Quantum Key Search for Ternary LWE

Iggy van Hoof[1], Elena Kirshanova[1,2], and Alexander May[1]

[1] Horst Görtz Institute for IT-Security, Ruhr University Bochum
[2] Immanuel Kant Baltic Federal University, Kaliningrad, Russia
iggy.hoof, elena.kirshanova, alex.may{@rub.de}

Abstract. Ternary LWE, i.e., LWE with coefficients of the secret and the error vectors taken from $\{-1, 0, 1\}$, is a popular choice among NTRU-type cryptosystems and some signatures schemes like BLISS and GLP.
In this work we consider *quantum* combinatorial attacks on ternary LWE. Our algorithms are based on the quantum walk framework of Magniez-Nayak-Roland-Santha. At the heart of our algorithms is a combinatorial tool called *the representation technique* that appears in algorithms for the subset sum problem. This technique can also be applied to ternary LWE resulting in faster attacks. The focus of this work is quantum speed-ups for such representation-based attacks on LWE.
When expressed in terms of the search space $\mathcal{S}$ for LWE keys, the asymptotic complexity of the representation attack drops from $\mathcal{S}^{0.24}$ (classical) down to $\mathcal{S}^{0.19}$ (quantum). This translates into noticeable attack's speed-ups for concrete NTRU instantiations like NTRU-HRSS [CHES'17] and NTRU Prime [SAC'17].
Our algorithms do not undermine current security claims for NTRU or other ternary LWE based schemes, yet they can lay ground for improvements of the combinatorial subroutines inside hybrid attacks on LWE.

Keywords. small secret LWE, representations, quantum random walk

1 Introduction

The Learning with Errors problem (LWE) [Reg03] asks to find the secret vector $s \in \mathbb{Z}_q^n$, given $(A, b = As + e \mod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, where the A is a matrix with entries taken uniformly at random from $\mathbb{Z}_q$ and e is a short "error" vector. Being an average-case hard problem, LWE is at least as hard as some worst-case problems on lattices [Reg03,SSTX09,LPR10]. That allowed LWE to serve as a foundation to numerous cryptographic schemes [BDK+18,PFH+19,Lyu12,BCLv17], some of which chose to use the secret s and the error e with bounded $\ell_\infty-$norm. This allows for more efficient constructions and shorter keys. Among such schemes is the famous NTRU cryptosystems [HRSS17,BCLv17] and some efficient signature schemes like [DDLL13,GLP12]. These schemes rely on the hardness of LWE with *ternary* secret and the error, i.e, $s_i, e_i \in \{-1, 0, 1\}$.

The fact that s and e are small does not significantly undermine the security of LWE: there exists a reduction from "standard" LWE to binary secret LWE [BLP+13], yet this reduction loses a $\log(n)$-factor in the secret dimension.

On the other hand, there exist attacks that exploit the fact that the secret is small [BG14,KF15], thus impacting, although mildly, the concrete security of such schemes.

Small-secret LWE opens up a path for combinatorial attacks.[1] In this direction, the most prominent attacks were proposed by Odlyzko [HPS98,HGSW03] and Howgrave-Graham [How07], where the authors give a Meet-in-the-Middle attack on NTRU keys. Recently, May in [May21] noticed that these MitM attacks can be significantly improved using the so-called representation technique that originates from attacks on subset sum [HJ10,BCJ11,BBSS20]. In this work, we investigate how the MitM algorithm Meet-LWE from [May21] can be sped up on a quantum computer.

Our contributions. Building upon the work of May [May21], we

- instantiate representation-based combinatorial algorithms for ternary LWE in the quantum walk framework setting from [MNRS11], thereby using techniques from [BJLM13,HM18],
- study the impact of our quantum algorithms on concrete parameters of NTRU [HRSS17,BCLv17], BLISS(I+II) [DDLL13], and GLP [GLP12], all of which rely on ternary LWE,
- obtain time-memory tradeoffs for our quantum walk based algorithms and show concrete bit complexities of the above schemes when we only have polynomial classical and polynomial quantum memory.

Our quantum walk-based algorithm, called QMeet-LWE, provides (in its optimized instantiation qRep-1) the following asymptotic improvements: for search space size $\mathcal{S}$ for ternary LWE key, we improve from roughly $\mathcal{S}^{0.24}$ (classically) downto roughly $\mathcal{S}^{0.19}$ (quantumly). This translates into considerable speed-ups (by factors in the range $2^{50} - 2^{130}$) for concrete security estimates. We provide such estimates for NTRU-HRSS [HRSS17], NTRU-Prime [BCLv17], signatures BLISS(I+II) [DDLL13], and GLP [GLP12] parameters. For our low-memory quantum algorithms the concrete savings are even larger.

Our estimates are currently inferior to the best quantum lattice-based estimates, thus our analysis does not invalidate the aforementioned schemes' security claims. However, our quantum algorithm, yet being heuristic, relies on *different* rather mild assumptions than the numerous heuristics for lattice-based attacks. Further, we believe that our quantum LWE Key search algorithm QMeet-LWE might be used as an improved building block inside more involved algorithms, e.g. for potentially speeding up the so-called Lattice Hybrid attack [How07].

2 Preliminaries

2.1 LWE-keys

In this work, we only consider ternary LWE keys, defined as follows.

[1] By 'combinatorial' here we exclude BKW-like algorithms [KF15,GJS15], since these apply only to LWE with $m \gg n$, which is not the case for NTRU-type schemes.

Definition 1 (Ternary LWE Key). *An LWE Key consists of three public parameters q, $A \in \mathbb{Z}_q^{m \times n}$, $b \in \mathbb{Z}_q^m$, and two secret parameters $s \in \mathbb{Z}_q^n$ and (error) $e \in \mathbb{Z}_q^m$ that satisfy the identity $As = b + e \mod q$. We call s and e* ternary keys *if $||s||_\infty = ||e||_\infty = 1$. We denote by $\mathcal{T}^n$ the set of n-dimensional ternary keys.*

Throughout the paper, we only consider ternary keys s, e as well as square A with $m = n$. Practical implications of LWE-based cryptosystems of the NTRU-type [HPS98,GLP12,DDLL13,BCLv17] also limit the number of non-zero entries in the secrets.

Definition 2 (Weight). *Let $s = (s_1, ..., s_n)$ be a vector in $\mathbb{F}^n$. The weight w of this vector s is defined as its Hamming Weight $w = \Sigma_{s_i \neq 0} 1$. Relative to n we also define the relative weight $\omega = w/n$ where $0 \leq \omega \leq 1$. The set of ternary weight-w keys with an even number $w/2$ of ± 1-entries each is denoted by $\mathcal{T}^n(w/2)$. For ease of notation, in the following we omit any roundings.*

Current security analysis suggests an optimal relative weight in the range $\omega \in [\frac{1}{3}, \frac{2}{3}]$ [HRSS17,BCLv17] with $\omega = \frac{3}{8}$ and $\omega = \frac{1}{2}$ being prominent choices for NTRU-type schemes.

We approximate the search space $\mathcal{S}$ for ternary key using the following standard formula that holds up to a small polynomial factor of $\frac{1}{\sqrt{n}}$. In general, we omit small polynomial factors throughout this paper.

Lemma 1 (Multinomial approximation). *Let $D = \{d_1, ..., d_k\} \subset \mathbb{Z}_q$ be a digit set of cardinality k. The number of vectors $s \in \mathbb{Z}_q^n \cap D^n$ having exactly $c_i n$ many d_i-entries with $\sum_{i=1}^k c_i = 1$, is*

$$\binom{n}{c_1 n, ..., c_k n} \approx 2^{H(c_1,...,c_k)n} \text{ with entropy } H(c_1, ..., c_k) = \sum_{i=1}^{k} c_i \log_2 \left(\frac{1}{c_i}\right).$$

For ease of notation, for multinomial coefficients $\binom{n}{c_1 n, ..., c_k n}$ we write $\binom{n}{c_1 n, ..., c_{k-1} n, \cdot}$, where $\cdot$ represents the last term $c_k n = n - c_1 n - ... - c_{k-1} n$. Analogous, we write $H(c_1, ..., c_k)$ more compactly as $H(c_1, ..., c_{k-1}, \cdot)$.

2.2 Quantum Walk

To translate the classical MEET-LWE algorithm to the quantum setting, we utilize the quantum walk framework by Magniez-Nayak-Roland-Santha [MNRS11].

Classical Random Walks. Classical random walks search for a marked vertex in some graph in 3 steps:

1. **Set up** a single explicit vertex v in set up time T_S.
2. **Update** v by walking to a random adjacent vertex $1/\delta$ times, where a single update takes time T_U.

3. **Check** whether the resulting vertex is marked in checking time T_C. If not marked, go back to 2.

Here, the spectral gap δ of the graph tells us how many steps we need to perform, until we can arrive at some (almost) uniformly random vertex that we check. Thus, if an ε-fraction of vertices is marked, we have total time complexity

$$T_{\mathtt{RW}} = T_S + \frac{1}{\varepsilon}\left(T_C + \frac{1}{\delta}T_U\right).$$

Quantum Walks. Rather than walking to a random adjacent vertex, we can walk to a superposition of all adjacent vertices. We need to repeat this only $1/\sqrt{\delta}$ times. And rather than checking whether a vertex is marked, we can change the phase of states with a marked vertex. Repeating this $1/\sqrt{\varepsilon}$ times we measure a marked node within quantum time complexity

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

Johnson graphs are useful to minimize update costs.

Definition 3 (Johnson graph). *Let L be a set of size N. For some $r \leq N$, the Johnson graph $J(N,r)$ is an undirected graph $G_J = (V_J, E_J)$ with $|V_J| = \binom{N}{r}$ vertices representing the size-r subsets of L. We have $\{v, v'\} \in E_J$ iff $v, v' \in V_J$ represent subsets S, S' that differ by a single element, i.e. $|S \cap S'| = r - 1$.*

We may combine several Johnson graphs via Cartesian products.

Definition 4. *Given graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ we define the Cartesian product $G_1 \times G_2 = (V, E)$ as:*
$V = V_1 \times V_2 = \{v_1 v_2 | v_1 \in V_1, v_2 \in V_2\}$ and
$E = \{v_1 v_2, v_1' v_2' | (v_1 = v_1', (v_2, v_2') \in E_2)$ or $((v_1, v_1') \in E_1, v_2 = v_2')\}$.

In the Cartesian product of m Johnson graphs two vertices are adjacent iff all m subsets represented by their vertices are equal, except for a single pair of subsets that differs by one element. The spectral gap of the Cartesian product of m Johnson graphs $J^m(N,r) = \bigtimes_{i=1}^{m} J(N,r)$ can be approximated with the following formula due to Kachigar and Tillich [KT17]:

$$\delta(J^m(N,r)) \geq \frac{1}{m}\delta(J(N,r)) = \Omega\left(\frac{1}{r}\right) \text{ for fixed } m. \tag{2}$$

Heuristics on Quantum Random Walks. We use the quantum random walk framework of [BBSS20] that guarantees that update costs are within a polynomial factor of their expected time. Ignoring polynomial factors, this allows us to work in the following *solely using expected costs.*

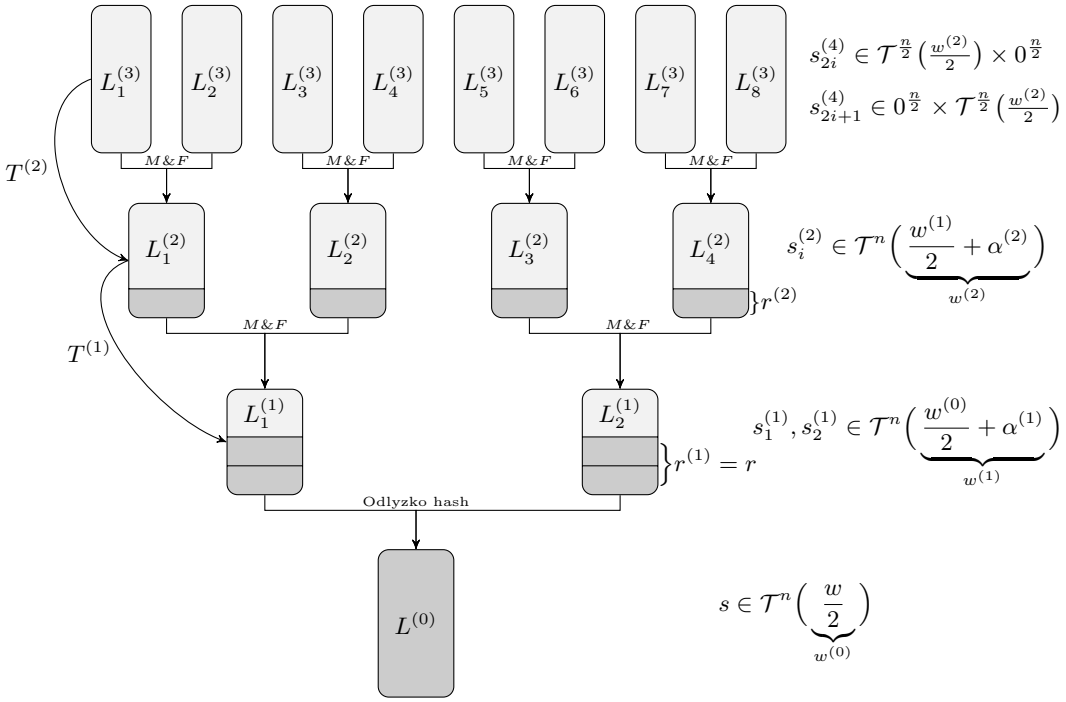

Fig. 1: The classical MEET-LWE algorithm

3 Quantum Meet-LWE - High Level Idea

3.1 Classical Meet-LWE

Let us decribe the high-level idea of May's classical MEET-LWE algorithm, see also Figure 1. Let s be a ternary weight-w LWE secret key with even number of ± 1. Let $w^{(0)} = w/2$, then $s \in \mathcal{T}^n(w^{(0)})$. We write $s = s_1^{(1)} + s_2^{(2)}$ with $s_1^{(1)}, s_2^{(1)} \in T^n(w^{(1)})$, where $w^{(1)} \geq w^{(0)}/2$.

We rewrite the LWE identity $As = b + e \bmod q$ as

$$As_1^{(1)} + e_1 = b - As_1^{(1)} + e_2 \bmod q \text{ for some } e_1, e_2 \in \{0,1\}^n. \tag{3}$$

In a nutshell, MEET-LWE constructs candidate solutions $s_1^{(1)}, s_2^{(1)}$ that fulfill Equation (3) on r coordinates. Let $R^{(1)}$ be the number of representations to write s as a sum $s_1^{(1)} + s_2^{(2)}$. Let us set $r = \lfloor \log_q(R^{(1)}) \rfloor$, and fix a random target $t \in \mathbb{Z}_q^r$. We denote by $\pi_r : \mathbb{Z}_q^n \to \mathbb{Z}_q^r$ the projection to the first r coordinates. Then on expectation at least one representation of s satisfies

$$\pi_r(As_1^{(1)} + e_1) = t = \pi_r(b - As_1^{(1)} + e_2) \bmod q. \tag{4}$$

Notice that Equation (4) can be checked if we know $\pi_r(e) \in \mathcal{T}^r$, which in turn gives us $\pi_r(e_1)$, $\pi_r(e_2)$. Thus, MEET-LWE involves a guessing step that guesses r coordinates of e.

Eventually, once all candidates $s_1^{(1)} \in L_1^{(1)}$ and $s_2^{(1)} \in L_2^{(2)}$ satisfying Equation (4) have been constructed, we have to find a pair (s_1, s_2) such that $A(s_1 + s_2) - b \mod q \in \mathcal{T}^n$. Algorithmically, this can be done via Odlyzko's locality sensitive hash function [HGSW03]. The resulting MEET-LWE is described in Algorithm 1.

Algorithm 1 Classical MEET-LWE

Require: LWE public key $(A, b) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$, weight $w \in \mathbb{N}$
Ensure: ternary weight-w s satisfying $e = As - \vec{b} \bmod q \in \mathcal{T}^n$
1: Let $R^{(1)}$ be the number of representations $s = s_1 + s_2$. Let $r = \lfloor \log_q(R^{(1)}) \rfloor$.
2: **for** all $\pi_r(e) \in \mathcal{T}^r$ **do**
3: Construct $L_1^{(1)}, L_2^{(1)}$ using some tree-based list construction.
4: Find s_1, s_2 with $s_1 + s_2 \in \mathcal{T}^n(w/2)$ and $A(s_1 + s_2) - b \in \mathcal{T}^n$ via Odlyzko hashing.
5: **end for**
6: **return** $s = s_1 + s_2$

Run time analysis. MEET-LWE has an outer **for**-loop that guesses r coordinates of $e \in \mathcal{T}^n$ in time $T_g = 3^r$, and an inner loop for list construction with run time T_ℓ. The overall run time complexity is then $T = T_g + T_\ell$. In [May21], it was shown that $T_\ell = 2^{\Theta(n)}$ whereas $T_g = 2^{\mathcal{O}(\frac{n}{\log n})}$. Thus, asymptotically we may omit the guessing cost T_g.

In the following, we describe how to compute T_ℓ, since this is crucial for the analysis of our MEET-LWE's quantum walk version. The classical run time analysis follows the typical *Match-and-Filter* approach, denoted $M\&F$ in Figure 1, within the representation technique. Namely, on each level $0 \leq j < 3$ of the search tree we filter out all vectors $s_i^{(j)}$ that do not have the correct weight distribution $\mathcal{T}^n(w^{(j)})$.

Let $L^{(j)}$ be the expected list size on level j. We show how to compute these values in the next sections. The time to compute each level-3 list is $T^{(3)} = L^{(3)}$.

Define $r^{(3)} = 0$. Then the *Match-and-Filter* approach constructs every level-j list for $j = 1, 2$ in time

$$T^{(j)} = \frac{(L^{(j+1)})^2}{q^{r^{(j)} - r^{(j+1)}}}.$$

Once we have the level-1 lists $L_1^{(1)}, L_2^{(2)}$ we construct the solution via Odlyzko's approximate matching. As we already exactly matched elements on $r^{(1)} = \lfloor \log_q R^{(1)} \rfloor$ elements, it remains to approximately match on $n - r^{(1)}$ coordinates. This can be done in time

$$T^{(0)} = \frac{(L^{(1)})^2}{2^{n - r^{(1)}}}.$$

The list construction time T_ℓ and memory complexity M is then in total

$$T_\ell = \max\{T^{(0)}, \ldots, T^{(3)}\} \text{ and } M = \max\{L^{(1)}, \ldots, L^{(3)}\}.$$

3.2 Quantum Meet-LWE (QMeet-LWE)

To translate Meet-LWE to a quantum random walk QMeet-LWE we use the same techniques that have been introduced in the subset sum context [HJ10,BCJ11,BBSS20].

Assume that we have a level-d search tree, see Figure 1 for an example with $d = 3$. Let $L^{(d)}$ be the size of our level-d list. For a quantum walk, in the setup phase we choose random subsets $U_i^{(d)} \subseteq L_i^{(d)}$ $(i = 1, ..., 2^d)$, each having $U^{(d)} := (L^{(d)})^\gamma$, $\gamma < 1$ elements. One then simply runs Meet-LWE with the new depth-d lists $U_i^{(d)}$.

Recall that the parameter choice for $L_i^{(d)}$ guarantees on expectation a representation $s = s_1^{(d)} \ldots + s_{2^d}^{(d)}$ that survives all Match-and-Filter steps up to the root list $L^{(0)}$. However, by construction we have $s_i \in U_i^{(d)}$ for all $i = 1, \ldots, 2^d$ only with probability

$$\epsilon = \left(\frac{U^{(d)}}{L^{(d)}}\right)^{2^d} = (L^{(d)})^{(\gamma-1)2^d}. \tag{5}$$

Let us define $N = L^{(d)}, r = U^{(d)}$. For all lists $L_i^{(d)}$, $i = 1, \ldots, 2^d$, we define their corresponding Johnson graph $J_i(N, r)$. We then perform our random walk on the graph

$$J(N, r) = J_1(N, r) \times \ldots \times J_{2^d}(N, r).$$

Using Equation (6), the spectral gap of $J(N, r)$ is

$$\delta(J(N, r)) = \Omega\left(\frac{1}{r}\right) = \Omega\left(\frac{1}{U^{(d)}}\right). \tag{6}$$

Let $v \in J(N, r)$ be a node defined by the Cartesian product of the size-r subsets $U_1^{(d)} \times \ldots \times U_{2^d}^{(d)}$. Then we label v with $U_1^{(d)} \times \ldots \times U_{2^d}^{(d)}$. Every node v contains the complete classical Meet-LWE search tree from Section 3.1 build with its label as level-d lists.

We call v *marked* if its corresponding level-0 list $U^{(0)}$ is non-empty, i.e. it contains a representation of s that survived all Match-and-Filter steps and Odlyzko's hash function. Notice that $\Pr[v \text{ is marked}] = \epsilon$. Checking whether a node is marked can be done in time $\mathcal{O}(1)$ and, thus, is asymptotically neglected.

Walking to a neighbor node in $J(N, r)$ implies by the definition of a Johnson graph that we exchange exactly one element in one of the depth-d lists $L_i^{(d)}$. We will detail the *update costs* of updating a Meet-LWE tree by such an exchange in the following section.

4 QRep-0: A First Implementation of QMeet-LWE

Let us instantiate the quantum random walk QMeet-LWE from Section 3.2, which in turn uses a variant of the classical Meet-LWE from Section 3.1. The latter is first instantiated by chosing tree depth $d = 2$ with optimization parameter $\alpha^{(1)} = 0$. This easiest representation setting is called Rep-0 in [May21].

QMeet-LWE*'s Setup Cost T_S.* Using $\alpha^{(1)} = 0$, the Meet-LWE level-2 list sizes are $L^{(2)} = \binom{n/2}{w/2,w/2,\cdot}$. For the quantum walk, we choose parameter $\gamma = \frac{4}{5}$. We see in the following that this choice balances quantum walk costs. Thus, we obtain in QMeet-LWE level-2 lists of size

$$U^{(2)} = (L^{(2)})^{\gamma} = \binom{n/2}{w/8, w/8, \cdot}^{\gamma} \approx 2^{\frac{2}{5}H(\omega/4,\omega/4,\cdot)n}.$$

We structure the level-2 lists of size $U^{(2)}$ as follows

$$\begin{aligned}
U_1^{(2)} &= \{(\pi_r(As_1^{(2)} + e_1), s_1^{(2)}) \mid s_1^{(2)} \in 0^{n/2} \times \mathcal{T}^{n/2}(w/8)\},\\
U_2^{(2)} &= \{(\pi_r(As_2^{(2)} + e_1) \bmod q, s_2^{(2)}) \mid s_2^{(2)} \in \mathcal{T}^{n/2}(w/8) \times 0^{n/2}\},\\
U_3^{(2)} &= \{(\pi_r(b - As_3^{(2)} + e_2), s_3^{(2)}) \mid s_3^{(2)} \in 0^{n/2} \times \mathcal{T}^{n/2}(w/8)\},\\
U_4^{(2)} &= \{(\pi_r(b - As_4^{(2)} + e_2) \bmod q, s_4^{(2)}) \mid s_4^{(2)} \in \mathcal{T}^{n/2}(w/8) \times 0^{n/2}\}.
\end{aligned}$$

On level 1, we obtain $R^{(1)} = \binom{w/2}{w/4}^2 \approx 2^{\omega n}$ representations of $s \in \mathcal{T}^n(w/2)$ as sums of the form $s_1^{(1)} + s_2^{(1)}$ with $s_i^{(1)} \in \mathcal{T}^n(w/4)$. Let $r = \lfloor \log_q R^{(1)} \rfloor$, and let $t \in \mathcal{T}^r$ be our random target vector. Moreover, let ℓ denote Odlyzko's locality sensitive hash function. Then level-1 lists are defined as

$$\begin{aligned}
U_1^{(1)} &= \{(\ell(As_1^{(1)}), s_1^{(1)}) \mid (x, s_1^{(2)}) \in U_1^{(2)}, (t - x, s_2^{(2)}) \in U_2^{(2)}, s_1^{(1)} = s_1^{(2)} + s_2^{(2)}\},\\
U_2^{(1)} &= \{(\ell(b - As_2^{(1)}), s_2^{(1)}) \mid (x, s_1^{(2))}) \in U_1^{(2)}, (t - x, s_2^{(2)}) \in U_2^{(2)}, s_1^{(1)} = s_1^{(2)} + s_2^{(2)}\}.
\end{aligned}$$

The expected size of level-1 lists is

$$U^{(1)} = \frac{(U^{(2)})^2}{R^{(1)}} \approx 2^{(\frac{4}{5}H(\omega/4,\omega/4,\cdot)-\omega)n}.$$

Finally for layer 0 we obtain the list

$$U_1^{(0)} = \{s = s_1^{(1)} + s_2^{(1)} \in \mathcal{T}(w/2) \mid (x, s_1^{(1)}) \in U_1^{(1)}, (x, s_2^{(1)}) \in U_2^{(1)}, As_1^{(1)} = b - As_2^{(1)}\}.$$

The expected size of this list is $U^{(0)} \leq \frac{(U^{(1)})^2}{2^{n-r}} \approx 2^{(\frac{8}{5}H(\omega/4,\omega/4,\cdot)-2\omega-1)n+r}$. In summary, the expected setup cost of QMeet-LWE is

$$T_S = \max\{U^{(2)}, U^{(1)}, U^{(0)}\} = U^{(2)} \text{ for all } \omega \in \left[\frac{1}{3}, \frac{2}{3}\right]. \tag{7}$$

A node in our Johnson graph is marked if $U^{(0)}$ contains at least one element, which can be checked in time $T_C = \mathcal{O}(1)$.

QMEET-LWE*'s Update Cost T_U.* The update algorithm requires us to insert into and delete an element from one of the $U_i^{(2)}$. For example, say we want to exchange an element $s_2^{(2)}$ in $U_2^{(2)}$. The update of $U_2^{(2)}$ can be done in time $\mathcal{O}(1)$.

The update impacts the list $U_1^{(1)}$ below if there are matching $s_1^{(2)}$'s such that $\pi_r(A(s_1^{(2)} + s_2^{(2)}) + e_1) \mod q = t$. The expected number of those elements is $U^{(2)}/R$, both for deletion and insertion. For the bottom list $U_1^{(0)}$, we expect $U^{(1)}/2^{n-r}$ deletions/insertions for each of the $U^{(2)}/R$ elements. In total, the expected update cost is

$$T_U = \max\left\{1, \frac{U^{(2)}}{R}, \frac{U^{(1)}U^{(2)}}{2^{n-r}R}\right\} = 1 \text{ for all } \omega \in \left[\frac{1}{3}, \frac{2}{3}\right]. \tag{8}$$

QMEET-LWE *Random Walk Cost.* Plugging Equations (5) to (8) into Equation (1), we obtain a quantum walk runtime of

$$\begin{aligned} T_\ell &\leq T_S + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}} \cdot T_U + T_C\right) = |U^{(2)}| + \left(\frac{L^{(2)}}{U^{(2)}}\right)^2 \left(\sqrt{U^{(2)}} \cdot 1 + 1\right) \\ &= (L^{(2)})^{\frac{4}{5}} + (L^{(2)})^{2-\frac{6}{5}}. \end{aligned}$$

Thus, our choice $\gamma = \frac{4}{5}$ balances the setup time T_S with the cost of the random walk until we hit a marked node. Neglecting low order terms, we achieve QMEET-LWE run time

$$(L^{(2)})^{\frac{4}{5}} = \binom{n/2}{w/8, w/8, \cdot}^{\frac{4}{5}} \approx 2^{\frac{2}{5}H(\omega/4, \omega/4, \cdot)n}. \tag{9}$$

In Section 6, we provide QMEET-LWE's asymptotic costs T_ℓ and non-asymptotic costs $T_g \cdot T_\ell$.

5 QREP-1: Optimized QMeet-LWE

We now optimize the parameters $\alpha^{(i)}$ as well as the tree depth d in QMEET-LWE, and, therefore, also in MEET-LWE. We advise the reader to follow Figure 1. As depicted in Figure 1, we first describe QMEET-LWE in depth $d = 3$, and then provide the necessary adjustments for depth $d = 4$.

As shown in [May21], for MEET-LWE we obtain level-j, $j = 1, 2$, list sizes $L^{(j)} = S^{(j)}/R^{(j)}$ where

$$\mathcal{S}^{(j)} = \binom{n}{w^{(j)}, w^{(j)}, \cdot} \text{ and } R^{(j)} = \binom{w^{(j-1)}}{w^{(j-1)}/2}^2 \binom{n - 2w^{(j-1)}}{\alpha_j, \alpha_j, \cdot}.$$

Further, we have level-3 list size $L^{(3)} = \binom{n/2}{w^{(2)}/2, w^{(2)}, \cdot} \approx \sqrt{S^{(2)}}$. Let $r = \lfloor \log_q R^{(1)} \rfloor$.

QMEET-LWE*'s Setup Cost T_S.* We choose $\gamma = \frac{8}{9}$ for depth $d = 3$. This yields for QMEET-LWE level-3 list sizes

$$U^{(3)} = (L^{(3)})^{\gamma} = \binom{n/2}{w^{(2)}/2, w^{(2)}/2, \cdot}^{\frac{8}{9}} \approx 2^{\frac{4}{9}H(\omega^{(2)}/2, \omega^{(2)}/2, \cdot)}.$$

Analogously to Sections 3.1 and 4, we obtain in the Match-and-Filter construction expected costs

$$U^{(2)} \leq \frac{(U^{(3)})^2}{R^{(2)}},\ U^{(1)} \leq \frac{(U^{(2)})^2 R^{(2)}}{R^{(1)}} \leq \frac{(U^{(3)})^4}{R^{(1)}R^{(2)}}, U^{(0)} \leq \frac{(U^{(1)})^2}{2^{n-r}} \leq \frac{(U^{(3)})^8}{2^{n-r}(R^{(1)}R^{(2)})^2}.$$

Thus, the setup cost can be bounded as

$$T_S \leq \max\left\{U^{(3)}, \frac{(U^{(3)})^2}{R^{(2)}}, \frac{(U^{(3)})^4}{R^{(1)}R^{(2)}}, \frac{(U^{(3)})^8}{2^{n-r}(R^{(1)}R^{(2)})^2}\right\}.$$

An analysis similar to Section 4 yields an update cost of

$$T_U \leq \max\left\{1, \frac{U^{(3)}}{R^{(2)}}, \frac{(U^{(3)})^3}{R^{(1)}R^{(2)}}, \frac{(U^{(3)})^7}{2^{n-r}(R^{(1)}R^{(2)})^2}\right\}.$$

QMEET-LWE *Random Walk Cost.* Notice that if we estimate T_S, T_U via their upper bounds, then we obtain the relation $T_S = U^{(3)}T_U$. This helps us to estimate the random walk cost as

$$\begin{aligned} T_\ell &\leq T_S + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}} \cdot T_U + T_C\right) = U^{(3)}T_U + \left(\frac{L^{(3)}}{U^{(3)}}\right)^4 \left(\sqrt{U^{(2)}} \cdot T_U + 1\right) \\ &= (L^{(3)})^{\frac{8}{9}}T_U + (L^{(3)})^{4-\frac{28}{9}}T_U = 2(L^{(3)})^{\frac{8}{9}}T_U. \end{aligned} \tag{10}$$

Thus, again our choice of $\gamma = \frac{8}{9}$ balances setup costs with the cost to find a marked node.

QMEET-LWE *level-4 cost.* If we use QMEET-LWE with depth $d = 4$, then a similar analysis yields quantum random walk cost

$$T_\ell \leq (L^{(4)})^{\frac{16}{17}} \cdot T_U \text{ with } L^{(4)} = \binom{n/2}{w^{(3)}/2, w^{(3)}/2, \cdot} \tag{11}$$

and

$$T_U \leq \max\left\{1, \frac{U^{(4)}}{R^{(3)}}, \frac{(U^{(4)})^3}{R^{(2)}R^{(3)}}, \frac{(U^{(4)})^7}{R^{(1)}R^{(2)}(R^{(3)})^2}, \frac{(U^{(3)})^{15}}{2^{n-r}(R^{(1)}R^{(2)})^2(R^{(3)})^4}\right\}.$$

Notice that the exponent γ converges to 1 for increasing depth. Thus, QMEET-LWE degrades to MEET-LWE.

6 Quantum Complexity Estimates

Asymptotics. Recall that QMEET-LWE's run time is $T = T_g \cdot T_\ell$, where T_g is the time to guess r coordinates of e, and T_ℓ is the quantum walk time. As mentioned in Section 3.1 asymptotically the guessing cost T_g can be neglected.

Thus, here we simply use the formulas for T_ℓ derived for our QREP-0 instantiation in Section 4, Equation (9) and for our QREP-1 instantiation in Section 5 for depth 3 in Equation (10) and for depth 4 in Equation (11).

The results are given in Table 1 as a function of ω, and asymptotically in n. Here we compare our quantum complexities with a classical instantiation of MEET-LWE with optimized α_i as in Section 5, called CREP-1. Let $\mathcal{S}$ denote the size of the search space for the secret LWE key s, and T the complexity of our algorithms for recovering s. Then we give in Table 1 the values $\log_{\mathcal{S}} T$. We obtained optimal values for QREP-1 in depth 4 using Equation (11). The optimization parameters $\zeta_i := \frac{\alpha_i}{n}$ are also provided in Table 1.

ω	QREP-0	CREP-1	QREP-1	ζ_1	ζ_2	ζ_3
0.3	0.257	0.238	**0.191**	0.030	0.017	0.005
0.375	0.266	0.243	**0.184**	0.029	0.017	0.005
0.441	0.274	0.237	**0.182**	0.027	0.017	0.005
0.5	0.283	0.235	**0.182**	0.025	0.017	0.005
0.62	0.305	0.243	**0.188**	0.021	0.017	0.005
0.667	0.316	0.244	**0.193**	0.019	0.017	0.005

Table 1: Improvement in the exponent of the search space of QREP-1.

Notice that whereas MEET-LWE achieves complexities slight below $\mathcal{S}^{\frac{1}{4}}$ its quantum random walk version QMEET-LWE improves these complexity below $\mathcal{S}^{\frac{1}{5}}$. The improvement in the exponent is in the range of $20 - 25\%$.

Non-asymptotical real-world results. For estimating the quantum costs of attacks on real-world cryptosystems we have to take the guessing costs T_g into account. Quantumly, we Grover search [Gro96] for r coordinates of a random e in time $T_g = 3^{\frac{r}{2}}$.

QREP-0. Table 2 provides the cost of our QREP-0 attack from Section 4 in comparison to the corresponding classical attack CREP-0 for NTRU-type encryption schemes NTRU-HPS [HRSS17] and NTRUPrime [BCLv17], and for signatures BLISS [DDLL13] and GLP [GLP12]. [2]

The costs in Table 2 are in bit complexity format in the form $\log_2 T = \log_2 T_\ell + \log_2 T_g$.

[2] The scripts can be provided upon request to any author.

	(n, q, w)	cRep-0	qRep-0+Grover
NTRU-Enc	$(509, 2048, 254)$	$248 = 212 + 36$	$230 = 212 + 18$
	$(677, 2048, 254)$	$377 = 341 + 36$	$254 = 236 + 18$
	$(821, 4096, 510)$	$555 = 488 + 67$	$425 = 391 + 34$
	$(701, 8192, 468)$	$491 = 434 + 57$	$377 = 348 + 29$
NTRU-Prime	$(653, 4621, 288)$	$383 = 346 + 37$	$271 = 252 + 19$
	$(761, 4591, 286)$	$420 = 384 + 36$	$284 = 265 + 19$
	$(857, 5167, 322)$	$479 = 439 + 40$	$322 = 302 + 20$
BLISS I+II	$(512, 12289, 154)$	$257 = 240 + 18$	$174 = 165 + 9$
GLP I	$(512, 8383489, 342)$	$338 = 214 + 24$	$264 = 252 + 12$

Table 2: Bit complexities of qRep-0 in comparison to cRep-0.

Notice that qRep-0 gives large savings compared to cRep-0, often by more than 100 bits. In bit complexity, we save typically around 30%.

qRep-1. For qRep-1 we achieved best results for depth-4 trees (except for BLISS), similar to cRep-1. The results are provided in Table 3, where we also give the optimization parameters $[\alpha_1, \alpha_2, \alpha_3]$.

	(n, q, w)	cRep-1	$[\alpha]_i$	Rep-1+Grover	$[\alpha]_i$
NTRU-Enc	$(509, 2048, 254)$	$267 = 193 + 74$	$[34, 15, 4]$	$188 = 155 + 33$	$[22, 11, 3]$
	$(677, 2048, 254)$	$313 = 235 + 78$	$[28, 12, 3]$	$223 = 191 + 32$	$[16, 8, 2]$
	$(821, 4096, 510)$	$449 = 336 + 113$	$[44, 20, 4]$	$320 = 268 + 52$	$[32, 16, 4]$
	$(701, 8192, 468)$	$387 = 295 + 92$	$[41, 20, 8]$	$278 = 235 + 43$	$[29, 15, 4]$
NTRU-Prime	$(653, 4621, 288)$	$309 = 236 + 73$	$[28, 12, 2]$	$225 = 190 + 35$	$[24, 12, 3]$
	$(761, 4591, 286)$	$344 = 265 + 79$	$[32, 14, 3]$	$245 = 206 + 39$	$[30, 15, 4]$
	$(857, 5167, 322)$	$383 = 294 + 89$	$[37, 15, 2]$	$274 = 236 + 38$	$[23, 10, 2]$
BLISS I+II	$(512, 12289, 154)$	$206 = 168 + 38$	$[15, 4]$	$149 = 133 + 16$	$[9, 3]$
GLP I	$(512, 8383489, 342)$	$250 = 210 + 40$	$[36, 15, 5]$	$193 = 175 + 18$	$[20, 11, 3]$

Table 3: Improvements from qRep-1 in comparison to cRep-1.

We obtain large savings of around 100 bits for the encryption schemes, and over 50 bits for the signature schemes. Although our results are still relatively far from the bit complexities offered by quantum lattice-based attacks (by around a factor of 2, see the estimator from [ACD+18]), our QMeet-LWE might serve as a useful quantum building block to speed up more advanced algorithms.

References

ACD+18. Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer, *Estimate all the LWE, NTRU schemes!*, SCN 18 (Dario Catalano and Roberto De Prisco, eds.), LNCS, vol. 11035, Springer, Heidelberg, September 2018, pp. 351–367.

BBSS20. Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen, *Improved classical and quantum algorithms for subset-sum*, ASIACRYPT 2020, Part II (Shiho Moriai and Huaxiong Wang, eds.), LNCS, vol. 12492, Springer, Heidelberg, December 2020, pp. 633–666.

BCJ11. Anja Becker, Jean-Sébastien Coron, and Antoine Joux, *Improved generic algorithms for hard knapsacks*, EUROCRYPT 2011 (Kenneth G. Paterson, ed.), LNCS, vol. 6632, Springer, Heidelberg, May 2011, pp. 364–385.

BCLv17. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal, *NTRU prime: Reducing attack surface at low cost*, SAC 2017 (Carlisle Adams and Jan Camenisch, eds.), LNCS, vol. 10719, Springer, Heidelberg, August 2017, pp. 235–260.

BDH+05. Harry Buhrman, Christoph Dürr, Mark Heiligman, Peter Høyer, Frédéric Magniez, Miklos Santha, and Ronald de Wolf, *Quantum algorithms for element distinctness*, SIAM Journal on Computing **34** (2005), no. 6, 1324–1330.

BDK+18. W. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, M. John Schanck, Peter Schwabe, and Damien Stehlé, *Crystals - kyber: a cca-secure module-lattice-based kem*, EuroS&P (2018), 353–367.

BG14. Shi Bai and Steven D. Galbraith, *Lattice decoding attacks on binary LWE*, ACISP 14 (Willy Susilo and Yi Mu, eds.), LNCS, vol. 8544, Springer, Heidelberg, July 2014, pp. 322–337.

BJLM13. Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer, *Quantum algorithms for the subset-sum problem*, Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013 (Philippe Gaborit, ed.), Springer, Heidelberg, June 2013, pp. 16–33.

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé, *Classical hardness of learning with errors*, 45th ACM STOC (Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, eds.), ACM Press, June 2013, pp. 575–584.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky, *Lattice signatures and bimodal Gaussians*, CRYPTO 2013, Part I (Ran Canetti and Juan A. Garay, eds.), LNCS, vol. 8042, Springer, Heidelberg, August 2013, pp. 40–56.

GJS15. Qian Guo, Thomas Johansson, and Paul Stankovski, *Coded-BKW: Solving LWE using lattice codes*, CRYPTO 2015, Part I (Rosario Gennaro and Matthew J. B. Robshaw, eds.), LNCS, vol. 9215, Springer, Heidelberg, August 2015, pp. 23–42.

GLP12. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann, *Practical lattice-based cryptography: A signature scheme for embedded systems*, CHES 2012 (Emmanuel Prouff and Patrick Schaumont, eds.), LNCS, vol. 7428, Springer, Heidelberg, September 2012, pp. 530–547.

Gro96. Lov K. Grover, *A fast quantum mechanical algorithm for database search*, 28th ACM STOC, ACM Press, May 1996, pp. 212–219.

HGSW03. Nick Howgrave-Graham, Joseph H Silverman, and William Whyte, *A meet-in-the-middle attack on an ntru private key*, Tech. report, Technical report, NTRU Cryptosystems, June 2003. Report, 2003.

HJ10. Nick Howgrave-Graham and Antoine Joux, *New generic algorithms for hard knapsacks*, EUROCRYPT 2010 (Henri Gilbert, ed.), LNCS, vol. 6110, Springer, Heidelberg, May / June 2010, pp. 235–256.

HM18. Alexander Helm and Alexander May, *Subset Sum Quantumly in* 1.17^n, 13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018), Leibniz International Proceedings in Informatics (LIPIcs), vol. 111, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 5:1–5:15.

How07. Nick Howgrave-Graham, *A hybrid lattice-reduction and meet-in-the-middle attack against NTRU*, CRYPTO 2007 (Alfred Menezes, ed.), LNCS, vol. 4622, Springer, Heidelberg, August 2007, pp. 150–169.

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman, *Ntru: A ring-based public key cryptosystem*, International Algorithmic Number Theory Symposium, Springer, 1998, pp. 267–288.

HRSS17. Andreas Hülsing, Joost Rijneveld, John M. Schanck, and Peter Schwabe, *High-speed key encapsulation from NTRU*, CHES 2017 (Wieland Fischer and Naofumi Homma, eds.), LNCS, vol. 10529, Springer, Heidelberg, September 2017, pp. 232–252.

KF15. Paul Kirchner and Pierre-Alain Fouque, *An improved BKW algorithm for LWE with applications to cryptography and lattices*, CRYPTO 2015, Part I (Rosario Gennaro and Matthew J. B. Robshaw, eds.), LNCS, vol. 9215, Springer, Heidelberg, August 2015, pp. 43–62.

KT17. Ghazal Kachigar and Jean-Pierre Tillich, *Quantum information set decoding algorithms*, Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017 (Tanja Lange and Tsuyoshi Takagi, eds.), Springer, Heidelberg, 2017, pp. 69–89.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev, *On ideal lattices and learning with errors over rings*, EUROCRYPT 2010 (Henri Gilbert, ed.), LNCS, vol. 6110, Springer, Heidelberg, May / June 2010, pp. 1–23.

Lyu12. Vadim Lyubashevsky, *Lattice signatures without trapdoors*, EUROCRYPT 2012 (David Pointcheval and Thomas Johansson, eds.), LNCS, vol. 7237, Springer, Heidelberg, April 2012, pp. 738–755.

May21. Alexander May, *How to meet ternary lwe keys*, Cryptology ePrint Archive, Report 2021/216, 2021, `https://eprint.iacr.org/2021/216`.

MNRS11. Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha, *Search via quantum walk*, SIAM Journal on Computing **40** (2011), no. 1, 142–164.

Niv04. Gabriel Nivasch, *Cycle detection using a stack*, Information Processing Letters **90** (2004), 135–140.

PFH+19. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang, *Falcon*, Tech. report, National Institute of Standards and Technology, 2019, available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

Pol75. J. M. Pollard, *A monte carlo method for factorization*, BIT Numerical Mathematics **15** (1975), 331–334.

Reg03. Oded Regev, *New lattice based cryptographic constructions*, 35th ACM STOC, ACM Press, June 2003, pp. 407–416.

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa, *Efficient public key encryption based on ideal lattices*, ASIACRYPT 2009 (Mitsuru Matsui, ed.), LNCS, vol. 5912, Springer, Heidelberg, December 2009, pp. 617–635.

Tan07. Seiichiro Tani, *In improved claw finding algorithm using quantum walk*, Mathematical Foundations of Computer Science 2007, 2007, pp. 536–547.

A Time-memory tradeoffs

A straightforward way to obtain time-memory tradeoffs in the quantum walk framework is to vary the parameter γ since it governs the size $U^{(i)}$ of the sublists. Our optimization so far was targeting optimal runtime assuming we have as may qubits as we need. For example, the expected quantum memory complexity of the Rep-1 algorithm from Section 5 is asymptotically T_S – the expected time of the setup phase – a value, which is exponential in n.

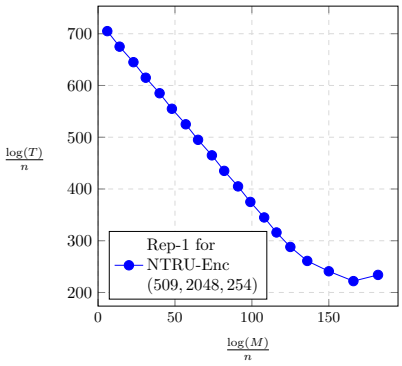

Fig. 2: Time-memory tradeoffs

If we have a fixed (but still exponential in n) number of qubits, which is smaller than what the optimal runtime needs, we can still instantiate and run the quantum walk but with smaller $U^{(i)}$'s. This will have the following effect on the runtime: the complexity of the setup phase will become smaller, but ε, the probability that a vertex contains a solution, will also decrease. Since the optimal runtime already balances the costs in Eq. 1, lowering the memory will necessarily incur larger runtime. For instance, Figure 2 shows time-memory tradeoffs for the NTRU-Enc parameter set $(509, 2048, 254)$.

Turning to the realm of polynomial memory, [May21] shows how to phrase the problem of finding s_1, s_2 that satisfy the MiTM Eq. (3) as a claw-finding problem. Namely, he defines two functions $f_1 : s_1 \mapsto \pi_r(\ell(As_1))$ and $f_2 : s_2 \mapsto \pi_r(\ell(b - As_2))$, where π_r is the projection function defined in Section 3.1 and ℓ is Odlyzko's hash function [HGSW03]. The domain of f_1, f_2 is the search space for s_1, s_2 of size $\mathcal{S}$, and the range can be (almost) bijectively mapped to a set of ternary vectors of the same size as the domain if we choose $r = \lceil \log_3(\mathcal{S}) \rceil$. A claw for f_1, f_2 is a pair (s_1, s_2) that gives the correct $s = s_1 + s_2$. Thus the number of claws is the number of representations.

To find a claw classically we use collision-finding algorithms like [Niv04,Pol75]. Assuming f_1, f_2 behave like random functions, thus expecting S collisions between them, out of which R collisions are good, we find a good collision in expected time $T_{\text{class}} = \sqrt{S} \cdot S/R$, where $\sqrt{S}$ is the expected time to find any

collision. For NTRU, BLISS, and GLP the concrete complexities under this polynomial memory attack are given in Table 4.

Quantumly, the claw-finding problem has been studied in [BDH$^+$05,Tan07]. Most of these works apply quantum random walk technique, resulting in large quantum memory requirement. The work of Buhrman et al. [BDH$^+$05] uses Grover's algorithms allowing for polynomial memory regime. In this regime, the algorithm simply creates a superposition over all (s_1, s_2) and applies Grover's algorithm to find a good pair. The checking function for Grover's routine verifies if $(As_1-(b-As_2))$ is ternary and the corresponding (s_1, s_2) have the right weight. Since we expect R good pairs in the search space of size $\mathcal{S}^2$, Grover's algorithm outputs a solution in expected time $T_{\text{quant}} = S/\sqrt{R}$. This is a algorithm uses only polynomial classical and quantum memory. The concrete runtimes for NTRU, BLISS, and GLP are given in Table 4.

	(n, q, w)	classical	quantum
NTRU-Enc	$(509, 2048, 254)$	491	401
	$(677, 2048, 254)$	581	463
	$(821, 4096, 510)$	856	708
	$(701, 8192, 468)$	751	620
NTRU-Prime	$(653, 4621, 288)$	600	486
	$(761, 4591, 286)$	654	521
	$(857, 5167, 322)$	736	586
BLISS I+II	$(512, 12289, 308)$	396	309
GLP I	$(512, 8383489, 342)$	548	453

Table 4: Bit complexities for classical and quantum claw-finding algorithms from Section A with polynomial classical and quantum memory.