Optimized Solving Strategy and Two-Step Estimator for LWE with Hints

Hanbin Yang, Jinzheng Cao*, Risheng Yin, and Qingfeng Cheng

Information Engineering University, Zhengzhou 450001, China
caojinzheng@126.com

Abstract. Side-channel hints can reduce the cost of solving the Learning with Errors (LWE) problem. We optimize a two-step strategy for LWE with hints that uses BKZ reduction followed by final lattice sieving. Given a lattice constructed from the hints and a specified pair of reduction and sieving algorithms, the strategy jointly selects BKZ block sizes, tour counts, and a final sieving configuration. We also introduce a two-step estimator that predicts the reduced-basis profile and estimates the probability of secret recovery. We combine these estimates with cost models to select a strategy that meets the success-probability and resource constraints. Experiments show that our strategy significantly outperforms the methods of Cao et al. (CRYPTO 2025) and May and Nowakowski (ASIACRYPT 2023). For example, using BLASterBKZ for lattice reduction and the BGJ sieve for final sieving, our strategy recovers the secret for a CRYSTALS-KYBER 512 instance with 190 perfect hints in 7 minutes, compared with 1.6 hours reported by Cao et al. and 7.4 hours reported by May and Nowakowski. With 64 threads, our strategy recovers the secret with 170 perfect hints in 82 minutes. These results highlight the importance of accounting for side-channel leakage when assessing lattice-based cryptographic schemes.

Keywords: Learning with Errors · side-channel hints · lattice reduction · lattice sieving

1 Introduction

The Learning with Errors (LWE) problem [3] is a fundamental problem in lattice-based cryptography. The security of many lattice-based cryptographic schemes, including the NIST post-quantum standards ML-KEM and ML-DSA, relies on the hardness of LWE or related problems. To evaluate the security of these schemes, it is important to estimate the computational cost of solving LWE and its variants.

In practical implementations, side-channel leakage may reveal partial information about the secret vector and make LWE easier to solve. Dachman-Soled, Ducas, Gong, and Rossi [12] introduced a systematic framework that represents

* Corresponding author.

such information as hints. The main types are perfect hints, approximate hints, modular hints, and mod-q hints. These hints impose additional constraints on the secret and can be used to construct a new lattice for the attack. Depending on their type and number, the hints can reduce the lattice dimension, change its determinant, or shorten the target vector. May and Nowakowski [13] used linear algebra to process mod-q hints and studied attacks with many hints. Dachman-Soled, Gong, Hanson, and Kippen [15] revisited security estimation for LWE with hints from a geometric perspective. Cao, Jiang, and Cheng [14] used Gaussian elimination to construct lattices from different types of hints. Once the lattice has been constructed, however, an efficient attack still requires suitable algorithms and parameters for finding its target vector.

BKZ reduction and lattice sieving are two important tools for finding short vectors in lattices. BKZ [16,17] improves the quality of a lattice basis by solving SVP instances on a sequence of projected blocks. Its block size and number of tours determine both the reduction effort and the quality of the resulting basis. Lattice sieving [20,32,33] searches for short vectors by maintaining and processing a collection of lattice vectors. These algorithms can be combined in a two-step solving procedure [18,24,27]. In the first step, BKZ reduces the lattice basis. In the second step, a sieve searches for the projected target vector in a sublattice of the reduced basis, after which the target is lifted to the full lattice. The two steps involve a trade-off. Stronger BKZ reduction generally takes more time but may reduce the cost of the final sieve, whereas weaker reduction is faster but may require a more expensive sieving call.

Selecting an efficient two-step strategy therefore requires more than determining a single BKZ block size. One must choose the reduction schedule, including the block sizes and tour counts, as well as the dimension and parameters of the final sieving call. For LWE with hints, these choices depend on the lattice produced from the available hints and on the distribution of the target vector after projection. Existing estimators for LWE with hints mainly predict the BKZ block size needed for secret recovery [12,14], while previous studies of two-step strategy optimization primarily consider LWE without hints [24,27]. Consequently, the trade-off between BKZ reduction and final sieving has not been systematically evaluated for hint-dependent instances in these frameworks. In this work, we study how to combine BKZ reduction and lattice sieving to solve LWE with hints efficiently. We develop a solving strategy and a two-step estimator to compare candidate configurations by their estimated computational cost and success probability, subject to specified time and memory limits.

1.1 Related Work

Previous studies on LWE with hints have mainly focused on incorporating the available hints into a lattice and estimating the difficulty of recovering the secret. Dachman-Soled, Ducas, Gong, and Rossi (DDGR) [12] introduced a systematic framework for analyzing LWE with hints. Their framework supports perfect hints, approximate hints, modular hints, and mod-q hints, and incorporates the hints into the primal attack lattice one at a time. However, processing each hint

requires repeated computations of the dual lattice basis, which can be expensive when many hints are available. DDGR also developed a security estimator that uses the expected length of the target vector to determine the lattice reduction needed for secret recovery. Dachman-Soled et al. [15] later studied LWE with hints from a geometric perspective and further analyzed how the available information affects the target vector.

May and Nowakowski (MN) [13] improved the efficiency of lattice construction for LWE with hints. In particular, they proposed a linear algebra method for processing mod-q hints. Given k independent mod-q hints, this method can reduce the secret dimension from n to $n-k$. For perfect hints and general modular hints, however, their approach relies on LLL reduction, which is more expensive than the corresponding linear algebra operations. MN also studied the "too many hints" regime, in which sufficiently many hints make it possible to recover the secret with relatively inexpensive lattice reduction. Their results show that the number and type of hints can have a significant effect on the cost of solving an LWE instance.

Cao, Jiang, and Cheng (CJC) [14] subsequently proposed a refined attack that processes different types of hints using Gaussian elimination. Their method first constructs a $\mathbb{Z}$-SIS lattice from the hints and then combines this lattice with the LWE samples when necessary. Since the $\mathbb{Z}$-SIS problem and the LWE instance share the same secret, the resulting lattice contains a short vector from which the secret can be recovered. Instead of applying LLL to the entire lattice basis, CJC compute the row echelon form of a smaller matrix block. This provides a more efficient construction for perfect, approximate, modular, and mod-q hints. Based on the constructed lattice, they also developed a complexity estimator to predict the BKZ block size required for secret recovery. Their work improves the construction and the block-size estimate, but does not optimize a separate sieving search after BKZ reduction.

After constructing the lattice, the attack must recover its target vector. BKZ reduction [16] is widely used for this purpose. It repeatedly applies a short-vector search to projected blocks of a lattice basis and improves the basis through a sequence of reduction tours. The block size determines the strength and cost of each local search, while the number of tours affects the quality of the resulting basis. BKZ 2.0 [17] introduced improvements to enumeration and preprocessing, together with a simulator for predicting the quality of a reduced basis. These developments made BKZ an important component of practical lattice attacks, but a large block size can be expensive, especially when it is used before the basis has been sufficiently reduced.

Progressive reduction offers an alternative to applying a large block size from the beginning. It starts with smaller blocks and gradually increases the block size as the basis improves. Small-block tours are relatively inexpensive and can reduce the cost of subsequent tours with larger blocks. Aono et al. [18] proposed ProBKZ, which selects a progressive reduction sequence by estimating the cost of moving between different basis-quality levels. Rather than following a fixed sequence of block sizes, ProBKZ searches for a lower-cost reduction path. These

studies show that the order and number of BKZ tours can affect the total reduction time, even when the final block size is the same.

Lattice sieving provides another way to find short vectors. Unlike enumeration, which searches through a tree of candidate vectors, a sieve maintains a collection of lattice vectors and combines them to obtain shorter ones. Early practical sieving methods were developed by Nguyen and Vidick [20], and later algorithms improved the running time by using more efficient nearest-neighbor searches [32,33]. Sieving has therefore become an important option for solving high-dimensional SVP instances, although its vector database can require substantial memory. The General Sieve Kernel (G6K) [22] made it possible to use a sieve as the local short-vector solver inside lattice reduction. PnJBKZ uses Pump as its local solver, and G6K can also run a standalone Pump after reduction [22,24,25]. In contrast, a BKZ implementation can also be paired with a separate sieve that runs only after reduction. BLASter [28] provides practical lattice reduction routines, while the BGJ sieve [32,33] is one possible solver for the final search. The choice of the two algorithms and their parameters determines the total running time and memory use.

A two-step lattice attack uses a separate search after reduction. The first step applies lattice reduction to improve the basis. The second step runs an SVP solver on the reduced basis or on one of its projected sublattices. Stronger reduction generally increases the preprocessing time but can make the final search easier. Weaker reduction has the opposite effect. Therefore, minimizing the reduction time alone does not necessarily minimize the total running time. Another important choice is the dimension of the final SVP search: searching in a larger projected sublattice takes more time, but searching in a smaller one may fail to recover the target. The reduction schedule and the search dimension must be considered together.

Several methods have studied this trade-off for LWE without hints. Some earlier estimators choose a BKZ block size using a condition on the expected length of the projected target vector, and expect the target to be recovered during BKZ reduction [26]. ProBKZ [18] uses a model-based method to optimize a progressive reduction sequence, followed by full-dimensional enumeration. G6K-based solvers [22,23] instead combine PnJBKZ reduction with a final Pump search on a projected sublattice, while their reduction schedules and stopping points are typically selected by heuristic rules. These methods differ in how they divide the work between reduction and final search, and the same LWE instance can have different solving costs under different choices.

Xia et al. [24,27] studied two-step strategy optimization for primal attacks on LWE. They represented a solving strategy by a sequence of reduction operations and a final SVP search, and developed methods for comparing the estimated costs of different strategies. Their framework uses a reduction simulator to predict the quality of the basis after a chosen schedule, and an SVP-dimension estimator to select the dimension of the final search. They also introduced a search procedure for finding a low-cost strategy within the specified strategy space and cost model. This work provides a systematic alternative to choosing

BKZ block sizes and stopping points by fixed rules. However, its models are developed for LWE instances without hints and do not directly account for the changes caused by hint information.

These two lines of research address different parts of the problem. Existing attacks on LWE with hints provide methods for constructing the lattice and estimating the BKZ block size needed to recover the secret [12,13,14]. Studies of two-step attacks show how to reduce the overall cost by choosing the reduction schedule and final search jointly [18,24,27]. For LWE with hints, however, the lattice dimension, determinant, and target-vector distribution depend on the available hints. The probability of recovering the secret also depends on whether the final sieve returns the projected target and whether this target can be lifted to the full lattice. Consequently, a block-size estimate alone is not enough to compare different two-step strategies combining BKZ reduction and final sieving. A two-step estimator for LWE with hints must evaluate both the reduction and the final search, including their success probabilities and computational costs. This motivates the time- and memory-aware strategy optimization developed in this work.

1.2 Contributions

We study the joint selection of BKZ reduction and final sieving parameters for LWE with hints. We also develop a two-step estimator for evaluating candidate strategies. Figure 1 provides an overview of our attack framework. It shows how success probability, running-time, and memory estimates are used to select a solving strategy for the lattice constructed from the available hints. Our main contributions are as follows.

- **An optimized solving strategy for LWE with hints.** We propose a two-step solving strategy that combines BKZ reduction with a final lattice sieve. Given an LWE-with-hints instance and a specified pair of reduction and sieving algorithms, we consider different BKZ schedules and final sieving configurations. Each schedule specifies the block sizes and tour counts of its BKZ stages. The final configuration specifies the projected sublattice dimension and the parameters of the sieve. We select these choices jointly rather than optimizing the reduction stage alone. In this way, the strategy balances the running time of BKZ reduction against the running time of the final sieving step.
- **A two-step estimator for LWE with hints.** We develop an estimator that predicts the success probability, running time, and memory use of a given strategy. For each BKZ schedule, it predicts the Gram–Schmidt profile of the reduced basis and estimates the probability of secret recovery using the projected lattice and the target distribution conditioned on the hints. The success estimate accounts for both finding the projected target by sieving and recovering the secret after lifting. Together with the reduction-time, sieving-time, and memory models, this allows us to select the strategy with the lowest expected running time among those meeting the required success probability and memory limit.

– **New predictions and experimental results for LWE with hints.** We instantiate the proposed strategy with BLASterBKZ and the BGJ sieve and evaluate it on CRYSTALS-KYBER parameter sets. The estimator selects BKZ schedules and final sieving dimensions for different numbers of perfect hints. For CRYSTALS-KYBER 512 with 190 perfect hints, our optimized strategy takes 7 minutes with a single thread, compared with the 1.6 hours reported by Cao, Jiang, and Cheng [14]. Our strategy also solves CRYSTALS-KYBER 512 with 170 perfect hints in 82 minutes using multiple threads. These results demonstrate that optimizing the BKZ reduction stages and the final sieving call can reduce the solving time.

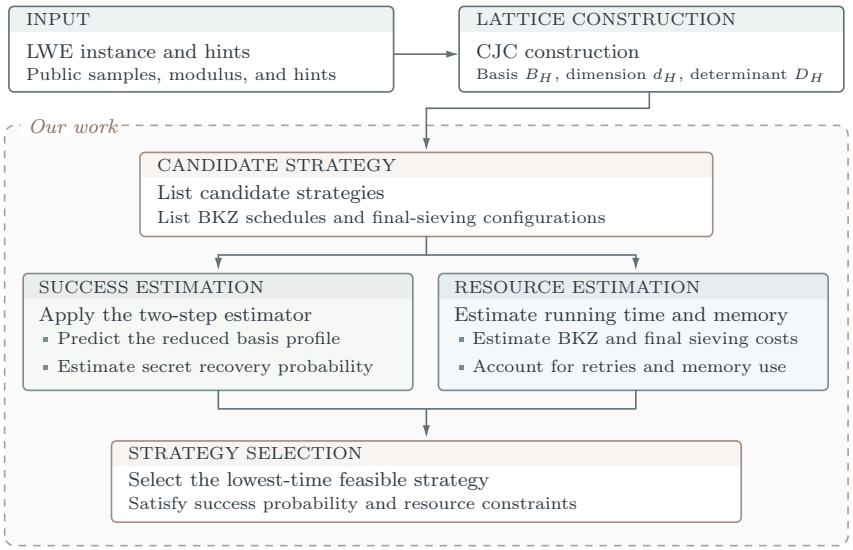

Fig. 1. Overview of our strategy optimization framework for LWE with hints.

1.3 Organization of the Paper

The rest of this paper is organized as follows. Section 2 introduces the necessary preliminaries, including lattices, BKZ reduction, lattice sieving, and the lattice construction for LWE with hints. Section 3 presents our optimized solving strategy and formulates the strategy selection problem. Section 4 develops the two-step estimator, including the prediction of the Gram–Schmidt profile and the estimation of the success probability of secret recovery. Section 5 describes cost estimation and strategy selection. Section 6 presents the numerical predictions and experiments on CRYSTALS-KYBER. Finally, Section 7 concludes the paper.

2 Preliminaries

In this section, we recall the basic definitions and notation used in this paper. We follow the lattice notation in [24,26], and use the formulation of LWE with hints from [12,14]. We use lowercase bold letters for vectors and uppercase bold letters for matrices. All vectors are treated as column vectors. The Euclidean norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|$, and the inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$. For a lattice basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_d]$, we denote its Gram–Schmidt vectors by $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$. Let $g_i = \|\mathbf{b}_i^*\|$. The logarithmic Gram–Schmidt profile is denoted by $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)$, where $\ell_i = \log_2 g_i$.

We also introduce the probability notation used in our estimator. For a random vector $\mathbf{X}$, $\mathbb{E}[\mathbf{X}]$ and $\mathrm{Cov}(\mathbf{X})$ denote its expectation and covariance matrix, respectively. For a scalar random variable X, $\mathrm{Var}(X)$ denotes its variance. We write $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The chi-squared distribution with η degrees of freedom is denoted by χ^2_η, and Φ denotes the cumulative distribution function of the standard normal distribution. Given a set of hints H, we use $\Pr[E \mid H]$ and $\mathbb{E}[\mathbf{X} \mid H]$ to denote conditional probability and conditional expectation, respectively.

2.1 Lattices and Lattice Problems

We first recall the definitions of lattices, the shortest vector problem, and the Learning with Errors problem. We use the standard lattice notation in [24,26].

Definition 1 (Lattice). *Let $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^d$ be linearly independent vectors. The lattice generated by these vectors is*

$$\mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{d} z_i \mathbf{b}_i \;\middle|\; z_i \in \mathbb{Z} \right\}, \tag{1}$$

where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_d] \in \mathbb{R}^{d \times d}$ is a basis of the lattice.

For a lattice basis $\mathbf{B}$, the Gram–Schmidt orthogonalization (GSO) produces orthogonal vectors $\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*$ defined by

$$\begin{aligned} \mathbf{b}_i^* &= \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \\ \mu_{i,j} &= \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}. \end{aligned} \tag{2}$$

The determinant of $\mathcal{L}(\mathbf{B})$ is

$$\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})| = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|. \tag{3}$$

The determinant is independent of the choice of basis.

For $1 \leq t \leq d$, let π_t denote the orthogonal projection onto the orthogonal complement of $\mathrm{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{t-1})$. The projected lattice $\pi_t(\mathcal{L}(\mathbf{B}))$ has dimension $\eta = d - t + 1$, and its determinant satisfies

$$\det\left(\pi_t(\mathcal{L}(\mathbf{B}))\right) = \prod_{i=t}^{d} \|\mathbf{b}_i^*\|. \tag{4}$$

The Gaussian heuristic (GH) is commonly used to estimate the length of a shortest nonzero vector in a random lattice [24,26]. For a d-dimensional lattice $\mathcal{L}$, it is given by

$$\begin{aligned} \mathrm{GH}(\mathcal{L}) &= \frac{\Gamma(d/2+1)^{1/d}}{\sqrt{\pi}} \det(\mathcal{L})^{1/d} \\ &\approx \sqrt{\frac{d}{2\pi e}} \det(\mathcal{L})^{1/d}, \end{aligned} \tag{5}$$

where Γ denotes the Gamma function. The Gaussian heuristic gives an estimate rather than an exact bound.

Definition 2 (Shortest Vector Problem (SVP)). *Given a basis* $\mathbf{B}$ *of a lattice* $\mathcal{L}$*, the Shortest Vector Problem asks to find a nonzero vector* $\mathbf{v} \in \mathcal{L}$ *such that*

$$\|\mathbf{v}\| = \lambda_1(\mathcal{L}), \tag{6}$$

where $\lambda_1(\mathcal{L})$ *denotes the length of a shortest nonzero vector in* $\mathcal{L}$*.*

SVP is NP-hard to approximate within polynomial factors [4,5,6], and the unique-SVP and bounded-distance-decoding problems that arise in lattice attacks are hard as well [7].

Definition 3 (Search Learning with Errors (Search-LWE)). *Let* n, m *be positive integers, let* $q \geq 2$ *be an integer, and let* χ_s *and* χ_e *be distributions for the secret and error, respectively. Given a uniformly sampled matrix* $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ *and a vector* $\mathbf{b} \in \mathbb{Z}_q^m$ *satisfying*

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}, \tag{7}$$

where $\mathbf{s} \in \mathbb{Z}_q^n$ *is sampled according to* χ_s *and* $\mathbf{e} \in \mathbb{Z}^m$ *is sampled according to* χ_e*, the search-LWE problem is to recover* $\mathbf{s}$ *from the given instance* $(\mathbf{A}, \mathbf{b}, q)$ *[3].*

In this paper, we consider the small-secret setting used in [14], where both the secret and error vectors have small entries.

2.2 Lattice Reduction

Lattice reduction algorithms aim to find a shorter and more orthogonal basis of a given lattice. The Lenstra–Lenstra–Lovász (LLL) algorithm [8] runs in polynomial time and is commonly used as a preprocessing step.

The Block Korkine–Zolotarev (BKZ) algorithm [16,17] extends LLL by applying an SVP solver to local blocks of the lattice basis. For a given block size β, BKZ calls an SVP solver on projected sublattices of dimension at most β and updates the basis using the short vectors found. A complete pass over the basis is called a BKZ tour. In general, a larger block size produces a better reduced basis but requires more computational time.

The quality of a reduced basis is commonly measured by the root Hermite factor.

Definition 4 (Root Hermite Factor). *Let* $\mathbf{B}$ *be a basis of a* d*-dimensional lattice* $\mathcal{L}$*. The root Hermite factor (RHF) of* $\mathbf{B}$ *is defined as*

$$\delta(\mathbf{B}) = \left(\frac{\|\mathbf{b}_1\|}{\det(\mathcal{L})^{1/d}} \right)^{1/d}. \tag{8}$$

A smaller root Hermite factor indicates a shorter first basis vector relative to the lattice determinant. For BKZ reduction with block size β, the root Hermite factor is commonly estimated as [17,24,26]

$$\delta(\beta) = \left(\frac{\beta}{2\pi e} (\pi\beta)^{1/\beta} \right)^{1/(2(\beta-1))}. \tag{9}$$

The Gram–Schmidt norms of a reduced basis can be approximated using the Geometric Series Assumption (GSA) [26].

Heuristic 1 (Geometric Series Assumption). *Let* $\mathbf{B}$ *be a lattice basis after lattice reduction. The Gram–Schmidt norms are assumed to approximately form a geometric sequence:*

$$\|\mathbf{b}_{i+1}^*\| \approx \rho \|\mathbf{b}_i^*\|, \qquad 1 \le i < d, \quad 0 < \rho < 1. \tag{10}$$

Combining the GSA with the root Hermite factor gives

$$\rho = \delta(\mathbf{B})^{-2d/(d-1)} \approx \delta(\mathbf{B})^{-2}. \tag{11}$$

Therefore, under the GSA, the logarithmic Gram–Schmidt profile $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)$ is approximately linear, with slope $\log_2 \rho$.

In practice, BKZ can also be performed with progressively increasing block sizes. Progressive BKZ [18] uses smaller blocks in the early stages and larger blocks in later stages. PnJBKZ [24,25] uses sieving-based SVP solvers during reduction, while BLASter [28] provides an efficient implementation of lattice reduction. AH-BKZ [30] combines BKZ with asynchronous hybrid processing to reduce the cost of the local SVP calls.

The Gram–Schmidt profile obtained after a finite number of BKZ tours may differ from the GSA prediction. In Section 4, we use the Gram–Schmidt profile produced by a given reduction schedule to estimate the quality of the reduced basis.

2.3 Lattice Sieving Algorithms

Lattice sieving is a method for finding short vectors in a lattice. A sieving algorithm maintains a list of lattice vectors and repeatedly combines vectors to obtain shorter ones [9,20,32,33]. Unlike lattice enumeration in the style of Kannan [19], sieving algorithms generally require exponential memory.

The BGJ sieve uses bucketing techniques to reduce the number of vector pairs that need to be checked. Vectors are assigned to buckets according to filtering conditions, and the search for reducing pairs is performed within these buckets.

Progressive sieving [21] and the "dimensions for free" technique [34] further reduce the cost of a sieve by reusing short vectors across dimensions.

In this paper, we use the optimized BGJ sieve proposed by Zhao, Ding, and Yang [35]. Their implementation improves the efficiency of sieving by reducing random memory accesses. In particular, the BGJ sieve uses

$$2^{0.2075\eta+o(\eta)} \tag{12}$$

streaming main-memory accesses in dimension η under the heuristic analysis of [35]. This expression counts streaming memory accesses; it does not directly give the total running time. A disk-based variant of this sieve extends the approach to larger lattice records [29].

In lattice-based attacks, sieving can be used as an SVP solver within BKZ reduction or as an independent SVP solver after lattice reduction. For a reduced basis $\mathbf{B}$ of dimension d and a final sieving dimension η, the terminal sieve is applied to

$$\pi_{d-\eta+1}(\mathcal{L}(\mathbf{B})). \tag{13}$$

The vectors returned by the sieve can then be lifted to the full lattice to recover the target vector. The corresponding running time and success probability are estimated in Sections 4 and 5.

2.4 LWE with Hints

Dachman-Soled et al. [12] introduced a framework that represents additional information about the LWE secret vector as hints. Following [12,14], we consider the following four types of hints:

- **Perfect hint:** A tuple $\bar{\mathbf{v}} = (\mathbf{v}, l) \in \mathbb{Z}^n \times \mathbb{Z}$ with $\langle \mathbf{v}, \mathbf{s} \rangle = l$.
- **Approximate hint:** A tuple $\bar{\mathbf{v}} = (\mathbf{v}, l) \in \mathbb{Z}^n \times \mathbb{Z}$ with $\langle \mathbf{v}, \mathbf{s} \rangle = l + \varepsilon$, where ε denotes the hint error.
- **Modular hint:** A tuple $\bar{\mathbf{v}} = (\mathbf{v}, l, m_i) \in \mathbb{Z}^n \times \mathbb{Z} \times \mathbb{N}$ with $\langle \mathbf{v}, \mathbf{s} \rangle \equiv l \pmod{m_i}$.
- **Mod-q hint:** A modular hint $\bar{\mathbf{v}} = (\mathbf{v}, l, m_i)$ is called a mod-q hint if $m_i = q$.

Side-channel attacks can reveal partial information about the LWE secret. Bos et al. [1] analyzed a single power trace of secret-dependent multiplication in Frodo. This leakage can be used to obtain approximate hints. Albrecht, Deo, and Paterson [2] studied cold-boot attacks that recover noisy secret-key data from memory. A recovered secret coefficient gives a perfect hint, while a noisy estimate

can give an approximate hint. Hamming-weight leakage can provide modular and approximate hints. A known linear relation modulo q gives a mod-q hint.

Cao, Jiang, and Cheng (CJC) [14] proposed a method that converts hints into a $\mathbb{Z}$-Small Integer Solution ($\mathbb{Z}$-SIS) problem and then combines the resulting lattice with the LWE samples. Recent work exploits the structure of the secret, such as its low Hamming weight, to recover it from perfect and approximate hints [31].

Definition 5 ($\mathbb{Z}$-Small Integer Solution ($\mathbb{Z}$-SIS)). *Given a matrix $\mathbf{V} \in \mathbb{Z}^{n\times k}$, a vector $\mathbf{l} \in \mathbb{Z}^k$, and a bound $\nu > 0$, the $\mathbb{Z}$-SIS problem asks to find a vector $\mathbf{x} \in \mathbb{Z}^n$ such that*

$$\mathbf{V}^T\mathbf{x} = \mathbf{l}, \qquad \|\mathbf{x}\| \le \nu. \tag{14}$$

For k perfect hints $(\mathbf{v}_i, l_i)$, let $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ and $\mathbf{l} = (l_1, \ldots, l_k)^T$. The secret vector satisfies $\mathbf{V}^T\mathbf{s} = \mathbf{l}$. CJC use Gaussian elimination to construct a $\mathbb{Z}$-SIS lattice with a short target vector. The construction also supports approximate and modular hints, as described in [14].

The $\mathbb{Z}$-SIS lattice is then combined with the LWE samples to obtain a lattice $\mathcal{L}_H$ containing the target vector. Let n_0 denote the remaining secret dimension after processing mod-q hints, and let k_p and k_a denote the numbers of independent perfect hints and approximate hints, respectively. Under the rank conditions of the CJC construction, the dimension of $\mathcal{L}_H$ is

$$d_H = n_0 + m - k_p + 1, \tag{15}$$

where m is the number of LWE samples. The corresponding target vector is

$$\mathbf{w}_H = \begin{pmatrix} -\mathbf{e} \\ \boldsymbol{\varepsilon} \\ \mathbf{s}_{\text{rem}} \\ -1 \end{pmatrix}, \tag{16}$$

where $\mathbf{e}$ is the LWE error vector, $\boldsymbol{\varepsilon}$ contains the approximate-hint errors, and $\mathbf{s}_{\text{rem}}$ contains the remaining secret coordinates.

Let $\mathbf{B}_{\text{SIS}}$ be the basis obtained from the hints. According to [14], the determinant of the combined lattice satisfies

$$D_H = \det(\mathcal{L}_H) = q^m |\det(\mathbf{B}_{\text{SIS}})|. \tag{17}$$

The dimension, determinant, and target vector of $\mathcal{L}_H$ are used in our solving strategy and two-step estimator.

3 Optimized Solving Strategy for LWE with Hints

We optimize BKZ schedules and final-sieving configurations for LWE with hints. The method lists admissible strategies, estimates their secret-recovery probabilities, expected running times, and peak memory, and selects the feasible strategy with the lowest expected running time. Table 1 lists the symbols used in this procedure.

Table 1. Notation for the four-step strategy selection procedure.

Symbol	Meaning
$\mathcal{I}$; $\mathcal{L}_H$	LWE instance with hints; constructed attack lattice.
(R, S)	Reduction and final-sieving algorithms.
Ω_{BKZ}; Ω_{sieve}	Admissible BKZ schedules; admissible final-sieving configurations.
$\boldsymbol{\xi}$; (η, θ_S, r)	BKZ schedule; final-sieving configuration.
x; $\mathcal{X}$; $\mathcal{F}$	Strategy; candidate set; feasible strategy set.
$\widehat{p}_x$	Estimated secret-recovery probability.
$\widehat{T}_x$; $\widehat{M}_x$	Expected running time; peak memory.
p_0; $M_{\max}$; x^*	Success threshold; memory limit; selected strategy.

Using these symbols, Fig. 2 shows the four steps of the selection procedure. A candidate is discarded if its estimated success probability is below the threshold or its resource requirements exceed the limits.

The following subsections describe each step. Section 4 gives the success probability estimates, while Section 5 gives the cost models. The concrete search procedure is described in Section 5.2.

3.1 Step 1: List Candidate Strategies

The attack lattice is constructed from the hints before the strategy parameters are selected. Let $\mathcal{I} = (\mathbf{A}, \mathbf{b}, q, H)$ be an LWE instance with hints. We use the CJC construction from Section 2.4 to obtain the attack lattice

$$\mathcal{L}_H = (\mathbf{B}_H, d_H, D_H, \mathcal{D}_H, \mathrm{Rec}_H),$$

where $\mathbf{B}_H$ is a lattice basis, d_H and D_H are its dimension and determinant, $\mathcal{D}_H$ describes the target distribution conditioned on the hints, and Rec_H denotes secret reconstruction. This lattice remains fixed while candidate strategies are listed.

For a fixed algorithm pair (R, S), let Ω_{BKZ} be the set of admissible BKZ reduction schedules and let Ω_{sieve} be the set of admissible final-sieving configurations. These two sets are determined by the prescribed upper bounds on the BKZ and sieving parameters, respectively. A candidate strategy pairs one BKZ schedule with one final-sieving configuration.

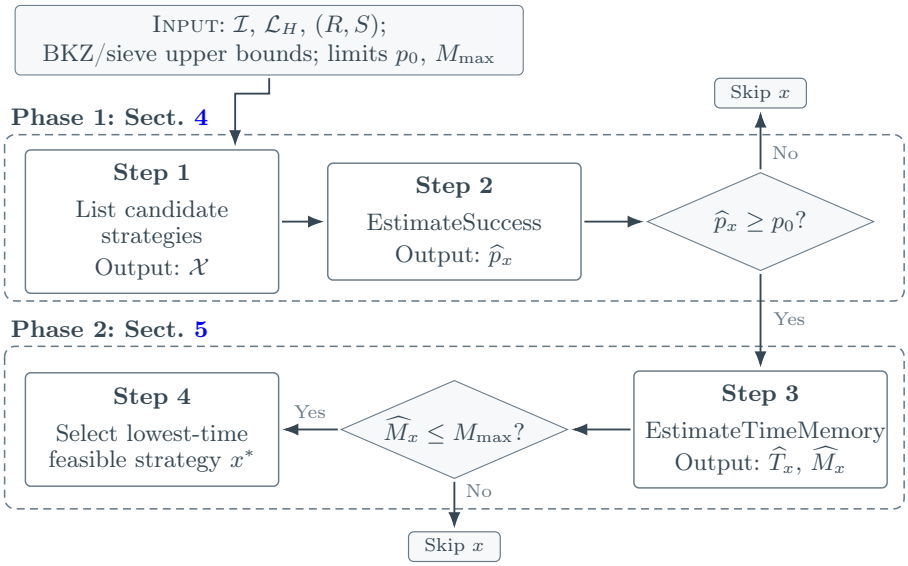

Fig. 2. Four-step strategy selection: list parameter combinations, estimate success probability, time and memory, and select the feasible strategy with the lowest expected time.

Definition 6 (Solving strategy). *For a fixed attack lattice and algorithm pair (R, S), a solving strategy is*

$$x = (R, S, \boldsymbol{\xi}, \eta, \theta_S, r).$$

Here, $\boldsymbol{\xi}$ is the BKZ reduction schedule, η is the dimension of the projected sublattice, θ_S contains the final search and lifting parameters, and r is the maximum number of final search attempts.

The BKZ reduction schedule consists of K stages:

$$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K), \qquad \mathbf{B}_j = \mathcal{R}_R(\mathbf{B}_{j-1}; \xi_j), \quad 1 \leq j \leq K, \tag{18}$$

where $\mathbf{B}_0 = \mathbf{B}_H$ and $\xi_j = (\beta_j, \tau_j, \vartheta_{R,j})$ specifies the block size, number of tours, and other reduction parameters. A final-sieving configuration is a tuple $(\eta, \theta_S, r) \in \Omega_{\text{sieve}}$.

The candidate set is the Cartesian product of the two parameter sets:

$$\mathcal{X}(\mathcal{I}; R, S) = \{(R, S, \boldsymbol{\xi}, \eta, \theta_S, r) : \boldsymbol{\xi} \in \Omega_{\text{BKZ}},\ (\eta, \theta_S, r) \in \Omega_{\text{sieve}}\}.$$

If reduction-only strategies are included, we also list $(R, \varnothing, \boldsymbol{\xi}, 0, \varnothing, 0)$ for each $\boldsymbol{\xi} \in \Omega_{\text{BKZ}}$.

Algorithm 1: List Candidate Strategies

Input: Algorithms (R, S); upper bounds on BKZ and final-sieving parameters
Output: Candidate strategy set $\mathcal{X}$

$\mathcal{X} \leftarrow \varnothing$;
foreach *BKZ schedule* $\boldsymbol{\xi} \in \Omega_{\mathrm{BKZ}}$ **do**
  foreach *final-sieving configuration* $(\eta, \theta_S, r) \in \Omega_{\mathrm{sieve}}$ **do**
    $x \leftarrow (R, S, \boldsymbol{\xi}, \eta, \theta_S, r)$;
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$;
  end
end
if *reduction-only strategies are included* **then**
  foreach *BKZ schedule* $\boldsymbol{\xi} \in \Omega_{\mathrm{BKZ}}$ **do**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(R, \varnothing, \boldsymbol{\xi}, 0, \varnothing, 0)\}$;
  end
end
return $\mathcal{X}$;

Thus, Step 1 only lists the admissible BKZ–sieving parameter combinations. It does not evaluate their success probability or resource requirements; these are considered in Steps 2 and 3.

3.2 Step 2: Estimate Secret Recovery Probability

For each candidate x, we estimate the probability of recovering the secret. The estimator is based on the two-step method developed in Section 4. Its only output at the strategy-selection level is the estimated success probability $\widehat{p}_x$.

Definition 7 (Strategy success probability). *Let* $\mathrm{Solve}_x(\mathcal{I}; \omega)$ *be the verified secret returned by strategy* x*, where* ω *contains the algorithmic randomness. For the target secret* $\mathbf{s}_H$*, define*

$$p_{\mathrm{succ}}(x \mid \mathcal{I}) = \Pr\big[\mathrm{Solve}_x(\mathcal{I}; \omega) = \mathbf{s}_H \mid \mathcal{I}\big].$$

The probability includes the target distribution conditioned on the hints and the randomness of the attack. We denote its estimate by $\widehat{p}_x = \widehat{p}_{\mathrm{succ}}(x \mid \mathcal{I})$.

The attack can recover the secret during BKZ reduction or during the final search. Let p_{red} be the probability of recovery during reduction and let $p_{\mathrm{term|red\ failure}}$ be the final-search recovery probability conditioned on reduction failure. Then

$$p_{\mathrm{succ}} = p_{\mathrm{red}} + (1 - p_{\mathrm{red}}) p_{\mathrm{term|red\ failure}}.$$

The final-search probability also includes successful lifting and verification. For its estimation, we use a heuristic approximation: given the instance $\mathcal{I}$ and the final reduced basis $\mathbf{B}$, we do not model how unsuccessful BKZ recovery changes the target distribution. Thus, we approximate the final-search recovery probability conditioned on BKZ failure by a probability computed from the target

distribution conditioned only on $\mathcal{I}$ and $\mathbf{B}$. This is a modeling assumption, not an independence result. Let $t = d_H - \eta + 1$ and let $F_{H,\mathbf{B},t}$ denote this model distribution of the projected target. The function $K_S(\mathbf{z}; \mathbf{B}, t, \theta_S)$ models the probability that the sieve returns $\mathbf{z}$, and $h_S(\mathbf{z}; \mathbf{B}, t, \theta_S)$ models successful lifting and verification once $\mathbf{z}$ is found. The estimated probability for one search attempt is

$$p_S^{(1)}(\mathbf{B}, t, \theta_S) = \int K_S(\mathbf{z}; \mathbf{B}, t, \theta_S) h_S(\mathbf{z}; \mathbf{B}, t, \theta_S)\, dF_{H,\mathbf{B},t}(\mathbf{z}). \tag{19}$$

For repeated attempts, let $s_S(\mathbf{w}, \mathbf{B})$ be the conditional recovery probability of one attempt for a fixed target and basis. If the solver uses independent randomness in each attempt, the probability within r attempts is

$$p_S^{(r)}(\mathbf{B}) = \mathbb{E}_{\mathbf{w}_H|\mathcal{I},\mathbf{B}}\left[1 - (1 - s_S(\mathbf{w}_H, \mathbf{B}))^r\right], \tag{20}$$

where the expectation uses the target model conditioned on $\mathcal{I}$ and $\mathbf{B}$, without additionally conditioning on BKZ failure. The formula assumes independent solver randomness for a fixed target and basis, not independent recovery events after averaging over targets. Section 4 develops the GSO, projected-target, sieve coverage, and lifting estimates used to evaluate these expressions.

The estimator returns only

$$\widehat{p}_x \leftarrow \text{EstimateSuccess}(\mathcal{L}_H, x).$$

The conditional recovery probabilities for individual BKZ stages and final search attempts are internal quantities of the success model. They are evaluated again in Step 3 to determine the probability of reaching each operation; no separate summary variable is required.

Algorithm 2: Estimate Secret Recovery Probability

Input: Attack lattice $\mathcal{L}_H$; candidate $x = (R, S, \boldsymbol{\xi}, \eta, \theta_S, r)$
Output: Estimated success probability $\widehat{p}_x$

$\widehat{\boldsymbol{\ell}} \leftarrow \text{PredictGSO}(\mathcal{L}_H, R, \boldsymbol{\xi})$;
$\widehat{p}_{\text{red}} \leftarrow \text{EstimateBKZRecovery}(\mathcal{L}_H, R, \boldsymbol{\xi}, \widehat{\boldsymbol{\ell}})$;
if $S = \varnothing$ **then**
  $\widehat{p}_x \leftarrow \widehat{p}_{\text{red}}$;
else
  $\widehat{p}_{\text{term}} \leftarrow \text{EstimateFinalRecovery}(\mathcal{L}_H, x, \widehat{\boldsymbol{\ell}})$;
  $\widehat{p}_x \leftarrow \widehat{p}_{\text{red}} + (1 - \widehat{p}_{\text{red}})\widehat{p}_{\text{term}}$;
end
return $\widehat{p}_x$;

The function EstimateBKZRecovery uses the algorithm R and the predicted BKZ profiles to estimate the conditional recovery probabilities $\widehat{q}_{\text{red},j}$ of the individual stages. The function EstimateFinalRecovery estimates final sieving, lifting, and verification success using the target model described above. These conditional probabilities are also used in Step 3 to calculate operation reach probabilities.

3.3 Step 3: Estimate Time and Memory

For each candidate that satisfies the success threshold, we estimate its expected total running time and peak memory. The running time covers one execution from the LWE instance through lattice construction, BKZ reduction, and any final sieving, lifting, and verification, until the secret is recovered or the configured search attempts are exhausted. It accounts for early recovery: an operation contributes to the expected time only with the probability that the execution reaches it.

Definition 8 (Expected running time and peak memory). *For a strategy x, let T_x be the running time of one execution and let M_x be its peak memory use. The estimated expected running time and peak memory are denoted by $\widehat{T}_x$ and $\widehat{M}_x$, respectively. The expectation is taken over the target distribution conditioned on $\mathcal{I}$ and the randomness of the attack.*

We use the interface

$$(\widehat{T}_x, \widehat{M}_x) \leftarrow \text{EstimateTimeMemory}(\mathcal{I}, \mathcal{L}_H, x).$$

Let $\widehat{t}_{\text{CJC}}$ and $\widehat{M}_{\text{CJC}}$ be the estimated time and memory for constructing the attack lattice. For BKZ stage j, let $\widehat{t}_{R,j}$ and $\widehat{M}_{R,j}$ be its estimated time and memory. For final search attempt a, let $\widehat{t}_{S,a}$ and $\widehat{M}_{S,a}$ be the corresponding estimates. The time $\widehat{t}_{S,a}$ is the expected time of one attempt, including sieving and the probability-weighted lifting and verification steps.

The conditional recovery probabilities from Section 4 determine the probability that each operation is reached. Let $\widehat{q}_{\text{red},j}$ be the conditional recovery probability at BKZ stage j. Let $\widehat{q}_{S,a}$ be the conditional recovery probability on final search attempt a, given that BKZ reduction and all earlier final-search attempts have failed. The reach probabilities are

$$\begin{aligned} \widehat{\pi}_{\text{red},j} &= \prod_{k=1}^{j-1} (1 - \widehat{q}_{\text{red},k}), \\ \widehat{\pi}_{S,a} &= (1 - \widehat{p}_{\text{red}}) \prod_{h=1}^{a-1} (1 - \widehat{q}_{S,h}). \end{aligned} \tag{21}$$

An empty product equals one. Lattice construction is always reached, so its reach probability is one. The expected total running time is

$$\widehat{T}_x = \widehat{t}_{\text{CJC}} + \sum_{j=1}^{K} \widehat{\pi}_{\text{red},j} \widehat{t}_{R,j} + \sum_{a=1}^{r} \widehat{\pi}_{S,a} \widehat{t}_{S,a}, \tag{22}$$

where the final sum is absent for a reduction-only strategy. The peak memory is

$$\widehat{M}_x = \max \left\{ \widehat{M}_{\text{CJC}}, \max_{1 \le j \le K} \widehat{M}_{R,j}, \max_{1 \le a \le r} \widehat{M}_{S,a} \right\}, \tag{23}$$

with the final-search term omitted when $S = \varnothing$.

Algorithm 3: Estimate Time and Memory

Input: LWE instance $\mathcal{I}$; attack lattice $\mathcal{L}_H$; candidate $x = (R, S, \boldsymbol{\xi}, \eta, \theta_S, r)$

Output: Expected running time $\widehat{T}_x$; peak memory $\widehat{M}_x$

Estimate $\widehat{t}_{\mathrm{CJC}}$ and $\widehat{M}_{\mathrm{CJC}}$ for lattice construction;
Estimate $\widehat{p}_{\mathrm{red}}$ and the stage probabilities $\widehat{q}_{\mathrm{red},j}$ using the BKZ recovery model;
for $j = 1, \ldots, K$ **do**
  Estimate $\widehat{t}_{R,j}$ and $\widehat{M}_{R,j}$ for BKZ stage j;
  $\widehat{\pi}_{\mathrm{red},j} \leftarrow \prod_{k=1}^{j-1}(1 - \widehat{q}_{\mathrm{red},k})$;
end
if $S \neq \varnothing$ **then**
  for $a = 1, \ldots, r$ **do**
    Estimate $\widehat{q}_{S,a}$, $\widehat{t}_{S,a}$, and $\widehat{M}_{S,a}$ for final search attempt a;
    $\widehat{\pi}_{S,a} \leftarrow (1 - \widehat{p}_{\mathrm{red}}) \prod_{h=1}^{a-1}(1 - \widehat{q}_{S,h})$;
  end
end
$\widehat{T}_x \leftarrow \widehat{t}_{\mathrm{CJC}} + \sum_{j=1}^{K} \widehat{\pi}_{\mathrm{red},j}\widehat{t}_{R,j} + \sum_{a=1}^{r} \widehat{\pi}_{S,a}\widehat{t}_{S,a}$;
$\widehat{M}_x \leftarrow \max\{\widehat{M}_{\mathrm{CJC}}, \max_j \widehat{M}_{R,j}\}$;
if $S \neq \varnothing$ **then**
  $\widehat{M}_x \leftarrow \max\{\widehat{M}_x, \max_{1 \le a \le r} \widehat{M}_{S,a}\}$;
end
return $(\widehat{T}_x, \widehat{M}_x)$;

For a reduction-only strategy, $S = \varnothing$ and $r = 0$, so no final-search term is included. The memory estimate is the maximum memory required by lattice construction, any BKZ stage, or any final-search attempt. The same success and time models are used for both reduction-only and BKZ-plus-sieving strategies.

3.4 Step 4: Select the Lowest-Time Feasible Strategy

The final step selects a strategy that meets the success-probability and memory requirements. We use the success estimate from Step 2 and the time and memory estimates from Step 3. This selection corresponds to Step 4 in Fig. 2.

Definition 9 (Optimized solving strategy). *Let $p_0 \in (0, 1)$ be the required recovery probability and $M_{\max}$ the memory limit. The feasible strategy set is*

$$\mathcal{F}(\mathcal{I}; R, S) = \left\{ x \in \mathcal{X}(\mathcal{I}; R, S) : \widehat{p}_x \geq p_0,\ \widehat{M}_x \leq M_{\max} \right\}. \tag{24}$$

We select

$$x^* \in \underset{x \in \mathcal{F}(\mathcal{I}; R, S)}{\arg\min}\ \widehat{T}_x.$$

If the feasible set is empty, no candidate meets the given requirements. The success constraint uses the heuristic estimate developed in Section 4; it does not certify an actual recovery probability of at least p_0.

The output specifies the BKZ schedule and, when used, the final sieving configuration. Algorithm 4 in Section 5.2 gives the corresponding selection procedure.

4 Phase 1: Strategy Listing and Success Estimation

Phase 1 of the framework in Fig. 2 lists candidate attack strategies and estimates their secret-recovery probabilities. We first specify the admissible parameter combinations and then estimate their success probabilities using the reduced-basis geometry and the conditional target distribution.

4.1 Strategy Listing

For a fixed LWE instance with hints, the CJC construction gives an attack lattice $\mathcal{L}_H$ with basis $\mathbf{B}_H$ and dimension d_H. The attack lattice and the algorithm pair (R, S) remain fixed while the BKZ reduction schedule and final-sieving configuration vary. Candidate listing therefore involves parameter enumeration rather than executing BKZ reduction or the final sieve.

The prescribed parameter bounds determine two finite sets. The set Ω_{BKZ} contains admissible BKZ schedules $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$, where each ξ_j specifies a block size β_j, a number of tours τ_j, and any additional reduction parameters. The set Ω_{sieve} contains admissible configurations (η, θ_S, r), specifying the projected dimension, the search and lifting parameters, and the maximum number of search attempts. The projection dimension η is restricted by the attack-lattice dimension d_H.

We list every pair consisting of a schedule $\boldsymbol{\xi} \in \Omega_{\mathrm{BKZ}}$ and a final-sieving configuration $(\eta, \theta_S, r) \in \Omega_{\mathrm{sieve}}$. Each pair defines a candidate $x = (R, S, \boldsymbol{\xi}, \eta, \theta_S, r)$ in $\mathcal{X}(\mathcal{I}; R, S)$. When reduction-only strategies are considered, the corresponding configurations with $S = \varnothing$ are included as well. The candidate set and its enumeration procedure are defined in Section 3.1. Success probability and memory requirements are evaluated after enumeration, not used to construct the parameter sets.

4.2 Conditional Distribution of the Projected Target Vector

The success estimate for each candidate depends on the lattice basis obtained after its BKZ schedule and, when final sieving is used, the selected projection dimension. We first predict the Gram–Schmidt profile of the reduced basis, then use it to describe the projected lattice and the distribution of the target vector.

The final sieve operates on a projected sublattice of the reduced basis. Its geometry depends on the Gram–Schmidt norms. We therefore predict the profile produced by each candidate BKZ schedule.

Let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$ be the reduction schedule defined in Section 3.1, with $\mathbf{B}_0 = \mathbf{B}_H$. For the basis $\mathbf{B}_j$ obtained after stage j, let $g_i^{(j)} = \|(\mathbf{b}_i^{(j)})^*\|$ denote its Gram–Schmidt norms and $\ell_i^{(j)} = \log_2 g_i^{(j)}$ the corresponding logarithmic profile. Since BKZ reduction preserves the lattice determinant, the profile satisfies

$$\sum_{i=1}^{d_H} \ell_i^{(j)} = \log_2 D_H.$$

For a fixed block size β, the GSA introduced in Section 2.2 provides an approximation of the Gram–Schmidt profile after BKZ reduction. Under this assumption, consecutive Gram–Schmidt norms satisfy $g_{i+1}/g_i \approx \rho_\beta$, where

$$\rho_\beta \approx \delta(\beta)^{-2d_H/(d_H-1)}, \qquad \log_2 \rho_\beta \approx -\frac{2d_H}{d_H-1} \log_2 \delta(\beta). \tag{25}$$

Here, $\delta(\beta)$ denotes the root Hermite factor estimated in (9). Consequently, the logarithmic Gram–Schmidt profile can be approximated by

$$\begin{aligned} \ell_i &\approx \ell_1 + (i-1)\log_2 \rho_\beta, \\ \ell_1 &= \frac{\log_2 D_H}{d_H} - \frac{d_H - 1}{2} \log_2 \rho_\beta. \end{aligned} \tag{26}$$

For a multi-stage reduction schedule, we use a BKZ simulator to predict the Gram–Schmidt profile after each stage. Starting from the initial profile $\widehat{\boldsymbol{\ell}}^{(0)}$, the predicted profile is updated according to

$$\widehat{\boldsymbol{\ell}}^{(j)} = \mathrm{Sim}_R\left(\widehat{\boldsymbol{\ell}}^{(j-1)}, \xi_j; H\right), \qquad 1 \le j \le K. \tag{27}$$

The resulting profile $\widehat{\boldsymbol{\ell}}^{(K)}$ represents the predicted Gram–Schmidt norms after the complete reduction schedule. The simulation takes into account the block sizes, tour counts, and additional reduction parameters specified by $\boldsymbol{\xi}$. The predicted profile also satisfies the determinant constraint

$$\sum_{i=1}^{d_H} \widehat{\ell}_i^{(K)} = \log_2 D_H.$$

For a final sieving dimension η, let $t = d_H - \eta + 1$ be the starting index of the projected sublattice. By (4) and (5), its Gaussian heuristic is estimated as

$$\widehat{G}_t(\boldsymbol{\xi}, \eta) = \frac{\Gamma(\eta/2+1)^{1/\eta}}{\sqrt{\pi}} \left(\prod_{i=t}^{d_H} \widehat{g}_i^{(K)}\right)^{1/\eta},$$

where $\widehat{g}_i^{(K)} = 2^{\widehat{\ell}_i^{(K)}}$. Thus, the reduction schedule determines the Gram–Schmidt profile used to estimate the projected lattice, while the final sieving dimension determines which part of the profile is considered. The resulting Gaussian heuristic provides the lattice-length scale used to evaluate the final search in Section 4.3.

With the projected-lattice geometry determined, we now model the projected target vector itself. The distribution of its norm then gives the probability that the target lies within the search radius. Let $t = d_H - \eta + 1$, as in Section 3.2. Write $\mathbf{Q}_t$ for a matrix whose columns form an orthonormal basis of the projection space. The projected target vector and its Euclidean norm are

$$\mathbf{z}_t = \mathbf{Q}_t^T \mathbf{w}_H, \qquad X_t = \|\mathbf{z}_t\|.$$

The distribution of the projected vector depends on the conditional target distribution and on the projection defined by the reduced basis. Let

$$\boldsymbol{\mu}_{H,\mathbf{B}} = \mathbb{E}[\mathbf{w}_H \mid \mathcal{I}, \mathbf{B}], \qquad \boldsymbol{\Sigma}_{H,\mathbf{B}} = \mathrm{Cov}(\mathbf{w}_H \mid \mathcal{I}, \mathbf{B})$$

denote the conditional mean and covariance matrix of the target vector.

Proposition 1 (Second moment of the projected target). *For a fixed reduced basis* $\mathbf{B}$ *and a target distribution with finite second moments, the expected squared length of the projected target vector is*

$$\mathbb{E}[X_t^2 \mid \mathcal{I}, \mathbf{B}] = \|\mathbf{Q}_t^T \boldsymbol{\mu}_{H,\mathbf{B}}\|^2 + \mathrm{tr}\left(\mathbf{Q}_t^T \boldsymbol{\Sigma}_{H,\mathbf{B}} \mathbf{Q}_t\right). \tag{28}$$

Proof. The projected vector $\mathbf{z}_t$ has conditional mean $\mathbf{Q}_t^T \boldsymbol{\mu}_{H,\mathbf{B}}$ and covariance matrix $\mathbf{Q}_t^T \boldsymbol{\Sigma}_{H,\mathbf{B}} \mathbf{Q}_t$. Applying the identity $\mathbb{E}[\|\mathbf{z}\|^2] = \|\mathbb{E}[\mathbf{z}]\|^2 + \mathrm{tr}(\mathrm{Cov}(\mathbf{z}))$ gives the result.

We define the effective standard deviation of the projected target by

$$\sigma_{\mathrm{eff}} = \sqrt{\frac{\mathbb{E}[X_t^2 \mid \mathcal{I}, \mathbf{B}]}{\eta}}. \tag{29}$$

Thus, $\sqrt{\eta}\sigma_{\mathrm{eff}}$ is the root-mean-square norm of the projected target vector. The vector model also gives the distribution of this norm, which we use to calculate the probability that the target lies within a specified search radius.

To estimate this probability, we consider a Gaussian approximation to the conditional target distribution,

$$\mathbf{w}_H \mid \mathcal{I}, \mathbf{B} \sim \mathcal{N}(\boldsymbol{\mu}_{H,\mathbf{B}}, \boldsymbol{\Sigma}_{H,\mathbf{B}}).$$

Under this model, the projected target follows

$$\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\nu}_t, \mathbf{V}_t),$$

where

$$\boldsymbol{\nu}_t = \mathbf{Q}_t^T \boldsymbol{\mu}_{H,\mathbf{B}}, \qquad \mathbf{V}_t = \mathbf{Q}_t^T \boldsymbol{\Sigma}_{H,\mathbf{B}} \mathbf{Q}_t.$$

Let $\mathbf{V}_t = \mathbf{U}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_\eta)\mathbf{U}^T$ be its eigendecomposition. The squared norm of the projected target vector is then distributed as

$$X_t^2 \overset{d}{=} \sum_{j=1}^{\eta} \left(\sqrt{\lambda_j} Z_j + (\mathbf{U}^T \boldsymbol{\nu}_t)_j\right)^2, \qquad Z_j \sim \mathcal{N}(0,1),$$

where the Z_j are independent standard normal random variables. The conditional mean and covariance matrix determine the projected vector distribution and, in turn, the distribution of its norm under the Gaussian model. In the centered isotropic case, where $\boldsymbol{\nu}_t = \mathbf{0}$ and $\mathbf{V}_t = \sigma_H^2 \mathbf{I}_\eta$, it simplifies to

$$\frac{X_t^2}{\sigma_H^2} \sim \chi_\eta^2.$$

For a search radius $R > 0$, let

$$p_{\text{in}}(R \mid \mathcal{I}, \mathbf{B}) = \Pr[X_t \leq R \mid \mathcal{I}, \mathbf{B}]$$

denote the probability that the projected target lies within the corresponding Euclidean ball. This probability can be evaluated using the distribution of X_t^2. When only its first two moments are available, we can obtain a lower bound using Cantelli's inequality.

In particular, let

$$u_t = \mathbb{E}[X_t^2 \mid \mathcal{I}, \mathbf{B}] = \operatorname{tr}(\mathbf{V}_t) + \|\boldsymbol{\nu}_t\|^2$$

and

$$v_t = \operatorname{Var}(X_t^2 \mid \mathcal{I}, \mathbf{B}) = 2\operatorname{tr}(\mathbf{V}_t^2) + 4\boldsymbol{\nu}_t^T \mathbf{V}_t \boldsymbol{\nu}_t.$$

For a required probability $p_g \in (0, 1)$, Cantelli's inequality gives

$$R^2 \geq u_t + \sqrt{\frac{p_g}{1 - p_g} v_t} \quad \Longrightarrow \quad p_{\text{in}}(R \mid \mathcal{I}, \mathbf{B}) \geq p_g.$$

When samples from the conditional target distribution are available, the distribution of the projected target vector can be estimated from the corresponding projected samples. The probability that its norm is below a given radius can then be estimated from those samples.

Section 4.3 combines the resulting inclusion probability with the sieve coverage and lifting models to estimate secret recovery.

4.3 Two-Step Success Probability Estimation

The two-step estimator evaluates secret recovery during BKZ reduction and the subsequent final sieving and lifting. It uses the reduced-basis profile and the conditional distribution of the projected target vector developed in Section 4.2 to estimate the corresponding recovery probabilities. We analyze these two attack steps in turn.

Recovery during BKZ reduction. Let $E_{\text{red},j}$ denote the event that the correct secret is recovered and verified during the j-th BKZ stage. We define the conditional recovery probability as

$$q_{\text{red},j} = \Pr\left[E_{\text{red},j} \mid \mathcal{I}, \bigcap_{k=1}^{j-1} E_{\text{red},k}^c\right].$$

For a reduction schedule $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$, the probability of recovering the secret during reduction is

$$p_{\mathrm{red}} = 1 - \prod_{j=1}^{K} (1 - q_{\mathrm{red},j}).$$

The stage probabilities are estimated from the recovery model of the selected BKZ implementation, using the reduction parameters and the predicted Gram–Schmidt profiles. Their estimates determine $\widehat{p}_{\mathrm{red}}$, the reduction-stage success probability used in strategy evaluation.

Recovery through the final sieve. If reduction does not recover the secret, the final sieve is applied to the projected sublattice $\pi_t(\mathcal{L}(\mathbf{B}_K))$, where $t = d_H - \eta + 1$. The Gaussian heuristic of this sublattice is estimated from the predicted profile:

$$\widehat{G}_t(\boldsymbol{\xi}, \eta) = \frac{\Gamma(\eta/2+1)^{1/\eta}}{\sqrt{\pi}} \left(\prod_{i=t}^{d_H} \widehat{g}_i^{(K)} \right)^{1/\eta}.$$

We express the effective search radius of the final sieve as

$$R_S = \rho_S \widehat{G}_t(\boldsymbol{\xi}, \eta),$$

where ρ_S depends on the sieving algorithm and its configuration, including the search dimension, database size, and search budget.

Using the effective standard deviation defined in (29), the root-mean-square length of the projected target can be compared with this radius:

$$\sqrt{\eta}\, \sigma_{\mathrm{eff}} \le \rho_S \frac{\Gamma(\eta/2+1)^{1/\eta}}{\sqrt{\pi}} \left(\prod_{i=t}^{d_H} \widehat{g}_i^{(K)} \right)^{1/\eta}. \tag{30}$$

This condition compares the root-mean-square norm of the projected target with the modeled search radius. The probability that the target lies within this radius is obtained from the projected target-vector model.

Let $\mathbf{z}_t = \mathbf{Q}_t^T \mathbf{w}_H$ and $X_t = \|\mathbf{z}_t\|$, as defined in Section 4.2. We define the geometric inclusion event

$$E_g = \{0 < X_t \le R_S\}$$

and its model probability

$$p_g = \Pr[E_g \mid \mathcal{I}, \mathbf{B}_K].$$

We evaluate this probability using the target distribution from Section 4.2. In the final-search estimator, we use this distribution as a heuristic approximation to the distribution after unsuccessful BKZ recovery.

Under the centered isotropic Gaussian model, the inclusion probability is

$$p_g = F_{\chi^2_\eta}\left(\frac{R_S^2}{\sigma_H^2} \right),$$

where $F_{\chi^2_\eta}$ is the cumulative distribution function of the chi-squared distribution with η degrees of freedom. For a general Gaussian target distribution, the probability is evaluated using the quadratic-form distribution derived in Section 4.2.

The sieve coverage probability is described by the function K_S introduced in Section 3.2. We use the following approximation:

$$K_S(\mathbf{z}; \mathbf{B}_K, t, \theta_S) \approx q_S \, \mathbf{1}_{\{0<\|\mathbf{z}\|\le R_S\}},$$

where $q_S \in [0,1]$ is the conditional probability that the solver returns the projected target within the modeled search region. Together, ρ_S and q_S characterize the coverage of the selected sieving configuration.

In our experiments with the BGJ sieve, a projected target inside the modeled search region was recovered with high probability in a single attempt. We therefore use $q_S \approx 1$ as a heuristic approximation for this sieving configuration.

Lifting and secret recovery. Once the sieve returns the projected target vector, the corresponding vector in the original lattice is reconstructed by lifting. We consider nearest-plane lifting using the first $t-1$ basis vectors.

Proposition 2 (Sufficient condition for nearest-plane lifting). *Suppose that the final sieve returns the projected target vector and its coordinates in the projected sublattice. Let*

$$\mathbf{q}_i = \frac{\mathbf{b}_i^*}{g_i}, \qquad g_i = \|\mathbf{b}_i^*\|.$$

If

$$|\langle \mathbf{w}_H, \mathbf{q}_i \rangle| < \frac{g_i}{2}, \qquad 1 \le i < t,$$

then nearest-plane lifting recovers the correct target vector, provided that the rounding operations are performed with sufficient precision.

Proof. The coordinates of the projected target determine the suffix coefficients of the target vector. The remaining coefficients are recovered in reverse order using the Gram–Schmidt vectors of the basis prefix. At step i, changing the corresponding integer coefficient changes the component along $\mathbf{q}_i$ by g_i. The stated inequality ensures that nearest-integer rounding selects the coefficient of the target vector. Repeating this process for $i = t-1, \ldots, 1$ recovers the full target vector.

Let

$$E_b = \bigcap_{i=1}^{t-1} \left\{ |\langle \mathbf{w}_H, \mathbf{q}_i \rangle| < \frac{g_i}{2} \right\}$$

denote the event that the sufficient lifting conditions hold.

Under the Gaussian model of Section 4.2, the i-th prefix coordinate has mean and variance

$$a_i = \mathbf{q}_i^T \boldsymbol{\mu}_{H,\mathbf{B}_K}, \qquad s_i^2 = \mathbf{q}_i^T \boldsymbol{\Sigma}_{H,\mathbf{B}_K} \mathbf{q}_i.$$

For $s_i > 0$, the probability that the corresponding lifting condition fails is

$$\epsilon_i = 1 - \left[\Phi\left(\frac{g_i/2 - a_i}{s_i}\right) - \Phi\left(\frac{-g_i/2 - a_i}{s_i}\right)\right].$$

When $s_i = 0$, the probability is determined by the corresponding fixed coordinate. For the target distribution conditioned on $\mathcal{I}$ and $\mathbf{B}_K$, the union bound gives

$$\Pr[E_g \cap E_b \mid \mathcal{I}, \mathbf{B}_K] \geq \max\left\{0,\, p_g - \sum_{i=1}^{t-1} \epsilon_i\right\}.$$

This bound is valid within the assumed target model. It is not a certified bound on the probability conditioned on BKZ failure. We use the following model-based lower estimate of the joint geometric and lifting probability:

$$\widehat{p}_{g,b} = \max\left\{0,\, \widehat{p}_g - \sum_{i=1}^{t-1} \widehat{\epsilon}_i\right\}.$$

In the centered isotropic case, the prefix failure probabilities simplify to

$$\epsilon_i = \operatorname{erfc}\left(\frac{g_i}{2\sqrt{2}\sigma_H}\right).$$

Combining the geometric inclusion probability, the lifting condition, and the sieve coverage model gives the following estimate for one final search attempt:

$$\widehat{p}_S^{(1)}(\mathbf{B}_K) = q_S\, \widehat{p}_{g,b}.$$

This approximation combines geometric inclusion, sieve coverage, and successful lifting. It gives a computable form of the final-search probability in (19).

For r attempts with independent solver randomness on the same reduced basis, the corresponding estimate is

$$\widehat{p}_S^{(r)}(\mathbf{B}_K) = \widehat{p}_{g,b}\left[1 - (1 - q_S)^r\right].$$

This is the uniform-coverage specialization of the retry model in (20).

Finally, combining BKZ reduction and the final search gives the estimated success probability of a strategy:

$$\widehat{p}_{\text{succ}}(x \mid \mathcal{I}) = \widehat{p}_{\text{red}} + (1 - \widehat{p}_{\text{red}})\widehat{p}_{\text{term|red failure}}. \tag{31}$$

In this estimate, $\widehat{p}_{\text{term|red failure}}$ is approximated by $\widehat{p}_S^{(r)}(\mathbf{B}_K)$, which does not condition the target distribution on BKZ failure. The two-stage probability decomposition is exact for the true conditional probabilities, but this substitution makes $\widehat{p}_{\text{succ}}$ a heuristic prediction, not a guaranteed lower bound on the actual recovery probability. We use this estimate for the success probability constraint in Definition 9.

4.4 Two-Step Estimation for Single-Block BKZ

The estimator in Sections 4.2 and 4.3 applies to multi-stage BKZ schedules. We now consider a single block size β. This gives a simpler form of the geometric recovery condition and allows a direct comparison with earlier block-size estimates for LWE with hints.

Let $d = d_H$, and suppose that the Gram–Schmidt profile after BKZ reduction is approximated by the GSA in (25)–(26). The logarithmic Gram–Schmidt norms satisfy

$$\ell_i \approx \frac{\log_2 D_H}{d} + \left(i - \frac{d+1}{2}\right) \log_2 \rho_\beta,$$

where

$$\rho_\beta = \delta(\beta)^{-\frac{2d}{d-1}}.$$

For a final sieving dimension η, let $t = d - \eta + 1$. Summing the logarithmic Gram–Schmidt norms over the projected sublattice gives

$$\frac{1}{\eta} \sum_{i=t}^{d} \ell_i \approx \frac{\log_2 D_H}{d} + \frac{d-\eta}{2} \log_2 \rho_\beta.$$

Therefore,

$$\begin{aligned} \left(\prod_{i=t}^{d} g_i\right)^{1/\eta} &\approx D_H^{1/d} \rho_\beta^{(d-\eta)/2} \\ &= D_H^{1/d} \delta(\beta)^{\frac{d(\eta-d)}{d-1}}. \end{aligned}$$

Substituting this expression into the Gaussian heuristic of the projected sublattice yields

$$\widehat{G}_t(\beta, \eta) \approx c_\eta D_H^{1/d} \delta(\beta)^{\frac{d(\eta-d)}{d-1}},$$

where

$$c_\eta = \frac{\Gamma(\eta/2+1)^{1/\eta}}{\sqrt{\pi}}.$$

Under the centered isotropic target model with standard deviation σ, we have $\sigma_{\text{eff}} \approx \sigma$. The geometric condition in (30) then becomes

$$\sqrt{\eta}\,\sigma \le \rho_S c_\eta D_H^{1/d} \delta(\beta)^{\frac{d(\eta-d)}{d-1}}.$$

Using the determinant of the CJC lattice in (17),

$$D_H = q^m |\det(\mathbf{B}_{\text{SIS}})|,$$

we obtain

$$\sqrt{\eta}\,\sigma \le \rho_S c_\eta \delta(\beta)^{\frac{d(\eta-d)}{d-1}} q^{m/d} |\det(\mathbf{B}_{\text{SIS}})|^{1/d}.$$

For comparison with the conventional single-block-size estimate, we use the approximations $c_\eta \approx \delta(\eta)^\eta$ and $d/(d-1) \approx 1$, together with the unit-radius benchmark $\rho_S = 1$. Taking base-2 logarithms gives

$$f(\eta) = \log_2 \Big(\delta(\eta)^\eta \delta(\beta)^{\eta-d} q^{m/d} \cdot |\det(\mathbf{B}_{\mathrm{SIS}})|^{1/d} \Big) - \log_2(\sqrt{\eta}\,\sigma). \tag{32}$$

For a fixed block size β, the condition $f(\eta) \geq 0$ indicates that the estimated search radius reaches the root-mean-square norm of the projected target under these approximations. Setting $\eta = \beta$ gives a block-size criterion of the form used in the CJC estimator [14].

The probabilistic estimator of Section 4.3 extends this geometric comparison by evaluating the distribution of the projected target vector. Under the centered isotropic Gaussian model, the geometric inclusion probability is

$$\widehat{p}_g(\beta, \eta) = F_{\chi^2_\eta}\left(\frac{\rho_S^2 \widehat{G}_t(\beta, \eta)^2}{\sigma^2} \right).$$

For a required geometric inclusion probability $p_g^{\min}$, the corresponding radius satisfies

$$\rho_S \widehat{G}_t(\beta, \eta) \geq \sigma \sqrt{F^{-1}_{\chi^2_\eta}(p_g^{\min})}.$$

The success probability of the complete strategy is then estimated using the sieve coverage, lifting, and BKZ recovery probabilities described in Section 4.3.

For a multi-stage BKZ schedule, we use the simulated profile predicted in Section 4. The recovery model then evaluates each schedule together with its final sieving dimension.

5 Phase 2: Time Estimation and Strategy Selection

Phase 2 of the framework in Fig. 2 estimates the expected running time and peak memory of candidate strategies, then selects the feasible strategy with the lowest expected running time. We first estimate the time and memory requirements of BKZ reduction and final sieving, and then combine these estimates with the success probabilities from Phase 1 to apply the success-probability and memory constraints. Section 5.1 presents the time and memory models, and Section 5.2 describes the strategy selection procedure. The numerical predictions and experiments are presented in Section 6.

5.1 Time Estimation for BKZ Reduction and Final Sieving

We use existing complexity models to estimate the running time and memory of the candidate strategies in Section 3.1. For a fixed algorithm pair (R, S), these estimates depend on the BKZ schedule and final solver configuration. This formulation allows the strategy optimization framework to be applied to different combinations of reduction and sieving algorithms.

BKZ reduction time. Let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_K)$ be a reduction schedule, where $\xi_j = (\beta_j, \tau_j, \vartheta_{R,j})$ specifies the block size, number of tours, and additional parameters of stage j. The estimated running time of the reduction schedule is

$$\widehat{T}_{\mathrm{red}}(\boldsymbol{\xi}) = \sum_{j=1}^{K} \widehat{T}_{R,j}(\mathbf{B}_{j-1}, \xi_j),$$

where $\widehat{T}_{R,j}$ denotes the estimated running time of the j-th reduction stage.

A commonly used time-complexity estimate for a BKZ tour with block size β is [24]

$$T_{\mathrm{BKZ}}(d_H, \beta) \approx N_{\mathrm{SVP}}(d_H, \beta)\, T_{\mathrm{SVP},R}(\beta), \tag{33}$$

where $N_{\mathrm{SVP}}(d_H, \beta) \approx \max\{1, d_H - \beta\}$ counts the local SVP calls in a tour and $T_{\mathrm{SVP},R}(\beta)$ estimates the running time of the selected reduction implementation's local SVP solver. This also assigns one SVP call to the full-dimensional case. Thus, a stage with τ_j tours has estimated running time

$$\widehat{T}_{R,j} \approx \tau_j N_{\mathrm{SVP}}(d_H, \beta_j)\, T_{\mathrm{SVP},R}(\beta_j).$$

Implementation-specific estimates can replace these approximations when available.

Final sieving time. Let η be the dimension of the projected sublattice, and let θ_S specify the parameters of the final SVP solver. We denote the estimated running time of one final search attempt by

$$\widehat{T}_{\mathrm{term}}(\mathbf{B}_K, \eta, \theta_S).$$

For sieving algorithms, the running time depends primarily on the dimension of the search and the implementation of the sieve. Table 2 summarizes the asymptotic time and memory complexities of several sieving algorithms from previous studies.

Table 2. Heuristic time and memory complexities of lattice sieving algorithms in dimension η.

Sieve	**Time**	**Memory**
Gauss sieve [20]	$2^{0.415\eta+o(\eta)}$	$2^{0.2075\eta+o(\eta)}$
List sieve [10]	$2^{0.383\eta+o(\eta)}$	$2^{0.2075\eta+o(\eta)}$
BGJ sieve [32,33]	$2^{0.292\eta+o(\eta)}$	$2^{0.2075\eta+o(\eta)}$
Quantum sieve [11]	$2^{0.265\eta+o(\eta)}$	$2^{0.2075\eta+o(\eta)}$

For example, the BGJ sieve has a heuristic time complexity of

$$T_{\mathrm{BGJ}}(\eta_{\mathrm{act}}) = 2^{0.292\eta_{\mathrm{act}}+o(\eta_{\mathrm{act}})},$$

with a database size of approximately

$$N_{\mathrm{BGJ}}(\eta_{\mathrm{act}}) = 2^{0.2075\eta_{\mathrm{act}}+o(\eta_{\mathrm{act}})},$$

where η_{act} is the dimension of the active sieving computation. For a standalone sieve operating directly on the projected sublattice, $\eta_{\mathrm{act}} = \eta$. The estimated memory requirement also depends on the storage required for each database vector and the working memory of the implementation.

The corresponding time and memory estimates for other sieving algorithms are obtained from their respective complexity models. For each selected solver S, the time and memory estimates are evaluated using the same solver configuration θ_S as the success probability estimate in Section 4.3.

Total strategy time and memory. The running time of a complete strategy includes lattice construction, BKZ reduction, final sieving, and the associated lifting and verification steps. As defined in Section 3.3, the expected total time weights each operation by the probability that it is reached. The peak memory is the maximum memory required by any operation along an execution of the strategy.

The predictions in Section 6 use a common time-cost unit derived from the reduction and sieving time-complexity estimates. When implementation-specific timing estimates are available, the same model gives elapsed running time. In both cases, strategy selection minimizes the expected total time subject to the success-probability and memory constraints.

5.2 Strategy Selection

Based on the success-probability estimates in Section 4 and the time and memory estimates in Section 5.1, we select a solving strategy from the candidate set defined in Section 3.4. For a given LWE instance with hints and a specified combination of reduction algorithm R and final SVP solver S, the selection procedure evaluates the listed BKZ schedules and final-sieving configurations, and chooses the feasible strategy with the lowest expected running time.

The candidate set is formed by listing the supported BKZ schedules and final-sieving configurations within their prescribed parameter limits. Each BKZ schedule $\boldsymbol{\xi}$ is paired with each admissible final-sieving configuration (η, θ_S, r) to form $\mathcal{X}(\mathcal{I}; R, S)$.

For each schedule, the simulator described in Section 4 predicts the Gram–Schmidt profile. We evaluate reduction-stage recovery and final sieve coverage using the models associated with the chosen algorithms. This profile is used to estimate the geometry of the projected sublattice for each final sieving dimension η. The success probability of the resulting strategy is then evaluated using the projected target distribution, sieve coverage probability, and lifting probability derived in Sections 4.2 and 4.3, together with the probability of recovery during BKZ reduction.

We estimate candidate running times and peak memory using Section 5.1. The expected running time accounts for early recovery during BKZ reduction

or final search. Candidates satisfying the success-probability and memory constraints in (24) form the feasible set.

The selection procedure is summarized in Algorithm 4.

Algorithm 4: Strategy selection for LWE with hints

Input: LWE instance with hints $\mathcal{I}$; algorithms (R, S); candidate set $\mathcal{X}(\mathcal{I}; R, S)$; success threshold p_0; memory limit $M_{\max}$
Output: Selected solving strategy x^*, or no feasible strategy

Construct the attack lattice $\mathcal{L}_H$ using the CJC method in Section 2.4;
$\mathcal{F} \leftarrow \varnothing$;
foreach *candidate strategy* $x \in \mathcal{X}$ **do**
  Estimate the success probability $\widehat{p}_x$ using Step 2;
  if $\widehat{p}_x \geq p_0$ **then**
    Estimate the expected running time $\widehat{T}_x$ and peak memory $\widehat{M}_x$ using Step 3;
    if $\widehat{M}_x \leq M_{\max}$ **then**
      Add x to the feasible set $\mathcal{F}$;
    end
  end
end
if $\mathcal{F} = \varnothing$ **then**
  return no feasible strategy;
end
return $x^* \in \arg\min_{x \in \mathcal{F}} \widehat{T}_x$;

We reuse each predicted Gram–Schmidt profile across final sieving configurations. The selected strategy specifies the BKZ reduction stages and the final sieving parameters to be used in the attack.

The procedure also applies to other BKZ and final-solver combinations when their success and cost models are available. In our experiments, we instantiate the framework with BLASterBKZ and the BGJ sieve. The resulting strategy predictions are presented in Section 6.

6 Experimental Results

We evaluate our method on CRYSTALS-KYBER 512 and CRYSTALS-KYBER 768 instances with perfect hints.

Setup. For CRYSTALS-KYBER 512 and CRYSTALS-KYBER 768, we construct the corresponding $\mathbb{Z}$-SIS lattices from perfect hints using the CJC construction, with $n = 512$ and $n = 768$, respectively, and $q = 3329$. The resulting lattice dimensions are $d = 513 - k_{\mathrm{p}}$ and $d = 769 - k_{\mathrm{p}}$, respectively. Both parameter sets use the same preprocessing pipeline: `flatter` followed by `BLASterDeepLLL`. We then apply `BLASterBKZ` and the BGJ sieve.

Hardware. Experiments used an Intel Xeon Gold 6254 CPU at 3.10 GHz with 192 GB RAM; its single-core performance is comparable to the platforms reported by MN and CJC. The 190–210-hint CRYSTALS-KYBER 512 experiments used one thread; the 170–189-hint CRYSTALS-KYBER 512 and 325–330-hint CRYSTALS-KYBER 768 experiments used 64 threads. Each run measures lattice construction, BKZ reduction and, when needed, BGJ sieving, lifting, and verification, stopping upon secret recovery. For each setting, we report average total running time over 20 independent runs.

Results. Table 3 reports the selected BKZ schedules and final BGJ sieving dimensions η for CRYSTALS-KYBER 512. Each BKZ stage uses two tours. As more perfect hints reduce the lattice dimension, the selected block sizes and sieving dimensions decrease.

Table 3. Estimated BKZ schedules and sieving dimensions for CRYSTALS-KYBER 512.

k_{p}	**BKZ schedule**	η
150	$107 \to 111 \to 113$	136
155	$91 \to 95 \to 97$	130
160	$83 \to 85 \to 89$	119
165	$71 \to 77 \to 79$	111
170	$65 \to 67 \to 69$	102
175	$55 \to 57 \to 59$	94
180	$47 \to 49 \to 51$	82

Figure 3 compares single-core running times for CRYSTALS-KYBER 512 with 190–210 perfect hints. Our method completes the attack within a few minutes throughout this range. At 190 hints, it takes about 7 minutes, compared with 1.6 hours for CJC and 7.4 hours for MN. As more hints reduce the dimension of the constructed lattice, BKZ can recover the secret without final sieving, and the running-time gap between our method and CJC narrows.

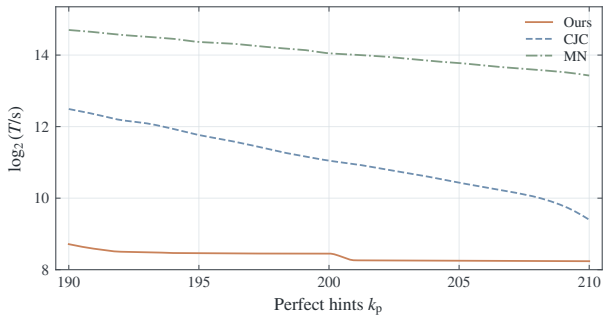

Fig. 3. Single-core running times for CRYSTALS-KYBER 512.

Figure 4 shows the estimated BKZ block sizes for 150–180 perfect hints. All three estimates decrease as the number of hints increases. For our method, the final BKZ block size decreases from 113 at 150 hints to 51 at 180 hints, and its estimates remain between those of CJC and MN.

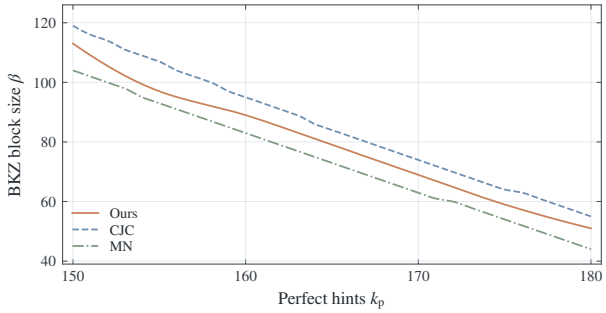

Fig. 4. Estimated BKZ block sizes for CRYSTALS-KYBER 512.

Figure 5 shows the total running time of our 64-thread CRYSTALS-KYBER 512 attack for 170–189 perfect hints. As the number of hints increases, the measured time decreases from roughly 80 minutes at 170 hints to about 3 minutes at 189 hints.

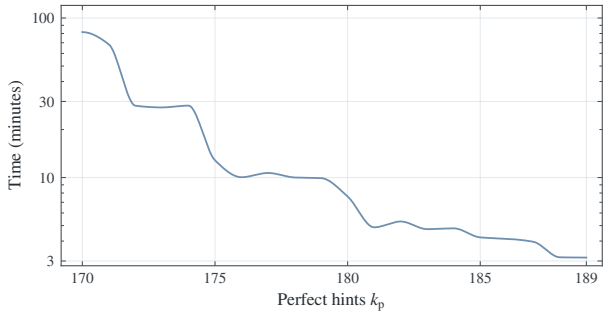

Fig. 5. Total running times for CRYSTALS-KYBER 512 (64 threads).

Figure 6 shows the predicted BGJ memory for 150–180 perfect hints. The estimate decreases from several hundred GiB to below 0.1 GiB as the hint count increases. This decline accompanies the smaller final-sieving dimensions selected for the lower-dimensional attack lattices.

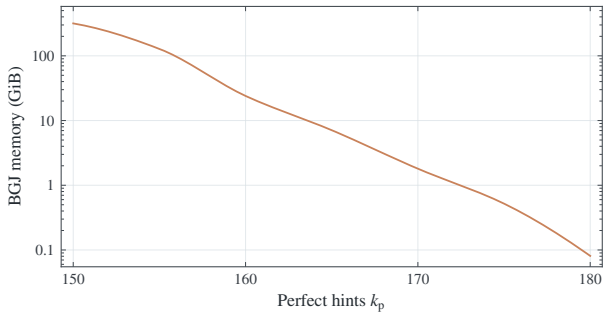

Fig. 6. Predicted BGJ memory for CRYSTALS-KYBER 512.

For 180 perfect hints, Fig. 7 examines the balance between BKZ reduction time and final-stage time in the single-block BKZ setting. Increasing β raises the BKZ reduction time, while the final-stage time decreases as the reduced basis improves. The total running time reaches its minimum at $\beta = 55$, giving the lowest-time single-block setting among those shown.

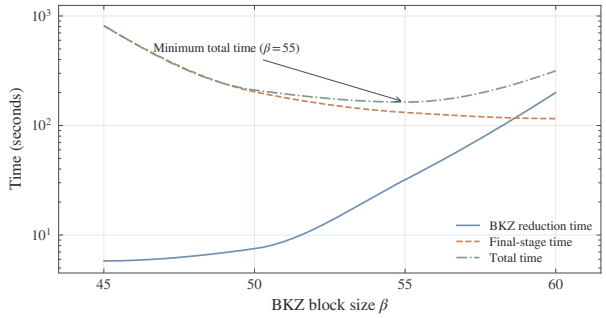

Fig. 7. Time components for CRYSTALS-KYBER 512 with 180 hints.

We also evaluate CRYSTALS-KYBER 768 using 64 threads and a fixed BKZ block size of $\beta = 40$. Table 4 lists the lattice dimensions and total running times for 325–330 perfect hints. Across these six settings, the running times range from 4.6 to 5.4 hours, with the shortest time observed at 329 hints. The results show that the total running time stays roughly constant over the tested hint range.

Table 4. CRYSTALS-KYBER 768 attack results.

k_p	Lattice dimension d	BKZ β	Threads	Time (h)
325	444	40	64	5.2
326	443	40	64	4.8
327	442	40	64	5.1
328	441	40	64	5.4
329	440	40	64	4.6
330	439	40	64	4.9

7 Conclusion

We presented a strategy optimization framework and a two-step estimator for LWE with hints. Given a BKZ and final-solver combination, the framework selects the BKZ reduction schedule and final sieving configuration using estimates of secret-recovery probability, computational cost, and memory requirements. The estimator combines the predicted Gram–Schmidt profile with the projected target distribution, sieve coverage, and lifting probability. We instantiated the framework with BLASterBKZ and the BGJ sieve and evaluated it on CRYSTALS-KYBER 512 and CRYSTALS-KYBER 768 with perfect hints. For CRYSTALS-KYBER 512, our single-core attack took about 7 minutes with 190 hints, while the 64-thread total running time decreased from roughly 80 minutes at 170 hints to about 3 minutes at 189 hints. For CRYSTALS-KYBER 768, the 64-thread attacks with 325–330 hints completed in 4.6–5.4 hours. These results illustrate how the proposed estimator combines recovery, time, and memory estimates to guide the selection of attack parameters.

Acknowledgements

The work of Qingfeng Cheng was supported by the National Natural Science Foundation of China under Grant No. 62472438.

References

1. Bos, J.W., Friedberger, S., Martinoli, M., Oswald, E., Stam, M.: Assessing the feasibility of single trace power analysis of Frodo. In: Cid, C., Jacobson, M.J. (eds.) SAC 2018. LNCS, vol. 11349, pp. 216–234. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10970-7_10
2. Albrecht, M.R., Deo, A., Paterson, K.G.: Cold boot attacks on ring and module LWE keys under the NTT. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(3), 173–213 (2018). https://doi.org/10.13154/tches.v2018.i3.173-213
3. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), Article 34 (2009). https://doi.org/10.1145/1568318.1568324
4. Ajtai, M.: The shortest vector problem in L_2 is NP-hard for randomized reductions. In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, STOC 1998, pp. 10–19. ACM, New York (1998). https://doi.org/10.1145/276698.276705
5. Khot, S.: Hardness of approximating the shortest vector problem in lattices. J. ACM 52(5), 789–808 (2005). https://doi.org/10.1145/1089023.1089027
6. Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC 2007, pp. 469–477. ACM, New York (2007). https://doi.org/10.1145/1250790.1250859
7. Lyubashevsky, V., Micciancio, D.: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 577–594. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_34

8. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261(4), 515–534 (1982). https://doi.org/10.1007/BF01457454
9. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, STOC 2001, pp. 601–610. ACM, New York (2001). https://doi.org/10.1145/380752.380857
10. Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 1468–1480. SIAM (2010). https://doi.org/10.1137/1.9781611973075.119
11. Laarhoven, T., Mosca, M., van de Pol, J.: Finding shortest lattice vectors faster using quantum search. Des. Codes Cryptogr. 77(2–3), 375–400 (2015). https://doi.org/10.1007/s10623-015-0067-5
12. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 329–358. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_12
13. May, A., Nowakowski, J.: Too many hints – when LLL breaks LWE. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14441, pp. 106–137. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8730-6_4
14. Cao, J., Jiang, H., Cheng, Q.: Refined attack on LWE with hints: constructing lattice via Gaussian elimination. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025. LNCS, vol. 16000, pp. 385–416. Springer, Cham (2025). https://doi.org/10.1007/978-3-032-01855-7_13
15. Dachman-Soled, D., Gong, H., Hanson, T., Kippen, H.: Revisiting security estimation for LWE with hints from a geometric perspective. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. LNCS, vol. 14085, pp. 748–781. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38554-4_24
16. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. 66(1–3), 181–199 (1994). https://doi.org/10.1007/BF01581144
17. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_1
18. Aono, Y., Wang, X., Hayashi, T., Takagi, T.: Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 789–819. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_30
19. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC 1983, pp. 193–206. ACM, New York (1983). https://doi.org/10.1145/800061.808749
20. Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. Math. Cryptol. 2(2), 181–207 (2008). https://doi.org/10.1515/jmc.2008.009
21. Laarhoven, T., Mariano, A.: Progressive lattice sieving. In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018. LNCS, vol. 10786, pp. 292–311. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-79063-3_14
22. Albrecht, M.R., Ducas, L., Herold, G., Kirshanova, E., Postlethwaite, E.W., Stevens, M.: The general sieve kernel and new records in lattice reduction. In:

Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 717–746. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_25
23. Ducas, L., Stevens, M., van Woerden, W.: Advanced lattice sieving on GPUs, with tensor cores. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 249–279. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_9
24. Xia, W., Wang, L., Wang, G., Gu, D., Wang, B.: Optimizing two-step primal LWE attacks with PnJBKZ simulation. IACR Cryptology ePrint Archive, Report 2022/1343 (2022). https://eprint.iacr.org/2022/1343
25. Wang, L., Wang, Y., Wang, B.: A trade-off SVP-solving strategy based on a sharper pnj-BKZ simulator. In: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, ASIA CCS 2023, pp. 664–677. ACM, New York (2023). https://doi.org/10.1145/3579856.3595802
26. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. 9(3), 169–203 (2015). https://doi.org/10.1515/jmc-2015-0016
27. Xia, W., Wang, L., Wang, G., Gu, D., Wang, B.: A refined hardness estimation of LWE in two-step mode. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. LNCS, vol. 14603, pp. 3–35. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57725-3_1
28. Ducas, L., Pulles, L.N., Stevens, M.: Towards a modern LLL implementation. In: Hanaoka, G., Yang, B.-Y. (eds.) ASIACRYPT 2025. LNCS, vol. 16247, pp. 65–99. Springer, Singapore (2025). https://doi.org/10.1007/978-981-95-5099-9_3
29. Zhao, Z., Ding, J.: Towards large-scale lattice attack: new lattice records by disk-based sieving. In: EUROCRYPT 2026. LNCS, vol. 16544, pp. 154–182. Springer, Cham (2026). https://doi.org/10.1007/978-3-032-25327-9_6
30. Hashimoto, R., Shikata, J., Takayasu, A., Wang, Y.: AH-BKZ: a lattice reduction algorithm with asynchronous hybrid processing. In: ICICS 2026. LNCS, vol. 17214, pp. 3–23. Springer, Singapore (2026). https://doi.org/10.1007/978-981-92-6831-3_1
31. Hhan, M., Hong, G.H., Kim, J., Lee, C., Lee, J.: From perfect to approximate hints: efficient LWE secret recovery leveraging low Hamming weight. In: Proceedings of the 2026 IEEE Symposium on Security and Privacy, SP 2026. IEEE (2026). https://doi.org/10.1109/SP63933.2026.00239
32. Becker, A., Gama, N., Joux, A.: Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. IACR Cryptology ePrint Archive, Report 2015/522 (2015). https://eprint.iacr.org/2015/522
33. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pp. 10–24. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch2
34. Ducas, L.: Shortest vector from lattice sieving: a few dimensions for free. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 125–145. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_5
35. Zhao, Z., Ding, J., Yang, B.-Y.: Sieving with streaming memory access. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2025(2), 362–384 (2025). https://doi.org/10.46586/tches.v2025.i2.362-384