	(586,14)	1.30	123207	1784745	17967	1.26	22147	259085	21611	0.59	85	1296	156444
KEM-1024	(829,14)	1.84	174237	2523708	25257	1.78	31283	365683	28888	0.84	86	1315	221214
DSA-44	(394,25)	2.36	133774	1886632	21392	2.36	24731	276168	32188	1.15	128	1913	295435
DSA-65	(587,25)	3.52	201553	2851082	31434	3.51	36824	410733	42210	1.71	129	1931	440001
DSA-87	(818,25)	4.91	280786	3971432	43446	4.89	51391	572927	53957	2.38	130	1952	613014
SMAUG-128	(410,12)	0.71	74342	1090126	10970	0.68	13676	162704	13727	0.32	76	1165	84286
SMAUG-192	(617,13)	1.22	121691	1778326	17626	1.17	21936	258416	20772	0.55	81	1243	144959
SMAUG-256	(860,13)	1.69	169562	2477680	24430	1.63	30528	359339	27621	0.76	82	1257	202147
HAETAE-120	(428,18)	1.44	109305	1552149	16807	1.42	19961	228536	22596	0.68	100	1515	177571
HAETAE-180	(810,18)	2.73	209959	2992902	31329	2.69	38179	437263	37225	1.29	102	1547	335554
HAETAE-260	(988,18)	3.33	256061	3649900	38093	3.28	46540	532802	43890	1.57	102	1555	409245

9 Conclusion

We reproduced and validated the concrete quantum NV sieve of Kim et al. and the asymptotic quantum 3-/4-sieves of Chailloux and Loyer, corrected errors in both, and built the first complete, simulated and counted quantum circuits for the 3-sieve and 4-sieve. The configuration-test arithmetic we developed (polarisation with signed symmetric squaring, folded constants, logical-AND carry-save trees with measurement-based uncomputation, prefix sign extraction) also improves the published NV oracle by an order of magnitude in T-count and three orders in T-depth.

References

[1] A. Chailloux and J. Loyer, "Classical and quantum 3 and 4-sieves to solve SVP with low memory," in *PQCrypto 2023*, LNCS 14154, pp. 225–255, Springer, 2023. HAL:hal-04276492.

[2] H. Kim, K. Jang, A. Baksi, S. Wang, S. Chakraborty, A. Chattopadhyay and H. Seo, "Concrete quantum sieve for shortest vector problem with application to lattice-based cryptography," *IEEE Access*, vol. 14, pp. 102027–102045, 2026. doi:10.1109/ACCESS.2026.3706759.

[3] H. Kim, S. Lim, K. Jang, S. Wang, A. Baksi, A. Chattopadhyay and H. Seo, "Tree-based quantum carry-save adder," IACR ePrint 2025/626.

[4] T. G. Draper, S. A. Kutin, E. M. Rains and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Inf. Comput.* 6(4–5), 2006; arXiv:quant-ph/0406142.

[5] C. Gidney, "Halving the cost of quantum addition," *Quantum* 2, 74, 2018.

[6] S. Jaques, M. Naehrig, M. Roetteler and F. Virdia, "Implementing Grover oracles for quantum key search on AES and LowMC," in *EUROCRYPT 2020*, LNCS 12106, pp. 280–310.

[7] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler and H. Neven, "Encoding electronic spectra in quantum circuits with linear T complexity," *Phys. Rev. X* 8, 041015, 2018.

[8] G. H. Low, V. Kliuchnikov and L. Schaeffer, "Trading T-gates for dirty qubits in state preparation and unitary synthesis," arXiv:1812.00954, 2018.

[9] D. W. Berry, C. Gidney, M. Motta, J. R. McClean and R. Babbush, "Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization," *Quantum* 3, 208, 2019 (Appendix C: measurement-based uncomputation of table lookups).

[10] P. Mukhopadhyay, "A quantum random access memory (QRAM) using a polynomial encoding of binary strings," *Sci. Rep.* 15, 11002, 2025.

[11] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca and P. V. Srinivasan, "On the robustness of bucket brigade quantum RAM," *New J. Phys.* 17, 123010, 2015.

[12] J. F. Doriguello, G. Giapitzakis, A. Luongo and A. Morolia, "On the practicality of quantum sieving algorithms for the shortest vector problem," arXiv:2410.13759, 2024; PQCrypto 2026.

[13] M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite and J. M. Schanck, "Estimating quantum speedups for lattice sieves," in *ASIACRYPT 2020*, LNCS 12492, pp. 583–613.

[14] M. Prokop, P. Wallden and D. Joseph, "Grover's oracle for the shortest vector problem and its application in hybrid classical-quantum solvers," arXiv:2402.13895, 2024.

[15] S.-M. Cho, C. Lee and S.-H. Seo, "Quantum security analysis of module-LWE PQC based on practical cost estimates," *IACR TCHES* 2025(4), pp. 437–462.

[16] L. Engelberts, Y. Chen, A. S. Gilani, M.-I. van Hoof, S. Jeffery and R. de Wolf, "An improved quantum algorithm for 3-tuple lattice sieving," in *CRYPTO 2026*; arXiv:2510.08473 / IACR ePrint 2025/2189.

[17] S. Bai, T. Laarhoven and D. Stehlé, "Tuple lattice sieving," *LMS J. Comput. Math.* 19(A), pp. 146–162, 2016.

[18] G. Herold and E. Kirshanova, "Improved algorithms for the approximate k-list problem in Euclidean norm," in *PKC 2017*, pp. 16–40.

[19] G. Herold, E. Kirshanova and T. Laarhoven, "Speed-ups and time–memory trade-offs for tuple lattice sieving," in *PKC 2018*, pp. 407–436.

[20] E. Kirshanova, E. Mårtensson, E. W. Postlethwaite and S. Roy Moulik, "Quantum algorithms for the approximate k-list problem and their application to lattice sieving," in *ASIACRYPT 2019*, pp. 521–551.

[21] A. Becker, L. Ducas, N. Gama and T. Laarhoven, "New directions in nearest neighbor searching with applications to lattice sieving," in *SODA 2016*, pp. 10–24.

[22] P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *J. Math. Cryptol.* 2(2), pp. 181–207, 2008.

[23] G. Brassard, P. Høyer, M. Mosca and A. Tapp, "Quantum amplitude amplification and estimation," *Contemp. Math.* 305, pp. 53–74, 2002.

[24] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *STOC 1996*, pp. 212–219.

[25] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent and J. Schanck, "Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3," in *SAC 2016*, pp. 317–337.

[26] A. G. Fowler, S. J. Devitt and C. Jones, "Surface code implementation of block code state distillation," *Sci. Rep.* 3, 1939, 2013.

[27] D. Litinski, "Magic state distillation: not as costly as you think," *Quantum* 3, 205, 2019.

[28] C. Gidney, N. Shutty and C. Jones, "Magic state cultivation: growing T states as cheap as CNOT gates," arXiv:2409.17595, 2024.

[29] A. G. Fowler, M. Mariantoni, J. M. Martinis and A. N. Cleland, "Surface codes: towards practical large-scale quantum computation," *Phys. Rev. A* 86, 032324, 2012.

[30] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.* C-22(12), pp. 1045–1047, 1973.

[31] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza* 34, pp. 349–356, 1965.

[32] P. Gossett, "Quantum carry-save arithmetic," arXiv:quant-ph/9808061, 1998.

[33] S. A. Cuccaro, T. G. Draper, S. A. Kutin and D. P. Moulton, "A new quantum ripple-carry addition circuit," arXiv:quant-ph/0410184, 2004.

[34] NIST, "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process," 2016.

[35] T. Laarhoven, *Search problems in cryptography*, PhD thesis, TU Eindhoven, 2016.

[36] A. Chailloux and J. Loyer, "Lattice sieving via quantum random walks," in *ASIACRYPT 2021*, pp. 63–91.