information

IMPACT FACTOR 2.9

CITESCORE 6.5

Article

Ternary LWE Key Search: A New Frontier for Quantum Combinatorial Attacks

Yang Li

MDPI

https://doi.org/10.3390/info16121085

information

MDPI

Article

Ternary LWE Key Search: A New Frontier for Quantum Combinatorial Attacks

Yang Li

Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China; liyang@ios.ac.cn

Abstract

The Learning with Errors (LWE) problem, particularly its efficient ternary variant where secrets and errors are small, is a fundamental building block for numerous post-quantum cryptographic schemes. Combinatorial attacks provide a potent approach to cryptanalyzing ternary LWE. While classical attacks have achieved complexities close to their asymptotic $\mathcal{S}^{0.25}$ bound for a search space of size $\mathcal{S}$, their quantum counterparts have faced a significant gap: the attack by van Hoof et al. (vHKM) only reached a concrete complexity of $\mathcal{S}^{0.251}$, far from its asymptotic promise of $\mathcal{S}^{0.193}$. This work introduces an efficient quantum combinatorial attack that substantially narrows this gap. We present a quantum walk adaptation of the locality-sensitive hashing algorithm by Kirshanova and May, which fundamentally removes the need for guessing error coordinates—the primary source of inefficiency in the vHKM approach. This crucial improvement allows our attack to achieve a concrete complexity of approximately $\mathcal{S}^{0.225}$, markedly improving over prior quantum combinatorial methods. For concrete parameters of major schemes including NTRU, BLISS, and GLP, our method demonstrates substantial runtime improvements over the vHKM attack, achieving speedup factors ranging from 2^{16} to 2^{60} across different parameter sets and establishing the new state-of-the-art for quantum combinatorial attacks. As a second contribution, we address the challenge of polynomial quantum memory constraints. We develop a hybrid approach combining the Kirshanova–May framework with a quantum claw-finding technique, requiring only $\mathcal{O}(n)$ qubits while utilizing exponential classical memory. This work provides the first comprehensive concrete security analysis of real-world LWE-based schemes under such practical quantum resource constraints, offering crucial insights for post-quantum security assessments. Our results reveal a nuanced landscape where our combinatorial attacks are superior for small-weight parameters, while lattice-based attacks maintain an advantage for others.

Keywords: LWE; quantum algorithms; quantum walks; amplitude amplification

Academic Editor: Haris Mouratidis

Received: 22 October 2025
Revised: 30 November 2025
Accepted: 5 December 2025
Published: 7 December 2025

Citation: Li, Y. Ternary LWE Key Search: A New Frontier for Quantum Combinatorial Attacks. *Information* **2025**, *16*, 1085. https://doi.org/10.3390/info16121085

1. Introduction

The Learning with Errors (LWE) problem [1] has become a cornerstone of post-quantum cryptography, serving as the security foundation for numerous cryptographic constructions [2–4]. In its standard form, LWE tries to find a secret vector $s \in \mathbb{Z}_q^n$ given $(A, b) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, where A is a random matrix and $b = As + e \mod q$ for some small error vector e. The hardness of LWE is supported by worst-case lattice reduction proofs [5], making it a robust basis for cryptographic constructions.

A particularly efficient variant is ternary LWE, where the entries of s and e are restricted to $\{-1, 0, 1\}$. This variant is employed in several practical schemes, including NTRU-based

cryptosystems [6–8] and signature protocols such as BLISS [9] and GLP [10]. While the small domain of the secret and error does not fundamentally break LWE's security, it does enable more efficient combinatorial attacks that exploit the sparsity and structure of the secrets.

Combinatorial attacks offer a distinct approach to cryptanalyzing LWE with small secrets, beginning with early Meet-in-the-Middle (MitM) attacks on NTRU by Odlyzko [11] and Howgrave-Graham [12]. A significant advancement came from May [13], who adapted the representation technique from subset-sum attacks [14–16] to reduce MitM complexity from $\mathcal{S}^{0.5}$ to approximately $\mathcal{S}^{0.25}$ for a search space of size $\mathcal{S}$. However, this improvement required guessing sub-linear error coordinates, which, while asymptotically negligible, raised concrete complexities to $\mathcal{S}^{0.3}$ for practical NTRU, BLISS, and GLP parameters. This limitation was later addressed by Kirshanova and May [17] through an Odlyzko-based locality-sensitive hashing (LSH) technique that eliminated error guessing, finally achieving complexities near the asymptotic $\mathcal{S}^{0.25}$ bound.

In the quantum setting, van Hoof et al. [18] investigated quantum speedups for May's algorithm. Their asymptotic analysis demonstrates that the approach achieves a time complexity close to the $\mathcal{S}^{0.193}$ bound; however, for concrete instantiations, the method only reaches a $\mathcal{S}^{0.251}$ bound. Despite employing quantum search to accelerate the guessing phase, their quantum approach inherits the same fundamental limitation: for concrete cryptographic parameters, the realized complexity substantially deviates from the asymptotic bound, failing to bridge the gap between theoretical and practical performance.

1.1. Our Contribution

This work introduces a quantum adaptation of the LSH-based algorithm by Kirshanova and May [17] for solving ternary LWE. We reformulate the key recovery problem within a graph search framework. By implementing an optimized quantum walk over this graph, we eliminate the need for guessing the error coordinates—a fundamental limitation inherent to the quantum approach of van Hoof et al. [18]. As a result, our method substantially narrows the gap between asymptotic and concrete complexity, achieving a concrete bound of approximately $S^{0.225}$ compared to the $S^{0.251}$ bound of vHKM. For concrete parameters of major schemes such as NTRU, BLISS, and GLP, our method demonstrates speedup factors ranging from 2^{16} to 2^{60} over the vHKM [18] attack.

Our algorithm establishes the state of the art for quantum combinatorial attacks on ternary LWE. A comparison with quantum sieving algorithm shows a split in dominance: our method is superior for small-weight parameters, while lattice-based attacks maintain the lead for most others.

Current LWE parameter selection relies heavily on lattice reduction heuristics (e.g., Gaussian Heuristic and Geometric Series Assumption) and diverse BKZ runtime models, potentially underestimating security. In contrast, our combinatorial approach builds on heuristic principles derived from subset-sum or knapsack problems, which possess a stronger theoretical foundation. For knapsack-type distributions, it can be rigorously proven that pathological instances form only an exponentially small fraction, enabling the construction of provable probabilistic algorithms that avoid heuristics entirely. This theoretical robustness, combined with experimental validation in prior work [15], provides runtime predictions with significantly enhanced precision compared to lattice reduction heuristics that depend on unproven assumptions.

Our second contribution focuses on LWE key recovery under polynomial quantum memory constraints. While previous quantum-walk-based approaches achieve optimal time complexity among combinatorial methods, they require exponential qubit resources and rely on the strong Quantum Random Access Quantum Memory (QRAQM) model.

Building on recent work by Benedikt [19], who applied the quantum claw-finding algorithm of Chailloux et al. [20] to ternary LWE problems, we adopt a more practical approach that combines the Kirshanova–May framework [17] with this claw-finding technique. This hybrid strategy operates efficiently with only $\mathcal{O}(n)$ qubits under the Quantum Random Access Classical Memory (QRACM) model.

While Benedikt [19] provided valuable asymptotic analysis, their work left open the question of concrete security estimates for practical cryptographic parameter sets. Our work specifically addresses this gap by performing the first comprehensive concrete security analysis of real-world LWE-based schemes under polynomial-qubit constraints, providing crucial insights for post-quantum security assessments.

Furthermore, we provide the time–classical memory trade-offs for concrete schemes and explicitly compare our approach with the vHKM [18] method under strict polynomial classical memory constraints. Notably, across all evaluated schemes, our algorithm consistently achieves lower time complexity than vHKM under the same polynomial memory bound, further establishing its practical relevance in constrained settings.

1.2. Organization

The rest of this paper is organized as follows. In Section 2, we provide the necessary background, including the definition of the Ternary LWE, an introduction to Generalized Odlyzko's Locality-Sensitive Hashing, and a review of the quantum primitives used in this work: amplitude amplification and quantum walks. Section 3 reviews the classical LSH-based MitM attack on ternary LWE by Kirshanova and May, which serves as the foundation for our quantum algorithms. In Section 4, we present our main contribution. We reformulate the ternary LWE key search problem as a graph search problem and introduce our novel quantum algorithm that combines the Kirshanova–May framework with a quantum walk. We provide a detailed description of the algorithm and a comprehensive concrete security analysis for major cryptographic schemes. Section 5 addresses the scenario of polynomial quantum memory constraints. We present a second algorithm that hybridizes the Kirshanova–May framework with a quantum claw-finding technique, requiring only a polynomial number of qubits. The concrete security of real-world schemes under this new algorithm is thoroughly analyzed. Finally, we conclude the paper in Section 6 with a summary of our findings.

2. Preliminaries

2.1. Notational Conventions

We adopt the following notation throughout this paper:

- Let $\mathbb{Z}_q$ represent the quotient ring of integers modulo $q \geq 2$.
- Vector quantities are denoted using lowercase letters (e.g., x, y), while matrices use uppercase letters (e.g., A, B).
- For any vector x, its ℓ_∞-norm is defined as $\|x\|_\infty = \max_i |x_i|$.
- The cardinality of a set S is denoted by $|S|$.
- For positive integers n and $\{n_i\}_{i \leq k}$ summing to n, the multinomial coefficient is given by the product

$$\binom{n}{n_1, \ldots, n_{k-1}, n_k} = \binom{n}{n_1} \binom{n - n_1}{n_2} \cdots \binom{n - \sum_{i=1}^{k-1} n_i}{n_k}. \tag{1}$$

A compact notation $\binom{n}{n_1, \ldots, n_{k-1}, \cdot}$ is commonly used, where the dot signifies the remaining value $n_k = n - \sum_{i=1}^{k-1} n_i$.

2.2. Ternary LWE Secret Key

We focus on LWE instances where both the secret and error vectors are ternary and matrix A is square. This specific case captures numerous practical implementations in lattice-based cryptography.

Definition 1 (Ternary LWE Secret Key)**.** *Let $(A, b) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ be an LWE public key satisfying $As = b + e \mod q$. The pair (s, e) is called a ternary LWE secret key if both s and e belong to the set $\mathcal{T}^n = \{-1, 0, 1\}^n$, i.e., $\|s\|_\infty = \|e\|_\infty = 1$.*

The ternary constraint ensures cryptographic efficiency while maintaining security guarantees. For a randomly chosen A, the correct solution s is uniquely identifiable with high probability through the condition that $As - b$ also falls within $\mathcal{T}^n$.

Practical implementations often impose additional structural constraints by fixing the number of non-zero entries in ternary vectors, leading to optimized security parameters and implementation characteristics.

Definition 2 (Weight)**.** *Let $s = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$. The weight w of s is defined as its Hamming weight $w := \sum_{s_i \neq 0} 1$. For ternary vectors with balanced non-zero coefficients, we define the specialized set:*

$$\mathcal{T}^n(w/2) = \{s \in \mathcal{T}^n \mid s \text{ has exactly } w/2 \text{ 1-entries and } w/2 \text{ } (-1)\text{-entries}\}. \tag{2}$$

Rounding considerations are handled appropriately in concrete parameterizations. Weight restrictions naturally emerge in the context of NTRU-type cryptosystems, where optimal values typically reside in the interval $\omega \in [\frac{1}{3}, \frac{2}{3}]$. The specific choice $\omega = \frac{3}{8}$ appears in several standardized parameter sets, representing a careful equilibrium between security and performance requirements.

2.3. Generalized Odlyzko's Locality-Sensitive Hashing

A key technique in combinatorial attacks on ternary LWE is the use of LSH to efficiently identify pairs of vectors that are close in the ℓ_∞-norm. Odlyzko's original LSH function [11] partitions $\mathbb{Z}_q$ into two halves and assigns binary labels, but it requires special handling of border values.

Kirshanova and May [17] generalized this approach to avoid explicit border checks and to enable more flexible bucketing. Their LSH family is defined as follows. For a bucket size parameter $B \in \{1, \ldots, q\}$ and a random shift vector $u \in \mathbb{Z}_q^n$, the hash function $h_{u,B} : \mathbb{Z}_q^n \to [0, \lceil q/B \rceil - 1]^n$ is given by

$$h_{u,B}(x) = \left(\left\lfloor \frac{x_1 + u_1}{B} \right\rfloor, \ldots, \left\lfloor \frac{x_n + u_n}{B} \right\rfloor \right). \tag{3}$$

This function maps each coordinate of x to a bucket index after a random cyclic shift. The probability that two vectors x, y with $\|x - y\|_\infty = 1$ collide under a random $h_{u,B}$ is $(1 - 1/B)^n$.

Algorithm 1 solves the close pairs problem in the ℓ_∞-norm: given two lists $L_1, L_2 \subset \mathbb{Z}_q^n$, find a $(1 - o(1))$-fraction of all pairs $(x, y) \in L_1 \times L_2$ such that $\|x - y\|_\infty = 1$.

Example 1. *We illustrate Algorithm 1 with $n = 4$, $q = 7$, and lists:*

$$L_1 = \{(1, 0, 2, 3), (0, 1, 1, 1), (2, 3, 0, 1)\},$$
$$L_2 = \{(1, 1, 2, 2), (3, 0, 1, 0), (0, 2, 1, 1)\}.$$

Choose $B = 3$, $u = (1, 2, 0, 1)$. *Hash computations:*

$$h_{u,B}(1,0,2,3) = (0,0,0,1), h_{u,B}(0,1,1,1) = (0,1,0,0), h_{u,B}(2,3,0,1) = (1,1,0,0);$$
$$h_{u,B}(1,1,2,2) = (0,1,0,1), h_{u,B}(3,0,1,0) = (1,0,0,0), h_{u,B}(0,2,1,1) = (0,1,0,0).$$

Matching hash $(0,1,0,0)$ *yields candidate pair* $(0,1,1,1), (0,2,1,1)$. *Verification:*

$$(0,1,1,1) - (0,2,1,1) = (0,-1,0,0) \Rightarrow \|(0,-1,0,0)\|_\infty = 1.$$

This pair is 1*-close. With* $(3/2)^4 \cdot 4 \log 4 \approx 41$ *repetitions using different random shifts* u*, Algorithm 1 finds all* 1*-close pairs with high probability.*

Algorithm 1 LSH-Odlyzko algorithm [17]

Input: Two lists L_1, L_2 of i.i.d. uniform vectors from $\mathbb{Z}_q^n$, each of size $|L|$
Output: $(1 - o(1))$ fraction of all pairs $(x_1, x_2) \in L_1 \times L_2$ such that $\|(x_1 - x_2) \bmod q\|_\infty = 1$

1: Choose $B \in \{1, \ldots, q\}$ suitably. Choose $u \xleftarrow{\$} \mathbb{Z}_q^n$.
2: Apply $h_{u,B}$ to all elements in L_1 and L_2. Sort L_1 and L_2 according to their hash values.
3: Merge the sorted lists and collect all pairs (x_1, x_2) with matching hash labels. Filter and output only those pairs satisfying $\|(x_1 - x_2) \bmod q\|_\infty = 1$.
4: Repeat Steps 1–3 $((B/(B-1))^n \cdot n \log n)$ times.

The space and time complexities of Algorithm 1 are given by

$$\begin{aligned} S &= \max\left\{ |L|,\ |L|^2 \cdot \left(\frac{3}{q}\right)^n \right\} \cdot \mathrm{poly}(n), \\ T_{\mathsf{LSH}}(|L|, n, B) &= \max\left\{ S,\ |L|^2 \cdot \left(\frac{B^2}{(B-1)q}\right)^n \cdot \mathrm{poly}(n) \right\}. \end{aligned} \tag{4}$$

When combining approximate matching on k_1 coordinates with exact matching on k_2 coordinates, the complexity becomes

$$\begin{aligned} S &= \max\left\{ |L|,\ |L|^2 \cdot \left(\frac{3}{q}\right)^{k_1} \cdot \left(\frac{1}{q}\right)^{k_2} \right\}, \\ T_{\mathsf{LSH+Exact}}(|L|, k_1, k_2, B) &= \max\left\{ S,\ |L|^2 \cdot \left(\frac{B^2}{(B-1)q}\right)^{k_1} \cdot \left(\frac{1}{q}\right)^{k_2} \cdot n \log n \right\}. \end{aligned} \tag{5}$$

This generalized LSH technique is crucial for efficiently merging partial solutions in Meet-in-the-Middle attacks on ternary LWE, as it allows for approximate matching without guessing error coordinates.

2.4. Amplitude Amplification

Amplitude amplification [21] is a fundamental technique in quantum computing that amplifies the probability amplitudes of the target states, which are identified by an oracle.

Consider a quantum algorithm $\mathcal{A}$ that prepares a uniform superposition $|\phi\rangle = \sum_{x \in \mathcal{X}} \frac{1}{|\mathcal{X}|} |x\rangle$ over some subset $\mathcal{X} \subseteq \{0,1\}^n$. Let $f : \mathcal{X} \to \{0,1\}$ be a computable Boolean function with its quantum oracle O_f acting as $O_f|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.

Define $\mathcal{X}_f = \{x \in \mathcal{X} \mid f(x) = 1\}$ as the set of "good" elements. The probability of obtaining a good element by measuring $|\phi\rangle$ is $p = |\mathcal{X}_f|/|\mathcal{X}|$. The amplitude amplification algorithm finds an element $x \in \mathcal{X}_f$ with high probability using only $\mathcal{O}(1/\sqrt{p})$ iterations, where each iteration comprises one call each to $\mathcal{A}$, $\mathcal{A}^\dagger$, and O_f.

Let $T_{\mathcal{A}}$ and T_f denote the time complexities of implementing $\mathcal{A}$ and O_f, respectively. The total time complexity of amplitude amplification is

$$T = \mathcal{O}\left(\frac{1}{\sqrt{p}} \cdot (T_{\mathcal{A}} + T_f)\right). \tag{6}$$

Amplitude amplification generalizes Grover search [22]. In the Grover setting, we take $\mathcal{X} = \{0,1\}^n$, and $\mathcal{A}$ applies a Hadamard gate to each qubit, which can be implemented efficiently with $T_{\mathcal{A}} = \mathcal{O}(1)$.

2.5. Quantum Walk Search on Graphs

Consider a connected, d-regular, undirected graph $G = (V, E)$ with a subset of marked vertices. The objective is to find a marked vertex. The classical random walk on the graph G is characterized by a symmetric transition matrix P of size $|V| \times |V|$, where the entry $P_{u,v}$ is defined as $1/d$ if $(u, v) \in E$, and zero otherwise. A classical random walk search proceeds as follows:

1. Sample a vertex $u \in V$ from an initial probability distribution over V and initialize an associated data structure $d(u)$.
2. Repeat the following t_1 times:
   (a) Check whether u is marked using $d(u)$. If marked, output u and halt.
   (b) Otherwise, perform t_2 steps of the random walk: at each step, move uniformly to a random neighbor v of the current vertex u, updating $d(u)$ to $d(v)$.

Let δ denote the spectral gap of G, defined as the difference between the two largest eigenvalues of its adjacency matrix. The mixing time required to approximate the uniform distribution from an arbitrary starting vertex is known to satisfy $t_2 = 1/\delta$.

Let ϵ be the fraction of marked vertices, so that a vertex sampled uniformly at random is marked with probability ϵ. Then, the expected number of trials needed to find a marked vertex is $t_1 = 1/\epsilon$.

We define the following cost parameters:

- Setup cost T_S: the cost of sampling the initial vertex and constructing the data structure $d(u)$.
- Update cost T_U: the cost incurred per random walk step, which encompasses moving from a node u to a neighbor v and updating $d(u)$ to $d(v)$.
- Checking cost T_C: the cost of checking whether u is marked using $d(u)$.

The overall cost of the classical random walk search is then given by

$$T_{\mathrm{RW}} = T_S + \frac{1}{\epsilon}\left(\frac{1}{\delta}T_U + T_C\right). \tag{7}$$

In the quantum setting, the walk is initialized by preparing a uniform superposition over all vertices, together with two auxiliary registers (we omit normalization for simplicity):

$$|\pi\rangle = \sum_{u \in V} |u\rangle |p_u\rangle |d(u)\rangle, \tag{8}$$

where $|p_u\rangle := \sum_v \sqrt{P_{uv}}|v\rangle$ represents the uniform superposition over the neighbors of u.

The coin register $|p_u\rangle$ uses the same number of qubits as the vertex register $|u\rangle$ and encodes the direction of the quantum walk. The data register $|d(u)\rangle$ contains the classical data structure associated with vertex u, which is used to determine if u is marked.

The quantum walk step is a unitary operation that coherently simulates one step of the random walk. It consists of a coin operation followed by a shift operation. The coin operation creates a superposition over the neighbors of the current vertex, while the

shift operation updates the vertex register accordingly. Throughout this process, the data structure in the register $|d(u)\rangle$ is updated coherently to reflect the new vertex. The checking operation is a phase oracle that applies a -1 phase to marked vertices.

We define the quantum analogues of the classical cost parameters as follows:

- Setup cost T_S: The cost of preparing the initial state $|\pi\rangle$.
- Update cost T_U: The cost of implementing one quantum walk step.
- Checking cost T_C: The cost of applying the checking oracle.

The quantum walk search algorithm achieves a quadratic speedup over classical approaches, requiring only $1/\sqrt{\epsilon}$ iterations via amplitude amplification. A key component of this algorithm is the implementation of a reflection operator about the stationary state of the quantum walk, which corresponds to the quantum analog of the classical stationary distribution. Remarkably, this reflection can be approximated using phase estimation applied to the quantum walk operator, which requires only $1/\sqrt{\delta}$ steps. This efficiency stems from the quadratic relationship between the classical spectral gap δ and the phase gap of the quantum walk: while the classical mixing time scales as $1/\delta$, the quantum phase estimation converges in time $1/\sqrt{\delta}$.

The cost of the quantum walk search is given by the following theorem:

Theorem 1 (Quantum walk search on graphs [23]). *Let $G = (V, E)$ be a regular graph with spectral gap $\delta > 0$, and suppose at least an $\epsilon > 0$ fraction of its vertices are marked. For a quantum walk on G with setup, update, and checking costs denoted by T_S, T_U, and T_C, respectively, there exists a quantum algorithm that finds a marked vertex with high probability in time:*

$$T_{QW} = T_S + \frac{1}{\sqrt{\epsilon}}\left(\frac{T_U}{\sqrt{\delta}} + T_C\right). \tag{9}$$

The graphs used in our quantum walk search algorithms are the Cartesian products of Johnson graphs.

Definition 3 (Johnson graph). *The Johnson graph, denoted as $J(N,k)$, is an undirected graph whose vertices correspond to all k-element subsets of a universe of size N. Two vertices S and S' are adjacent if and only if $|S \cap S'| = k-1$. This implies that S' can be obtained from S by replacing exactly one element. The spectral gap of $J(N,k)$ is given by*

$$\delta(J(N,k)) = \frac{N}{k(N-k)}. \tag{10}$$

Definition 4 (Cartesian product of Johnson graphs). *Let $J^m(N,k)$ denote the Cartesian product of m copies of the Johnson graph $J(N,k)$. Each vertex in $J^m(N,k)$ is an m-tuple $(S_1,\ldots,S_m)$ of k-element subsets. Two vertices $(S_1,\ldots,S_m)$ and $(S'_1,\ldots,S'_m)$ are adjacent if and only if they differ in exactly one coordinate i, and for that index, $|S_i \cap S'_i| = k-1$. The spectral gap satisfies*

$$\delta(J^m(N,k)) \geq \frac{1}{m} \cdot \delta(J(N,k)) = \Omega\left(\frac{1}{k}\right). \tag{11}$$

Remark 1 (Quantum Walk Heuristics [18]). *A key difference in the design of classical versus quantum subroutines lies in termination guarantees: quantum updates must complete within a predetermined time bound, while classical algorithms—though with negligible probability—may exhibit unbounded worst-case running times. This fundamental issue was systematically examined in (Section 6 in [24]), where it was demonstrated that enforcing termination within a polynomial multiple of the expected runtime introduces only negligible distortion to the resulting quantum*

states. Consequently, for complexity analysis purposes, evaluating quantum walk efficiency via expected costs remains valid up to polynomial overhead.

3. LSH-Based MitM Attacks for Ternary LWE

The combinatorial attack on ternary LWE by Kirshanova and May [17] introduces a MitM approach improved with LSH and representation techniques. We refer to the two instantiations of their attack as Rep-0 and Rep-1, which we will introduce in turn, as they form the basis for our quantum algorithms.

3.1. Rep-0: First Instantiation of LSH-Based MitM Attacks

Let $s \in \mathcal{T}^n(w/2)$ be a ternary LWE secret vector of weight w. Rep-0 represents the secret vector $s \in \mathcal{T}^n(w/2)$ into two vectors $s_1^{(1)}, s_2^{(1)} \in \mathcal{T}^n(w/4)$ such that $s = s_1^{(1)} + s_2^{(1)}$. Such a pair $(s_1^{(1)}, s_2^{(1)})$ is called a representation of s. There are $R = \binom{w/2}{w/4}^2$ such representations in total. The key idea is to leverage this representation to split the LWE equation $As = b + e \mod q$ into two parts:

$$As_1^{(1)} = -As_2^{(1)} + b + e \bmod q. \tag{12}$$

In the standard MitM approach, one would enumerate all possible $s_1^{(1)}$ and $s_2^{(1)}$ and look for pairs where $As_1^{(1)}$ and $-As_2^{(1)} + b$ differ by a ternary error vector e. However, this requires checking all pairs, which is inefficient. Instead, Rep-0 uses a tree-based search with LSH to efficiently find matching pairs.

Rep-0 constructs a depth-2 search tree as illustrated in Figure 1. At the top level (level 2), each $s_i^{(1)}$ is further splitted into two vectors, yielding a total of four lists:

$$\begin{aligned} L_1^{(2)} &= \left\{ s_1^{(2)} \in \mathcal{T}^{n/2}(w/8) \times 0^{n/2} \right\}, \\ L_2^{(2)} &= \left\{ s_2^{(2)} \in 0^{n/2} \times \mathcal{T}^{n/2}(w/8) \right\}, \\ L_3^{(2)} &= \left\{ s_3^{(2)} \in \mathcal{T}^{n/2}(w/8) \times 0^{n/2} \right\}, \\ L_4^{(2)} &= \left\{ s_4^{(2)} \in 0^{n/2} \times \mathcal{T}^{n/2}(w/8) \right\}. \end{aligned} \tag{13}$$

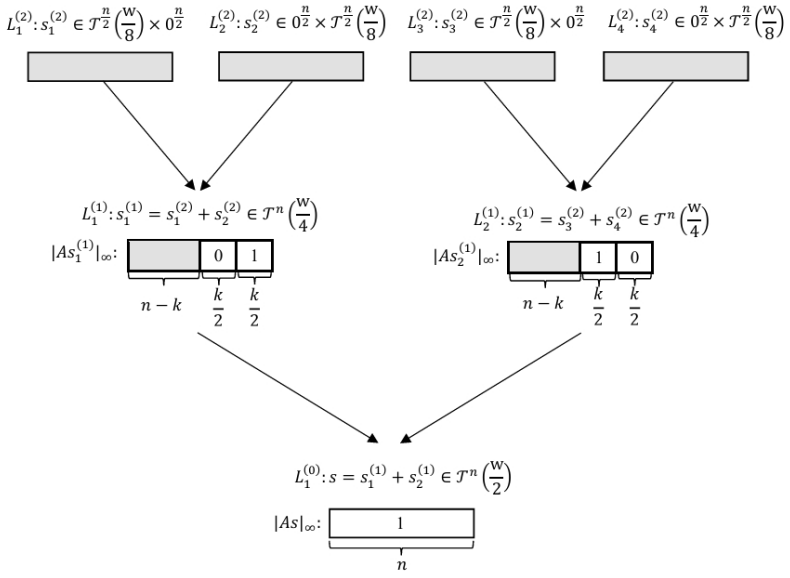

Figure 1. The tree structure of Rep-0. The shaded regions represent the parts where the matching operation has not been performed.

The merge process proceeds level by level. At each step, we perform both exact matching and approximate matching, employing the LSH-Odlyzko method from Section 2.3 for the latter. To ensure that expect one solution survives after merging, the parameter k must satisfy $q^{\frac{k}{2}}\left(\frac{q}{3}\right)^{\frac{k}{2}} \approx R$. More precisely, k is calculated as

$$k = \left\lfloor \frac{\log_2 R}{\log_2 q - 0.5 \log_2 3} \right\rfloor. \tag{14}$$

The pseudocode of Rep-0 is presented in Algorithm 2.

Algorithm 2 Rep-0

Input: An LWE public key $(A, b) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$, weight w

Output: Secret vector $s \in \mathcal{T}^n(w/2)$ satisfying $As - b \bmod q \in \mathcal{T}^n$ of $\perp$ if no such secret is found

1: Enumerate four level-2 lists $L_1^{(2)}, L_2^{(2)}, L_3^{(2)}, L_4^{(2)}$ as above.
2: Choose k that satisfies $q^{\frac{k}{2}}\left(\frac{q}{3}\right)^{\frac{k}{2}} \approx \binom{w/2}{w/4}^2$.
3: Merge $L_1^{(2)}$ and $L_2^{(2)}$ into $L_1^{(1)}$ using approximate matching on $k/2$ coordinates and exact matching on the other $k/2$ coordinates.
4: Merge $L_3^{(2)}$ and $L_4^{(2)}$ into $L_2^{(1)}$ similarly.
5: Merge $L_1^{(1)}$ and $L_2^{(1)}$ into $L_0^{(1)}$ using LSH on the remaining $n-k$ coordinates, and filter out any vector not in $\mathcal{T}^n(w/2)$.
6: If $L_1^{(0)}$ is non-empty, return any $s \in L_1^{(0)}$; otherwise, return $\perp$.

After Step 3–4 in Algorithm 2, we obtain two level-1 lists with the following structure:

$$\begin{aligned} L_1^{(1)} &= \left\{ s_1^{(1)} \in \mathcal{T}^n(w/4) : As_1^{(1)} \in \mathbb{Z}_q^{n-k} \times 0^{k/2} \times \{\pm 1, 0\}^{k/2} \right\}, \\ L_2^{(1)} &= \left\{ s_2^{(1)} \in \mathcal{T}^n(w/4) : As_2^{(1)} \in \mathbb{Z}_q^{n-k} \times \{\pm 1, 0\}^{k/2} \times 0^{k/2} \right\}. \end{aligned} \tag{15}$$

This means that vectors in $L_1^{(1)}$ satisfy $As_1^{(1)} = 0$ on $k/2$ coordinates (exact matching) and $As_1^{(1)} \in \{\pm 1, 0\}$ on another $k/2$ coordinates (approximate matching), while vectors in $L_2^{(1)}$ satisfy $As_2^{(1)} \in \{\pm 1, 0\}$ on $k/2$ coordinates and $As_2^{(1)} = 0$ on the other $k/2$ coordinates.

Step 5 in Algorithm 2 merges $L_1^{(1)}$ and $L_2^{(1)}$ using LSH-Odlyzko on the $n-k$ coordinates that were not involved in the previous matching steps. This finds pairs $(s_1^{(1)}, s_2^{(1)})$ such that $As_1^{(1)}$ and $b - As_2^{(1)}$ are 1-close in the ℓ_∞-norm, which ensures that $s = s_1^{(1)} + s_2^{(1)}$ satisfies $As = b + e$ with $e \in \mathcal{T}^n$.

The correctness of Algorithm 2 follows from the representation technique: each valid secret s has multiple representations as $s_1^{(1)} + s_2^{(1)}$, and we expect that one of these representations survives the merging process. The LSH-based matching efficiently identifies pairs that satisfy Equation (12) without explicitly checking all possible pairs.

Let $L^{(j)}$ denote the common size of the lists at level j, for $j = 0, 1, 2$. We have

$$\begin{aligned} L^{(2)} &= \binom{n/2}{w/8, w/8, \cdot}, \\ L^{(1)} &= \left(L^{(2)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}}, \\ L^{(0)} &= \frac{\left(L^{(1)}\right)^2}{2^{n-k}}. \end{aligned} \tag{16}$$

The space complexity is determined by the maximum list size encountered during the merging process:

$$S_{\text{Rep-0}} = \max\left\{L^{(2)}, L^{(1)}, L^{(0)}\right\}. \tag{17}$$

The time complexity is dominated by the largest list size and the cost of LSH-based merging at each level:

$$\begin{aligned} T_{\text{Rep-0}} &= \max\left\{S_{\text{Rep-0}}, T_{\mathsf{LSH+Exact}}\left(L^{(2)}, \frac{k}{2}, \frac{k}{2}, B\right), T_{\mathsf{LSH+Exact}}\left(L^{(1)}, n-k, 0, \frac{q}{2}\right)\right\} \\ &= \max\left\{S_{\text{Rep-0}},\ \left(L^{(2)}\right)^2 \cdot \left(\frac{B^2}{(B-1)q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}} \cdot n\log n,\ \frac{\left(L^{(1)}\right)^2 \cdot n\log n}{2^{n-k}}\right\}. \end{aligned} \tag{18}$$

Here, the last equality follows from Equation (5). The bucket size B is a parameter that needs to be optimized.

3.2. Rep-1: Optimized LSH-Based MitM Attacks

Rep-1 generalizes Rep-0 by allowing more flexible decompositions. In addition to representing non-zero entries of s as sums of non-zero entries in $s_1^{(1)}$ and $s_2^{(1)}$, Rep-1 also allows zero entries in s to be represented as $1+(-1)$ or $(-1)+1$. This increases the number of representations and allows for a more efficient search tree.

We now describe Rep-1 with depth-3, as illustrated in Figure 2. The ternary LWE secret vector $s \in \mathcal{T}^n(w/2)$ is decomposed as

$$s = s_1^{(1)} + s_2^{(1)}, \quad \text{where } s_1^{(1)}, s_2^{(1)} \in \mathcal{T}^n(w/4 + \varepsilon[1]). \tag{19}$$

Here, $\varepsilon[1] \geq 0$ is a parameter that controls the number of additional ± 1 entries used to represent zeros in s. The total number of representations at level 1 is

$$R^{(1)} = \binom{w/2}{w/4}^2 \cdot \binom{n-w}{\varepsilon[1], \varepsilon[1], \cdot}. \tag{20}$$

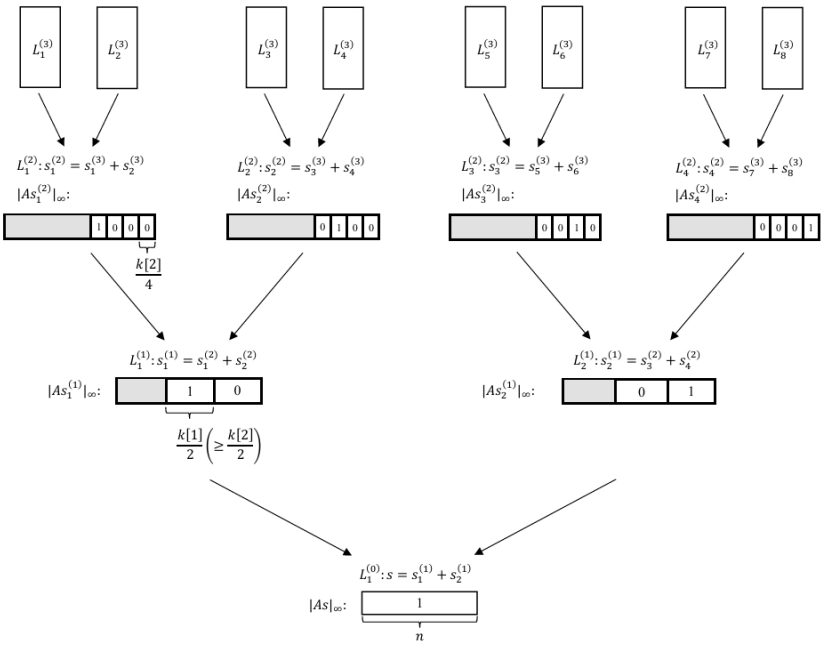

Figure 2. The tree structure of Rep-1 with depth 3. The shaded regions represent the parts where the matching operation has not been performed.

Each $s_i^{(1)}$ is further decomposed into two vectors at level 2:

$$s_i^{(1)} = s_{2i-1}^{(2)} + s_{2i}^{(2)}, \quad \text{where } s_{2i-1}^{(2)}, s_{2i}^{(2)} \in \mathcal{T}^n\left(\frac{w}{8} + \frac{\varepsilon[1]}{2} + \varepsilon[2]\right). \tag{21}$$

The number of representations at level 2 is

$$R^{(2)} = \binom{w/4 + \varepsilon[1]}{w/8 + \varepsilon[1]/2}^2 \cdot \binom{n - w/2 - 2\varepsilon[1]}{\varepsilon[2], \varepsilon[2], \cdot}. \tag{22}$$

Finally, at level 3, Rep-1 enumerates the vectors $s_j^{(3)} \in \mathcal{T}^{n/2}\left(\frac{w}{16} + \frac{\varepsilon[1]}{4} + \frac{\varepsilon[2]}{2}\right)$ directly. The merging process uses a combination of exact and approximate matching at each level, with the number of matched coordinates $k[i]$ chosen to satisfy

$$k[i] = \left\lfloor \frac{\log_2 R^{(i)}}{\log_2 q - 0.5^i \log_2 3} \right\rfloor. \tag{23}$$

The pseudocode of Rep-1 is outlined in Algorithm 3.

Algorithm 3 Rep-1 with Depth 3

Input: An LWE public key $(A, b) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$, weight w

Output: Secret vector $s \in \mathcal{T}^n(w/2)$ satisfying $As - b \bmod q \in \mathcal{T}^n$ of $\perp$ if no such secret is found

1: Enumerate eight level-3 lists $L_1^{(3)}, \ldots, L_8^{(3)}$ with weights as above.
2: Compute $k[2]$ from $R^{(2)}$ and merge pairs into four level-2 lists using exact and approximate matching.
3: Compute $k[1]$ from $R^{(1)}$ and merge into two level-1 lists using exact and approximate matching.
4: Merge level-1 lists into $L_1^{(0)}$ using LSH on the remaining $n - k[1]$ coordinates, and filter out any vector not in $\mathcal{T}^n(w/2)$.
5: If $L_1^{(0)}$ is non-empty, return any $s \in L_1^{(0)}$; otherwise, return $\perp$.

Let $L^{(j)}$ denote the common size of the lists at level j, for $j = 0, 1, 2, 3$. We have

$$\begin{aligned}
L^{(3)} &= \binom{n/2}{g(w, \varepsilon[1], \varepsilon[2]), g(w, \varepsilon[1], \varepsilon[2]), \cdot}, \\
L^{(2)} &= \left(L^{(3)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}}, \\
L^{(1)} &= \left(L^{(2)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}}, \\
L^{(0)} &= \frac{\left(L^{(1)}\right)^2}{2^{n-k}},
\end{aligned} \tag{24}$$

where $g(x, y, z) := x/16 + y/4 + z/2$.

The space complexity is determined by the maximum list size encountered during the merging process:

$$S_{\text{Rep-1}} = \max\left\{L^{(3)}, L^{(2)}, L^{(1)}, L^{(0)}\right\}. \tag{25}$$

The time complexity is dominated by the largest list size and the cost of LSH-based merging at each level:

$$T_{\text{Rep-1}} = \max\left\{T^{(3)}, T^{(2)}, T^{(1)}, T^{(0)}\right\}, \tag{26}$$

where $T^{(i)}$ is the cost at level i for $i = 0, 1, 2, 3$,

$$\begin{aligned}
T^{(3)} &= L^{(3)}, \\
T^{(2)} &= \max\left\{ L^{(3)}, L^{(2)}, \left(L^{(2)}\right)^2 \cdot \left(\frac{B[2]^2}{(B[2]-1)q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}} \cdot n \log n \right\}, \\
T^{(1)} &= \max\left\{ L^{(2)}, L^{(1)}, \left(L^{(1)}\right)^2 \cdot \left(\frac{B[1]^2}{(B[1]-1)q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot n \log n \right\}, \\
T^{(0)} &= \max\left\{ L^{(1)}, L^{(0)}, \frac{\left(L^{(1)}\right)^2}{2^{n-k}} \right\}.
\end{aligned} \tag{27}$$

The parameters $\varepsilon[1], \varepsilon[2], B[1], B[2]$ are optimized to minimize the overall complexity.

4. LSH-Based MitM Attacks on Ternary LWE via Quantum Walks

In this work, we develop new LSH-based MitM quantum algorithms for ternary LWE, which transform the key recovery problem into a graph search task amenable to quantum walks. We first present QRep-0, the quantum adaptation of Rep-0, to establish the foundational approach. We then introduce QRep-1, which builds upon Rep-1 and achieves a lower complexity through enhanced representation techniques.

4.1. QRep-0: Foundational Instantiation of LSH-Based MitM Quantum Algorithms

The ternary LWE problem aims to recover a ternary secret vector $s \in \mathcal{T}^n(w/2)$ satisfying $As = b + e \mod q$, with a ternary error vector e. We cast this cryptographic recovery problem as a graph search task solvable by quantum walks, where marked vertices represent valid LWE solutions.

Rep-0 features a level-2 search tree, as illustrated in Figure 1. The quantum walk operates on a graph constructed as the Cartesian product of four identical Johnson graphs. The construction begins with the four level-2 lists $L_j^{(2)} (1 \le j \le 4)$, each of size $L^{(2)} = \binom{n/2}{w/8, w/8, \cdot}$. From these, we define restricted subsets $U_j^{(2)} \subseteq L_j^{(2)}$ of size $U^{(2)} := (L^{(2)})^\gamma$, where $\gamma \in [0, 1]$ is a parameter to be optimized. Setting $N = L^{(2)}$ and $k = U^{(2)}$, the graph is formally given by

$$G_{\text{QRep-0}}(V_{\text{QRep-0}}, E_{\text{QRep-0}}) := J(N, k) \times J(N, k) \times J(N, k) \times J(N, k). \tag{28}$$

Each vertex $u \in V_{\text{Rep-0}}$ is a 4-tuple $(U_1^{(2)}, U_2^{(2)}, U_3^{(2)}, U_4^{(2)})$. Two vertices u and v are adjacent if and only if, for some index j, the components $U_j^{(2)}$ of u and v differ by exactly one element, while the other three components are identical.

The data structure associated with a vertex in $G_{\text{QRep-0}}$ comprises all intermediate lists $U_j^{(i)}$ generated during the execution of Rep-0, which takes the vertex's 4-tuple $(U_1^{(2)}, \ldots, U_4^{(2)})$ as its level-2 input lists. Here, $U_j^{(i)}$ corresponds to the j-th list at level-i, for $0 \le i \le 1$ and $1 \le j \le 2^i$.

A vertex is marked if its resultant top-level list $U_1^{(0)}$ contains at least one valid ternary secret s satisfying $s \in \mathcal{T}^n(w/2)$ and $As - b \mod q \in \mathcal{T}^n$. This direct correspondence guarantees that finding any marked vertex solves the original LWE problem.

We begin by explaining the role of the parameter γ: it governs the trade-off between the setup cost of QRep-0 and the cost of the quantum walk required to reach a marked vertex.

When γ is large (close to 1), the setup cost dominates the overall complexity. In the extreme case of $\gamma = 1$, all vertices become marked (since $U^{(2)} = L^{(2)}$), and QRep-0 reduces to the classical Rep-0 method, no longer relying on quantum walk.

Conversely, when γ is small (close to 0), the setup cost is minimized, but the fraction of marked vertices becomes negligible. As a result, the cost of amplifying these marked vertices via quantum walk becomes the dominant factor in the overall complexity.

We now analyze the quantum resources—specifically, circuit width and depth—required by QRep-0 as detailed in Algorithm 4.

Algorithm 4 QRep-0

Input: An LWE public key (A, b), weight w

Output: Secret vector $s \in \mathcal{T}^n(w/2)$ satisfying $As - b \bmod q \in \mathcal{T}^n$ or $\perp$ if no such secret is found.

1: Prepare the initial state (normalization omitted):

$$\sum_{u \in V_{\text{Rep-0}}} |u\rangle|p_u\rangle|d(u)\rangle.$$

2: Repeat $t_1 = \mathcal{O}(1/\sqrt{\epsilon})$ times:

(2.1) Apply the phase flip if u is marked:

$$|u\rangle|p_u\rangle|d(u)\rangle \mapsto \begin{cases} -|u\rangle|p_u\rangle|d(u)\rangle, & \text{if } u \text{ is marked,} \\ |u\rangle|p_u\rangle|d(u)\rangle, & \text{otherwise.} \end{cases}$$

(2.2) Perform the quantum walk for $t_2 = \mathcal{O}(1/\sqrt{\delta})$ steps.

3: Measure the register $|d(u)\rangle$.

4: If $U_1^{(0)}$ is non-empty, return any $s \in U_1^{(0)}$; otherwise, return $\perp$.

Let $U^{(i)}$ denote the size of level-i lists in QRep-0, for $i = 0, 1$. These sizes follow the recurrence relations:

$$\begin{aligned} U^{(2)} &= \left(L^{(2)}\right)^{\gamma} = \binom{n/2}{w/8, w/8, \cdot}^{\gamma}, \\ U^{(1)} &= \left(U^{(2)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}}, \\ U^{(0)} &= \frac{\left(U^{(1)}\right)^2}{2^{n-k}}. \end{aligned} \tag{29}$$

The quantum circuit width, which corresponds to the space complexity, is determined by three registers: the vertex register $|u\rangle$ encoding a 4-tuple of subsets $(U_1^{(2)}, \ldots, U_4^{(2)})$, requiring $4U^{(2)}$ qubits; the coin register $|p_u\rangle$ of similar size; and the data register $|d(u)\rangle$, which dominates the space complexity with $2U^{(1)} + U^{(0)}$ qubits. Hence, the overall circuit width (space complexity) is given by

$$S_{\text{QRep-0}} = \mathcal{O}\left(\max\left\{U^{(2)}, U^{(1)}, U^{(0)}\right\}\right). \tag{30}$$

The circuit depth, which corresponds to the time complexity, follows the quantum walk complexity:

$$T_{\text{QRep-0}} = T_S + \frac{1}{\sqrt{\epsilon}}\left(\frac{1}{\sqrt{\delta}} T_U + T_C\right), \tag{31}$$

where T_S encompasses the initial state preparation and data structure construction, T_U captures the cost per quantum walk step including data updates, and $T_C = \mathcal{O}(1)$ represents the phase oracle for marked vertices.

From Equation (11), the spectral gap of $G_{\text{QRep-0}}$ is

$$\delta = \Omega\left(\frac{1}{k}\right) = \Omega\left(\frac{1}{U^{(2)}}\right). \tag{32}$$

Since the classical Rep-0 algorithm yields, in expectation, a single element in $L_1^{(0)}$, the fraction of marked vertices ϵ corresponds to the probability that all four subsets $U_j^{(2)}$ contain the necessary elements to reconstruct s. This fraction is given by

$$\epsilon = \left(\frac{U^{(2)}}{L^{(2)}}\right)^4 = \left(L^{(2)}\right)^{4(\gamma-1)}. \tag{33}$$

To determine the circuit depth, we analyze the setup cost T_S and update cost T_U for the quantum walk in QRep-0. Following the methodology in Section 6 of [24], these costs correspond to the classical computational complexity of constructing and updating the hierarchical data structure used in Rep-0.

The setup cost T_S involves creating the hierarchical lists according to Rep-0. The level-2 lists $U_j^{(2)}$ (for $j = 1, 2, 3, 4$) are constructed by randomly sampling $U^{(2)}$ elements from $L_j^{(2)}$. The level-1 lists such as $U_1^{(1)}$ are formed by mergeing $U_1^{(2)}$ and $U_2^{(2)}$ via exact matching and LSH. Finally, the level-0 list $U_1^{(0)}$ is built from the two level-1 lists using LSH. Therefore,

$$\begin{aligned} T_S &= \max\left\{S_{\text{QRep-0}}, T_{\mathsf{LSH+Exact}}\left(U^{(2)}, \frac{k}{2}, \frac{k}{2}, B\right), T_{\mathsf{LSH+Exact}}\left(U^{(1)}, n-k, 0, \frac{q}{2}\right)\right\} \\ &= \max\left\{S_{\text{QRep-0}},\ \left(U^{(2)}\right)^2 \cdot \left(\frac{B^2}{(B-1)q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}} \cdot n\log n, \frac{\left(U^{(1)}\right)^2 \cdot n\log n}{2^{n-k}}\right\}. \end{aligned} \tag{34}$$

The update cost T_U involves involves inserting or deleting an element from one of the level-2 lists. Without loss of generality, we assume the element to be updated, denoted as x, belongs to $U_1^{(2)}$. The update of x in $U_1^{(2)}$ can be performed in time $\mathcal{O}(1)$.

The insertion or deletion of x subsequently triggers updates in the level-1 list $U_1^{(1)}$, which is formed by merging $U_1^{(2)}$ and $U_2^{(2)}$. Specifically, this requires inserting or deleting all elements $z \in U_1^{(1)}$ that are constructed by pairing x with some $y \in U_2^{(2)}$, where x and y satisfy the matching conditions on k coordinates: approximate matching on $k/2$ coordinates and exact matching on the other $k/2$ coordinates.

To analyze the number of such elements y (and thus the corresponding z) and the time complexity of locating them, we employ the following theorem:

Theorem 2. *Given an element $x \in \mathbb{Z}_q^n$ and a list L of size $|L|$ with independent and identically distributed elements drawn uniformly from $\mathbb{Z}_q^n$, there exists a classical algorithm that can find a $(1 - o(1))$-fraction of $y \in L$ satisfying $\max_{i\in I} |x_i - y_i| = 1$ for some set I of size k_1 and $\max_{i\in J} |x_i - y_i| = 0$ for some set J of size k_2. The time complexity of this algorithm is*

$$\max\left\{|L| \cdot \left(\frac{3}{q}\right)^{k_1} \cdot \left(\frac{1}{q}\right)^{k_2}, |L| \cdot \left(\frac{B^2}{(B-1)q}\right)^{k_1} \cdot \left(\frac{1}{q}\right)^{k_2} \cdot n\log n, \right\} \tag{35}$$

where the first term corresponds to the expected number of elements y that meet the above conditions.

Proof. A detailed derivation of this result is provided in Appendix A. □

Applying Theorem 2, the time required to update $U_1^{(1)}$ due to the modification of x is given by

$$T_U^{(1)} = \max\left\{ U^{(2)} \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}}, U^{(2)} \cdot \left(\frac{B^2}{(B-1)q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}} \cdot n \log n, \right\} \tag{36}$$

The number of elements $z \in U_1^{(1)}$ that need to be updated is $U^{(2)} \cdot (3/q)^{k/2} \cdot (1/q)^{k/2}$. For each such z, applying Theorem 2 again, the time required to update the level-0 list $U^{(0)}$ is $(U^{(1)} \cdot n \log n)/2^{n-k}$. Therefore, the total update time for the level-0 list is

$$T_U^{(0)} = U^{(2)} \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}} \cdot \frac{U^{(1)} \cdot n \log n}{2^{n-k}} \tag{37}$$

Consequently, the overall update cost is given by

$$T_U = \max\left\{ \mathcal{O}(1), T_U^{(1)}, T_U^{(0)} \right\}. \tag{38}$$

An important observation is that T_S and T_U satisfy the relation $T_S = U^{(2)} T_U$. Combining this observation with Equations (31)–(33) and $T_C = \mathcal{O}(1)$, we obtain the circuit depth (time complexity) of QRep-0:

$$\begin{aligned} T_{\text{QRep-0}} &= T_S + \frac{1}{\sqrt{\epsilon}} \left(\frac{1}{\sqrt{\delta}} T_U + T_C \right) \\ &= \mathcal{O}\left(U^{(2)} T_U + \left(\frac{L^{(2)}}{U^{(2)}} \right)^2 \left(\sqrt{U^{(2)}} T_U + 1 \right) \right) \\ &= \left(L^{(2)} \right)^{\gamma} T_U + \left(L^{(2)} \right)^{2(1-\gamma)+\frac{\gamma}{2}} T_U. \end{aligned} \tag{39}$$

The parameter γ is chosen to minimize the overall time complexity by balancing the two dominant terms in the expression: the setup cost $\left(L^{(2)}\right)^{\gamma} T_U$ and the cost of performing the quantum walk to find a marked vertex $\left(L^{(2)}\right)^{2(1-\gamma)+\frac{\gamma}{2}} T_U$.

To achieve this balance, we set the exponents of the two terms equal to each other:

$$\gamma = 2(1-\gamma) + \gamma/2. \tag{40}$$

Solving this equation for γ yields the optimal value $\gamma = 4/5$.

4.2. Concrete Security Analysis of QRep-0

We now present the concrete security analysis of our foundational QRep-0 algorithm and compare it with the vHKM's QRep-0. Table 1 evaluates both methods on major ternary-LWE-based cryptographic schemes that were also analyzed by van Hoof et al. [18], including NTRU variants (Encrypt and Prime), BLISS signatures, and GLP signatures. These schemes all rely on the hardness of ternary LWE, where both the secret and error vectors have entries in $\{-1, 0, 1\}$.

The parameter triple (n, q, w) for each scheme specifies the polynomial dimension n (ring dimension), the modulus q, and the weight w (number of non-zero entries in the ternary secret vector). The complexity is measured in $\log_2 T$, where T represents the time

complexity of the attack. For quantum walk-based combinatorial attacks, including both our approach and vHKM's method, the time complexity T and quantum space complexity M (measured in qubits) are essentially equivalent, i.e., $\log_2 T = \log_2 M$. This resource equivalence arises because both methods rely on quantum walks over Johnson graphs that require storing the entire quantum state during computation.

Table 1 provides concrete complexity estimates for QRep-0. Across all cryptographic schemes evaluated, our QRep-0 implementation demonstrates superior performance over the vHKM variant in all cases except for BLISS I+II. As will be shown subsequently, when enhanced with improved representation techniques, our quantum algorithm achieves comprehensive advantage over the vHKM approach.

Table 1. Comparative analysis of bit complexity: our QRep-0 vs. vHKM's QRep-0.

Scheme	Parameters (n, q, w)	Our QRep-0 $\log_2 T$	vHKM's QRep-0 [18] $\log_2 T$
NTRU-Enc	(509, 2048, 254)	212	230
	(677, 2048, 254)	241	254
	(821, 4096, 510)	390	425
	(701, 8192, 468)	347	377
NTRU-Prime	(653, 4621, 288)	251	271
	(761, 4591, 286)	273	284
	(857, 5167, 322)	319	322
BLISS I+II	(512, 12,289, 154)	194	174
GLP I	(512, 8,383,489, 342)	252	264

As shown in Table 1, our QRep-0 implementation outperforms the vHKM variant in all cases except for BLISS I+II. As will be shown subsequently, when enhanced with improved representation techniques, our quantum algorithm achieves comprehensive advantage over the vHKM approach.

4.3. QRep-1: Improved LSH-Based MitM Quantum Algorithms

Following the introduction of the foundational instantiation, we proceed to QRep-1—a superior instantiation of our quantum algorithm which leverages a more powerful representation technique. In this instantiation, the depth of the search tree is also treated as an optimization parameter.

We begin by describing QRep-1 with a depth of 3; its search tree structure is illustrated in Figure 2. Since the top level (level 3) contains 8 lists, the quantum walk is performed on the Cartesian product of 8 identical Johnson graphs. Consider subsets $U_j^{(3)} \subseteq L_j^{(3)}$ of size $U^{(3)} = \left(L^{(3)}\right)^{\gamma}$, where parameter $\gamma \in [0, 1]$ balances the quantum walk cost.

The spectral gap of the graph $J^8(L^{(3)}, U^{(3)})$ is $\Omega(1/U^{(3)})$, and the fraction of marked vertices is $(U^{(3)}/L^{(3)})^8$.

Analogously to Section 4.1, we can recursively derive the list sizes from level 2 down to level 0:

$$\begin{aligned} U^{(2)} &= \left(U^{(3)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}}, \\ U^{(1)} &= \left(U^{(2)}\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}}, \\ U^{(0)} &= \frac{\left(U^{(1)}\right)^2}{2^{n-k}}. \end{aligned} \tag{41}$$

The space complexity of QRep-1 with depth 3 is

$$S_{\text{QRep-1}} = \mathcal{O}\left(\max\left\{U^{(3)}, U^{(2)}, U^{(1)}, U^{(0)}\right\}\right). \tag{42}$$

Similarly, the setup cost of the quantum walk for QRep-1 with depth 3 is

$$T_S = \max\left\{T_S^{(3)}, T_S^{(2)}, T_S^{(1)}, T_S^{(0)}\right\}, \tag{43}$$

where $T_S^{(i)}$ is the setup cost at level i for $i = 0, 1, 2, 3$,

$$\begin{aligned}
T_S^{(3)} &= U^{(3)}, \\
T_S^{(2)} &= \max\left\{U^{(3)}, U^{(2)}, \left(U^{(2)}\right)^2 \cdot \left(\frac{B[2]^2}{(B[2]-1)q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}} \cdot n\log n\right\}, \\
T_S^{(1)} &= \max\left\{U^{(2)}, U^{(1)}, \left(U^{(1)}\right)^2 \cdot \left(\frac{B[1]^2}{(B[1]-1)q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot n\log n\right\}, \\
T_S^{(0)} &= \max\left\{U^{(1)}, U^{(0)}, \frac{\left(U^{(1)}\right)^2}{2^{n-k}}\right\}.
\end{aligned} \tag{44}$$

The parameters $\varepsilon[1], \varepsilon[2], B[1], B[2]$ are optimized to minimize the overall complexity.

The update cost of the quantum walk for QRep-1 with depth 3 is

$$T_U = \max\left\{T_U^{(3)}, T_U^{(2)}, T_U^{(1)}, T_U^{(0)}\right\}, \tag{45}$$

where $T_U^{(i)}$ represents the update cost at level i for $i = 0, 1, 2, 3$.

Following an analysis similar to that in Section 4.1 and applying Theorem 2, we establish the relationship between the update cost and setup cost at each level: $T_S^{(i)} = U^{(3)} \cdot T_U^{(i)}$, and consequently $T_S = U^{(3)} \cdot T_U$. This relationship enables us to estimate the time complexity of QRep-1 as follows:

$$\begin{aligned}
T_{\text{QRep-1}} &= T_S + \frac{1}{\sqrt{\epsilon}}\left(\frac{1}{\sqrt{\delta}}T_U + T_C\right) \\
&= \mathcal{O}\left(U^{(3)}T_U + \left(\frac{L^{(3)}}{U^{(3)}}\right)^4\left(\sqrt{U^{(3)}}T_U + 1\right)\right) \\
&= \left(L^{(3)}\right)^{\gamma}T_U + \left(L^{(3)}\right)^{4(1-\gamma)+\frac{\gamma}{2}}T_U.
\end{aligned} \tag{46}$$

Similarly, for QRep-1, we balance the setup cost $\left(L^{(3)}\right)^{\gamma}T_U$ and the search cost $\left(L^{(3)}\right)^{4(1-\gamma)+\frac{\gamma}{2}}T_U$ to find the optimal γ. The balancing equation is $\gamma = 4(1-\gamma) + \frac{\gamma}{2}$. Solving this equation gives the optimal value $\gamma = 8/9$.

A similar analysis at depth 4 gives $\gamma = 16/17$. This demonstrates that the performance of QRep-1 converges to that of Rep-1 as the depth of the classical search tree increases.

4.4. Concrete Security Analysis of QRep-1

This subsection presents a comprehensive security analysis of our optimized QRep-1 algorithm, comparing it against previous quantum combinatorial attacks and lattice-based quantum sieving methods. Table 2 extends our evaluation to include additional ternary-

LWE-based schemes from the NTRU-IEEE family, using the same methodology established in Section 4.2.

Table 2. Comparative analysis of bit complexity for ternary-LWE schemes.

Scheme	Parameters (n, q, w)	vHKM's QRep-1 [18] $\log_2 T$	vHKM's QRep-1 [18] $\log_S T$ *	Our QRep-1 $\log_2 T$	Our QRep-1 $\log_{\mathcal{S}} T$	Quantum Sieving † [25] $\log_2 T$
NTRU-IEEE	(401, 2048, 226)	–	–	144	0.234	$70 + c$
	(449, 2048, 268)	–	–	149	0.214	$82 + c$
	(677, 2048, 314)	–	–	202	0.206	$134 + c$
	(1087, 2048, 240)	–	–	220	0.207	$215 + c$
	(541, 2048, 98)	–	–	113	0.244	$96 + c$
	(613, 2048, 110)	–	–	132	0.254	$112 + c$
	(887, 2048, 162)	–	–	169	0.221	$174 + c$
	(1171, 2048, 212)	–	–	221	0.220	$243 + c$
	(659, 2048, 76)	–	–	110	0.267	$118 + c$
	(761, 2048, 84)	–	–	114	0.247	$140 + c$
	(1087, 2048, 126)	–	–	167	0.245	$214 + c$
	(1499, 2048, 158)	–	–	203	0.231	$300 + c$
NTRU-Enc	(509, 2048, 254)	188	0.249	155	0.205	$95 + c$
	(677, 2048, 254)	223	0.249	177	0.198	$133 + c$
	(821, 4096, 510)	320	0.248	290	0.225	$159 + c$
	(701, 8192, 468)	278	0.251	239	0.216	$122 + c$
NTRU-Prime	(653, 4621, 288)	225	0.243	184	0.199	$116 + c$
	(761, 4591, 286)	245	0.244	185	0.184	$139 + c$
	(857, 5167, 322)	274	0.242	217	0.192	$158 + c$
BLISS I+II	(512, 12,289, 154)	149	0.249	133	0.222	$79 + c$
GLP I	(512, 8,383,489, 342)	193	0.240	169	0.210	$38 + c$

* $\mathcal{S}$ is the size of the search space. † The term c denotes a constant in quantum sieving complexities.

All evaluated schemes rely on the hardness of ternary LWE with secret and error vectors restricted to $\{-1, 0, 1\}$ entries. The parameter triple (n, q, w) for each scheme specifies the polynomial dimension, modulus, and weight of the secret vector, respectively. The complexity is measured in two forms: $\log_2 T$ represents the base-2 logarithm of the time complexity, while $\log_S T$ expresses the time complexity relative to the search space size S, providing a normalized measure of efficiency.

Our QRep-1 algorithm achieves a concrete complexity bound of approximately $\mathcal{S}^{0.225}$, substantially improving upon the $\mathcal{S}^{0.251}$ bound of vHKM's quantum combinatorial attack. This represents a significant reduction in the gap between asymptotic predictions and concrete performance. As shown in Table 2, our method demonstrates dramatic runtime improvements over vHKM, with speedup factors ranging from 2^{30} for NTRU-Enc-821 to 2^{60} for NTRU-Prime-761. For signature schemes, we observe speedups of 2^{16} for BLISS I+II and 2^{24} for GLP I.

The comparison with quantum sieving reveals a nuanced security landscape. While lattice-based attacks maintain advantages for most parameter sets, our combinatorial approach demonstrates superiority for small-weight parameters. This divergence highlights the importance of considering both attack paradigms in security assessments.

It is crucial to distinguish the resource models between these approaches. Quantum walk-based combinatorial attacks (including both our QRep-1 and vHKM's) exhibit equivalence between time and quantum space complexity due to the quantum walk framework requiring storage of the entire quantum state. In contrast, quantum sieving employs a hybrid classical-quantum model where only the locality-sensitive filtering step is quan-

tumized, resulting in a two-component resource model with both classical and quantum memory requirements.

Our combinatorial method leverages representation techniques derived from subset-sum or knapsack problems. This approach has strong theoretical foundations: for knapsack-type distributions, it can be rigorously proven that pathological instances constitute an exponentially small fraction, enabling the construction of provable probabilistic algorithms that avoid heuristics [15]. Experimental validation in prior work [15] confirms that observed runtimes align closely with theoretical predictions, providing enhanced precision in security estimates compared to lattice reduction heuristics that rely on unproven assumptions like the Geometric Series Assumption.

Our QRep-1 algorithm establishes the new state of the art for quantum combinatorial attacks on ternary LWE, offering both theoretical advances and practical security implications for post-quantum cryptography standardization.

5. LSH-Based MitM Quantum Algorithms with Polynomial Qubits

5.1. The Impact of Memory Models

Section 4 established the state of the art for quantum combinatorial attacks in terms of time complexity, operating under the strong QRAQM assumption. This model, which allows for coherent, efficient queries to an exponentially large quantum memory, is fundamental to the optimal performance of quantum walk-based approaches like our QRep0 and QRep1. However, the QRAQM model presents a significant barrier to practical implementability, as it requires maintaining and coherently manipulating a number of qubits that grows exponentially with the problem size—a requirement far beyond the capabilities of current and foreseeable quantum hardware due to constraints on qubit count, coherence time, and error rates.

In contrast, the QRACM model offers a more pragmatic alternative for early quantum cryptanalysis. The QRACM model assumes that large amounts of data can be stored in classical memory (which is cheap and abundant) and accessed in a quantum superposition—an operation often considered more feasible than full QRAQM. While this access may incur a polynomial overhead, it drastically reduces the quantum hardware requirements to only $\mathcal{O}(n)$ qubits. This distinction between QRAQM and QRACM is critical for assessing the practical viability of quantum algorithms. Our work directly addresses this implementability challenge by introducing a hybrid algorithm that functions efficiently under the more realistic QRACM assumption.

This leads to two distinct algorithmic paradigms for quantum combinatorial attacks: one prioritizing theoretical time efficiency (using quantum walks under QRAQM), and the other minimizing quantum hardware requirements for practical realization (using claw-finding under QRACM). The existence of this trade-off underscores that both asymptotic complexity and concrete resource constraints must be considered for a comprehensive security assessment.

Building on Benedikt's [19] application of the Chailloux et al. [20] (CNPS) quantum claw-finding technique to ternary LWE, we extend this line of work through a novel integration with the Kirshanova–May [17] framework. This hybrid construction operates efficiently with only $\mathcal{O}(n)$ qubits under the QRACM model. We now detail this approach.

5.2. Reformulating LWE Key Recovery as a Consistent Claw-Finding Problem

Let $s \in \mathcal{T}^n(w/2)$ be a ternary LWE secret vector, and consider a representation $s = s_1 + s_2$, where $s_1, s_2 \in \mathcal{T}^n(w/4 + \alpha)$ for some $\alpha \geq 0$. We rewrite the LWE identity given in Equation (12) for convenience:

$$As_1 = -As_2 + b + e \bmod q. \tag{47}$$

Let $S_1, S_2 \subseteq \mathcal{T}^n$. We define a pair of random, efficiently computable functions $f_1 : S_1 \to \mathbb{Z}_q^n$ and $f_2 : S_2 \to \mathbb{Z}_q^n$ (i.e., $T_{f_i} = \text{poly}(n)$) given by

$$f_1 : s_1 \mapsto As_1 \bmod q \ \text{ and } \ f_2 : s_2 \mapsto b - As_2 \bmod q. \tag{48}$$

A 1-close claw for f_1, f_2 is defined as a pair $(s_1, s_2) \in S_1 \times S_2$ that satisfies $\|f_1(s_1) - f_2(s_2)\|_\infty = 1$.

While every representation of the secret s yields a 1-close claw, the converse is not true: a 1-close claw pair does not necessarily satisfy $s_1 + s_2 \in \mathcal{T}^n(w/2)$. Define a 1-close claw pair (s_1, s_2) as consistent when $s_1 + s_2 \in \mathcal{T}^n(w/2)$. The representation of s is then in one-to-one correspondence with a consistent 1-close claw (s_1, s_2).

Consequently, the problem of recovering LWE secret s reduces to finding a consistent 1-close claw. The CNPS algorithm serves as an effective approach for solving the claw finding problem, as it utilizes only $O(n)$ qubits, which aligns with our objectives. We now present a variant of CNPS, recently proposed by Benedikt and tailored for addressing the LWE problem, as detailed in Algorithm 5.

Algorithm 5 Quantum Consistent 1-Close Claw Finding

Input: Random functions $f_1 : s_1 \mapsto As_1 \bmod q$ and $f_2 : s_2 \mapsto b - As_2 \bmod q$

Output: A consistent 1-close claw (s_1, s_2)

1: Define $S_r^{f_i} := \{s_i \in S_i \mid f_i(s_i) = 0 \bmod q^r\}$ for $i = 1, 2$.
Construct a sorted list (by the second entry):

$$L = \{(s_1, f_1(s_1)) \in S_r^{f_1} \times \mathbb{Z}_q^n\} \text{ with } |L| = 2^\ell.$$

2: Execute amplitude amplification using the following subroutines:

(2.1) $\mathcal{A}$: Prepare the quantum state

$$|\phi_r\rangle := \frac{1}{\sqrt{|S_r^{f_2}|}} \sum_{x_2 \in S_r^{f_2}} |s_2, f_2(s_2)\rangle |0\rangle.$$

(2.2) O_{f_L}: Apply the set-membership oracle O_{f_L}

$$O_{f_L}(|\phi_r\rangle) := \frac{1}{\sqrt{|S_r^{f_2}|}} \sum_{x_2 \in S_r^{f_2}} |s_2, f_2(s_2)\rangle |f_L(s_2)\rangle,$$

where

$$f_L(s_2) := \begin{cases} 1 & \text{if } \exists (s_1, f_1(s_1)) \in L \text{ s.t. } f_1(s_1) = f_2(s_2) \wedge s_1 + s_2 \in \mathcal{T}^n(\frac{w}{2}) \\ 0 & \text{else} \end{cases}.$$

3: Amplitude amplification eventually yields a state $|s_2, f_2(s_2)\rangle |1\rangle$. Measure this state.

4: For the measured s_2, search for a corresponding s_1 such that (s_1, s_2) forms a consistent 1-close claw.

Algorithm 5 identifies a consistent 1-close claw (s_1, s_2) with $O(n)$ qubits in two phases. First, it constructs a classically sorted list L. Then, it applies amplitude amplification to process a superposition of s_2 candidates. A set-membership oracle for L is used to find an L-suitable s_2, i.e., one for which a corresponding s_1 exists such that $(s_1, f_1(s_1)) \in L$ and (s_1, s_2) constitute a consistent 1-close claw.

Let R denote the number of consistent 1-close claws. The parameter r must satisfy the constraint $r \leq \log_q R$ to guarantee the existence of at least one claw satisfying $f_1(s_1) = 0 \bmod q^r$. Since a uniformly random $s_1 \in S_1$ can be extended to a consistent 1-close claw with probability $R/|S_1|$, the list size parameter ℓ must consequently satisfy $2^\ell \geq |S_1|/R$.

Regarding the construction of the classic list L, the original CNPS algorithm builds the list in an element-wise manner using Grover search. However, Benedikt observed that Grover search offers no asymptotic advantage when multiple solutions are required. Consequently, his method employs a classical algorithm proposed by May to construct L. To achieve more accurate security estimates for practical cryptographic schemes, we adopt the LSH-based MitM technique introduced by Kirshanova and May for building L.

Recall the search tree structure of Rep-0 and Rep-1 from Section 3. We can set $L := L_1^{(1)}$ and $S_r^{f_2} := \{s_2 \mid (s_2, f_2(s_2)) \in L_2^{(1)}\}$. This ensures that $L_1^{(1)}$ contains at least one pair $(s_1, f_1(s_1))$, and the set derived from $L_2^{(1)}$ contains a $L_1^{(1)}$-suitable s_2 such that (s_1, s_2) forms a consistent 1-close claw.

5.3. Quantum Consistent 1-Close Claw Finding for Rep-0

To make the better use of the representation technique, we reduce the size of list L by narrowing the search space for s_1. This is accomplished by adjusting the weight constraints to $L_1^{(1)} \subseteq S_1 = \mathcal{T}^n(cw/4)$ and $L_2^{(1)} \subseteq S_2 = \mathcal{T}^n((2-c)w/4)$, where $c \in [0, 1]$. Consequently, the number of representations becomes

$$R_{\text{Rep-0}} := \binom{w/2}{cw/4}^2. \tag{49}$$

With this adjustment, the sizes of the level-2 and level-1 lists are given as follows:

$$\begin{aligned} |L_1^{(2)}| = |L_2^{(2)}| &= \binom{n/2}{cw/8, cw/8, \cdot}, \\ |L_3^{(2)}| = |L_4^{(2)}| &= \binom{n/2}{(2-c)w/8, (2-c)w/8, \cdot}, \\ |L_1^{(1)}| &= |L_1^{(2)}| \cdot |L_2^{(2)}| \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}}, \\ |L_2^{(1)}| &= |L_3^{(2)}| \cdot |L_4^{(2)}| \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}}. \end{aligned} \tag{50}$$

The corresponding parameter k satisfies

$$k = \left\lfloor \frac{\log_2 R_{\text{Rep-0}}}{\log_2 q - 0.5 \log_2 3} \right\rfloor. \tag{51}$$

It follows that

$$2^\ell = |L_1^{(1)}| = \binom{n}{cw/4, cw/4, \cdot} \cdot \left(\frac{3}{q}\right)^{\frac{k}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k}{2}} \geq \frac{|\mathcal{T}^n(cw/4)|}{R_{\text{Rep-0}}} = \frac{|S_1|}{R_{\text{Rep-0}}}, \tag{52}$$

which is the constraint that the parameter ℓ must satisfy.

Using Rep-0, the list $L = L_1^{(1)}$ is constructed classically, requiring time and space complexities of

$$T_1 = \max\left\{|L_1^{(2)}|, |L_1^{(1)}|, T_{\mathsf{LSH+Exact}}\left(|L_1^{(2)}|, \frac{k}{2}, \frac{k}{2}, B\right)\right\};\ M_1 = \max\left\{|L_1^{(2)}|, |L_1^{(1)}|\right\}. \quad (53)$$

Let $r := \log_q R_{\text{Rep-0}}$. In Step 2.1 of Algorithm 5, subroutine $\mathcal{A}$ begins by preparing a uniform superposition over S_2, which corresponds to a ternary Dicke state. This state can be constructed using $\mathcal{O}(n)$ qubits in linear time. Subsequently, $\mathcal{A}$ executes a Grover search—without the final measurement—to generate the state $|\phi_r\rangle$. The overall qubit requirement for $\mathcal{A}$ remains $\mathcal{O}(n)$. Due to the randomness of f_2, any $s_2 \in S_2$ satisfies $f_2(s_2) = 0 \bmod q^r$ with probability q^{-r}, leading to a runtime of

$$T_{\mathcal{A}} = \mathcal{O}(q^{r/2}) = \sqrt{R_{\text{Rep-0}}}. \quad (54)$$

For Step 2.2 of Algorithm 5, the quantum set-membership oracle for L operates with time complexity $T_{f_L} = \mathcal{O}(n \cdot 2^\ell) = \mathcal{O}(n \cdot |L_1^{(1)}|)$ and utilizes $2n + 1$ qubits [19].

A uniformly random $s_1 \in S_1$ can be extended to a consistent 1-close claw (s_1, s_2) with probability $R_{\text{Rep-0}}/|S_1|$. This probability distribution remains unchanged when sampling from $S_r^{f_1}$. Consequently, the expected number of consistent 1-close claws (s_1, s_2) with $(s_1, f_1(s_1)) \in L$ is $|L| \cdot R_{\text{Rep-0}}/|S_1|$. It follows that for any $x_2 \in S_r^{f_2}$, the probability of $f_L(s_2) = 1$ is given by

$$p = \frac{|L| \cdot R_{\text{Rep-0}}}{|S_r^{f_2}| \cdot |S_1|} = \frac{|L_1^{(1)}| \cdot R_{\text{Rep-0}}}{|L_2^{(1)}| \cdot |S_1|} = \frac{1}{|L_2^{(1)}|}. \quad (55)$$

Applying Equation (6), the runtime complexity for Step 2 of Algorithm 5 becomes

$$T_2 = \mathcal{O}\left(\frac{1}{\sqrt{p}} \cdot \left(T_{\mathcal{A}} + T_{f_L}\right)\right) = \mathcal{O}\left(\sqrt{|L_2^{(1)}|} \cdot \left(\sqrt{R_{\text{Rep-0}}} + n \cdot |L_1^{(1)}|\right)\right). \quad (56)$$

The space requirement for this step is $\mathcal{O}(n)$ qubits.

Step 3 of Algorithm 5, which leverages the sorted structure of L, executes in time $\mathcal{O}(\ell)$. The overall time complexity of the algorithm for Rep-0 is therefore bounded by

$$T = \mathcal{O}(\max\{T_1, T_2\}), \quad (57)$$

with total resource requirements of $\mathcal{O}(2^\ell)$ classical memory and $\mathcal{O}(n)$ qubits.

5.4. Concrete Security Analysis of Rep-0 Instantiation

We provide concrete complexity estimates for the quantum consistent 1-close claw finding subroutine (Rep-0 instantiation) in Table 3.

The parameter triple (n, q, w) for each scheme specifies the polynomial dimension, the modulus, and the weight of the secret vector, respectively. The complexity is measured in $\log_2 T$, where T represents the time complexity of the classical–quantum hybrid attack. The quantum space complexity, corresponding to the number of qubits required, is polynomial in n for all instances. The classical memory complexity, measured in $\log_2 M$, is often exponential; a value of 0 in Table 3 indicates that the required classical memory is polynomial.

Table 3. Bit complexities for the quantum consistent 1-close claw finding subroutine (Rep-0 instantiation).

Scheme	Parameters (n, q, w)	Time $\log_2 T$	Classical Memory $\log_2 M$
NTRU-IEEE	(401, 2048, 226)	297	73
	(449, 2048, 268)	344	75
	(677, 2048, 314)	472	124
	(1087, 2048, 240)	526	50
	(541, 2048, 98)	239	0
	(613, 2048, 110)	265	17
	(887, 2048, 162)	383	34
	(1171, 2048, 212)	503	50
	(659, 2048, 76)	211	0
	(761, 2048, 84)	236	0
	(1087, 2048, 126)	349	19
	(1499, 2048, 158)	445	37
NTRU-Enc	(509, 2048, 254)	361	107
	(677, 2048, 254)	433	114
	(821, 4096, 510)	614	247
	(701, 8192, 468)	536	236
NTRU-Prime	(653, 4621, 288)	449	103
	(761, 4591, 286)	488	107
	(857, 5167, 322)	556	131
BLISS I+II	(512, 12,289, 154)	300	66
GLP I	(512, 8,383,489, 342)	387	200

We further consider the trade-off between time and classical memory. By constraining the available classical memory, we can optimize the time complexity of our algorithm. We present the trade-off curves for five representative examples, as shown in Figure 3. A similar analysis can be conducted for other parameter sets.

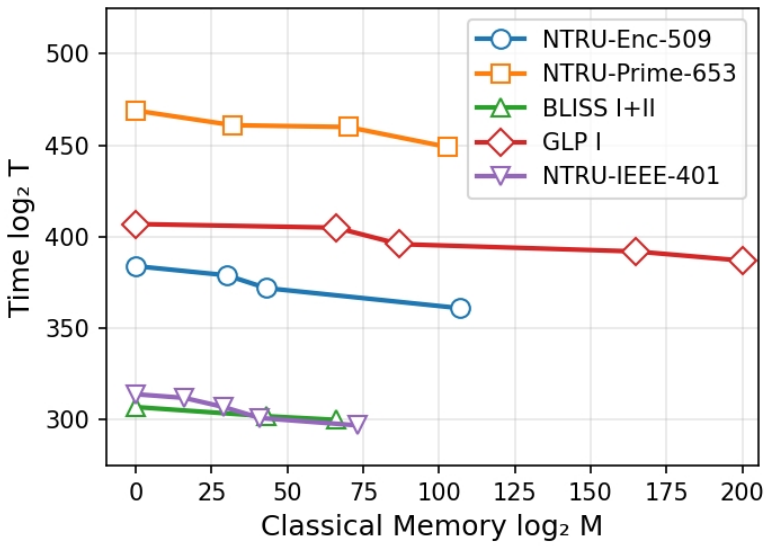

Figure 3. Time–classical memory trade-offs for five representative schemes under the Rep-0 instantiation.

5.5. Quantum Consistent 1-Close Claw Finding for Rep-1

Consider quantum consistent 1-close claw finding for Rep-1 with depth 3. Similar to Section 5.3, we adjusting the weight constraints to $L_1^{(1)} \subseteq S_1 = \mathcal{T}^n(cw/4 + \epsilon[1])$ and

$L_2^{(1)} \subseteq S_2 = \mathcal{T}^n((2-c)w/4 + \epsilon[1])$, where $c \in [0,1]$. Consequently, the number of representations becomes

$$\begin{aligned} R_{\text{Rep-1}}^{(1)} &:= \binom{w/2}{cw/4}^2 \cdot \binom{n-w}{\varepsilon[1],\varepsilon[1],\cdot}, \\ R_{\text{Rep-1}}^{(2)} &:= \binom{cw/4+\varepsilon[1]}{cw/8+\varepsilon[1]/2}^2 \cdot \binom{n-w/2-2\varepsilon[1]}{\epsilon[2],\epsilon[2],\cdot}. \end{aligned} \tag{58}$$

With this adjustment, the sizes of the level-i lists (for $1 \le i \le 3$) are given as follows:

$$\begin{aligned} |L_j^{(3)}| &= \binom{n/2}{g(cw,\varepsilon[1],\varepsilon[2]), g(cw,\varepsilon[1],\varepsilon[2]),\cdot}, \text{ for } j = 1,2,3,4, \\ |L_j^{(3)}| &= \binom{n/2}{g((2-c)w,\varepsilon[1],\varepsilon[2]), g((2-c)w,\varepsilon[1],\varepsilon[2]),\cdot}, \text{ for } j = 5,6,7,8, \\ |L_j^{(2)}| &= \left(|L_1^{(3)}|\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}}, \text{ for } j = 1,2, \\ |L_j^{(2)}| &= \left(|L_5^{(3)}|\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[2]}{4}} \cdot \left(\frac{1}{q}\right)^{\frac{3k[2]}{4}}, \text{ for } j = 3,4, \\ |L_1^{(1)}| &= \left(|L_1^{(2)}|\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}}, \\ |L_2^{(1)}| &= \left(|L_3^{(2)}|\right)^2 \cdot \left(\frac{3}{q}\right)^{\frac{k[1]-k[2]}{2}} \cdot \left(\frac{1}{q}\right)^{\frac{k[1]-k[2]}{2}}, \end{aligned} \tag{59}$$

where $g(x,y,z) := x/16 + y/4 + z/2$.

The corresponding parameter $k[i]$ (for $i = 1,2$) satisfies

$$k[i] = \left\lfloor \frac{\log_2 R_{\text{Rep-1}}^{(i)}}{\log_2 q - 0.5^i \log_2 3} \right\rfloor. \tag{60}$$

Similarly, it follows that $2^\ell = |L_1^{(1)}| \ge |S_1|/R_{\text{Rep-1}}^{(1)}$.

Using Rep-1, the list $L = L_1^{(1)}$ is constructed classically, requiring time and space complexity of

$$\begin{aligned} T_1 = \max\Bigg\{ & M_1, T_{\mathsf{LSH+Exact}}\left(|L_1^{(3)}|, \frac{k[2]}{4}, \frac{3k[2]}{4}, B[2]\right), \\ & T_{\mathsf{LSH+Exact}}\left(|L_1^{(2)}|, \frac{k[1]-k[2]}{2}, \frac{k[1]-k[2]}{2}, B[1]\right)\Bigg\}, \end{aligned} \tag{61}$$

where $M_1 = \max\left\{|L_1^{(3)}|, |L_1^{(2)}|, |L_1^{(1)}|\right\}$ represents the classical memory.

Let $r := \log_q R_{\text{Rep-1}}^{(1)}$. Similar to the analysis from Section 5.3, we get

$$\begin{aligned} T_{\mathcal{A}} &= \mathcal{O}(q^{r/2}) = \sqrt{R_{\text{Rep-1}}^{(1)}}, \\ T_{f_L} &= \mathcal{O}(n \cdot 2^\ell) = \mathcal{O}(n \cdot |L_1^{(1)}|), \\ p &= \frac{|L| \cdot R_{\text{Rep-1}}^{(1)}}{|S_r^{f_2}| \cdot |S_1|} = \frac{|L_1^{(1)}| \cdot R_{\text{Rep-1}}^{(1)}}{|L_2^{(1)}| \cdot |S_1|} = \frac{1}{|L_2^{(1)}|}. \end{aligned} \tag{62}$$

Thus, the runtime complexity for Step 2 of Algorithm 5 becomes

$$T_2 = \mathcal{O}\left(\frac{1}{\sqrt{p}} \cdot \left(T_{\mathcal{A}} + T_{f_L}\right)\right) = \mathcal{O}\left(\sqrt{|L_2^{(1)}|} \cdot \left(\sqrt{R_{\text{Rep-1}}^{(1)}} + n \cdot |L_1^{(1)}|\right)\right). \tag{63}$$

The overall time complexity of Algorithm 5 for Rep-1 with depth 3 is therefore bounded by

$$T = \mathcal{O}(\max\{T_1, T_2\}), \tag{64}$$

with total resource requirements of $\mathcal{O}(2^{\ell})$ classical memory and $\mathcal{O}(n)$ qubits.

5.6. Concrete Security Analysis of Rep-1 Instantiation

We provide concrete complexity estimates for the quantum consistent 1-close claw finding subroutine (Rep-1 instantiation) in Table 4.

Table 4. Bit complexities for the quantum consistent 1-close claw finding subroutine (Rep-1 instantiation).

Scheme	Parameters (n, q, w)	Time		Classical Memory
		$\log_2 T$	$\log_S T$ *	$\log_2 M$
NTRU-IEEE	(401, 2048, 226)	239	0.388	166
	(449, 2048, 268)	282	0.404	192
	(677, 2048, 314)	377	0.385	268
	(1087, 2048, 240)	421	0.397	321
	(541, 2048, 98)	186	0.403	127
	(613, 2048, 110)	217	0.413	157
	(887, 2048, 162)	303	0.396	185
	(1171, 2048, 212)	415	0.414	295
	(659, 2048, 76)	182	0.443	142
	(761, 2048, 84)	196	0.426	149
	(1087, 2048, 126)	291	0.427	192
	(1499, 2048, 158)	364	0.414	285
NTRU-Enc	(509, 2048, 254)	300	0.397	200
	(677, 2048, 254)	345	0.387	234
	(821, 4096, 510)	530	0.412	516
	(701, 8192, 468)	466	0.423	466
NTRU-Prime	(653, 4621, 288)	364	0.392	231
	(761, 4591, 286)	383	0.381	240
	(857, 5167, 322)	423	0.374	264
BLISS I+II	(512, 12,289, 154)	231	0.385	157
GLP I	(512, 8,383,489, 342)	328	0.408	238

* $\mathcal{S}$ is the size of the search space.

The parameter triple (n, q, w) for each scheme specifies the polynomial dimension, modulus, and weight of the secret vector, respectively. The complexity is measured in two forms: $\log_2 T$ represents the base-2 logarithm of the time complexity, while $\log_S T$ expresses the time complexity relative to the search space size S, providing a normalized measure of efficiency. The classical memory requirement is given by $\log_2 M$.

In the work of Benedikt [19], an asymptotic analysis of quantum claw-finding was provided for $w/n \in [0.375, 0.668]$, concluding that the time complexity T falls within the range $[S^{0.379}, S^{0.397}]$. In our concrete analysis shown in Table 4, for schemes where $w/n \in [0.375, 0.668]$, the time complexity T ranges from $S^{0.381}$ to $S^{0.423}$, which aligns reasonably well with the asymptotic predictions, demonstrating the consistency between our concrete implementations and theoretical asymptotic analysis.

We further examine the trade-off between time and classical memory by optimizing the time complexity under constrained classical memory resources. Figure 4 illustrates this trade-off for five representative examples, showing how attack time varies with available classical memory. Similar analysis can be extended to other parameter sets, providing a comprehensive understanding of the resource-performance landscape.

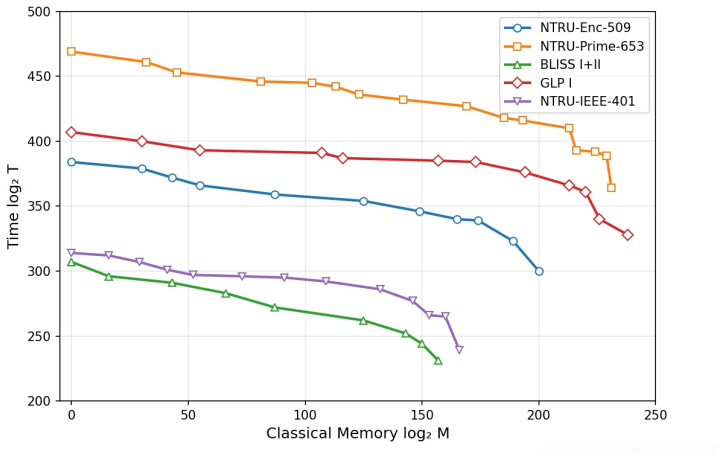

Figure 4. Time–classical memory trade-offs for five representative schemes under the Rep-1 instantiation.

Finally, we consider the scenario where both quantum and classical memory complexities are constrained to polynomial bounds. Table 5 compares our method with the vHKM [18] approach under such strict polynomial memory constraints. The results demonstrate that our algorithm achieves superior performance across all evaluated schemes, consistently requiring fewer computational resources than the vHKM method while maintaining the same polynomial memory footprint.

Table 5. Comparison of time complexity under polynomial memory.

Scheme	Parameters (n, q, w)	vHKM [18] $\log_2 T$	Ours $\log_2 T$
NTRU-Enc	(509, 2048, 254)	401	384
	(677, 2048, 254)	463	454
	(821, 4096, 510)	708	650
	(701, 8192, 468)	620	558
NTRU-Prime	(653, 4621, 288)	486	469
	(761, 4591, 286)	521	510
	(857, 5167, 322)	586	574
BLISS I+II	(512, 12,289, 154)	309	307
GLP I	(512, 8,383,489, 342)	453	407

6. Conclusions

In this work, we have advanced the frontier of quantum combinatorial cryptanalysis for ternary LWE. We introduced a novel quantum key search algorithm that effectively addresses the critical limitation of prior quantum approaches—the costly need to guess error coordinates. By reformulating the problem as a graph search and integrating the classical LSH-based Meet-in-the-Middle framework of Kirshanova and May with a quantum walk, our attack achieves a significant reduction in concrete time complexity. This allows it to bridge the gap between the asymptotic promise of quantum combinatorial attacks and their practical performance, establishing a new state of the art. Our comprehensive security analysis provides revised, and more precise, quantum security estimates for a wide range

of practical cryptographic schemes, including NTRU, BLISS, and GLP. The primary and most effective defense against the quantum combinatorial attacks presented in this work is to appropriately increase the size of the search space. This can be achieved by scaling parameters such as the polynomial dimension n or the weight w of the secret vector.

Furthermore, we addressed the critical issue of quantum resource constraints by presenting a second algorithm tailored for a polynomial-qubit setting. By combining the Kirshanova–May framework with a quantum claw-finding technique, we demonstrated that effective attacks remain feasible even with minimal quantum memory, albeit at the cost of increased classical storage and time. This analysis provides the first concrete security assessment of major schemes under such practical quantum hardware limitations.

Funding: This work was supported by the Innovation Program for Quantum Science and Technology under Grant No. 2024ZD0300502 and the Beijing Nova Program under Grants No. 20220484128 and 20240484652.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Source code for reproducing our results is available at: https://gitee.com/li-yang777777/lwe (accessed on 22 October 2025).

Acknowledgments: The author thank the reviewers for their valuable feedback and insightful suggestions, which greatly improved this manuscript.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LWE	Learning with Errors
vHKM	van Hoof–Kirshanova–May algorithm
NTRU	Number Theory Research Unit
BLISS	Bimodal Lattice Signature Scheme
GLP	Güneysu–Lyubashevsky–Pöppelmann signature scheme
MitM	Meet-in-the-Middle
LSH	Locality-Sensitive Hashing
QRAQM	Quantum Random Access Quantum Memory
QRACM	Quantum Random Access Classical Memory
CNPS	Chailloux–Naya–Plasencia–Schrottenloher algorithm

Appendix A. Proof of Theorem 2

We begin by introducing the combined approximate and exact matching hash function, which will be used in the proof of Theorem 2.

Let $x \in \mathbb{Z}_q^n$ be a vector. We partition the coordinates into three disjoint sets:

- I: set of k_1 coordinates for approximate matching
- J: set of k_2 coordinates for exact matching
- Remaining $n - k_1 - k_2$ coordinates for subsequent processing

The combined hash function $H_{u,B} : \mathbb{Z}_q^n \to \mathbb{Z}^{k_1} \times \mathbb{Z}_q^{k_2}$ is defined as:

$$H_{u,B}(x) = \left(h_{u,B}(x_I), Id(x_J)\right) \tag{A1}$$

where:

- Approximate hash component for coordinates I:

$$h_{u,B}(x_I) = \left(\left\lfloor \frac{x_i + u_i}{B} \right\rfloor \right)_{i \in I} \tag{A2}$$

with $B < q$ and random shift $u \in \mathbb{Z}_q^n$. Each coordinate maps to $\{0, 1, \ldots, \lceil q/B \rceil - 1\}$.

- Exact hash component for coordinates J:

$$Id(x_J) = (x_i)_{i \in J} \tag{A3}$$

This is simply the identity function on the exact matching coordinates.

The overall hash label is the concatenation of these two components, forming a $(k_1 + k_2)$-dimensional vector.

We now proceed to prove Theorem 2.

Proof. We analyze the complexity of Algorithm A1 for finding vectors that are approximately close to x on k_1 coordinates and exactly matching on k_2 coordinates.

In Step 2, the operations of hashing and sorting require both time and memory on the order of $|L| \log |L|$.

The expected count of elements in L whose hash values coincide with that of x is $|L| \cdot (B/q)^{k_1} \cdot (1/q)^{k_2}$. Step 3 retrieves these candidates in time proportional to $|L| \cdot (B/q)^{k_1} \cdot (1/q)^{k_2} \cdot \log |L|$, after which we apply a filter to exclude vectors failing the condition $\max_{i \in I} |x_i - y_i| = 1$.

The expected number of valid elements $y \in L$ satisfying both the approximate condition on k_1 coordinates and exact equality on k_2 coordinates is $|L| \cdot (3/q)^{k_1} \cdot (1/q)^{k_2}$. This arises because each coordinate in the approximate matching set admits 3 possible differences $\{-1, 0, 1\}$ with probability $3/q$, while exact matches occur with probability $1/q$ per coordinate.

Each iteration captures only a subset of valid elements. We demonstrate that the number of iterations specified in Step 4 suffices to recover nearly all valid vectors with high probability.

Consider a valid element $y \in L$ and define event E as $h_{u,B}(x) = h_{u,B}(y)$. Then

$$\Pr[E] = \Pi_{i \in I} \left(1 - \Pr\left[\left\lfloor \frac{x_i + u_i}{B} \right\rfloor \neq \left\lfloor \frac{y_i + u_i}{B} \right\rfloor \right] \right) = \left(1 - \frac{q/B}{q} \right)^{k_1} = (1 - 1/B)^{k_1}. \tag{A4}$$

After $(\Pr[E])^{-1} n \log n$ repetitions, E happens with probability

$$1 - (1 - \Pr[E])^{(\Pr[E])^{-1} n \log n} \leq 1 - e^{-n \log n}. \tag{A5}$$

Applying a union bound over all $\exp(n)$ potentially valid vectors $y \in L$ guarantees that, with high probability, a $(1 - o(1))$-fraction of all valid elements will be recovered.

□

Algorithm A1 LSH-Odlyzko with Combined Approximate and Exact Matching

Input: A vector $x \in \mathbb{Z}_q^n$, a list L of size $|L|$ with i.i.d. uniform elements from $\mathbb{Z}_q^n$, sets I ($|I| = k_1$) and J ($|J| = k_2$).

Output: $(1 - o(1))$-fraction of $y \in L$ satisfying $\max_{i \in I} |(x - y) \bmod q| = 1$ and $x_j = y_j$ for all $j \in J$.

1: Choose $B \in \{1, \ldots, q\}$ suitably. Choose $u \xleftarrow{\$} \mathbb{Z}_q^n$.
2: Apply $H_{u,B}$ to x and all vectors in L. Sort L according to their hash values.
3: Output all y satisfying $H_{u,B}(x) = H_{u,B}(y)$ and $\max_{i \in I} |x_i - y_i| = 1$.
4: Repeat Steps 1-3 $\left((B/(B-1))^{k_1} \cdot n \log n \right)$ times.

References

1. Regev, O. New lattice-based cryptographic constructions. *J. ACM* **2004**, *51*, 899–942. [CrossRef]
2. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory* **2014**, *6*, 1–36. [CrossRef]
3. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 353–367.
4. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 238–268. [CrossRef]
5. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Advances in Cryptology—EUROCRYPT 2010, Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, French, 30 May–3 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
6. *IEEE Std 1363.1-2008*; IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. IEEE: Piscataway, NJ, USA, 2009. Available online: https://ieeexplore.ieee.org/document/4800404 (accessed on 10 March 2009).
7. Bernstein, D.J.; Chuengsatiansup, C.; Lange, T.; van Vredendaal, C. NTRU Prime: Reducing Attack Surface at Low Cost. In *Selected Areas in Cryptography—SAC 2017, Proceedings of the SAC 2017, 24th International Conference, Ottawa, ON, Canada, 16–18 August 2017*; Adams, C., Camenisch, J., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 10719.
8. Hülsing, A.; Rijneveld, J.; Schanck, J.; Schwabe, P. High-speed key encapsulation from NTRU. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 232–252.
9. Ducas, L.; Durmus, A.; Lepoint, T.; Lyubashevsky, V. Lattice signatures and bimodal Gaussians. In *Advances in Cryptology, Proceedings of the CRYPTO 2013: Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 40–56.
10. Güneysu, T.; Lyubashevsky, V.; Pöppelmann, T. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems—CHES 2012, Proceedings of the 14th International Workshop, Leuven, Belgium, 9–12 September 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7428, pp. 530–547.
11. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Proceedings of the Third International Symposium, Portland, OR, USA, 21–25 June 1998*; Buhler, J.P., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
12. Howgrave-Graham, N. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Annual International Cryptology—CRYPTO 2007, Proceedings of the 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 150–169.
13. May, A. How to meet ternary LWE keys. In *Advances in Cryptology—CRYPTO 2021, Proceedings of the 41st Annual International Cryptology Conference, Virtual Event, 16–20 August 2021*; Springer International Publishing: Cham, Switzerland, 2021; pp. 701–731.
14. Howgrave-Graham, N.; Joux, A. New generic algorithms for hard knapsacks. In *Advances in Cryptology—EUROCRYPT 2010, Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, French, 30 May–3 June 2010*; Springer International Publishing: Berlin/Heidelberg, Germany, 2010; pp. 235–256.
15. Becker, A.; Coron, J.; Joux, A. Improved generic algorithms for hard knapsacks. In *Advances in Cryptology—EUROCRYPT 2011, Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011*; Springer International Publishing: Berlin/Heidelberg, Germany, 2011; pp. 364–385.
16. Bonnetain, X.; Bricout, R.; Schrottenloher, A.; Shen, Y. Improved classical and quantum algorithms for subset-sum. In *Advances in Cryptology—ASIACRYPT 2020, Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, 7–11 December 2020*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 633–666.
17. Kirshanova, E.; May, A. How to Find Ternary LWE Keys Using Locality Sensitive Hashing. In Proceedings of the IMA International Conference on Cryptography and Coding, Oxford, UK, 14–16 December 2021.
18. van Hoof, I.; Kirshanova, E.; May, A. Quantum Key Search for Ternary LWE. In *Post-Quantum Cryptography, Proceedings of the 12th International Workshop, Daejeon, South Korea, 20–22 July 2021*; Springer International Publishing: Cham, Switzerland, 2021; pp. 117–132.
19. Benedikt, B.J. Reducing the Number of Qubits in Solving LWE. In *Post-Quantum Cryptography, Proceedings of the 16th International Workshop, Taipei, Taiwan, 8–10 April 2025*; Springer International Publishing: Cham, Switzerland, 2025; pp. 231—263.

20. Chailloux, A.; Naya-Plasencia, M.; Schrottenloher, A. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *Advances in Cryptology—ASIACRYPT 2017, Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017*; Springer International Publishing: Cham, Switzerland, 2017; pp. 211–240.
21. Brassard, G.; Hoyer, P.; Mosca, M.; Tapp, A. Quantum amplitude amplification and estimation. *arXiv* **2000**, arXiv:quant-ph/0005055. [CrossRef]
22. Grover, L.K. Quantum computers can search arbitrarily large databases by a single query. *Phys. Rev. Lett.* **1997**, *79*, 4709–4712. [CrossRef]
23. Magniez, F.; Nayak, A.; Roland, J.; Santha, M. Search via quantum walk. *SIAM J. Comput.* **2011**, *40*, 142–164. [CrossRef]
24. Ambainis, A. Quantum walk algorithm for element distinctness. *SIAM J. Comput.* **2007**, *37*, 210–239. [CrossRef]
25. Bonnetain, X.; Chailloux, A.; Schrottenloher, A.; Shen, Y. Finding many collisions via reusable quantum walks: Application to lattice sieving. In *Advances in Cryptology—EUROCRYPT 2023, Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, 23–27 April 2023*; Springer International Publishing: Cham, Switzerland, 2023; pp. 221–251.