How to Enumerate LWE Keys as Narrow as in Kyber/Dilithium

Timo Glaser, Alexander May

Ruhr-University Bochum, Germany
{timo.glaser, alex.may}@rub.de

Abstract. In the Learning with Errors (LWE) problem we are given a matrix $A \in \mathbb{Z}_q^{N \times N}$ and a target vector $\boldsymbol{t} \in \mathbb{Z}_q^N$ such that there exists small-norm $\boldsymbol{s}, \boldsymbol{e} \in \mathbb{Z}_q^N$ satisfying $A \cdot \boldsymbol{s} = \boldsymbol{t} + \boldsymbol{e} \bmod q$. Modern cryptosystems often sample $\boldsymbol{s}, \boldsymbol{e}$ from narrow distributions that take integer values in a small range $[-\eta, \eta]$. Kyber and Dilithium both choose $\eta = 2$ and $\eta = 3$ using either a Centered Binomial distribution (Kyber), or a uniform distribution (Dilithium).

In this work, we address the fundamental question how hard the enumeration of LWE secret keys for narrow distributions with $\eta \leq 3$ is. At Crypto 21, May proposed a representation-based algorithm for enumerating ternary keys, i.e. the case $\eta = 1$, with a fixed number of ± 1 entries. In this work, we extend May's algorithm in several ways.

First, we show how to deal with keys sampled from a probability distribution as in many modern systems like Kyber and Dilithium, rather than with keys having a fixed number of entries.

Second, we generalize to larger values $\eta = 2, 3$, thereby achieving asymptotic key guess complexities that are not far off from lattice estimates.

E.g. for Kyber's Centered Binomial distribution we achieve heuristic time/memory complexities of $\mathcal{O}(2^{0.36N})$ for $\eta = 2$, and $\mathcal{O}(2^{0.37N})$ for $\eta = 3$. For Dilithium's uniform distribution we achieve heuristic complexity $\mathcal{O}(2^{0.38N})$ for $\eta = 2$.

Let $\mathcal{S}$ be the Shannon entropy of Kyber/Dilithium keys. Then our algorithms runs in time about $\mathcal{S}^{\frac{1}{6}}$, which greatly improves over the standard combinatorial Meet-in-the-Middle attack with complexity $\mathcal{S}^{\frac{1}{2}}$.

Our results also compare well to current lattice asymptotics of $2^{0.29\beta}$, where the lattice parameter β is roughly of size $\frac{4}{5}N$. Thus, our analysis supports that Kyber secret keys are indeed hard to enumerate. Yet, we find it remarkable that a purely combinatorial key search is almost competitive with highly evolved lattice sieving techniques.

Keywords: LWE Key Search, Representation Technique, Asymptotics.

1 Introduction

Since the introduction of the Learning with Errors (LWE) problem by Regev [Reg05] into the cryptographic community, LWE has shown its amazing power to realize efficient cryptographic constructions, some being groundbreaking as the Gödel Prize 22 award Fully Homomorphic Encryption schemes [BV14,BGV14].

It does not come as a big surprise that LWE-type constructions play a central role in the current NIST initiative for identifying encryption/signature schemes resistant to quantum computers [BDK$^+$18,DKSRV18,HPS98,CDH$^+$19]. As solving LWE implies the solution to worst-case lattice problems, LWE is usually considered a lattice problem. However, this does not imply that lattice algorithms necessarily provide the best way for solving LWE. Moreover, many cryptosystems choose especially small secret keys for efficiency reasons, and to keep the probability of decryption errors low.

In this work we study the combinatorial complexity of recovering LWE keys chosen from a narrow range $\{-\eta, \dots, \eta\}$. Our analysis applies to common variants of LWE, such as Ring-LWE or Module-LWE, but we make no use of the additional structure that these LWE variants provide.

Previous Work on LWE Key Enumeration. Very little is known about directly enumerating LWE keys. A brute-force attack enumerates $\boldsymbol{s} \in \mathbb{Z}_q^N$, and checks whether $A\boldsymbol{s} - \boldsymbol{t}$ yields a small-norm error vector $\boldsymbol{e}$. If $\boldsymbol{s}$ has Shannon entropy $\mathcal{S}$, then the brute-force attack takes (expected) time $\mathcal{S}$, up to a polynomial runtime factor for checking key correctness. Throughout the paper, for ease of notation we ignore polynomial factors and round run time exponents upwards.

In a standard Meet-in-the-Middle attack, attributed to Odlyzko [SO97], we split $\boldsymbol{s}$ in two $N/2$-dimensional vectors $\boldsymbol{s}_1, \boldsymbol{s}_2$ and check whether $A\boldsymbol{s}_1 \approx \boldsymbol{t} - A\boldsymbol{s}_2 \bmod q$. The approximate matching of $A\boldsymbol{s}_1$ and $\boldsymbol{t} - A\boldsymbol{s}_2$ is usually realized by a locality-sensitive hash function. Up to polynomial factors, Odlyzko's attack takes time $\mathcal{S}^{\frac{1}{2}}$.

Recently, May [May21] showed that *ternary LWE keys* $\boldsymbol{s}, \boldsymbol{e} \in \{-1, 0, 1\}^N$ can be enumerated more efficiently in time roughly $\mathcal{S}^{\frac{1}{4}}$. His algorithm for NTRU-type schemes beats lattice reduction if $\boldsymbol{s}$ is overly sparse. May's technique is a natural recursive generalization of Odlyzko's Meet-in-the-Middle attack to search trees, using the so-called representation technique. This technique has been introduced in [HJ10] and successfully applied in the cryptographic context of decoding algorithms [MMT11,BJMM12].

Our Technical Contributions. We extend May's LWE key recovery algorithm in several ways.

– We first show that May's algorithm can be applied for LWE keys sampled from a probabilistic distribution. Since the purely combinatorial analysis in [May21] requires to know for every element in $\{-\eta, \dots, \eta\}$ the exact number of appearances in $\boldsymbol{s}$, we define for *any probability distribution* $\mathcal{P} = (p_{-\eta}, \dots, p_\eta)$ a so-called *core set* of vectors.
  We then show that length-N LWE keys randomly sampled coordinate-wise from $\mathcal{P}$ are in the core set with probability inverse polynomial in N. This core set density shows that our key enumeration already applies to a polynomial fraction of all keys.

- We then strengthen our result to almost all LWE keys $\boldsymbol{s}, \boldsymbol{e}$ by transforming almost any LWE instance in subexponential time $2^{\mathcal{O}(\sqrt{N})}$ to a permuted, weight-preserving LWE instance with keys $\boldsymbol{s}', \boldsymbol{e}'$ such that $\boldsymbol{s}'$ lies in the core set. Since our subsequent enumeration of $\boldsymbol{s}'$ takes time $2^{\mathcal{O}(N)}$, the transformation's subexponential overhead contributes only an $o(1)$-term to the run time exponent.
- We generalize the combinatorics of [May21] such that we can analyze secret vectors from $\{-2, \dots, 2\}$ and even from $\{-3, \dots, 3\}$. This introduces run time optimization parameters whose amount grows quadratically with the digit sets and linearly in the search tree depth. The optimization complexity is the reason that we only analyze up to $\eta \le 3$.
- Along this way we also generalize the ways in which the secret $\boldsymbol{s}$ can be represented. This is crucial in the representation technique, since more representations usually lead to better results. See as comparison the related subset sum literature that optimized run times by solely analyzing more powerful representations starting from $\{0, 1\}$ [HJ10], over $\{-1, 0, 1\}$ [BCJ11], to $\{-1, 0, 1, 2\}$ [BBSS20]. In this work, we introduce four different representations, called Rep-0 to Rep-3, with increasing complexity. Rep-3 representations are most powerful, and we eventually use Rep-3 to show our best results for the Kyber/Dilithium distributions.
- We analyze different probability distributions $\mathcal{P} = (p_{-\eta}, \dots, p_{\eta})$. For $\eta = 1$, we revisit weighted ternary key distributions and slightly improve over [May21] by using larger search tree depths. For $\eta = 2, 3$ we study the *Centered Binomial* distribution $\mathcal{B}(\eta)$ used in Kyber, and the uniform distribution used in Dilithium.

Our Results. Figure 1 shows our run times for $\boldsymbol{s}, \boldsymbol{e} \in \mathcal{B}(\eta)^N$, Kyber's Centered Binomial distribution. Kyber uses $\eta = 3$ in combination with $N = 256 \cdot 2 = 512$, and $\eta = 2$ in combination with $N = 256 \cdot 3 = 768$ and $N = 256 \cdot 4 = 1024$.

η	$\mathcal{T}$	$\mathcal{S}$	$\log_{\mathcal{S}}(\mathcal{T})$
1	$2^{0.337N}$	$2^{1.500N}$	**0.225**
2	$\mathbf{2^{0.357N}}$	$\mathbf{2^{2.031N}}$	**0.176**
3	$\mathbf{2^{0.371N}}$	$\mathbf{2^{2.334N}}$	**0.159**

Fig. 1. Run time $\mathcal{T}$ and entropy $\mathcal{S}$ of our LWE key enumeration algorithm for $\boldsymbol{s}$ sampled from a *Centered Binomial* distribution $\mathcal{B}(\eta)^N$, $\eta = 1, 2, 3$.

As one would expect with increasing η — i.e., broader distributions — the key entropy $\mathcal{S}$ and our run time $\mathcal{T}$ both increase. However, let us express our run time as a polynomial function of the entropy $\mathcal{T} = \mathcal{S}^c$ for some constant $c = \log_{\mathcal{S}}(\mathcal{T})$. Then we see that the run time exponent c actually decreases in Fig. 1 monotonously in η.

For $\eta = 2$ and $\eta = 3$ we have complexities around only $\mathcal{S}^{\frac{1}{6}}$, as opposed to the $\eta = 1$ ternary key case with complexity $\mathcal{S}^{0.225}$ (slightly improving over $\mathcal{S}^{0.232}$ achieved in [May21]). Thus, our generalizations for larger digit sets are for larger η *more effective*. This seems to be an artifact of the representation method. Our analysis shows that the entropy growth with larger digit sets is over-compensated by the growth of the number of representations, resulting in decreased run time exponents $c = \log_{\mathcal{S}}(\mathcal{T})$.

These results demonstrate the power of our new combinatorial LWE key search algorithm. Recall that the so far best known combinatorial Meet-in-the-Middle algorithm by Odlyzko achieved square root complexity $\mathcal{S}^{\frac{1}{2}}$, independent of η. For the case $\eta = 1$ the exponent was lowered to $c = 0.232$ in [May21]. Our work indicates that for Centered Binomials c as a function of η might be strictly decreasing.

The effect of a strictly decreasing exponent $c(\eta)$ is also reflected in the absolute run times $\mathcal{T}$ in Fig. 1. More precisely, when choosing keys from $\mathcal{B}(3)^N$ rather than ternary keys $\mathcal{B}(1)^N$, then our key enumeration algorithm's run time only mildly increases from $2^{0.337N}$ to $2^{0.371N}$. In other words, although we significantly increase the key entropy from $2^{1.5N}$ to $2^{2.334N}$ we do *not significantly increase the key security.* [1]

Other distributions. Besides the Centered Binomial distribution we also study the enumeration of randomly sampled ternary LWE keys of different weight, thereby slightly improving the results of [May21], and extending them to the probabilistic setting.

Moreover, we also study the uniform distribution $\mathcal{U}(\eta) = (p_{-\eta}, \dots, p_{\eta}) = \frac{1}{2\eta+1}$, widely used in cryptography. For example, some NTRU variants [CDH+19] sample their secret LWE keys from $\mathcal{U}(1)^N$. Dilithium chooses $\boldsymbol{s}, \boldsymbol{e} \in \mathcal{U}(2)^N$ with $N = 1024$ and $N = 2048$.

Our results for the uniform distribution are provided in Fig. 2. When comparing with Fig. 1, the lower entropy, more sharply zero-centered Binomial distribution yields slightly better run times than the uniform distribution in Fig. 2, as one would expect. However, maybe somewhat surprisingly, our results for $\mathcal{U}(1)^N$ and $\mathcal{U}(2)^N$ are not far off, only $\mathcal{U}(3)^N$ is significantly worse.

Relative to the entropy $\mathcal{S}$ we achieve for $\eta = 2$ again run time $\mathcal{S}^{\frac{1}{6}}$, but as opposed to the Centered Binomial distribution $c = \log_{\mathcal{T}} \mathcal{S}$ is for the uniform distribution not strictly decreasing with growing η.

Notice that we achieve for Dilithium's $\mathcal{U}(2)^N$ a run time $\mathcal{T}$ similar to Kyber's $\mathcal{B}(3)^N$. However, since Dilithium proposes much larger key lengths, our key enumeration is way more effective for Kyber parameter sets.

Asymptotics. We would like to stress that our LWE key search algorithm is at this point mainly of theoretical interest. Our run time analysis is asymptotic, and throughout our work we do not only generously suppress polynomial run time

[1] This conclusion is of course only valid relative to our algorithm. Relative to other algorithms like lattice reduction the key security might be behave differently.

η	$\mathcal{T}$	$\mathcal{S}$	$\log_{\mathcal{S}} \mathcal{T}$
1	$2^{0.345N}$	$2^{1.585N}$	0.218
2	$\mathbf{2^{0.378N}}$	$\mathbf{2^{2.322N}}$	**0.163**
3	$2^{0.493N}$	$2^{2.808N}$	0.176

Fig. 2. Run time $\mathcal{T}$ and entropy $\mathcal{S}$ of our enumeration algorithm for LWE keys sampled from a *uniform* distribution $\mathcal{U}(\eta)^N$, $\eta = 1, 2, 3$.

factors in soft-Oh notation $\tilde{\mathcal{O}}(\cdot)$, but also we also suppress two subexponential factors. First, as in [May21] we have to guess $r = (N/\log N)$ coordinates of $\boldsymbol{e}$ in slightly subexponential time 2^r. Second, our transformation to the core set of $\boldsymbol{s}$ introduces another subexponential $2^{\mathcal{O}(\sqrt{N})}$ run time factor.

Our analysis solely focuses on minimizing the run time exponent. Our memory consumption is almost as large as the run time exponent. Time-memory tradeoffs are possible, as in [May21], but we do not consider them in this work.

Significant further work would be required to bring our results to practice. We would like to draw an analogy to decoding algorithms, which also first solely focused on asymptotic improvements [MMT11,BJMM12]. It took a decade that these algorithms nowadays define the state-of-the-art in practical attacks against code-based cryptosystems [EMZ22].

LWE Representation Heuristic. Our LWE key enumeration uses the standard heuristic from representation based algorithms. Namely, we iteratively construct in our key search partial solutions as vectors sums, and treat these sums as independent in our analysis. This heuristic has been extensively verified experimentally in the context of subset sum and decoding algorithms [BBSS20,EMZ22]. On the theoretical side, it has been shown in [DRX17] that the dependence between vectors sums merely affects the overall runtime exponent by an $o(1)$-term. Thus, our heuristic can be considered quite mild.

In addition, we require that the LWE public key A is randomly chosen from $\mathbb{Z}_q^{N \times N}$, which is the case for standard LWE. In the case of Module-LWE (MLWE) we heuristically assume that A's structure does not affect our analysis.

When formulating theorems, we refer to these two heuristic assumptions as *MLWE Representation Heuristic.*

Lattices. Our results are slightly inferior to current lattice asymptotics of $2^{0.29\beta}$ from BKZ reduction, where the BKZ block length β is roughly of size $\frac{4}{5}N$. We find it quite remarkable that combinatorial key enumeration, at least in the case of quite narrow LWE keys as in Kyber and Dilithium, is not far off from highly evolved lattice reduction techniques. Even if key enumeration eventually cannot outperform lattice reduction, there exist other attack scenarios where direct key enumeration might be preferable over lattices, e.g. in the setting of partially known keys [BBSS20,EMVW22].

Organization of paper. After fixing notations in Section 2, we introduce in Section 3 a high-level version of our algorithm Search-LWE, adapted from [May21].

In Section 4, we show how to extend Search-LWE's analysis to probabilistically sampled keys. In Section 5, we provide a first simple instantiation of Search-LWE, called Rep-0, for an introduction into the representation technique. We then strengthen our results by introducing more elaborated representations Rep-1 to Rep-3 in Section 6. Section 7 contains an overview of our results for the weighted ternary, the Centered Binomial, and the uniform distribution.

We provide the source code for calculating our run times via
`https://anonymous.4open.science/r/HTELWEK-E491/`

2 Preliminaries

Unless explicitly stated otherwise, any log in this paper is base 2. For simplicity, we denote all vectors as column vectors and omit transposing them.

Binomial and Multinomial Coefficients. For $N_1, \ldots, N_\ell \in \mathbb{N}_0$ and $N = \sum_{i=1}^{\ell} N_i$, we denote the multinomial coefficient with

$$\tbinom{N}{N_1,\ldots,N_{\ell-1},\cdot} := \tbinom{N}{N_1,\ldots,N_\ell} = \tfrac{N!}{\prod_{i=1}^{\ell} N_i!}.$$

For $\ell = 2$, we write $\binom{N}{N_1}$ instead.

Shannon Entropy. We denote with H the *n-ary entropy function* [MU17] where for some $p_i, i \le n$ with $p_1 + \cdots + p_n = 1$

$$H(p_1, \ldots, p_{n-1}, \cdot) := H(p_1, \ldots, p_n) = -\textstyle\sum_{i=1}^{n} p_i \log(p_i).$$

For $n = 2$, we write $H(p)$ instead of $H(p, 1-p)$ or $H(p, \cdot)$. We always upper bound binomial and multinomial coefficients asymptotically using the formula

$$\tbinom{N}{p_1 N, \cdots, p_n N} = \Theta\left(N^{-\frac{n-1}{2}} \cdot 2^{H(p_1,\cdots,p_n)N}\right) = \tilde{\Theta}(2^{H(p_1,\cdots,p_n)N}). \tag{1}$$

Distributions. Probability distributions are denoted $\mathcal{P} = (p_{-\eta}, \ldots, p_\eta)$ where p_i is the probability to sample $i \in \{-\eta, \ldots, \eta\}$. We exclusively consider distributions that are symmetric around 0, i.e. distributions where $p_i = p_{-i}$, so indices are generally unsigned.

The weight of i in some vector $\boldsymbol{v}$, i.e. the amount of times i appears in $\boldsymbol{v}$, is denoted $\mathrm{wt}_i(\boldsymbol{v})$.

Sampling from a probability distribution $\mathcal{P}$ will be denoted with $s \sim \mathcal{P}$. If an element $\boldsymbol{s} \in \mathbb{Z}_q^N$ has its N coefficients drawn independently according to $\mathcal{P}$, we write $\boldsymbol{s} \sim \mathcal{P}^N$.

For some $\eta \in \mathbb{N}$, we denote the *Centered Binomial distribution* with

$$p_i = \frac{\binom{2\eta}{\eta+i}}{2^{2\eta}}. \tag{2}$$

LWE Keys. We attack standard LWE keys, where both $\boldsymbol{s}$ and $\boldsymbol{e}$ are randomly drawn from some narrow probability distribution over $\mathbb{Z}_q$. More precisely, for some prime $q \in \mathbb{N}$ and some $N \in \mathbb{N}$, given a random $A \in \mathbb{Z}_q^{N \times N}$ and $\boldsymbol{t} \in \mathbb{Z}_q^N$ where $\boldsymbol{t} = A\boldsymbol{s} + \boldsymbol{e}$ for $\boldsymbol{s}, \boldsymbol{e} \in \mathbb{Z}_q^N$ drawn from $\mathcal{P}^N$, we want to find the secret pair $(\boldsymbol{s}, \boldsymbol{e})$.

If we replace $\mathbb{Z}_q$ with ${}^{\mathbb{Z}_q[X]}/{}_{X^n-1}$ and N with k, this becomes Module-LWE, abbreviated MLWE. Our results can be applied to MLWE with $N = nk$.

3 How to Enumerate LWE Keys with May's Algorithm

Before we give a brief introduction into May's algorithm for LWE key enumeration [May21], let us briefly recall some basic techniques.

3.1 Brute-Force and Meet-in-the-Middle LWE Key Enumeration

Let $q \in \mathbb{N}$ be a public modulus, and $(A, \boldsymbol{t}) \in \mathbb{Z}_q^{N \times N} \times \mathbb{Z}_q^N$ be an LWE-instance satisfying $A\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{t} \bmod q$ for some $\boldsymbol{s}, \boldsymbol{e} \in \mathbb{Z}_q^N$ that have small coefficients (compared to q).

A *Brute-Force* LWE key enumeration searches over all potential secrets $\boldsymbol{s} \in \mathbb{Z}_q^N$, and checks whether the resulting error term $\boldsymbol{t} - A\boldsymbol{s}$ is sufficiently small. By construction, there is usually a unique $\boldsymbol{s}$ that satisfies this condition. If the potential $\boldsymbol{s}$ come from an exponential search space of size $\mathcal{S}$, then one has to iterate over $\Theta(\mathcal{S})$ potential $\boldsymbol{s}$, where each candidate can be tested in polynomial time. Thus, *Brute-Force* runs in time $\tilde{\mathcal{O}}(\mathcal{S})$. E.g. for random ternary $\boldsymbol{s} \in \{-1, 0, 1\}^N$ Brute-Force takes time $\tilde{\mathcal{O}}(3^N)$.

A classical *Meet-in-the-Middle* (MitM) LWE key enumeration equally splits $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2) \in \mathbb{Z}_q^{N/2} \times \mathbb{Z}_q^{N/2}$ and $A = (A_1, A_2) \in \mathbb{Z}_q^{N \times N/2} \times \mathbb{Z}_q^{N \times N/2}$. One then enumerates pairs $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ and checks whether $A_1\boldsymbol{s}_1$ approximately matches $\boldsymbol{t} - A_2\boldsymbol{s}_2$ modulo q, up to a small error term. The benefit is that $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ have half the dimension of $\boldsymbol{s}$, and the terms $A_1\boldsymbol{s}_1$, $\boldsymbol{t} - A_2\boldsymbol{s}_2$ can be computed independently. The approximate matching (up to the small error term) of $A_1\boldsymbol{s}_1$, $\boldsymbol{t} - A_2\boldsymbol{s}_2$ that finds the right pairs $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ can usually be done in polynomial time, using a locality-sensitive hashing approach due to Odlyzko [SO97]. This implies that classical MitM runs for secret $\boldsymbol{s}$ from a search space of size $\mathcal{S}$ in time $\tilde{O}(\sqrt{\mathcal{S}})$. For instance, for random ternary $\boldsymbol{s} \in \{-1, 0, 1\}^N$, classical MitM takes time $\tilde{\mathcal{O}}(3^{N/2})$.

3.2 High-Level Idea of the Algorithm

May's LWE key enumeration [May21] can be seen as a Meet-in-the-Middle attack, where we *additively* split $\boldsymbol{s} = \boldsymbol{s}_1 + \boldsymbol{s}_2$ with $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathbb{Z}_q^N$. As opposed to classical MitM the benefit of $\boldsymbol{s}$'s splitting does *not* come from dimension reduction, but from the following three properties.

Reduced Search Space. $\boldsymbol{s}_1, \boldsymbol{s}_2$ are usually easier to enumerate, i.e. they are defined over smaller search spaces. For instance, [May21] uses for enumerating ternary keys $\boldsymbol{s}_1, \boldsymbol{s}_2$ of roughly half the Hamming weight of $\boldsymbol{s}$.

Recursion. [May21] recursively splits $\boldsymbol{s}_1, \boldsymbol{s}_2$ as sums of N-dimensional vectors that are (yet) defined over smaller search spaces. This recursion eventually results in a complete binary search tree of some optimal depth d. The optimization of the search spaces over all tree levels is a non-trivial optimization problem.

Ambiguous Representations. The secret $\boldsymbol{s}$ can be expressed in exponentially many ways as a sum $\boldsymbol{s}_1 + \boldsymbol{s}_2$. The algorithm uses these so-called *representations* of $\boldsymbol{s}$ to fix a special representation s.t. $A_1\boldsymbol{s}_1$, $\boldsymbol{t} - A_2\boldsymbol{s}_2$ take a fixed predefined value on certain coordinates.

In order to use representations, for some candidate $\boldsymbol{s}_1, \boldsymbol{s}_2$ we thus have to fix the values $A_1\boldsymbol{s}_1$, $\boldsymbol{t} - A_2\boldsymbol{s}_2$ on certain r coordinates. Let us fix zeros on these coordinates for simplicity. Recall however that the values $A_1\boldsymbol{s}_1$, $\boldsymbol{t} - A_2\boldsymbol{s}_2$ still differ by the unknown error vector $\boldsymbol{e}$. Thus, May's algorithm first guesses r coordinates of $\boldsymbol{e}$. The algorithm's high-level structure is described in Algorithm 1.

Algorithm 1: LWE-Search [May21]

```
  Input  : $A \in \mathbb{Z}_q^{N \times N}, \boldsymbol{t} \in \mathbb{Z}_q^N$
  Output: Small norm $\boldsymbol{s} \in \mathbb{Z}_q^N$ s.t. $\boldsymbol{e} := A\boldsymbol{s} - \boldsymbol{t}$ has small norm
1 Guess $r$ coordinates of $\boldsymbol{e}$, denoted $\boldsymbol{e}_r$.
2 for all $\boldsymbol{s}_1, \boldsymbol{s}_2$ such that $A\boldsymbol{s}_1 = \boldsymbol{0}^r = \boldsymbol{t} - A\boldsymbol{s}_2 + \boldsymbol{e}_r$ on these $r$ coordinates and
    $A\boldsymbol{s}_1 \approx \boldsymbol{t} - A\boldsymbol{s}_2$ on the remaining $n - r$ coordinates do
3     Output $\boldsymbol{s} = \boldsymbol{s}_1 + \boldsymbol{s}_2$
4 end
```

Algorithm 1 was instantiated and analyzed in [May21] only for ternary vector $\boldsymbol{s} \in \{-1, 0, 1\}^N$ with a predefined number of ± 1-entries. However, the algorithm may as well be instantiated with any notion of smallness of $\boldsymbol{s}, \boldsymbol{e}$ (in comparison to q). Throughout the paper, we assume that $\boldsymbol{s}, \boldsymbol{e}$ are sampled from a constant size range $\{-\eta, \ldots, \eta\}$. I.e., the max-norm of $\boldsymbol{s}, \boldsymbol{e}$ does not grow as a function of q, as opposed to e.g. Regev's original cryptosystem [Reg05].

The narrow max-norm distributions that we address in this work are typical for highly practical lattice-based schemes like Kyber and Dilithium. In the narrow max-norm distribution setting for $\boldsymbol{e}$ (only, we do not need constant max-norm $\boldsymbol{s}$ at this point) the following holds.

Subexponential Key Guessing. Let $\mathcal{R}$ be the number of representations of $\boldsymbol{s}$. In Algorithm 1 we choose r, the number of guessed error coordinates, such that on expectation at least one representation survives. The probability that a representation $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ satisfies the condition $A\boldsymbol{s}_1 = \mathbf{0}^r$ in Algorithm 1 is q^{-r}. Thus, a representation survives if $\mathcal{R}q^{-r} \geq 1$. Since in the following $\mathcal{R} = 2^{\mathcal{O}(N)}$ and $q = \Theta(N)$, we obtain $r = \mathcal{O}(\frac{N}{\log N})$. Thus, for every constant max-norm $\boldsymbol{e}$, Algorithm 1 requires for the key guessing in step 1 subexponential time

$$(\mathcal{O}(1))^r = 2^{\mathcal{O}(\frac{N}{\log N})}.$$

As a consequence all our (exponential) run times that we achieve throughout this work include an additional slightly subexponential run time factor. Asymptotically, this contributes a factor of $(1 + o(1))$ to the exponent, and can asymptotically be ignored by rounding the run time exponent upwards. In practice, such a factor might significantly affect concrete run time estimates.

Efficient LSH. An approximate matching $A_1\boldsymbol{s}_1 \approx \boldsymbol{t} - A_2\boldsymbol{s}_2 \bmod q$ with Odlyzko's locality sensitive hash function (LSH) includes a constant run time overhead $\mathcal{O}(1)$ over an exact matching of lists (via sorting), when we match up to a constant max-norm error vector $\boldsymbol{e}$, see [May21]. Thus, in the following, we ignore LSH costs.

4 Enumerating Keys from a Probabilistic Distribution

As we discussed in the previous section, LWE-Search (Algorithm 1) was originally instantiated and analyzed with ternary $\boldsymbol{s}, \boldsymbol{e} \in \{-1, 0, 1\}^N$, where $\boldsymbol{s}$ has a fixed number of $-1, 0, 1$ that sum to 0. In our notation, we have to know the weights $\mathrm{wt}_{-1}(\boldsymbol{s}), \mathrm{wt}_0(\boldsymbol{s}), \mathrm{wt}_1(\boldsymbol{s})$.

Larger Range. In subsequent sections, we extend to vectors $\boldsymbol{s}, \boldsymbol{e} \in \{-\eta, \ldots, \eta\}^N$, where $\eta = 2, 3$. While the analysis already gets quite cumbersome for $\eta > 1$, these calculations with fixed weights $\mathrm{wt}_{-\eta}(\boldsymbol{s}), \ldots, \mathrm{wt}_{\eta}(\boldsymbol{s})$ still can be seen as a generalization of May's original analysis for ternary keys.

Handling Probabilism. Furthermore —motivated by our applications Kyber and Dilithium— we want to deal with keys $\boldsymbol{s}, \boldsymbol{e}$ that are sampled from a *probabilistic distribution* $\mathcal{P}$. This causes problems, since we do not know the weights of $\boldsymbol{s}$ anymore, and these weights also vary in a certain range. The case of probabilistic distributions is not covered by [May21], and per se it is not clear that LWE-Search permits a proper analysis in this case.

In this section, we first show that for *any* probabilistic distribution $\mathcal{P} = (p_{-\eta}, \ldots, p_{\eta})$ with constant η, a polynomial fraction of all secret keys $\boldsymbol{s} \in \mathcal{P}^N$ has weights $\mathrm{wt}_{-\eta}(\boldsymbol{s}) = p_{-\eta}N, \ldots, \mathrm{wt}_{\eta}(\boldsymbol{s}) = p_{\eta}N$. I.e., all weights achieve their expected values. Notice that we might have to round p_iN to the next integer. Since the effect of rounding can (asymptotically in N) be neglected, for ease of notation we always assume that all p_iN are integers.

Hence, if we analyze LWE-Search with all weights fixed to their expectation, we already obtain a key enumeration attack that succeeds for a *polynomial fraction of all keys.*

Attacking (Almost) All Keys. We further extend this result by introducing a *permutation technique*. In a nutshell, we permute the entries of $\boldsymbol{s}, \boldsymbol{e} \in \mathcal{P}^N$, until $\boldsymbol{s}$ achieves its expected weights. We then show that this event happens on expectation within a subexponential number $2^{\mathcal{O}(\sqrt{N})}$ of iterations for all but a (tunably very small) constant fraction of keys. Hence, for almost all keys we can run LWE-Search on a subexponential number of permuted instances, where each instance enumerates (the same expected) weights. This adds only a subexponential overhead $2^{\mathcal{O}(\sqrt{N})}$ to our exponential time algorithm, whose cost we hide in an $o(1)$-term in subsequent sections.

In other words, we show that, for *any* probability distribution, one may analyze LWE-Search w.l.o.g. with the weights of $\boldsymbol{s}$ fixed to their expectation. As a consequence, our results hold for a $(1 - o(1))$-fraction of all probabilistically sampled $\boldsymbol{s}, \boldsymbol{e} \in \mathcal{P}^N$.

4.1 A Polynomial Fraction of All Keys Achieves Expectations

Let us define what we mean by the event that a vector $\boldsymbol{v}$ sampled from some probability distribution $\mathcal{P}$ achieves its expected number of entries. We call the set of these vectors a *core set*.

Definition 1 (core set). *Let $N, \eta \in \mathbb{N}$. Let $\mathcal{P} = (p_{-\eta}, \ldots, p_\eta)$ be a probability distribution. We define the* core set *of N-dimensional vectors over $\mathcal{P}$ as*

$$\mathcal{C}(\mathcal{P}) := \{\boldsymbol{v} \in \{-\eta, \ldots, \eta\}^N \mid \mathrm{wt}_i(\boldsymbol{v}) = p_i N \text{ for all } -\eta \le i \le \eta\},$$

where w.l.o.g. (asymptotically in N) we assume that $p_i N \in \mathbb{N}$ for all i.

Next, we show that for any discrete probability distribution $\mathcal{P} = (p_{-\eta}, \ldots, p_\eta)$ with constant η, a length-N vector $\boldsymbol{v}$ randomly sampled coordinate-wise according to $\mathcal{P}$ belongs to the core set $\mathcal{C}(\mathcal{P})$ with probability inverse polynomial in N.

Lemma 1. *Let $\mathcal{P} = (p_{-\eta}, \ldots, p_\eta)$ be some probability distribution, and let $N \in \mathbb{N}$ be such that $N_i := p_i N \in \mathbb{N}$ for all i. Then, $\boldsymbol{v} \sim \mathcal{P}^N$ is in the core set $\mathcal{C}(\mathcal{P})$ with probability at least $\Omega(\frac{1}{N^\eta})$.*

Proof. Let $\boldsymbol{v} \sim \mathcal{P}^N$. Then we have for all i that $\mathrm{wt}_i(\boldsymbol{v}) := N_i := p_i N$ with probability

$$\mathbb{P}[\boldsymbol{v} \in \mathcal{C}(\mathcal{P})] = \binom{N}{N_{-\eta}, \cdots, N_\eta} \cdot \prod_{-\eta \le i \le \eta} p_i^{N_i}.$$

We bound the multinomial coefficient using Eq. (1) as

$$\begin{aligned}\mathbb{P}[\boldsymbol{v} \in \mathcal{C}(\mathcal{P})] &= \Omega\left(\frac{1}{N^\eta} \cdot 2^{H(p_{-\eta}, \cdots, p_\eta)N}\right) \cdot 2^{\sum_{-\eta \le i \le \eta} N_i \log p_i} \\ &= \Omega\left(\frac{1}{N^\eta}\right) \cdot 2^{H(p_{-\eta}, \cdots, p_\eta)N} \cdot 2^{-H(p_{-\eta}, \cdots, p_\eta)N} = \Omega\left(\frac{1}{N^\eta}\right). \quad \square\end{aligned}$$

By Lemma 1, any attack that works for LWE keys $\boldsymbol{s}$ from the core set $\mathcal{C}(\mathcal{P})$ with probability ε also works for any key $\boldsymbol{s} \sim \mathcal{P}^N$ with probability $\Omega(\frac{1}{N^\eta}) \cdot \varepsilon$, where the last probability is taken over the random choice of $\boldsymbol{s}$.

Algorithm 2: PERMUTE-LWE

```
  Input  : LWE instance $(A, \boldsymbol{t})$ such that $A\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{t} \bmod q$
  Output: LWE instance $(A', \boldsymbol{t}')$ such that $A'\boldsymbol{s}' + \boldsymbol{e}' = \boldsymbol{t}' \bmod q$ , where $(\boldsymbol{s}', \boldsymbol{e}')$
          is a permutation of $(\boldsymbol{s}, \boldsymbol{e})$
1 repeat
2     Choose a random permutation matrix $P \in \mathbb{Z}_q^{2N \times N}$. ;
3     Compute $(B \mid C) = (A \mid I_n) \cdot P^{-1}$ with $B, C \in \mathbb{Z}_q^{N \times N}$.
4 until $C$ is invertible;
5 $(A', \boldsymbol{t}') := (C^{-1}B, C^{-1}\boldsymbol{t})$ ;
```

4.2 Attacking Almost All Keys via Permutations

In order to attack almost all keys we devise a simple permutation technique that exchanges coordinates in $\boldsymbol{s}$ and $\boldsymbol{e}$. Our goal is to show that for almost all keys a subexponential number of permutations yields a permuted secret $\boldsymbol{s}'$ from the core set.

Permutation Technique. Let $A\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{t} \bmod q$ be an LWE instance with a square $A \in \mathbb{Z}_q^{N \times N}$. We can rewrite this equation in the form

$$(A \mid I_N) \cdot (\boldsymbol{s} \mid \boldsymbol{e}) = \boldsymbol{t} \bmod q.$$

Let $P \in \mathbb{Z}_q^{2N \times 2N}$ be a permutation matrix, and let $(A \mid I_n) \cdot P^{-1} = (B \mid C)$, where $B, C \in \mathbb{Z}_q^{N \times N}$. Then clearly

$$(B \mid C) \cdot P(\boldsymbol{s} \mid \boldsymbol{e}) = \boldsymbol{t} \bmod q.$$

Assume that C is invertible with inverse $C^{-1} \in \mathbb{Z}_q^{N \times N}$. Then

$$(C^{-1}B|I_N) \cdot P(\boldsymbol{s} \mid \boldsymbol{e}) = C^{-1}\boldsymbol{t} \bmod q.$$

Define $A' := C^{-1}B \in Z_q^{N \times N}$, $(\boldsymbol{s}', \boldsymbol{e}') := P(\boldsymbol{s} \mid \boldsymbol{e})$, and $\boldsymbol{t}' = C^{-1}\boldsymbol{t}$. Then we obtain a new LWE instance

$$A'\boldsymbol{s}' + \boldsymbol{e}' = \boldsymbol{t}',$$

where the coordinates of $(\boldsymbol{s}', \boldsymbol{e}')$ are a permutation of the coordinates of $(\boldsymbol{s}, \boldsymbol{e})$. Notice that a random matrix is invertible over $\mathbb{Z}_q$ with probability $\prod_{i=1}^{N}(1 - q^{-i}) \geq \frac{1}{4}$ [Wat87].

This gives us the following algorithm PERMUTE-LWE (Algorithm 2) with expected polynomial run time $\mathcal{O}(N^3)$.

Any LWE key $\boldsymbol{v} = (\boldsymbol{s}, \boldsymbol{e}) \sim \mathcal{P}^{2N}$ has expected weight $\mathrm{wt}_i(\boldsymbol{v}) = 2Np_i$ for all i. Intuitively, if $\mathrm{wt}_i(\boldsymbol{v})$ is not significantly smaller than $2Np_i$ for any i, then PERMUTE-LWE should have a good chance to produce some $\boldsymbol{s}' \in \mathcal{C}(\mathcal{P})$. This motivates our following definition of well-balanced vectors.

Definition 2. *Let $\mathcal{P} = (p_{-\eta}, \dots, p_\eta)$ be a probability distribution. We call an LWE key $(\boldsymbol{s}, \boldsymbol{e}) \sim \mathcal{P}^{2N}$* c-well-balanced *if for any $-\eta \leq i \leq \eta$ and some constant c, $(\boldsymbol{s}, \boldsymbol{e})$ contains at least $2Np_i - c\sqrt{Np_i}$ many i-entries.*

We now show that any randomly sampled vector $(\boldsymbol{s}, \boldsymbol{e}) \in \mathcal{P}^N$ is with constant probability c-well-balanced. With growing c, this probability goes exponentially fast to 1. T.i. for sufficiently large c, almost all keys $(\boldsymbol{s}, \boldsymbol{e})$ are c-well-balanced.

Lemma 2. *Let $\mathcal{P} = (p_{-\eta}, \dots, p_\eta)$ be a probability distribution. Then an LWE-key $(\boldsymbol{s}, \boldsymbol{e}) \sim \mathcal{P}^{2N}$ is* c-well-balanced *with probability at least $1 - (2\eta + 1)e^{-c^2/2}$.*

Proof. Let X_i be a random variable for the number of i-entries in $(\boldsymbol{s}, \boldsymbol{e}) \sim \mathcal{P}^{2N}$. Then $\mu := \mathbb{E}[X_i] = 2Np_i$. We apply the Chernoff bound

$$\Pr[X_i \leq (1-\delta)\mu] \leq e^{-\frac{\mu\delta^2}{2}}$$

with the choice $\delta = \frac{c}{\sqrt{Np_i}}$, which yields

$$\Pr[X_i \leq 2Np_i - c\sqrt{Np_i}] \leq e^{-\frac{c^2}{2}}.$$

An application of the union bound shows the statement. $\square$

Now that we know that almost all keys are c-well-balanced, we show that any LWE instance with c-well-balanced keys $(\boldsymbol{s}, \boldsymbol{e})$ can in subexponential time turned via Permute-LWE into an LWE-instance with a secret $\boldsymbol{s}'$ in the core set, for which we analyze our instantiations of LWE-Search (Algorithm 1) in subsequent sections.

Lemma 3. *Let $\mathcal{P} = (p_{-\eta}, \dots, p_\eta)$ be a probability distribution, and $(A, \boldsymbol{t})$ be an LWE instance with* c-well-balanced *LWE-key $(\boldsymbol{s}, \boldsymbol{e}) \sim \mathcal{P}^{2N}$. Then on expectation* Permute-LWE *outputs an LWE instance $(A, \boldsymbol{t}')$ with $\boldsymbol{s}' \in \mathcal{C}(\mathcal{P})$ after $2^{\mathcal{O}(\sqrt{N})}$ trials.*

Proof. Since $(\boldsymbol{s}, \boldsymbol{e})$ is c-well-balanced, we have $\mathrm{wt}_i(\boldsymbol{s}, \boldsymbol{e}) \geq 2Np_i - c\sqrt{Np_i}$ for any $i \in \{-\eta, \dots, \eta\}$. Thus, we obtain

$$\Pr[\boldsymbol{s} \in \mathcal{C}(\mathcal{P})] \geq \frac{\binom{2Np_{-\eta} - c\sqrt{Np_{-\eta}}}{Np_{-\eta}} \cdot \ldots \cdot \binom{2Np_\eta - c\sqrt{Np_\eta}}{Np_\eta}}{\binom{2N}{N}}.$$

Using Eq. (1) and neglecting polynomial terms we obtain for the exponent

$$\log \Pr[\boldsymbol{s} \in \mathcal{C}(\mathcal{P})] \geq -2N + \sum_{i=-\eta}^{\eta} H\left(\frac{1}{2 - \frac{c}{Np_i}}\right) \cdot \left(2 - \frac{c}{\sqrt{Np_i}}\right) Np_i.$$

For any $x \leq \frac{1}{2}$ we have $H(x) \geq 2(1-x)$, leading to

$$\begin{aligned}
\log \Pr[\boldsymbol{s} \in \mathcal{C}(\mathcal{P})] &\geq -2N + \sum_{i=-\eta}^{\eta} 2\left(1 - \frac{1}{2 - \frac{c}{Np_i}}\right) \cdot \left(2 - \frac{c}{\sqrt{Np_i}}\right) Np_i. \\
&= -2N + 2\sum_{i=-\eta}^{\eta} \left(1 - \frac{c}{\sqrt{Np_i}}\right) Np_i \\
&= -2c \sum_{i=-\eta}^{\eta} \sqrt{Np_i} = -\Theta(\sqrt{N}).
\end{aligned}$$

Thus, we expect that after $(\Pr[\boldsymbol{s} \in \mathcal{C}(\mathcal{P})])^{-1} = 2^{\mathcal{O}(\sqrt{N})}$ iterations PERMUTE-LWE outputs an LWE-instance with a secret $\boldsymbol{s}$ in the core set $\mathcal{C}(\mathcal{P})$. $\square$

5 Instantiating LWE-SEARCH with Simple (Rep-0) Representations

In this section, we show how to instantiate LWE-SEARCH (Algorithm 1) from Section 3 with both $\boldsymbol{s}, \boldsymbol{e}$ sampled from the Centered Binomial distribution $\mathcal{B}(3)^N$. In the previous Section 4 we showed that for any distribution $\mathcal{P}$ it suffices to instantiate LWE-SEARCH with secret $\boldsymbol{s}$ chosen from the core set $\mathcal{C}(\mathcal{P})$, that fixes all weights to their expectations, see Definition 1. Therefore, in the following we assume that $\boldsymbol{s} \in \mathcal{C}(\mathcal{P})^N$.

Our first LWE-SEARCH instantiation is mainly for didactic reasons. We assume that the reader is not familiar with the representation technique. Therefore, we define an especially simple representation, called REP-0, to illustrate the analysis. In subsequent sections, we further refine and parametrize our representations, called REP-1, REP-2 and REP-3. While these refinements complicate the analysis, they also lead to significantly stronger results.

Representation Warmup. To introduce the basic concept of representations, let us start for simplicity with a secret binary vector $\boldsymbol{s} \in \{0,1\}^N$ with an even weight $\mathrm{wt}_1(\boldsymbol{s}) \equiv 0 \mod 2$. We represent $\boldsymbol{s}$ as the sum of two vectors $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)} \in \{0,1\}^N$ where $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}$ both have half the weight $\mathrm{wt}_1(\boldsymbol{s}^{(1)}) = \mathrm{wt}_1(\boldsymbol{s}^{(2)}) = \frac{\mathrm{wt}_1(\boldsymbol{s})}{2}$.

i	Representations of i						
0				$\mathbf{0+0}$			
1				$\mathbf{0+1}$	$\mathbf{1+0}$		
2					$\mathbf{1+1}$		
3					$\mathbf{1+2}$	$\mathbf{2+1}$	

Fig. 3. REP-0 representations of $i \in \{0,\ldots,3\}$, $i < 0$ are represented by changing signs.

Now, every 1-coefficient of $\boldsymbol{s}$ can either be presented as $1+0$, i.e., a 1 in $\boldsymbol{s}^{(1)}$ and a 0 in $\boldsymbol{s}^{(2)}$, or vice versa as $0+1$. Every 0-coefficient of $\boldsymbol{s}$ is uniquely represent by $0+0$, i.e., by a zero in both coefficients of $\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}$.

As an example, for the weight-4 vector $\boldsymbol{s} = (1\ 1\ 1\ 1\ 0\ 0)$ we obtain the following $6 = \binom{4}{2}$ representations

$$\begin{aligned} \boldsymbol{s} &= (1\ 1\ 0\ 0\ 0\ 0) + (0\ 0\ 1\ 1\ 0\ 0) = (1\ 0\ 1\ 0\ 0\ 0) + (0\ 1\ 0\ 1\ 0\ 0) \\ &= (1\ 0\ 0\ 1\ 0\ 0) + (1\ 0\ 1\ 0\ 0\ 0) = (0\ 1\ 1\ 0\ 0\ 0) + (1\ 0\ 0\ 1\ 0\ 0) \\ &= (0\ 1\ 0\ 1\ 0\ 0) + (0\ 1\ 1\ 0\ 0\ 0) = (0\ 0\ 1\ 1\ 0\ 0) + (1\ 1\ 0\ 0\ 0\ 0). \end{aligned}$$

Rep-0. Let us define simple representations, called Rep-0, for $\boldsymbol{s} \in \{-3, \ldots, 3\}^N$. Our representations are illustrated in Fig.3. For instance, we represent a 3 in $\boldsymbol{s}$ as either $1+2$ or $2+1$, whereas a 2 is represented uniquely as $1+1$. For negative numbers we simply change sign, e.g. -1 is represented as either $0 + (-1)$ or $(-1) + 0$.

If a coefficient i has two representations, then we represent half of its occurrences with either representation. As an example, for $\boldsymbol{s} = (3\ 3\ 2\ 1\ 1\ 0\ 0)$, we obtain the $4 = \binom{2}{1} \cdot \binom{2}{1}$ representations

$$\begin{aligned} \boldsymbol{s} &= (2\ 1\ 1\ 1\ 0\ 0\ 0) + (1\ 2\ 1\ 0\ 1\ 0\ 0) = (1\ 2\ 1\ 1\ 0\ 0\ 0) + (2\ 1\ 1\ 0\ 1\ 0\ 0) \\ &= (2\ 1\ 1\ 0\ 1\ 0\ 0) + (1\ 2\ 1\ 1\ 0\ 0\ 0) = (1\ 2\ 1\ 0\ 1\ 0\ 0) + (2\ 1\ 1\ 1\ 0\ 0\ 0). \end{aligned}$$

Counting Representations. Let $\mathcal{R}$ denote the number of representations of $\boldsymbol{s} \in \{-3, \ldots, 3\}^N$. Let $\mathcal{R}_i$ denote the number of representations for entry i. Then $\mathcal{R} = \prod_{i=-3}^{3} \mathcal{R}_i$. Since we only consider distributions that are symmetric in 0, we obtain $\mathcal{R}_{-i} = \mathcal{R}_i$. For $i \in \{-2, 0, 2\}$ we have unique representations and therefore $\mathcal{R}_0 = \mathcal{R}_2 = \mathcal{R}_{-2} = 1$. For $i \in \{-3, -1, 1, 3\}$ we have two representations with equal splits, i.e.,

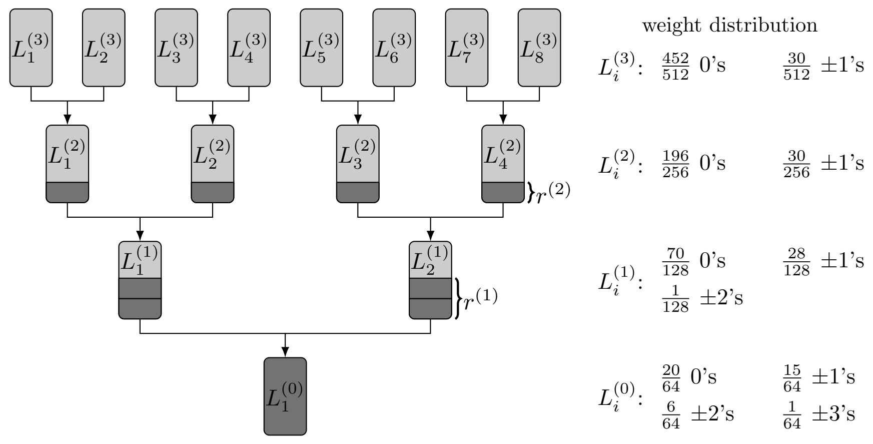
$$\mathcal{R}_1 = \mathcal{R}_{-1} = \binom{\mathrm{wt}_1(\boldsymbol{s})}{\frac{\mathrm{wt}_1(\boldsymbol{s})}{2}}, \qquad \mathcal{R}_3 = \mathcal{R}_{-3} = \binom{\mathrm{wt}_3(\boldsymbol{s})}{\frac{\mathrm{wt}_3(\boldsymbol{s})}{2}}.$$

As a conclusion, using Eq. (1) the number $\mathcal{R}$ of Rep-0 representations is

$$\mathcal{R} = \binom{\mathrm{wt}_1(\boldsymbol{s})}{\frac{\mathrm{wt}_1(\boldsymbol{s})}{2}}^2 \cdot \binom{\mathrm{wt}_3(\boldsymbol{s})}{\frac{\mathrm{wt}_3(\boldsymbol{s})}{2}}^2 = \tilde{\Theta}(2^{2\,\mathrm{wt}_1(\boldsymbol{s}) + 2\,\mathrm{wt}_3(\boldsymbol{s})}). \tag{3}$$

5.1 Rep-0 Instantiation of LWE-Search

In a nutshell, LWE-Search enumerates candidates for the LWE secret $\boldsymbol{s}$ in a list $L_1^{(0)}$. The candidates for $\boldsymbol{s}$ are represented as sums of $\boldsymbol{s}_1^{(1)}$ and $\boldsymbol{s}_2^{(1)}$, enumerated in lists $L_1^{(1)}$ and $L_2^{(1)}$, respectively, see Fig. 4 for an illustration. LWE-Search constructs candidates recursively, i.e., on level j in the search tree of Fig. 4 we construct all candidates $\boldsymbol{s}_i^{(j)}$ in list $L_i^{(j)}$ as the sum of candidates $\boldsymbol{s}_{2i-1}^{(j+1)}$ and $\boldsymbol{s}_{2i}^{(j+1)}$ in lists $L_{2i-1}^{(j+1)}$ and $L_{2i}^{(j+1)}$. In the simplified illustration of Fig. 4 we stopped the recursion in depth $d = 3$, but in general we have to optimize d.



$L_1^{(3)}$ $L_2^{(3)}$ $L_3^{(3)}$ $L_4^{(3)}$ $L_5^{(3)}$ $L_6^{(3)}$ $L_7^{(3)}$ $L_8^{(3)}$
$L_1^{(2)}$ $L_2^{(2)}$ $L_3^{(2)}$ $L_4^{(2)}$
$r^{(2)}$
$L_1^{(1)}$ $L_2^{(1)}$
$r^{(1)}$
$L_1^{(0)}$
weight distribution
$L_i^{(3)}$: $\frac{452}{512}$ 0's $\frac{30}{512}$ ±1's
$L_i^{(2)}$: $\frac{196}{256}$ 0's $\frac{30}{256}$ ±1's
$L_i^{(1)}$: $\frac{70}{128}$ 0's $\frac{28}{128}$ ±1's $\frac{1}{128}$ ±2's
$L_i^{(0)}$: $\frac{20}{64}$ 0's $\frac{15}{64}$ ±1's $\frac{6}{64}$ ±2's $\frac{1}{64}$ ±3's

Fig. 4. LWE-Search tree in depth $d = 3$ with relative weights for $\mathcal{B}(3)$.

Weights. In the root list $L_1^{(0)}$ we eventually enumerate the candidates for $\boldsymbol{s}$. Recall that $\boldsymbol{s} \in \mathcal{B}(3)^N$ and $\mathcal{B}(3)$ has by Eq. (2) probability distribution

$$(p_{-3}, \dots, p_3) = \left(\frac{1}{64}, \frac{6}{64}, \frac{15}{64}, \frac{20}{64}, \frac{15}{64}, \frac{6}{64}, \frac{1}{64}\right).$$

As shown in Section 4 it suffices to enumerate in the root list $L_1^{(0)}$ only vectors $\boldsymbol{s} \in \mathcal{C}(\mathcal{B}(3)^N)$ from the core set with weights

$$\mathrm{wt}_{-3}(\boldsymbol{s}) = p_{-3}N, \dots, \mathrm{wt}_3(\boldsymbol{s}) = p_3 N.$$

For notational convenience let us define *relative (to N) weights* $\omega_i^{(j)} := \frac{\mathrm{wt}_i(\boldsymbol{s})}{N}$ for entry i on level j of our LWE-search tree, see also Fig. 4. On the root level, we have relative weights for $i = 0, \dots 3$

$$\omega_i^{(0)} := p_i \qquad \text{and} \qquad \omega_{-i}^{(0)} := \omega_i^{(0)}.$$

In general, we define for ease of exposition $\omega_{-i}^{(j)} := \omega_i^{(j)}$ on all levels j.

From Fig. 3 we deduce that for our Rep-0 representations we can recursively calculate the relative weights level on level $j < d$ as

$$\omega_0^{(j)} = \frac{2\omega_0^{(j-1)} + 2\omega_1^{(j-1)}}{2}, \quad \omega_1^{(j)} = \frac{\omega_1^{(j-1)} + 2\omega_2^{(j-1)} + \omega_3^{(j-1)}}{2},$$
$$\omega_2^{(j)} = \frac{\omega_3^{(j-1)}}{2}, \qquad \omega_3^{(j)} = 0.$$

On level d, LWE-Search uses a classical Meet-in-the-Middle strategy (without representations) that splits the weights evenly. Thus, we obtain

$$\omega_1^{(d)} = \frac{\omega_1^{(d-1)}}{2}, \omega_2^{(d)} = \frac{\omega_2^{(d-1)}}{2}, \omega_3^{(d)} = \frac{\omega_3^{(d-1)}}{2}, \text{ and } \omega_0^{(d)} = 1 - 2\left(\omega_1^{(d)} + \omega_2^{(d)} + \omega_3^{(d)}\right).$$

The values of all relative weights for $\mathcal{B}(3)$ on all search tree levels are summarized in Fig. 4.

As an example, in both level-1 lists $L_1^{(1)}$, $L_2^{(1)}$ all vectors $\boldsymbol{s}_i^{(1)}$ have $\mathrm{wt}_0(\boldsymbol{s}_i^{(1)}) = \frac{70}{128}N$ many 0-entries, $\mathrm{wt}_1(\boldsymbol{s}_i^{(1)}) = \frac{28}{128}N$ many ± 1-entries each, $\mathrm{wt}_2(\boldsymbol{s}_i^{(1)}) = \frac{1}{128}N$ many ± 2-entries each, and no ± 3-entries.

Search Spaces. Now that we fixed the weight distributions on all levels of our LWE-Search tree we can define ***search spaces***, i.e., the amount $\mathcal{S}^{(j)}$ of vectors on level j that satisfy our weight distributions. We obtain

$$\mathcal{S}^{(j)} = \binom{N}{\omega_0 N, \omega_1 N, \omega_1 N, \ldots, \omega_3 N, \omega_3 N} = \tilde{\Theta}(2^{H(\omega_0, \omega_1, \omega_1, \ldots, \omega_3, \omega_3)N}). \tag{4}$$

Representations and Lists. Recall that our secret $\boldsymbol{s}$ has many representations as the sum of two vectors. LWE-Search uses the representations to significantly reduce search spaces. More precisely, if on level j we have $\mathcal{R}^{(j)}$ representations, then we cut the search space by a random $\frac{1}{\mathcal{R}^{(j)}}$-factor such that on expectation only a single representation remains. This search space reduction is the representation technique's core idea.

From Eq. (3), we already know the amount of representations on level 1. Let $\mathcal{R}^{(j)}$ denote the amount of level-j representations, Then $\mathcal{R}^{(1)} := \mathcal{R}$, and Eq. (3) easily generalizes to

$$\mathcal{R}^{(j+1)} = \binom{\mathrm{wt}_1(\boldsymbol{s}^{(j)})}{\frac{\mathrm{wt}_1(\boldsymbol{s}^{(j)})}{2}}^2 \cdot \binom{\mathrm{wt}_3(\boldsymbol{s}^{(j)})}{\frac{\mathrm{wt}_3(\boldsymbol{s}^{(j)})}{2}}^2 = \tilde{\Theta}(2^{(2\omega_1^{(j)} + 2\omega_3^{(j)})N}). \tag{5}$$

Recall that in the root list $L_1^{(0)}$ we store candidate secret keys $\boldsymbol{s}$, i.e.,

$$L_1^{(0)} = \{\boldsymbol{s} \in \mathcal{C}(\mathcal{B}(3)) \mid A \cdot \boldsymbol{s} - \boldsymbol{t} \in \{-3, \ldots, 3\}^N\}.$$

At level 1 of the search tree, we have $\mathcal{R}^{(1)}$ many representations of $\boldsymbol{s}$. Therefore, we have to cut the search space $\mathcal{S}^{(1)}$ by an $\frac{1}{\mathcal{R}^{(1)}}$-fraction. Let

$$r^{(1)} := \lfloor \log_q(\mathcal{R}^{(1)}) \rfloor = \mathcal{O}\left(\frac{N}{\log N}\right).$$

Let

$$\pi_r : \mathbb{Z}_q^N \to \mathbb{Z}_q^r, \ (e_1, \ldots, e_N) \mapsto (e_1, \ldots, e_r)$$

denote the projection on the first r coordinates. In LWE-Search we guess $\boldsymbol{e}_r := \pi_{r^{(1)}}(\boldsymbol{e})$ in subexponential time $\mathcal{O}\left(7^{r^{(1)}}\right) = 2^{\mathcal{O}(\frac{N}{\log N})}$.

Let $\boldsymbol{s}_1^{(1)}, \boldsymbol{s}_2^{(1)}$ be a representation of the secret key $\boldsymbol{s}$. Then $A\boldsymbol{s}_1^{(1)} + \boldsymbol{e} = \boldsymbol{t} - A\boldsymbol{s}_2^{(1)}$, which implies

$$\pi_{r^{(1)}}\left(A\boldsymbol{s}_1^{(1)}\right) + \boldsymbol{e}_r = \pi_{r^{(1)}}\left(\boldsymbol{t} - A\boldsymbol{s}_2^{(1)}\right). \tag{6}$$

By the randomness of A, the left and right hand side of Eq. (6) takes random values in $\mathbb{Z}_q^{r^{(1)}}$. Thus, for every target value $\boldsymbol{v} \in \mathbb{Z}_q^{r^{(1)}}$ any representation $\boldsymbol{s}_1, \boldsymbol{s}_2$ of $\boldsymbol{s}$ takes on both sides of Eq. (6) value $\boldsymbol{v}$ with probability $q^{-r^{(1)}}$. As a consequence, we expect that $\mathcal{R}^{(1)} \cdot q^{-r^{(1)}} \geq 1$ representations take value $\boldsymbol{v}$. For ease of exposition, we choose $\boldsymbol{v} = \mathbf{0}^{r^{(1)}}$ in the following, but in a real implementation one could randomize target values $\boldsymbol{v}$.
Hence, we define level-1 lists

$$L_1^{(1)} := \{\boldsymbol{s}_1^{(1)} \mid \pi_{r^{(1)}}(A\boldsymbol{s}_1^{(1)}) = -\boldsymbol{e}_r\},$$
$$L_2^{(1)} := \{\boldsymbol{s}_2^{(1)} \mid \pi_{r^{(1)}}(\boldsymbol{t} - A\boldsymbol{s}_2^{(1)}) = \mathbf{0}^{r^{(1)}}\}.$$

On level 2 to $d-1$, we algorithmically take the same approach as on level 1. As an example, let us derive the level-2 list descriptions. On level 2, we have $\mathcal{R}^{(2)}$ representations, and thus cut the search space $\mathcal{S}^{(2)}$ by an $\frac{1}{\mathcal{R}^{(2)}}$-fraction. To this end, define $r^{(2)} := \lfloor \log_q(\mathcal{R}^{(2)}) \rfloor$. Let $\boldsymbol{s}_1^{(2)}, \boldsymbol{s}_2^{(2)}$ and $\boldsymbol{s}_3^{(2)}, \boldsymbol{s}_4^{(2)}$ be representations of $\boldsymbol{s}_1^{(1)}$ and $\boldsymbol{s}_2^{(1)}$, respectively. Then we obtain level-2 lists

$$L_1^{(2)} := \{\boldsymbol{s}_1^{(2)} \mid \pi_{r^{(2)}}(A\boldsymbol{s}_1^{(2)}) = \mathbf{0}^{r^{(2)}}\},$$
$$L_2^{(2)} := \{\boldsymbol{s}_2^{(2)} \mid \pi_{r^{(2)}}(A\boldsymbol{s}_2^{(2)} + \boldsymbol{e}_r) = \mathbf{0}^{r^{(2)}}\},$$
$$L_3^{(2)} := \{\boldsymbol{s}_3^{(2)} \mid \pi_{r^{(2)}}(\boldsymbol{t} - A\boldsymbol{s}_3^{(2)}) = \mathbf{0}^{r^{(2)}}\},$$
$$L_4^{(2)} := \{\boldsymbol{s}_4^{(2)} \mid \pi_{r^{(2)}}(A\boldsymbol{s}_4^{(2)})) = \mathbf{0}^{r^{(2)}}\}.$$

Eventually, all level-d lists are constructed in a standard Meet-in-the-Middle manner by splitting each $\boldsymbol{s}_i^{(d-1)}$ in two $N/2$-dimensional vectors $\boldsymbol{s}_{2i-1}^{(d)}, \boldsymbol{s}_{2i}^{(d)}$.

Run Time Analysis. On levels $1 \leq j < d$ we enumerate an $\frac{1}{\mathcal{R}^{(j)}}$-fraction of the search space size $\mathcal{S}^{(j)}$, i.e., we obtain level-j list sizes

$$\mathcal{L}^{(j)} = \frac{\mathcal{S}^{(j)}}{\mathcal{R}^{(j)}}. \tag{7}$$

The level-d lists are constructed by a classical square-root complexity Meet-in-the-Middle approach for a search space of size $\mathcal{S}^{(d-1)}$. Thus, we obtain

$$\mathcal{L}^{(d)} = \sqrt{\mathcal{S}^{(d-1)}}.$$

Since root list $L_1^{(0)}$ can be constructed on-the-fly and must not be stored, we obtain a total memory consumption of

$$\mathcal{M} = \max\{\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(d)}\}.$$

Let $r^{(d)} = 0$. For constructing lists on level $1 \leq j < d$, we match two neighboring lists $L_{2i-1}^{(j+1)}$, $L_{2i}^{(j+1)}$ of size $\mathcal{L}^{(j+1)}$ on $r^{(j)} - r^{(j+1)}$ coordinates into a list $L_i^{(j)}$.

d	$\log \mathcal{T}^{(0)}$	$\log \mathcal{T}^{(1)}$	$\log \mathcal{T}^{(2)}$	$\log \mathcal{T}^{(3)}$	$\log \mathcal{M}$
3	$1.090N$	$\mathbf{1.103}N$	$.583N$	-	$1.045N$
4	$1.090N$	$\mathbf{1.103}N$	$.605N$	$.405N$	$1.045N$

Fig. 5. Rep-0 complexity exponents for $\mathcal{B}(3)^N$ using LWE-Search with depths $d = 3, 4$. Bold exponents indicate the dominating term.

Neglecting low order terms (e.g. for sorting), this can be done in time

$$\mathcal{T}^{(j)} = \max\left\{\mathcal{L}^{(j+1)}, \frac{(\mathcal{L}^{(j+1)})^2}{q^{r^{(j)}-r^{(j+1)}}}\right\}. \tag{8}$$

We then filter out all $\boldsymbol{s}_i^{(j)} \in L_i^{(j)}$ that do not have the correct weight distribution, resulting in list size $\mathcal{L}^{(j)}$.

The root list $L_1^{(0)}$ results from approximately matching both level-1 lists of size $\mathcal{L}^{(1)}$ via Odlyzko's hash function on the remaining $N - r^{(1)}$ coordinates that were previously unmatched. This can be done in time

$$\mathcal{T}^{(0)} = \max\left\{\mathcal{L}^{(1)}, \frac{(\mathcal{L}^{(1)})^2}{2^{N-r^{(1)}}}\right\}. \tag{9}$$

We obtain as total run time complexity

$$\mathcal{T} = \max\left\{\mathcal{T}^{(0)}, \dots, \mathcal{T}^{(d-1)}\right\}. \tag{10}$$

We analyzed LWE-Search in depths $d = 3, 4$. All run time exponents are given in Fig. 5. We observe that depth 3 is already sufficient, since depth 4 does not reduce the maximal exponent. The analysis for $d = 3$ is detailed in the proof of the following theorem.

Theorem 1 (Rep-0). *Under the* MLWE Representation Heuristic *the following holds. Let* $(A, \boldsymbol{t}) \in \mathbb{Z}_q^{N\times N} \times \mathbb{Z}_q^N$ *be an (M)LWE instance with* $q = \Omega(N)$ *and secret keys* $\boldsymbol{s}, \boldsymbol{e} \sim \mathcal{B}(3)^N$*, where* $N = nk$ *for MLWE. Then* LWE-Search *instantiated with* Rep-0 *representations finds* $\boldsymbol{s}$ *within time* $\mathcal{O}(2^{1.103N})$.

Proof. The correctness of LWE-Search follows by the discussion above. It remains to show that LWE-Search terminates in time $\mathcal{O}(2^{1.103N})$.

We use the run time formulas from Eqs. (8), (9) and (10). List sizes are computed using Eq. (7), search spaces using Eq. (4) and representations using Eq. (5). This results in

$$\begin{aligned}
\mathcal{T}^{(0)} &= \tfrac{(\mathcal{L}^{(1)})^2}{2^{N-r^{(1)}}} = 2^{(2(H(\frac{1}{128},\frac{28}{128},\frac{70}{128},\frac{28}{128},\frac{1}{128})-2\frac{15}{64}-2\frac{1}{64})-1+o(1))N} = \mathcal{O}(2^{1.090N}),\\
\mathcal{T}^{(1)} &= \tfrac{(\mathcal{L}^{(2)})^2}{q^{r^{(1)}-r^{(2)}}} = \tilde{\Theta}(2^{(2H(\frac{30}{256},\frac{196}{256},\frac{30}{256})-2\frac{15}{64}-2\frac{1}{64}-2\frac{28}{128})N}) = \mathcal{O}(2^{1.103N}),\\
\mathcal{T}^{(2)} &= \tfrac{(\mathcal{L}^{(3)})^2}{q^{r^{(2)}-r^{(3)}}} = \tilde{\Theta}(2^{(2\frac{H(\frac{30}{256},\frac{196}{256},\frac{30}{256})}{2}-2\frac{28}{128})N}) = \mathcal{O}(2^{0.583N}).
\end{aligned}$$

The total running time is the maximum over all levels, i.e., LWE-Search terminates in time $\mathcal{O}(2^{1.103N})$. □

6 More Representations

In this section, we enhance our representations to significantly reduce the LWE-Search run time from Theorem 1. Our first refined representation Rep-1 can be seen as an introduction to parametrization in the representation technique, where we add additional ± 1's to represent 0-entries not only as $0+0$, but also as $1+(-1)$ and $(-1)+1$.

We then optimize and parametrize to the full extent by adding in additional ± 2's and ± 3's in Rep-2 and Rep-3.

i	Representations of i						
0			$-1+1$	$\mathbf{0+0}$	$1-1$		
1				$\mathbf{0+1}$	$\mathbf{1+0}$		
2					$\mathbf{1+1}$		
3					$\mathbf{1+2}$	$\mathbf{2+1}$	

Fig. 6. Rep-1 representations of $i \in \{0,1,2,3\}$, magenta-colored representations are new.

6.1 Rep-1 Representations

Our Rep-1 representations are illustrated in Fig. 6. We introduce a parameter $\varepsilon^{(j)} \in [0,1], 1 \leq j < d$ for the additional number of ± 1's on level j. I.e., if we have $\omega_0^{(j-1)}N$ many entries 0 on level $j-1$, we represent $\varepsilon^{(j)}N$ many as $1+(-1)$, and $\varepsilon^{(j)}N$ many as $(-1)+1$. The remaining $(\omega_0^{(j-1)} - 2\varepsilon^{(j)})N$ 0-entries are still represented as $0+0$.

Recall that Rep-0 represented $\boldsymbol{s} = (3\ 3\ 2\ 1\ 1\ 0\ 0)$ as

$$\begin{aligned}\boldsymbol{s} &= (2\ 1\ 1\ 1\ 0\ 0\ 0) + (1\ 2\ 1\ 0\ 1\ 0\ 0) = (1\ 2\ 1\ 1\ 0\ 0\ 0) + (2\ 1\ 1\ 0\ 1\ 0\ 0) \\ &= (2\ 1\ 1\ 0\ 1\ 0\ 0) + (1\ 2\ 1\ 1\ 0\ 0\ 0) = (1\ 2\ 1\ 0\ 1\ 0\ 0) + (2\ 1\ 1\ 1\ 0\ 0\ 0).\end{aligned}$$

With Rep-1 and $\varepsilon = \frac{1}{7}$, we obtain the $8 = \binom{2}{1,1,0}\binom{2}{1}\binom{2}{1}$ representations

$$\begin{aligned}\boldsymbol{s} &= (2\ 1\ 1\ 1\ 0\ 1\ {-1}) + (1\ 2\ 1\ 0\ 1\ {-1}\ 1) = (1\ 2\ 1\ 1\ 0\ 1\ {-1}) + (2\ 1\ 1\ 0\ 1\ {-1}\ 1) \\ &= (2\ 1\ 1\ 1\ 0\ {-1}\ 1) + (1\ 2\ 1\ 0\ 1\ 1\ {-1}) = (1\ 2\ 1\ 1\ 0\ {-1}\ 1) + (2\ 1\ 1\ 0\ 1\ 1\ {-1}) \\ &= (2\ 1\ 1\ 0\ 1\ 1\ {-1}) + (1\ 2\ 1\ 1\ 0\ {-1}\ 1) = (1\ 2\ 1\ 0\ 1\ 1\ {-1}) + (2\ 1\ 1\ 1\ 0\ {-1}\ 1) \\ &= (2\ 1\ 1\ 0\ 1\ {-1}\ 1) + (1\ 2\ 1\ 1\ 0\ 1\ {-1}) = (1\ 2\ 1\ 0\ 1\ {-1}\ 1) + (2\ 1\ 1\ 1\ 0\ 1\ {-1}).\end{aligned}$$

Most formulas from the previous Section 5 remain unchanged. We only increase the number of 0-representations

$$\mathcal{R}_0^{(j)} = \binom{\omega_0^{(j-1)}N}{\varepsilon^{(j)}N, \varepsilon^{(j)}N, \cdot}$$

at the cost of slightly increased search spaces $\mathcal{S}^{(j)}$, reflected by different weights of the $0, \pm 1$-entries

$$\begin{aligned} \omega_0^{(j)} &= \tfrac{2\omega_0^{(j-1)} + 2\omega_1^{(j-1)} - 4\varepsilon^{(j)}}{2}, \\ \omega_1^{(j)} &= \tfrac{\omega_1^{(j-1)} + 2\omega_2^{(j-1)} + \omega_3^{(j-1)} + 2\varepsilon^{(j)}}{2}. \end{aligned}$$

d	j	$\varepsilon^{(j)}$	$\log \mathcal{T}^{(j)}$	$\log \mathcal{L}^{(j)}$
2	0	-	$.810N$	0
	1	$.036$	$\mathbf{.813}N$	$.810N$
	2	-	-	$\mathbf{.813}N$
3	0	-	$.718N$	0
	1	$.073$	$\mathbf{.787}N$	$\mathbf{.718}N$
	2	$.028$	$.436N$	$.436N$
	3	-	-	$.655N$

d	j	$\varepsilon^{(j)}$	$\log \mathcal{T}^{(j)}$	$\log \mathcal{L}^{(j)}$
4	0	-	$.718N$	0
	1	$.073$	$\mathbf{.787}N$	$\mathbf{.718}N$
	2	$.028$	$.495N$	$.436N$
	3		$.503N$	$.503N$
	4	-	-	$.433N$

Fig. 7. Rep-1 complexity exponents for $\mathcal{B}(3)^N$ using LWE-Search with depths $d = 2, 3, 4$ and optimized $\varepsilon^{(j)}$. Bold exponents indicate the dominating term.

Theorem 2 (Rep-1). *Under the* MLWE Representation Heuristic *the following holds. Let $(A, \boldsymbol{t}) \in \mathbb{Z}_q^{N \times N} \times \mathbb{Z}_q^N$ be an (M)LWE instance with $q = \Omega(N)$ and secret keys $\boldsymbol{s}, \boldsymbol{e} \sim \mathcal{B}(3)^N$, where $N = nk$ for MLWE. Then* LWE-Search *instantiated with* Rep-1 *representations finds $\boldsymbol{s}$ within time $\mathcal{O}(2^{0.787N})$.*

Proof. Analogous to the proof of Theorem 1, optimized parameters $\varepsilon^{(j)}$ can be found in Fig. 7. The desired complexity $\mathcal{O}(2^{0.787N})$ can be achieved using LWE-Search with tree depth $d = 3$. □

Notice that the —in comparison to Rep-0— only slightly more advanced Rep-1 representations lowered the exponent $1.103N$ from Theorem 1 already significantly down to $0.787N$. In Section 6.2 we study way more advanced representations that lower to even $0.388N$ (Rep-2) and $0.371N$ (Rep-3). The small improvement from Rep-3 over Rep-2 however indicates that we are converging. We conjecture that even more complex representations would only provide marginal improvements over Rep-3.

i	Representations of i						
0	$-3+3$	$-2+2$	$-1+1$	$\mathbf{0+0}$	$1-1$	$2-2$	$3-3$
1		$-2+3$	$-1+2$	$\mathbf{0+1}$	$\mathbf{1+0}$	$2-1$	$3-2$
2			$-1+3$	$0+2$	$\mathbf{1+1}$	$2+0$	$3-1$
3				$0+3$	$\mathbf{1+2}$	$\mathbf{2+1}$	$3+0$

Fig. 8. Representations of $i \in \{0,1,2,3\}$ under Rep-2 and Rep-3; magenta-colored entries are added with Rep-1; orange-colored entries are added with Rep-2; blue-colored entries are added with Rep-3.

6.2 Rep-2, Rep-3 Representations

Our Rep-2 and Rep-3 representations are illustrated in Fig. 8.

While our Rep-1 representations only allowed for two more representations of 0, our Rep-2 and eventually Rep-3 representations heavily increase the number of representations (e.g. 7 representations for 0) for all elements (e.g. still 4 representations of 3).

Parametrization. Note that we define Rep-k such that for every $0 \le \ell \le k$ the parameters of Rep-ℓ are contained in Rep-k.

To express the amount of additionally added $\pm1, \pm2, \pm3$, we define parameters $\varepsilon_{10}^{(j)}, \varepsilon_{20}^{(j)}, \varepsilon_{21}^{(j)}, \varepsilon_{22}^{(j)}, \varepsilon_{30}^{(j)}, \varepsilon_{31}^{(j)}, \varepsilon_{32}^{(j)}, \varepsilon_{33}^{(j)} \in [0,1]$, where $\varepsilon_{10}^{(j)} = \varepsilon^{(j)}$ from Rep-1 and $\varepsilon_{30}^{(j)}, \varepsilon_{31}^{(j)}, \varepsilon_{32}^{(j)}, \varepsilon_{33}^{(j)}$ are Rep-3 parameters only.

These parameters are to be understood as follows. Let $\boldsymbol{s}_i^{(j-1)}$ by a level-$(j-1)$ vector, represented by $\boldsymbol{s}_{2i-1}^{(j)}$, $\boldsymbol{s}_{2i}^{(j)}$. On level j, we replace $2\varepsilon_{ik}^{(j)}N$ of the Rep-0 representations of k with

- $\varepsilon_{ik}^{(j)}N$ representations $i + (k-i)$ and
- $\varepsilon_{ik}^{(j)}N$ representations $(k-i) + i$

in $\boldsymbol{s}_{2i-1}^{(j)}, \boldsymbol{s}_{2i}^{(j)}$.

To unify notation, we always choose i such that $i \ge k - i$ (e.g. we denote $\varepsilon_{22}^{(j)}$ instead of $\varepsilon_{02}^{(j)}$). Note that $\varepsilon_{ik}^{(j)}$ is a parameter for Rep-ℓ if and only if $i \le \ell$.

Example. Let us have a look at $\boldsymbol{s}^{(j-1)} = (2\ 2\ 2\ 2\ 2\ 2)$. Let $\varepsilon_{22}^{(j)} = \varepsilon_{32}^{(j)} = \frac{1}{6}$ and $\varepsilon_{ik}^{(j)} = 0$ for all other possible ik. By definition of $\varepsilon_{ik}^{(j)}$, we represent exactly one coefficient of $\boldsymbol{s}$ as $2 = 3-1,\quad 2 = 2+0,\quad 2 = 0+2,\quad 2 = -1+3$ each. The two remaining coefficients are represented as $2 = 1+1$, i.e., the Rep-0 representation.

This leads to $\binom{6}{1,1,1,1,2} = 360$ representations, where for comparison with Rep-1 we only had the unique representation $\boldsymbol{s} = (1\ 1\ 1\ 1\ 1\ 1) + (1\ 1\ 1\ 1\ 1\ 1)$.

For fixed optimization parameters $\varepsilon_{ik}^{(j)}$, we calculate the new formulas for $\mathcal{R}_i^{(j)}$ as

$$\begin{aligned}
\mathcal{R}_0^{(j)} &= \binom{\omega_0^{(j-1)}N}{\varepsilon_{10}^{(j)}N,\varepsilon_{10}^{(j)}N,\varepsilon_{20}^{(j)}N,\varepsilon_{20}^{(j)}N,\varepsilon_{30}^{(j)}N,\varepsilon_{30}^{(j)}N,\cdot},\\
\mathcal{R}_1^{(j)} &= \binom{\omega_1^{(j-1)}N}{\varepsilon_{21}^{(j)}N,\varepsilon_{21}^{(j)}N,\varepsilon_{31}^{(j)}N,\varepsilon_{31}^{(j)}N,\frac{\omega_1^{(j-1)}-2\varepsilon_{21}^{(j)}-2\varepsilon_{31}^{(j)}}{2}N,\frac{\omega_1^{(j-1)}-2\varepsilon_{21}^{(j)}-2\varepsilon_{31}^{(j)}}{2}N},\\
\mathcal{R}_2^{(j)} &= \binom{\omega_2^{(j-1)}N}{\varepsilon_{22}^{(j)}N,\varepsilon_{22}^{(j)}N,\varepsilon_{32}^{(j)}N,\varepsilon_{32}^{(j)}N,\cdot},\\
\mathcal{R}_3^{(j)} &= \binom{\omega_3^{(j-1)}N}{\varepsilon_{33}^{(j)}N,\varepsilon_{33}^{(j)}N,\frac{\omega_3^{(j-1)}-2\varepsilon_{33}^{(j)}}{2}N,\frac{\omega_3^{(j-1)}-2\varepsilon_{33}^{(j)}}{2}N},
\end{aligned}$$

and for $\omega_i^{(j)}$ as

$$\begin{aligned}
\omega_1^{(j)} &= \frac{\omega_1^{(j-1)}+2\omega_2^{(j-1)}+\omega_3^{(j-1)}+2\varepsilon_{10}^{(j)}-4\varepsilon_{22}^{(j)}-2\varepsilon_{31}^{(j)}-2\varepsilon_{32}^{(j)}-2\varepsilon_{33}^{(j)}}{2},\\
\omega_2^{(j)} &= \frac{\omega_3^{(j-1)}+2\varepsilon_{20}^{(j)}+2\varepsilon_{21}^{(j)}+2\varepsilon_{22}^{(j)}+2\varepsilon_{31}^{(j)}-2\varepsilon_{33}^{(j)}}{2},\\
\omega_3^{(j)} &= \frac{2\varepsilon_{30}^{(j)}+2\varepsilon_{31}^{(j)}+2\varepsilon_{32}^{(j)}+2\varepsilon_{33}^{(j)}}{2},\\
\omega_0^{(j)} &= \frac{2\omega_0^{(j-1)}+2\omega_1^{(j-1)}-4\varepsilon_{10}^{(j)}-4\varepsilon_{20}^{(j)}-4\varepsilon_{21}^{(j)}+4\varepsilon_{22}^{(j)}-4\varepsilon_{30}^{(j)}-4\varepsilon_{31}^{(j)}+4\varepsilon_{33}^{(j)}}{2}.
\end{aligned}$$

For a consistency check, verify that

$$\omega_0^{(j)}+2\omega_1^{(j)}+2\omega_2^{(j)}+2\omega_3^{(j)} = \omega_0^{(j-1)}+2\omega_1^{(j-1)}+2\omega_2^{(j-1)}+2\omega_3^{(j-1)}.$$

By induction this implies an equal sum of relative weights on all levels j. Especially for $j=0$, we obtain by definition

$$\omega_0^{(0)}+2\omega_1^{(0)}+2\omega_2^{(0)}+2\omega_3^{(0)} = \sum_{-3}^{3} p_i = 1.$$

Optimization of parameters leads to our following main result.

Theorem 3 (main result). *Under the* MLWE Representation Heuristic *the following holds. Let $(A,\boldsymbol{t}) \in \mathbb{Z}_q^{N\times N}\times\mathbb{Z}_q^N$ be an (M)LWE instance with $q=\Omega(N)$ and secret keys $\boldsymbol{s},\boldsymbol{e}\sim\mathcal{B}(3)^N$, where $N=nk$ for MLWE. Then* LWE-Search *finds $\boldsymbol{s}$ within time $\mathcal{O}(2^{0.388N})$* (for Rep-2)*, respectively $\mathcal{O}(2^{0.371N})$* (for Rep-3)*.*

Proof. Analogous to the proof of Theorem 1. Necessary optimization parameters can be found in Fig. 9. □

7 Other Distributions – Ternary, $\mathcal{B}(2)$, and Uniform

We apply our in previous sections developed representation technique to other distributions of cryptographic interest. Throughout this section, we only focus on the best results that we achieve with Rep-3 representations.

Rep.	j	$\varepsilon_{10}^{(j)}$	$\varepsilon_{20}^{(j)}$	$\varepsilon_{21}^{(j)}$	$\varepsilon_{22}^{(j)}$	$\varepsilon_{30}^{(j)}$	$\varepsilon_{31}^{(j)}$	$\varepsilon_{32}^{(j)}$	$\varepsilon_{33}^{(j)}$	$\log \mathcal{T}^{(j)}$	$\log \mathcal{L}^{(j)}$
Rep-2	0	-								$.316N$	0
	1	.075	.018	.032	.023					$.388N$	$.316N$
	2	.061	.004	.014	.019					$\mathbf{.388}N$	$.350N$
	3	.053	.001	.004	.007					$.388N$	$.366N$
	4	.028		.001	.002					$.388N$	$.382N$
	5	.007								$.388N$	$\mathbf{.388}N$
	6	-								-	$.382N$
Rep-3	0	-								$.297N$	0
	1	.072	.011	.024	.020		.003	.003	.001	$.371N$	$.297N$
	2	.078	.004	.016	.015			.001	.001	$.371N$	$.316N$
	3	.070	.001	.007	.007					$\mathbf{.371}N$	$.329N$
	4	.046		.002	.002					$.371N$	$.348N$
	5	.023								$.371N$	$\mathbf{.360}N$
	6	.003								$.356N$	$.356N$
	7	-								-	$.316N$

Fig. 9. Rep-2 and Rep-3 complexity exponents for $\mathcal{B}(3)^N$ using LWE-Search with optimized depths $d = 6$ (Rep-2) and $d = 7$ (Rep-3), and optimized $\varepsilon_{ik}^{(j)}$. Bold exponents indicate the dominating term.

First, we analyze ternary keys $\boldsymbol{s} \in \{-1, 0, 1\}^N$ of varying weight, as used e.g. in the cryptosystems NTRU [CDH+19,BCLvV19], BLISS [DDLL13], and GLP [GLP12]. For large weight ternary keys we slightly improve over [May21] by using larger depths. Moreover as opposed to [May21], using the techniques of Section 4 our results also hold for probabilistic distributions.

Second, we study the Centered Binomial distribution $\mathcal{B}(\eta)$ for $\eta = 1, 2, 3$. Notice that $\mathcal{B}(3)^{nk}$ is used in Kyber with $nk = 512$, whereas $\mathcal{B}(2)^{nk}$ is used in Kyber with larger security parameters $nk = 768$ and $nk = 1024$.

Eventually, we study uniform distributions in the range $[-\eta, \ldots, \eta]$ for $\eta = 1, 2, 3$. Naturally, uniformly distributed keys are widely used in cryptography, a prominent example being Dilithium with secret keys uniformly sampled from $\{-2, \ldots, 2\}^{nk}$, where $nk = 1024$ or $nk = 2048$.

7.1 Ternary Keys — Featuring NTRU, BLISS and GLP

We define a *weighted ternary* distribution as follows.

Definition 3. *Let $\omega \in [0, 1]$. We denote by $\mathcal{T}(\omega)$ the* weighted ternary *distribution*

$$(p_{-1}, p_0, p_1) = \left(\frac{\omega}{2}, 1 - \omega, \frac{\omega}{2}\right).$$

Some NTRU versions [HPS98] sample keys from the core set $\mathcal{C}(\mathcal{T}(\omega))$, see Definition 1, i.e., with fixed expected weights. Other NTRU versions [CDH+19] sample directly from $\mathcal{T}(\omega)^N$. Our new techniques also apply to the later probabilistic versions.

Our ternary key results are summarized in Fig. 10. More detailed optimization parameters are provided in Fig. 13, Appendix A. In particular, Fig. 13 shows that for ternary keys Rep-2 is sufficient, and Rep-3 provides no further benefit.

ω	0.3			0.375			0.441		
	d	$\log \mathcal{T}$	$\log \mathcal{M}$	d	$\log \mathcal{T}$	$\log \mathcal{M}$	d	$\log \mathcal{T}$	$\log \mathcal{M}$
[May21]	4	$.295N$	$.294N$	4	$.318N$	$.316N$	4	$.334N$	$.333N$
Ours	4	**.295N**	$.294N$	5	**.315N**	$.312N$	6	**.326N**	$.320N$
ω	0.5			0.62			0.667		
	d	$\log \mathcal{T}$	$\log \mathcal{M}$	d	$\log \mathcal{T}$	$\log \mathcal{M}$	d	$\log \mathcal{T}$	$\log \mathcal{M}$
[May21]	4	$.348N$	$.346N$	4	$.371N$	$.371N$	4	$.379N$	$.379N$
Ours	5	**.337N**	$.337N$	6	**.342N**	$.336N$	6	**.345N**	$.338N$

Fig. 10. Ternary Key Results for different weights ω, and comparison with [May21].

Whereas [May21] analyzed only depths $d \leq 4$, for which we confirm the results, we obtain for increasing weights $\omega \geq 0.375$ slightly better run time exponents in depths 5 and 6. This is in particular interesting for weight $\omega = \frac{1}{2}$, which is the distribution $\mathcal{B}(1)^N$, and for weight $\omega = \frac{2}{3}$, the uniform distribution.

7.2 $\mathcal{B}(2)$ and $\mathcal{B}(3)$ — Featuring Kyber-512 and Kyber-768,1024

From Eq. (2) we obtain the Centered Binomial distributions

$$\begin{aligned}
\mathcal{B}(1) &= (p_{-1}, p_0, p_1) = \left(\frac{1}{4}, \frac{2}{4}, \frac{1}{4}\right), \\
\mathcal{B}(2) &= (p_{-2}, \ldots, p_2) = \left(\frac{1}{16}, \frac{4}{16}, \frac{6}{16}, \frac{4}{16}, \frac{1}{16}\right), \\
\mathcal{B}(3) &= (p_{-3}, \ldots, p_3) = \left(\frac{1}{64}, \frac{6}{64}, \frac{15}{64}, \frac{20}{64}, \frac{15}{64}, \frac{6}{64}, \frac{1}{64}\right).
\end{aligned}$$

We observe that $\mathcal{B}(1) = \mathcal{T}(\frac{1}{2})$. Our results for $\mathcal{B}(\eta)$, $\eta = 1, 2, 3$ are illustrated in Fig. 11. Our full optimization parameters can be found in Figure 14, Appendix A.

η	d	$\log \mathcal{T}$	$\log \mathcal{M}$	$\log \mathcal{S}$	$\log_{\mathcal{S}} \mathcal{T}$
1	5	**.337N**	$.337N$	$1.500N$	**.225**
2	7	**.357N**	$.357N$	$2.031N$	**.176**
3	7	**.371N**	$.360N$	$2.334N$	**.159**

Fig. 11. Results for Centered Binomial distributions $\mathcal{B}(\eta)$ for $\eta = 1, 2, 3$.

We find it remarkable that despite a significant growth in entropy from $\mathcal{B}(1)$ with exponent $1.5N$ to $\mathcal{B}(3)$ with exponent $2.3N$, the actual key security against Search-LWE with Rep-3 increases only slightly with exponent $0.034N$. Obviously, for our algorithm in the case of Centered Binomial distributions the number of representations grows much faster than the search space sizes.

As a consequence, whereas we obtain for $\mathcal{B}(1)^N$ roughly an $\mathcal{S}^{\frac{1}{4}}$ algorithm, for $\mathcal{B}(3)^N$ we obtain an $\mathcal{S}^{\frac{1}{6}}$ algorithm, i.e., we achieve the 6th root of the search space.

7.3 Uniform Distribution — Featuring Dilithium-1024,2048

We define the *uniform distribution* as follows.

Definition 4. *Let $\eta \in \mathbb{N}$. We denote by $\mathcal{U}(\eta)$ the* uniform distribution *having for all $i = -\eta, \dots, \eta$ constant*

$$p_i = \frac{1}{2\eta + 1}.$$

Notice that $\mathcal{U}(1) = \mathcal{T}(\frac{2}{3})$. Two Dilithium parameter sets use $\mathcal{U}(2)^{nk}$ for $nk = 1024$ and $nk = 2048$. Our $\mathcal{U}(\eta)$ results for $\eta = 1, 2, 3$ are provided in Fig. 12. Our optimization parameters can be found in Fig. 14, Appendix A.

η	d	$\log \mathcal{T}$	$\log \mathcal{M}$	$\log \mathcal{S}$	$\log_{\mathcal{S}} \mathcal{T}$
1	6	**.345N**	$.338N$	$1.585N$	**.218**
2	8	**.378N**	$.372N$	$2.322N$	**.163**
3	6	**.493N**	$.481N$	$2.808N$	**.176**

Fig. 12. Our results for uniform distributions $\mathcal{U}(\eta)$, $\eta = 1, 2, 3$.

The complexity exponent $0.378N$ for $\mathcal{U}(2)$ is of a similar size than $0.371N$ for $\mathcal{B}(3)$. But since Kyber uses significantly smaller key lengths $N = nk$ in comparison to Dilithium, our Search-LWE algorithm can be considered much more effective for Kyber keys.

References

BBSS20. Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. Improved classical and quantum algorithms for subset-sum. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 633–666. Springer, Heidelberg, December 2020.

BCJ11. Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 364–385. Springer, Heidelberg, May 2011.

BCLvV19. Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime: Round 2 Specification. 2019.

BDK^{+}18. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

BGV14. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

BJMM12. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, Heidelberg, April 2012.

BV14. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871, 2014.

CDH^{+}19. Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU Algorithm Specifications And Supporting Documentation. *Brown University and Onboard security company, Wilmington USA*, 2019.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013.

DKSRV18. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018.

DRX17. Srinivas Devadas, Ling Ren, and Hanshen Xiao. On iterative collision search for lpn and subset sum. In *Theory of Cryptography Conference*, pages 729–746. Springer, 2017.

EMVW22. Andre Esser, Alexander May, Javier Verbel, and Weiqiang Wen. Partial key exposure attacks on bike, rainbow and ntru. In *CRYPTO 2022*, Lecture Notes in Computer Science. Springer, 2022.

EMZ22. Andre Esser, Alexander May, and Floyd Zweydinger. Mceliece needs a break - solving mceliece-1284 and quasi-cyclic-2918 with modern ISD. In *EUROCRYPT (3)*, volume 13277 of *Lecture Notes in Computer Science*, pages 433–457. Springer, 2022.

GLP12. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, September 2012.

HJ10. Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 235–256. Springer, Heidelberg, May / June 2010.

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 267–288. Springer, Heidelberg, June 1998.

May21. Alexander May. How to meet ternary LWE keys. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 701–731, Virtual Event, August 2021. Springer, Heidelberg.

MMT11. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, Heidelberg, December 2011.

MU17. Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis.* Cambridge university press, 2017.

Reg05. Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.

SO97. Joseph H Silverman and A Odlyzko. A meet-in-the-middle attack on an ntru private key. *preprint*, 1997.

Wat87. William C Waterhouse. How often do determinants over finite fields vanish? *Discrete Mathematics*, 65(1):103–104, 1987.

A Full Parameter Sets: Ternary, Binomial, and Uniform

$\mathcal{D}$	j	$\varepsilon_{10}^{(j)}$	$\varepsilon_{20}^{(j)}$	$\varepsilon_{21}^{(j)}$	$\varepsilon_{22}^{(j)}$	$\varepsilon_{30}^{(j)}$	$\varepsilon_{31}^{(j)}$	$\varepsilon_{32}^{(j)}$	$\varepsilon_{33}^{(j)}$	$\log \mathcal{T}^{(j)}$	$\log \mathcal{L}^{(j)}$
$\mathcal{T}(0.3)$	0					-				$.239N$	0
	1	.050		.001						$.295N$	$.239N$
	2	.026								$\mathbf{.295}N$	$.283N$
	3	.006								$.294N$	$\mathbf{.294}N$
	4					-				-	$.288N$
$\mathcal{T}(0.375)$	0					-				$.251N$	0
	1	.052	.001	.003						$.313N$	$.251N$
	2	.031		.001	.001					$.315N$	$.299N$
	3	.012								$\mathbf{.315}N$	$\mathbf{.312}N$
	4	.001								$.275N$	$.275N$
	5					-				-	$.216N$
$\mathcal{T}(0.441)$	0					-				$.254N$	0
	1	.056	.001	.005						$.325N$	$.254N$
	2	.042		.001	.001					$\mathbf{.326}N$	$.298N$
	3	.019								$.326N$	$\mathbf{.320}N$
	4	.002								$.316N$	$.313N$
	5									$.220N$	$.220N$
	6					-				-	$.155N$
$\mathcal{T}(0.5)$	0					-				$.268N$	0
	1	.049	.001	.009						$.337N$	$.268N$
	2	.040	.001	.002	.002					$\mathbf{.337}N$	$.311N$
	3	.017		.001	.001					$.337N$	$\mathbf{.337}N$
	4	.002								$.333N$	$.333N$
	5					-				-	$.273N$
$\mathcal{T}(0.62)$	0					-				$.250N$	0
	1	.063	.001	.011						$.341N$	$.250N$
	2	.061	.001	.003	.002					$.342N$	$.290N$
	3	.036		.001	.001					$\mathbf{.342}N$	$.324N$
	4	.015								$.342N$	$\mathbf{.336}N$
	5	.001								$.313N$	$.313N$
	6					-				-	$.249N$
$\mathcal{T}(0.667)$	0					-				$.258N$	0
	1	.056	.001	.013						$\mathbf{.345}N$	$.258N$
	2	.060	.001	.004	.002					$.345N$	$.294N$
	3	.038		.001	.001					$.345N$	$.325N$
	4	.016								$.345N$	$\mathbf{.338}N$
	5	.001								$.321N$	$.321N$
	6					-				-	$.257N$

Fig. 13. Ternary distribution full parameter sets with Rep-2 (Rep-3 params all 0).

$\mathcal{P}$	j	$\varepsilon_{10}^{(j)}$	$\varepsilon_{20}^{(j)}$	$\varepsilon_{21}^{(j)}$	$\varepsilon_{22}^{(j)}$	$\varepsilon_{30}^{(j)}$	$\varepsilon_{31}^{(j)}$	$\varepsilon_{32}^{(j)}$	$\varepsilon_{33}^{(j)}$	$\log \mathcal{T}^{(j)}$	$\log \mathcal{L}^{(j)}$
$\mathcal{B}(1)$	0				-					$.268N$	0
	1	.049	.001	.009						$.337N$	$.268N$
	2	.040	.001	.002	.002					**.337**N	$.311N$
	3	.017		.001	.001					$.337N$	**.337**N
	4	.002								$.333N$	$.333N$
	5				-					-	$.273N$
$\mathcal{B}(2)$	0				-					$.264N$	0
	1	.076	.007	.022	.014			.001		$.357N$	$.264N$
	2	.084	.003	.010	.009					$.357N$	$.289N$
	3	.061	.001	.004	.004					**.357**N	$.315N$
	4	.038		.001	.001					$.357N$	$.340N$
	5	.015								$.357N$	**.351**N
	6	.002								$.316N$	$.316N$
	7				-					-	$.265N$
$\mathcal{B}(3)$	0				-					$.297N$	0
	1	.072	.011	.024	.020		.003	.003	.001	$.371N$	$.297N$
	2	.078	.004	.016	.015			.001	.001	$.371N$	$.316N$
	3	.070	.001	.007	.007					**.371**N	$.329N$
	4	.046		.002	.002					$.371N$	$.348N$
	5	.023								$.371N$	**.360**N
	6	.003								$.356N$	$.356N$
	7				-					-	$.316N$
$\mathcal{U}(1)$	0				-					$.258N$	0
	1	.056	.001	.013						**.345**N	$.258N$
	2	.060	.001	.004	.002					$.345N$	$.294N$
	3	.038		.001	.001					$.345N$	$.325N$
	4	.016								$.345N$	**.338**N
	5	.001								$.321N$	$.321N$
	6				-					-	$.257N$
$\mathcal{U}(2)$	0				-					$.308N$	0
	1	.046	.014	.029	.040	.001	.005	.010		$.377N$	$.308N$
	2	.072	.007	.024	.020		.001	.002	.001	**.378**N	$.322N$
	3	.071	.003	.014	.012					$.378N$	$.331N$
	4	.051	.001	.005	.006					$.378N$	$.351N$
	5	.031		.001	.002					$.375N$	$.370N$
	6	.010								$.378N$	**.372**N
	7	.001								$.307N$	$.307N$
	8				-					-	$.244N$
$\mathcal{U}(3)$	0				-					$.451N$	0
	1	.030	.018	.025	.022	.006	.011	.013	.028	**.493**N	$.451N$
	2	.056	.007	.025	.027		.001	.001	.002	$.492N$	$.475N$
	3	.048	.001	.007	.012					$.493N$	**.481**N
	4	.023		.001	.001					$.492N$	$.476N$
	5	.004								$.449N$	$.449N$
	6				-					-	$.423N$

Fig. 14. Centered Binomial $\mathcal{B}(\eta)$ and uniform $\mathcal{U}(\eta)$ full parameter sets with Rep-3.