How to Meet Ternary LWE Keys

Alexander May*

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
alex.may@rub.de

Abstract. The LWE problem with its ring variants is today the most prominent candidate for building efficient public key cryptosystems resistant to quantum computers. NTRU-type cryptosystems use an LWE-type variant with small max-norm secrets, usually with ternary coefficients from the set $\{-1, 0, 1\}$. The presumably best attack on these schemes is a hybrid attack that combines lattice reduction techniques with Odlyzko's Meet-in-the-Middle approach. Odlyzko's algorithm is a classical combinatorial attack that for key space size $\mathcal{S}$ runs in time $\mathcal{S}^{0.5}$. We substantially improve on this Meet-in-the-Middle approach, using the representation technique developed for subset sum algorithms. Asymptotically, our heuristic Meet-in-the-Middle attack runs in time roughly $\mathcal{S}^{0.25}$, which also beats the $\mathcal{S}^{\frac{1}{3}}$ complexity of the best known quantum algorithm.
For the round-3 NIST post-quantum encryptions NTRU-Encrypt and NTRU-Prime we obtain non-asymptotic instantiations of our attack with complexity roughly $\mathcal{S}^{0.35}$. As opposed to other combinatorial attacks, our attack benefits from larger LWE field sizes q, as they are often used in modern lattice-based signatures. For example, for BLISS signatures we obtain non-asymptotic combinatorial attacks in between $\mathcal{S}^{0.31}$ and $\mathcal{S}^{0.35}$, for GLP signatures in $\mathcal{S}^{0.3}$.
Our attacks do not invalidate the security claims of the aforementioned schemes. However, they establish improved combinatorial upper bounds for their security. We leave it is an open question whether our new Meet-in-the-Middle attack in combination with lattice reduction can be used to speed up the hybrid attack.

Keywords: Meet in the Middle, Representation Technique, NTRU/BLISS/GLP

1 Introduction

In the LWE problem [Reg03], we are given a (random) matrix $A \in \mathbb{Z}_q^{m \times n}$ and a target vector $\mathbf{b} \in \mathbb{Z}_q^m$ with the promise that there exist small $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ such that $A\mathbf{s} = \mathbf{b} + \mathbf{e} \bmod q$. In this paper, we consider only the case $m = n$, i.e. m equals the LWE dimension n, which is the standard setting in modern lattice-based encryption and signature schemes. In the Ring-LWE case [LPR10], one uses the algebraic structure of rings to compactly represent A. All results in

* Funded by DFG under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

this work also apply to the ring setting, and in fact all of our applications are in the ring setting, but for the sake of LWE generality we do not exploit any ring properties in our analysis.

The LWE problem, and especially its ring variants, are an extremely versatile source for the construction of cryptographic primitives [Reg03, Gen09, GPV08, BDK+17]. Due to its beautiful connection to worst-case lattice problems, one usually calls the resulting schemes lattice-based, although the LWE problem is per se more a combinatorial problem that asks to find a small solution $\mathbf{s}$ to some erroneous—by error $\mathbf{e}$—linear system of equations.

While asymptotically the worst-case connection of LWE to hard lattice problems guarantees security for sufficiently large dimension n, it is still a tricky business to instantiate LWE parameters (n, q) and the error distribution for $\mathbf{e}$ that lead to practical cryptographic schemes, which yet provide a concrete level of security. LWE security proofs usually utilize a discrete Gaussian distribution for $\mathbf{e}$ (and often also for $\mathbf{s}$). However, certain schemes prove security with Gaussians, and then in turn define especially efficient parameters sets where $\mathbf{s}, \mathbf{e}$ are binary or ternary vectors, such as BLISS [DDLL13] or GLP [GLP12]. The NTRU encryption scheme took the other way round, starting with an efficient scheme [HPS98], whose security was later proved for less efficient variants [SS11]. LWE with $\mathbf{s}, \mathbf{e} \in \{0,1\}^n$ is also known as binary-LWE, and its security has been studied recently [BLP+13, MP13, BG14, BGPW16]. In this paper, we focus on ternary vectors $\mathbf{s}, \mathbf{e} \in \{-1, 0, 1\}^n$, as they are frequently used in modern NTRU-type schemes. Limiting the distribution to vectors of small max-norm 1 (or any small constant) has several advantages for cryptographic schemes.

Efficiency and simplicity. The implementation of discrete Gaussian sampling is quite involved, and costs a reasonable amount of random bits [DN12]. Proper randomness is in practice often a scarce source. Instead, sampling ternary vectors is comparably simple and much less error-prone to implement. Moreover, the resulting keys are especially compact.

Correctness of decryption. The use of ternary vectors allows to define encryption schemes that always decrypt correctly. Among the remaining lattice-based encryption schemes in NIST competition's third round there are only two schemes, both NTRU variants with ternary secrets, that do not have decryption failure. This property is particularly important, since even smallish decryption failures give rise to powerful attacks [HNP+03, DRV20].

As a consequence, designers of recent cryptosystems often replace Gaussian error distributions by small max-norm secrets (still guarding against lattice attacks), with the argument that despite of 25 years NTRU-type cryptanalysis there is no combinatorial algorithm better than Odlyzko's Meet-in-the-Middle (MitM) attack—mentioned in the original 1996 NTRU paper [HPS98]— that can directly take advantage of small max-norm keys, such as ternary keys. Our work invalidates this argument. Our new MitM attack for ternary secrets heavily uses the small max-norm property and significantly improves over Odlyzko's algorithm.

The development of more involved combinatorial LWE MitM search algorithms usually directly influences the parameter choice of NTRU-type cryptosystems, since the presumably best known attack on these schemes—Howgrave-Graham's *Hybrid* approach [How07]—is up to now a combination of Odlyzko's MitM attack on a projected sub-key and lattice reduction on a projected sub-lattice. The Hybrid attack balances the cost of MitM and lattice reduction by properly adjusting the dimensions of these projections.

On the one hand, there is a long line of research that decreased the complexity of lattice reduction [NV08, Laa15, BDGL16, HKL18], using involved techniques. On the other hand, for the combinatorial part there is still only the comparatively simple and costly Odlyzko MitM with square root complexity of the search space. The best quantum attack is a quantum version of Odlyzko's MitM that achieves third root complexity of the search space [WMM13, dBDJW18]. This complexity imbalance currently puts large emphasis on lattice reduction in the Hybrid attack.

On the theoretical side, we cannot expect to fully break LWE with ternary keys. In 2013, Brakerski, Langlois, Peikert, Regev, Stehlé [BLP+13] and Micciancio, Peikert [MP13] provide reductions showing that LWE with binary or ternary secrets is indeed still hard. However, these reductions require to increase the LWE dimension from n to approximately $n \log q$. Our new MitM approach gives cryptanalytic indication that small norm secrets are indeed significantly easier to recover.

We would like to point out that other known algebraic/combinatorial attacks [GJS15, KF15, ACF+14] do not work in our scenario due to the limited number of samples $m = n$, or because they require superpolynomial q.

Our Results. We give the first significant progress for MitM attacks on ternary LWE keys since Odlyzko's attack from 1996 [HPS98, HGSW03]. Let the LWE secret key $\mathbf{s} \in \mathbb{Z}_q^n$ be taken from an exponential (in n) search space size $\mathcal{S}$. Then Odlyzko's attack recovers $\mathbf{s}$ in time $\mathcal{S}^{0.5}$. E.g. random ternary secrets $\mathbf{s} \in \{-1, 0, 1\}^n$ have $\mathcal{S} = 3^n$ and Odlyzko's attack runs in time $3^{n/2} = 2^{\log_2(3)n/2}$. A quantum version of Odlyzko's attack runs in time $\mathcal{S}^{\frac{1}{3}}$ [WMM13, dBDJW18].

Other attacks [MS01] use the structure of the public LWE key A in the ring setting, but these speedups are polynomial in n and thus lead to run time $\mathcal{S}^{0.5-o(1)}$. We show that the exponent 0.5 can be significantly reduced for small max-norm keys $\mathbf{s}$ and $\mathbf{e}$ in the supposedly hard LWE case where q is polynomial in n. Notice that for larger q efficient attacks are known [ABD16].

In a nutshell, our algorithm guesses r coordinates of $\mathbf{e} \in \mathbb{Z}_q$, where $r = \mathcal{O}(\frac{n}{\log q}) = \mathcal{O}(\frac{n}{\log n})$ is slightly sub-linear in the LWE dimension n. This can be done in slightly subexponential time $2^{\mathcal{O}(\frac{n}{\log q})}$. We then solve a vectorial subset sum problem $A\mathbf{s} = \mathbf{b} + \mathbf{e}$ on the known r coordinates. This is done by generalizing search tree-based subset sum algorithms [HJ10, BCJ11, BBSS20] to our setting, where the columns of A define the subset sum instance with target vector $\mathbf{b}$, and the max-norm of $\mathbf{s}$ defines which linear combinations are allowed.

In the original subset sum setting, we are allowed to take 0/1-combinations, whereas for ternary $\mathbf{s}$ we have to take $0/\pm 1$ combinations. Intuitely, although

a larger digit set for the linear combinations increases the combinatorial complexity, it might at the time weaken subset sum instances, since it introduces symmetries. The latter effect can be seen in our results. Our subset sum instances on r coordinates can be solved more efficiently than the original 0/1-instances from [HJ10, BCJ11, BBSS20], despite the fact that—as opposed to the subset sum setting—we also have to take the complexity of guessing r coordinates of $\mathbf{e}$ into account.

Eventually, our subset sum-type algorithm outputs two (sorted) lists of size $\mathcal{S}^c$, $c < 0.5$ from which the secret $\mathbf{s}$ can be recovered in time linear in the list size by using Odlyzko's algorithm on the remaining $n - r$ coordinates.

We give different instantiations of our algorithm using different representations of $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ as a sum of two vectors $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_q^n$. The more representations we have of $\mathbf{s}$, the larger is the number r of guessed coordinates, and the smaller gets c. Intuitely, choosing larger r in this tradeoff pays off, since key guessing is slightly subexponential, whereas $\mathcal{S}$ is fully exponentially in n.

This bias in the tradeoff can be seen in the instantiations of our MitM. We choose three different instantiations—called Rep-0, Rep-1, Rep-2—with an increasing number of representations. More representations yield smaller list sizes and therefore complexities $\mathcal{S}^c$ with smaller c. For Rep-0, Rep-1 and Rep-2 we optimize the search trees in our subset sum-type list construction to find a minimal c, while using relatively small search tree depths 3 or 4.

We also show that, despite the key guessing, our MitM leads to significantly improved non-asymptotic combinatorial attacks. As running example, we consider the current NIST round-3 candidate encryption schemes NTRU-Encrypt [CDH+19] and NTRU-Prime [BCLvV17, BCLvV], which both have ternary secrets $\mathbf{s}, \mathbf{e}$. As examples for signature schemes, we address the efficient GLP version [GLP12] of Lyubashevksy's scheme [Lyu12] and BLISS [DDLL13], both using ternary secrets. For BLISS we also analyze an instance with secret $\mathbf{s} \in \{-2, -1, 0, 1, 2\}^n$.

For these schemes, we illustrate the effects of our new MitM LWE key search, instantiated with Rep-0 to Rep-2. We compute the list sizes and therefore complexities *non-asymptotically* exact, but for ease of exposition throughout the paper we ignore polynomial factors that stem from list operations like sorting or hashing. These polynomial factors usually increase the run time, but tricks like using rotations in ring-LWE might also decrease our complexities by other polynomial factors.

Our non-asymptotic analysis nicely illustrates the tradeoff between guessing r coordinates and decreasing the complexity of the subset sum-type list constructions. Let us give some numerical example for NTRU-Encrypt. The largest of the three proposed NTRU-Encrypt parameter sets is $(n, q) = (821, 4096)$, where the ternary secret has exactly 255 1-coordinates and 255 (-1)-coordinates. This gives search space size $\mathcal{S} = \binom{821}{255,255,311} \approx 2^{1286}$. Thus, we obtain $\mathcal{S}^{0.5} \approx 2^{643}$, or 643 bit, complexity for Odlyzko's attack.

Using Rep-0, we decrease to 546 bit, where we require 480 bit for the subset sum-type list construction and 66 bit for key guessing.

Using Rep-1, we further decrease to 452 bit, using 339 bit for list construction and 113 bit for guessing.

Eventually using Rep-2, we even further decrease to 438 bit, using only 320 bit for list construction but 118 bit for guessing.

In total, our new purely combinatorial MitM on parameter set $(821, 4096)$ has complexity $2^{438} \approx \mathcal{S}^{0.34}$—with list construction time $2^{322} \approx S^{0.25}$—instead of Odlyzko's $2^{643} \approx \mathcal{S}^{\frac{1}{2}}$. For all six officially proposed instances of NTRU-Encrypt and NTRU-Prime we obtain complexity roughly $\mathcal{S}^{0.35}$. For BLISS signatures we obtain complexities between $\mathcal{S}^{0.31}$ and $\mathcal{S}^{0.35}$, and for GLP signatures even complexity $\mathcal{S}^{0.3}$. The reason for these improved complexities is that lattice-based signatures (as opposed to encryption schemes) typically use larger q, from which our LWE MitM key search algorithm benefits.

The memory requirement of our attack is roughly $\mathcal{S}^{0.25}$, asymptotically and non-asymptotically, as compared to $\mathcal{S}^{0.5}$ for Odlyzko's attack. However, we show that our techniques also lead to time-memory tradeoffs that improve over the best known time-memory tradeoff for Odlyzko's attack by van Vredendaal [vV16].

Although our MitM performs much better than the best known combinatorial attack, our results do not invalidate the security claims of current LWE-type systems like NTRU (at least for modern parameter settings), BLISS and GLP. The situation would change if our MitM attack could be used to speed up the lattice hybrid attack by the amount that we improve over Odlyzko's MitM. However, our attempts to speedup the lattice hybrid so far failed.

Nevertheless, our results show that the security of many NTRU instantions, such as many low weight instances in the NTRU standards [ntr08], is significantly lower than originally expected. E.g. the NTRU `ees659ep1` parameter set $(n, q) = (651, 2048)$ is estimated in [ntr08] (by a deliberate conservative underestimation) to achieve 251 bit security against lattice reduction and at least 138 bit security against the Hybrid attack. Plugging into our analysis, we achieve a purely classical MitM using complexity only 2^{138} for list construction and additionally 2^{25} for key guessing.

Moreover, we consider it important to establish new solid combinatorial upper bounds on the security of small key LWE-type schemes. Up to now, the LWE parameter selection is solely based on lattice reduction estimates which involve not only a good amount of heuristics like e.g. the Gaussian Heuristic and the Geometric Series Assumption, but also a variety of run time formulas for estimating BKZ lattice reduction. This is a shaky basis to solely rely on in a post-quantum standardization process—although our impression is that (for good reasons) the security of current LWE type schemes is rather underestimated by current lattice complexity estimates.

In contrast, our MitM uses only very mild heuristic assumptions from subset sum-type list constructions that have already been thoroughly experimentally verified in other settings [BCJ11, BJMM12], and our run time analysis can be considered very accurate.

Organization of our paper. We recall known MitM attacks from Odlyzko (Section 3) and Howgrave-Graham (Section 4, not to be confused with Hybrid). These two MitMs are sometimes mixed in the literature, although they are algorithmically different. We show that Howgrave-Graham's method is strictly inferior, but it can be used as the basis for our improved LWE MitM key search algorithm Meet-LWE in Section 5. We instantiate Meet-LWE with different representations Rep-0 (Section 6), Rep-1 (Section 7) and Rep-2 (Section 8). Some improved time-memory tradeoffs are given in Section 9. In Section 10, we discuss why Meet-LWE fails to be directly applicable to Howgrave-Graham's lattice Hybrid attack, and discuss possible work-arounds.

2 Preliminaries

2.1 LWE-Key and Max-Norm Key Search

Definition 1 (Small Max-Norm LWE Secret Key). *An LWE public key is a tuple* $(A, \mathbf{b}) \in \mathbb{Z}_q^{m \times n}$ *satisfying the identity*

$$A\mathbf{s} = \mathbf{b} + \mathbf{e} \bmod q$$

for some secret vectors $\mathbf{s} \in \mathbb{Z}_q^n$ *and* $\mathbf{e} \in \mathbb{Z}_q^m$*. We call* $\mathbf{s}, \mathbf{e}$ small max-norm LWE secret keys *if* $\|\mathbf{s}\|_\infty = \|\mathbf{e}\|_\infty = O(1)$*. We call max-norm 1 vectors* ternary*. We denote the set of* n*-dimensional ternary vectors by* $\mathcal{T}^n = \mathbb{Z}_q^n \cap \{-1, 0, 1\}^n$*.*

All small max-norm LWE keys allow for simple checking of key guess correctness. Namely, with overwhelming probability (over the randomness of A) there is a unique $\mathbf{s}$ such that $A\mathbf{s} - \mathbf{b}$ has small max-norm. Throughout the paper, we use square $A \in \mathbb{Z}_q^{n \times n}$, and assume that A's entries are uniformly at random from $\mathbb{Z}_q$. For Ring-LWE type cryptosystems the entries are in fact dependent, but we do not make use of any ring structure.

If not specified otherwise we use **ternary LWE keys**, since almost all prominent running examples of NTRU-type cryptosystems in this paper— NTRU-Encrypt [CDH+19], NTRU-Prime [BCLvV], BLISS [DDLL13] and GPL [GLP12]— use coefficients in $\{-1, 0, 1\}$. In principle, our technique applies to any max-norm and we analyze a max-norm 2 BLISS example in Section 8.3, but we consider our algorithms most effective for very small max-norms like 1 and 2. Moreover, our technique is currently harder to analyze with increasing max-norm.

Most NTRU-type systems such as the above examples do not only use *small max-norm keys*, but they also restrict the number of non-zero entries.

Definition 2 (Weight). *Let* $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$*. Then the* weight w *of* $\mathbf{s}$ *is defined as its Hamming weight* $w := \sum_{s_i \neq 0} 1$*. We often specify the weight* relative to n *as* $w = \omega n$ *for some* $0 \leq \omega \leq 1$*. We denote the set of* n*-dimensional ternary*

weight-w vectors that split their weight evenly in $w/2$ (-1)-entries and 1-entries by

$$\mathcal{T}^n(w/2) = \{\mathbf{s} \in \mathcal{T}^n \mid \mathbf{s} \text{ has } w/2 \ (\pm 1)\text{-entries each.}\}$$

For notational convenience we omit any roundings. Asymptotically, roundings can be neglected. For real-world security estimates we round appropriately.

NTRU's security analysis so far yields an optimal relative weight in the range $\omega \in [\frac{1}{3}, \frac{2}{3}]$ [CDH+19, BCLvV]. A prominent choice is $\omega = \frac{3}{8}$, which is used in one (out of three) suggested NTRU-Encrypt parameter sets with dimension $n = 677$ and in two (out of three) suggested NTRU Prime parameter sets with $n = 761$ and $n = 857$. Througout the paper when we speak of NTRU-Encrypt instances, we address parameter sets of the HPS variant, but in principle our attack also applies to HRSS, see [CDH+19]. Similar, with NTRU-Prime we address the Streamlined NTRU variant, but our attack also works for NTRU LPRime, see [BCLvV].

2.2 Search Space, Entropy and Representations

Obviously, there are 3^n ternary vectors $\mathbf{s} \in \mathcal{T}^n$. When using asymptotics (and only in this case!), we frequently approximate sets of vectors with a fixed number of certain coefficients by the following well-known Shannon entropy formula that stems from Stirling's approximation [MU17].

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

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

Notice that Lemma 1 approximates the number of ternary vectors having exactly $n/3$ coefficients for each of $-1, 0, 1$ as

$$2^{H(\frac{1}{3},\frac{1}{3},\frac{1}{3})n} = 2^{3\cdot\frac{1}{3}\log_2(3)n} = 3^n.$$

The straight-forward proof of Lemma 1 via Stirling approximation shows that the approximation suppresses a $\frac{1}{\sqrt{n}}$-factor, which implies that a $\frac{1}{\sqrt{n}}$-fraction of all ternary vectors splits its coefficients evenly among the entries $-1, 0, 1$. In our notation, $\mathcal{T}^n(n/3)$ is up to a (small) polynomial factor as large as $\mathcal{T}^n$.

For ease of notation, in the following we always assume that we search for ternary keys with a predefined portion of entries. This is true e.g. for NTRU-Encrypt, GPL and BLISS. However, we can easily generalize our attack to systems with arbitrary portions of entries, such as NTRU-Prime, by simply guessing each portion. Since we only consider constant max-norm, such a guessing costs only an $n^{O(1)}$-factor.

2.3 Asymptotics and Real-World Applications

Although our NTRU-applications in mind require real-world security estimates, we usually start our analysis with asymptotic notion, before giving non-asymptotics for concrete instances. Asymptotics often allows for much cleaner results, clearly indicating the dependence on the involved LWE parameters (n, q, w). In the asymptotic setting, we suppress all polynomial factors. For exponential run times, we simply round the run time exponent upwards, e.g. $n^2 \cdot 2^{n/3}$ is upper bounded as $2^{0.334n}$.

All modern NTRU-type cryptosystems require $q = \Omega(n)$, encryption schemes have such a lower bound to eliminate decryption errors. E.g. NTRU-Prime restricts the weight $w \geq \frac{1}{3}n$ and chooses $q \geq 16w + 1$, and NTRU-Encrypt recommends only parameter sets $q \in [\frac{8}{3}n, \frac{16}{3}n]$. Restricting $q = O(n)$ is used to obtain small ciphertexts/signatures.

3 Odlyzko's Meet-in-the-Middle algorithm

Odlyzko's attack was originally designed for binary vectors, but the following generalization to ternary (or even small max-norm) vectors is straight-forward.

In the following, we use the short-hand multinomial notion $\binom{n}{a_1,\ldots,a_k,\cdot}$, where $\cdot$ stands for the missing argument $n - a_1 - \ldots - a_k$. Analogous, we use the entropy notion $H(c_1, \ldots, c_k, \cdot)$, where $\cdot$ represents the missing arguments $1 - c_1 - \ldots - c_k$.

Let the search space consist of all ternary weight-w vectors $\mathbf{s} \in \mathcal{T}^n(w/2)$ with even number $w/2$ of ± 1. By Lemma 1, the search space size $\mathcal{S}$ can be approximated as

$$\mathcal{S} = \binom{n}{\frac{w}{2}, \frac{w}{2}, \cdot} \approx 2^{H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)n}.$$

We split $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2) \in \mathcal{T}^{n/2}(w/4) \times \mathcal{T}^{n/2}(w/4)$ in ternary weight-$w/2$ vectors $\mathbf{s}_1, \mathbf{s}_2$ with again an even split of ± 1. Notice that we may rerandomize the positions of (± 1)-entries in $\mathbf{s}$ via permutation of A's columns. The probability that a rerandomized $\mathbf{s}$ has the desired weight distribution split can be estimated via Lemma 1 as

$$\frac{\binom{\frac{n}{2}}{\frac{w}{4}, \frac{w}{4}, \cdot}^2}{\binom{n}{\frac{w}{2}, \frac{w}{2}, \cdot}} \approx 2^{2H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)\frac{n}{2} - H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)n} = 1.$$

Since $\approx$ suppresses polynomial factors, our probability is more precisely $1/\mathrm{poly}(n)$. Thus via permutation of A's columns we always achieve the desired distribution after $\mathrm{poly}(n)$ iterations. Therefore, without loss of generality we always assume throughout this paper that we can evenly split all coefficients of our secrets (up to minor rounding issues).

Let $A = (A_1 | A_2) \in \mathbb{Z}_q^{n \times n}$, where A_1 (respectively A_2) denote the left (respectively right) $n/2$ columns of A. From Definition 1 we obtain the identity $A_1\mathbf{s}_1 = \mathbf{b} - A_2\mathbf{s}_2 + \mathbf{e} \bmod q$. Thus, the terms $A_1\mathbf{s}_1$ and $\mathbf{b} - A_2\mathbf{s}_2$ differ by at most ± 1. We may rewrite this as

$$A_1\mathbf{s}_1 + \mathbf{e}_1 = \mathbf{b} - A_2\mathbf{s}_2 + \mathbf{e}_2 \bmod q \text{ where } \mathbf{e}_1, \mathbf{e}_2 \in \{0, 1\}^n. \tag{1}$$

Since we do not know the error vectors $\mathbf{e}_1, \mathbf{e}_2$, Odlyzko proposed a simple locality sensitive hashing approach. We assign to each $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ coordinate-wise the following binary hash labels $\ell(\mathbf{x})_i$ that can be interpreted as most significant bits of the x_i

$$\ell : \mathbb{Z}_q^n \to \{0,1\}^n \text{ with } \ell(\mathbf{x})_i = \begin{cases} 0 & \text{if } 0 \le x_i < \lfloor q/2 \rfloor - 1 \\ 1 & \text{if } \lfloor q/2 \rfloor \le x_i < q - 1 \end{cases}. \tag{2}$$

For any candidates $\mathbf{s}_1, \mathbf{s}_2$ we hash $A_1\mathbf{s}_1$ and $\mathbf{b} - A_2\mathbf{s}_2$. Notice that for the two *border values* $\lfloor q/2 \rfloor - 1$ and $q - 1$ the error vectors $\mathbf{e}_1, \mathbf{e}_2$ may result in a flip of the hash value. Therefore, we assign for these entries both labels 0 and 1.

Example: Let $q = 4096$. Then the vector $\mathbf{x} = (0, 2047, 3000, 4095)$ with border values $2047, 4095$ in positions $2, 4$ gets assigned all four labels from the set $\{0\} \times \{0,1\} \times \{1\} \times \{0,1\}$.

For all elements we store their labels (sorted). Let X be a random variable for the number of border values for each entry. Then every entry is stored in 2^X places. Since $\mathbb{E}[X] = \frac{2}{q}n = \Theta(1)$, every element occupies only linear space.

Odlyzko's MitM algorithm is given in Algorithm 1, complexities for our addressed cryptosystems can be found in Table 1.

Algorithm 1 Odlyzko's Meet-in-the-Middle

Input: LWE public key $(A, \mathbf{b}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^m$, weight $w \in \mathbb{N}$
Output: $\mathbf{s} \in \mathcal{T}^n(w/2)$ satisfying $\mathbf{e} := A\mathbf{s} - \mathbf{b} \bmod q \in \mathcal{T}^n$

1: **for** all $\mathbf{s}_1 \in \mathcal{T}^{n/2}(w/4)$ **do**
2: Store $(\mathbf{s}_1, \ell(A_1\mathbf{s}_1))$ in list L_1.
3: **for** all $\mathbf{s}_2 \in \mathcal{T}^{n/2}(w/4)$ **do**
4: Store $(\mathbf{s}_2, \ell(\mathbf{b} - A_2\mathbf{s}_2))$ in list L_2.
5: **for** all matches of $(\mathbf{s}_1, \cdot)$ and $(\mathbf{s}_2, \cdot)$ in the second component of $L_1 \times L_2$ **do**
6: **if** $A(\mathbf{s}_1, \mathbf{s}_2) - \mathbf{b} \bmod q \in \mathcal{T}^n$ **then return** $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$

3.1 Correctness

By definition of the hash function ℓ every candidate tuple $(\mathbf{s}_1, \mathbf{s}_2)$ that satisfies Equation (1) for some binary $\mathbf{e}_1, \mathbf{e}_2 \in \{0,1\}^n$ leads to colliding labels $\ell(A_1\mathbf{s}_1) = \ell(\mathbf{b} - A_2\mathbf{s}_2)$.

Contrary, let $(\mathbf{s}_1, \mathbf{s}_2)$ be a candidate tuple that does not satisfy Equation (1) for some binary $\mathbf{e}_1, \mathbf{e}_2 \in \{0,1\}^n$. By A's randomness we have colliding labels $\ell(A_1\mathbf{s}_1) = \ell(\mathbf{b} - A_2\mathbf{s}_2)$ with probability only (roughly) 2^{-n}.

Notice that Odlyzko's locality sensitive hashing makes use of a large field size q to separate wrong candidates $(\mathbf{s}_1, \mathbf{s}_2)$ from the unique correct solution.

	(n, q, w)	$\mathcal{S}$	Odlyzko
NTRU-Encrypt	(509,2048,254)	754 bit	377 bit
	(677,2048,254)	891 bit	445 bit
	(821,4096,510)	1286 bit	643 bit
NTRU-Prime	(653,4621,288)	925 bit	463 bit
	(761,4591,286)	1003 bit	502 bit
	(857,5167,322)	1131 bit	566 bit
BLISS I+II	(512,12289,154)	597 bit	299 bit
GLP I	(512,8383489,342)	802 bit	401 bit

Table 1: Odlyzko's MitM complexity.

3.2 Runtime

Runtime and memory consumption of Algorithm 1 is dominated by the list sizes

$$|L_1| = |L_2| = \binom{\frac{n}{2}}{\frac{w}{4}, \frac{w}{4}, \cdot} \approx 2^{H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)\frac{n}{2}} = \sqrt{\mathcal{S}}. \tag{3}$$

Notice that in line 5 of Algorithm 1 we expect $2^{-n} \cdot |L_1| \cdot |L_2|$ matches, which is larger than $|L_1|, |L_2|$, if $|L_1|, |L_2|$ are larger than 2^n. In this case, we may modify Odlyzko's hash function such that its range gets greater than the list sizes (by assigning more than two labels, see also Section 9).

4 Howgrave-Graham's MitM algorithm

There is a second Meet-in-the-Middle attack that was first analyzed in [HGSW03]. Howgrave-Graham described it in [How07] as Odlyzko's MitM, which let people mix (and confuse) both approaches in the literature.

We briefly discuss this second MitM, which we attribute to Howgrave-Graham. We show that it performs worse than Odlyzko's MitM for ternary vectors for every weight w. However, Howgrave-Graham's method is a first step to our new MitM, since both approaches are based on ambiguous sum representations, also known as the *representation technique*.

Whereas Odlyzko splits $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ *uniquely* as $n/2$-dimensional $\mathbf{s}_i$, Howgrave-Graham represents $\mathbf{s} \in \mathcal{T}^n(w/2)$ *ambiguously* as $\mathbf{s}_1 + \mathbf{s}_2$ with n-dimensional $\mathbf{s}_i \in \mathcal{T}^n(w/4)$. As a consequence, the search space for the $\mathbf{s}_i$ is of increased size

$$\mathcal{S}^{(1)} = \binom{n}{\frac{w}{4}, \frac{w}{4}, \cdot}, \tag{4}$$

as compared to $\binom{\frac{n}{2}}{\frac{w}{4}, \frac{w}{4}, \cdot}$ from (3) in Odlyzko's MitM. But the ambiguity also introduces $R^{(1)} = \binom{w/2}{w/4} \cdot \binom{w/2}{w/4}$ representations of the desired solution $\mathbf{s}$, since each of the $w/2$ 1-coordinates in $\mathbf{s}$ can be represented as $1 + 0$ or $0 + 1$, and analogous for the $w/2$ (-1)-coordinates.

Note that our MitM identity from Equation (1) now becomes $A\mathbf{s}_1 + \mathbf{e}_1 = \mathbf{b} - A\mathbf{s}_2 + \mathbf{e}_2$. We describe Howgrave-Graham's approach in Algorithm 2. Its correctness follows analogous to Section 3.1.

Algorithm 2 Howgrave-Graham's Meet-in-the-Middle

Input: LWE public key $(A, \mathbf{b}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^m$, weight $w \in \mathbb{N}$
Output: $\mathbf{s} \in \mathcal{T}^n(w/2)$ satisfying $\mathbf{e} := A\mathbf{s} - \mathbf{b} \bmod q \in \mathcal{T}^n$

1: **repeat**
2: Sample some $\mathbf{s}_1 \in \mathcal{T}^n(w/4)$. Store $(\mathbf{s}_1, \ell(A_1\mathbf{s}_1))$ in list L_1.
3: Sample some $\mathbf{s}_2 \in \mathcal{T}^n(w/4)$. Store $(\mathbf{s}_2, \ell(\mathbf{b} - A_2\mathbf{s}_2))$ in list L_2.
4: **until** there exists a match $(\mathbf{s}_1, \cdot), (\mathbf{s}_2, \cdot)$ in 2^{nd} component of $L_1 \times L_2$ with $\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{T}^n$
5: **if** $A(\mathbf{s}_1 + \mathbf{s}_2) - \mathbf{b} \bmod q \in \mathcal{T}^n$ **then return** $\mathbf{s} = (\mathbf{s}_1 + \mathbf{s}_2)$

Run Time. In each iteration of the **repeat**-loop we hit a vector $\mathbf{s}_1$ or $\mathbf{s}_2$ that is part of a representation $(\mathbf{s}_1, \mathbf{s}_2)$ of $\mathbf{s}$ with probability $p = \frac{R^{(1)}}{S^{(1)}}$. After $\sqrt{R^{(1)}}$ hits, by the birthday paradox we expect to have both parts $\mathbf{s}_1, \mathbf{s}_2$ of a representation in L_1, L_2. Thus the expected number of iterations in Algorithm 2 is

$$p^{-1} \cdot \sqrt{R^{(1)}} = \frac{S^{(1)}}{R^{(1)}} \cdot \sqrt{R^{(1)}} = \binom{n}{\frac{w}{4}, \frac{w}{4}, \cdot} \cdot \binom{\frac{w}{2}}{\frac{w}{4}}^{-1} \approx 2^{(H(\frac{\omega}{4}, \frac{\omega}{4}, \cdot) - \frac{\omega}{2})n}.$$

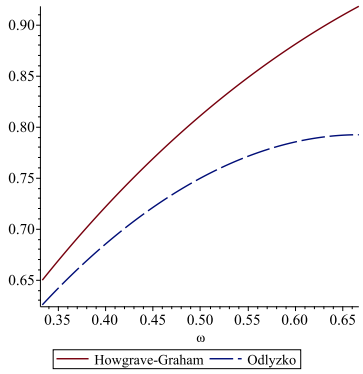

Fig. 1: Run Time Comparison Odlyzko vs Howgrave-Graham

We see in Figure 1 that the run time exponent $H(\frac{\omega}{4}, \frac{\omega}{4}, \cdot) - \frac{\omega}{2}$ is larger than Odlyzko's run time exponent $\frac{1}{2}H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)$ for every $\omega \in [\frac{1}{3}, \frac{2}{3}]$ (in fact this holds for all $0 \leq \omega \leq 1$). E.g. for the prominent NTRU setting $\omega = \frac{3}{8}$ we obtain exponents 0.697 versus 0.665. Hence, one should always prefer Odlyzko's algorithm for ternary secrets.

Intuitively, in a subset sum-type approach of the representation technique as in [HJ10], one would try to construct two lists L_1, L_2 with entries $(\mathbf{s}_1, \ell(A\mathbf{s}_1))$, $(\mathbf{s}_2, \ell(\mathbf{b} - A\mathbf{s}_2))$ recursively such that on expectation $L_1 \times L_2$ contains a single representation. However, the non-linearity of Odlyzko's hash function ℓ hinders such a direct recursive application of the representation technique. We solve this technical issue in the following section.

5 Our New MitM Algorithm – High Level Idea

Let us first state our new LWE key search algorithm Meet-LWE in a high-level manner, for an illustration see Figure 2. Moreover, we introduce some more scalable notation that will prove useful in subsequent sections. For ease of exposition, we again focus on ternary secret LWE keys $\mathbf{s}, \mathbf{e} \in \mathcal{T}^n$.

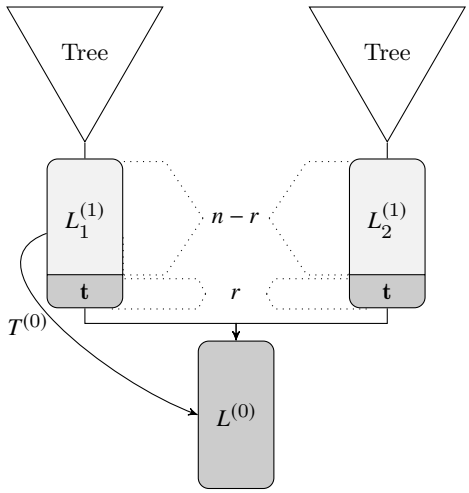

Fig. 2: Meet-LWE high level structure

As in Section 4, we represent a weight-w ternary $\mathbf{s} \in \mathcal{T}^n$ as a sum $\mathbf{s}_1 + \mathbf{s}_2$ of n-dimensional $\mathbf{s}_1, \mathbf{s}_2$ in $R^{(1)}$ ways. Here, $\mathbf{s}_1, \mathbf{s}_2$ may be ternary weight-$w/2$ vectors (Section 6), ternary vectors with weight larger than $w/2$ (Section 7), or even non-ternary vectors (Section 8). As a rule of thumb, the larger the search space for $\mathbf{s}_1, \mathbf{s}_2$, the larger also the number of representations $R^{(1)}$.

We use the LWE identity

$$A\mathbf{s}_1 + \mathbf{e}_1 = \mathbf{b} - A\mathbf{s}_2 + \mathbf{e}_2 \text{ for some binary } \mathbf{e}_1, \mathbf{e}_2 \in \{0,1\}^n. \tag{5}$$

Assume for a moment that we know the error vector $\mathbf{e}$, and thus $\mathbf{e}_1, \mathbf{e}_2$. Let $r = \lfloor \log_q R^{(1)} \rfloor$, and fix a randomly chosen target vector $\mathbf{t} \in \mathbb{Z}_q^r$. Moreover, let us

define the projection π_r on the first r coordinates as

$$\pi_r : \mathbb{Z}_q^n \to \mathbb{Z}_q^r,\ \mathbf{x} = (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_r), \tag{6}$$

which is a ring homomorphism (as opposed to Odlyzko's hash function).

Notice that the range $\mathbb{Z}_q^r$ of π_r has size $q^r < q^{\log_q R^{(1)}} = R^{(1)}$. Therefore, we expect that for at least one out of the $R^{(1)}$ representations $(\mathbf{s}_1, \mathbf{s}_2)$ of $\mathbf{s}$ its projection via π_r matches the random target $\mathbf{t}$, i.e.

$$\pi_r(A\mathbf{s}_1 + \mathbf{e}_1) = \mathbf{t} \bmod q.$$

By Equation (5) and π_r's ring homomorphism property, this automatically implies the second identity $\pi_r(\mathbf{b} - A\mathbf{s}_2 + \mathbf{e}_2) = \mathbf{t} \bmod q$, as well. Let us stress that for checking both identities it suffices to only know the first r coordinates $\pi_r(\mathbf{e})$ of $\mathbf{e}$.

Let $\ell : \mathbb{F}_q^n \to \{0,1\}^n$ be Odlyzko's hash function from Equation (2). Our goal is to construct lists $L_1^{(1)}, L_2^{(1)}$ (see also Figure 2) satisfying

$$\begin{aligned} L_1^{(1)} &= \{(\mathbf{s}_1, \ell(A\mathbf{s}_1)) \mid \pi_r(A\mathbf{s}_1 + \mathbf{e}_1) = \mathbf{t} \bmod q\}, \\ L_2^{(1)} &= \{(\mathbf{s}_2, \ell(\mathbf{b} - A\mathbf{s}_2)) \mid \pi_r(\mathbf{b} - A\mathbf{s}_2 + \mathbf{e}_2) = \mathbf{t} \bmod q\}. \end{aligned} \tag{7}$$

Our resulting LWE MitM key search is given in Algorithm 3, called Meet-LWE.

Algorithm 3 LWE Key Search Meet-LWE (High-Level)

Input: LWE key $(A, \mathbf{b}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$, weight $w \in \mathbb{N}$
Output: ternary weight-w $\mathbf{s}$ satisfying $\mathbf{e} = A\mathbf{s} - \mathbf{b} \bmod q \in \mathcal{T}^n$

1: We represent $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ using different vector sets for $\mathbf{s}_1, \mathbf{s}_2$ (see Sections 6 to 8). Let $R^{(1)}$ be the resulting number of representations. Let $r = \lfloor \log_q(R^{(1)}) \rfloor$.
2: **for** all $\pi_r(\mathbf{e}) \in \mathcal{T}^r$ **do**
3: Construct $L_1^{(1)}, L_2^{(1)}$ from Equation (7), using some tree-based list construction. ▷ For the list construction see Sections 6 to 8.
4: **for** all matches of $(\mathbf{s}_1, \cdot)$ and $(\mathbf{s}_2, \cdot)$ in the second component of $L_1 \times L_2$ **do**
5: **if** $(\mathbf{s} := \mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{T}^n$ has weight $w)$ and $(A\mathbf{s} - \mathbf{b} \bmod q \in \mathcal{T}^n)$ **then return** $\mathbf{s}$

5.1 Correctness

In a nutshell, Meet-LWE (Algorithm 3) constructs an $\mathbf{s}$ that fulfills $A\mathbf{s} = \mathbf{b} + \mathbf{e}$ on r coordinates exactly, and on the remaining $n - r$ coordinates approximately via Odlyzko's hash function. Thus, Meet-LWE's correctness follows from the discussion in Section 3 and 4.

While Odlyzko's matching of $\mathbf{s}_1, \mathbf{s}_2$ guarantees that $A(\mathbf{s}_1 + \mathbf{s}_2) - \mathbf{b} \in \mathcal{T}^n$ (with high probability), it does not ensure that $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ is a ternary weight-w vector. Therefore, we check for *consistency* of $\mathbf{s}$ in line 5 of Algorithm 3. Filtering out inconsistent solutions is called *Match-and-filter*, a standard technique for representations [BCJ11, BJMM12].

5.2 Runtime

Algorithm 3 has an outer **for**-loop that guesses r coordinates of $\mathbf{e}$, and an inner loop which is basically a subset sum-type list construction step.

Let us start with the outer loop's guessing complexity T_g. Assume $\mathbf{e}$ is a random ternary vector. Then the guessing complexity is

$$T_g = 3^r \le 3^{\log_q R^{(1)}} = 2^{\log_2(3)\frac{\log_2 R^{(1)}}{\log_2 q}}.$$

For low weight $\mathbf{e}$ we may improve by searching for a subset of r projected coordinates that are all zero. However, for the cryptosystems that we address in our applications $\mathbf{e}$ is not low weight.

In all our instantiations of Meet-LWE in the following sections, we have $\log_2 R^{(1)} = \mathcal{O}(n)$. Since $q = \Omega(n)$ (see Section 2), we obtain a guessing complexity of

$$T_g = 2^{\mathcal{O}\left(\frac{n}{\log q}\right)} = 2^{\mathcal{O}\left(\frac{n}{\log n}\right)}.$$

Since the inner loop has—as in Odlyzko's attack—list construction complexity $T_\ell = 2^{\mathcal{O}(n)}$, the overall *asymptotic* complexity $T = T_g \cdot T_\ell$ is fully determined by the inner loop's complexity T_ℓ, and guessing r coordinates of $\mathbf{e}$ just adds an $o(1)$-term to the inner loop's run time exponent!

In the following sections, we see that guessing $\pi_r(\mathbf{e})$ also leads to tolerable *non-asymptotic* overheads T_g for real-world parameters, for which we cannot simply neglect a $2^{\mathcal{O}(\frac{n}{\log n})}$-term.

In Section 6-8, we instantiate Algorithm 3 with varying representations of ternary $\mathbf{s}$ with increasing $R^{(1)}$. As a warm-up, we start in Section 6 with Howgrave-Graham's representation from Section 4 as $\mathbf{s} = \mathbf{s}_1^{(1)} + \mathbf{s}_2^{(1)} \in \mathcal{T}^n(w/2)$ with ternary $\mathbf{s}_1^{(1)}, \mathbf{s}_2^{(1)} \in \mathcal{T}^n(w/4)$. This already leads to an (asymptotical) complexity improvement from $\frac{S^{(1)}}{\sqrt{R^{(1)}}}$ downto $\frac{S^{(1)}}{R^{(1)}}$, superior to Odlyzko's attack.

6 Rep-0: First Instantiation of Meet-LWE

The reader is advised to compare Figures 2 and 3. Meet-LWE's tree-based list construction from Figure 2 is realized by a single additional tree layer in Figure 3.

Let $\mathbf{s} \in \mathcal{T}^n(w/2)$ be the weight-w ternary secret. We represent $\mathbf{s}$ as the sum of weight-$w/2$ ternary secrets $\mathbf{s}_1^{(1)}, \mathbf{s}_2^{(1)} \in \mathcal{T}^n(w/4)$.

Recall from Section 4, Equation (4) that the search space for the $\mathbf{s}_i^{(1)}$ is of size $S^{(1)} = \binom{n}{\frac{w}{4}, \frac{w}{4}, \cdot} \approx 2^{H(\frac{\omega}{4}, \frac{\omega}{4}, \cdot)n}$ and we have $R^{(1)} = \binom{w/2}{w/4}^2 \approx 2^{\omega n}$ representations. Hence the lists $L_1^{(1)}, L_2^{(1)}$ in Algorithm 3 are both of size

$$L^{(1)} = \frac{S^{(1)}}{R^{(1)}} = \binom{n}{\frac{w}{4}, \frac{w}{4}, \cdot}\binom{\frac{w}{2}}{\frac{w}{4}}^{-2} \approx 2^{(H(\frac{\omega}{4}, \frac{\omega}{4}, \cdot) - \omega)n}.$$

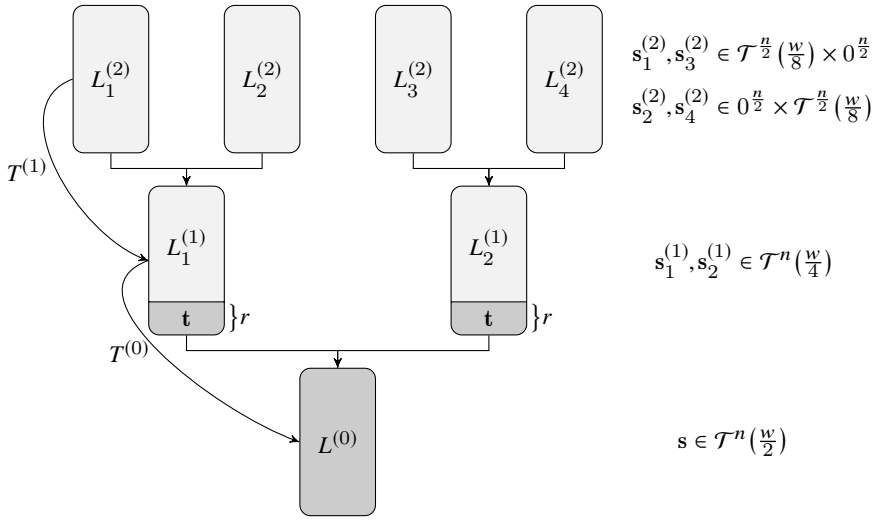

Fig. 3: Rep-0 instantiation of Meet-LWE

Let us construct $L_1^{(1)}, L_2^{(1)}$ in a standard MitM manner. Namely, we enumerate $\mathbf{s}_1^{(1)} \in \mathcal{T}^n(w/4)$ as the sum of

$$\mathbf{s}_1^{(2)} \in \mathcal{T}^{\frac{n}{2}}(w/8) \times 0^{\frac{n}{2}} \text{ and } \mathbf{s}_2^{(2)} \in 0^{\frac{n}{2}} \times \mathcal{T}^{\frac{n}{2}}(w/8).$$

Analogous, we proceed with $\mathbf{s}_2^{(1)}$. All four $\mathbf{s}_i^{(2)}$ are from a search space of size $S^{(2)} = \sqrt{S^{(1)}}$. Thus, on level 2 of our complete binary search tree we obtain four lists $L_1^{(2)}, \ldots, L_4^{(2)}$ each of size

$$L^{(2)} = S^{(2)} = \binom{\frac{n}{2}}{\frac{w}{8}, \frac{w}{8}, \cdot} \approx 2^{\frac{1}{2}H(\frac{\omega}{4}, \frac{\omega}{4}, \cdot)n}.$$

The time $T^{(1)}$ to construct the level-1 lists $L_1^{(1)}$, respectively $L_2^{(2)}$, from the (sorted) level-2 lists $L_1^{(2)}, L_2^{(2)}$, respectively $L_3^{(2)}, L_3^{(2)}$, is

$$T^{(1)} = \max\{L^{(2)}, L^{(1)}\}.$$

From Section 3.2, we know that Meet-LWE's final approximate matching on $n - r$ coordinates can be realized via Odlyzko's hash function in time

$$T^{(0)} = \max\{L^{(1)}, 2^{-(n-r)}(L^{(1)})^2\} = L^{(1)} \text{ for all } \omega \in [0, 1].$$

Thus, the total run time of list construction is $T_\ell = \max\{T^{(1)}, T^{(0)}\} = \max\{L^{(2)}, L^{(1)}\}$. This implies that we obtain run time exponent

$$\max\left\{\frac{1}{2}H\left(\frac{\omega}{4}, \frac{\omega}{4}, \cdot\right), H\left(\frac{\omega}{4}, \frac{\omega}{4}, \cdot\right) - \omega\right\},$$

which improves on Odlyzko's exponent $\frac{1}{2}H(\frac{\omega}{2}, \frac{\omega}{2}, \cdot)$ for every $\omega \in [0, 0.87]$.

Remark 1. We could slightly improve our Rep-0 attack such that the size of $L^{(1)}$ dominates the run time for all $\omega \leq \frac{2}{3}$, i.e. up to random $\mathbf{s} \in \mathcal{T}^n$. As described above, $L^{(2)}$ dominates for $\omega \geq 0.58$. However, such an improvement comes at the cost of adding a third tree layer.

As Rep-0 is mainly for didactical reasons to make the reader familiar with the technique, we chose to sacrifice optimality for the sake of a simpler algorithmic description. The Rep-0 results are superseeded anyway in subsequent sections, where we optimize our tree depth.

Theorem 1. *Let* $\mathbf{s}$ *be a ternary weight-*$w = \omega n$ *LWE key. Then* $\mathbf{s}$ *can be found in time and space (neglecting polynomial factors)*

$$3^{2\log_q\binom{w/2}{w/4}} \cdot \max\left\{\binom{\frac{n}{2}}{\frac{w}{8},\frac{w}{8},\cdot}, \binom{n}{\frac{w}{4},\frac{w}{4},\cdot}\binom{\frac{w}{2}}{\frac{w}{4}}^{-2}\right\} \approx 2^{\max\{\frac{1}{2}H(\frac{\omega}{4},\frac{\omega}{4},\cdot), H(\frac{\omega}{4},\frac{\omega}{4},\cdot)-\omega\}n}.$$

Proof. We have $R^{(1)} = \binom{w/2}{w/4}^2$ representations. In Meet-LWE we guess $r = \lfloor \log_q R^{(1)} \rfloor$ coordinates of $\mathbf{e} \in \{-1,0,1\}^n$ in time

$$T_g = 3^r \leq 3^{2\log_q\left(\binom{w/2}{w/4}\right)}.$$

The run time T_ℓ of Meet-LWE's list construction is dominated by the maximum of the sizes $L^{(2)}$, $L^{(1)}$ of the lists on level 2 and 1:

$$T_\ell = \max\left\{L^{(2)}, L^{(1)}\right\} = \max\left\{S^{(2)}, \frac{S^{(1)}}{R^{(1)}}\right\} = \max\left\{\binom{\frac{n}{2}}{\frac{w}{8},\frac{w}{8},\cdot}, \binom{n}{\frac{w}{4},\frac{w}{4},\cdot}\binom{\frac{w}{2}}{\frac{w}{4}}^{-2}\right\}.$$

Meet-LWE's total run time is $T = T_g \cdot T_\ell$. □

	(n,q,w)	Odlyzko	Rep-0 [bit]	$\log_{\mathcal{S}}$Rep-0
NTRU-Encrypt	(509,2048,254)	377 bit	323 = 287 + 35	0.43
	(677,2048,254)	445 bit	382 = 347 + 35	0.43
	(821,4096,510)	643 bit	546 = 480 + 66	0.42
NTRU-Prime	(653,4621,288)	463 bit	389 = 352 + 36	0.42
	(761,4591,286)	502 bit	426 = 390 + 36	0.42
	(857,5167,322)	565 bit	479 = 439 + 40	0.42
BLISS I+II	(512,12289,154)	299 bit	256 = 238 + 17	0.43
GLP I	(512,8383489,342)	401 bit	337 = 314 + 23	0.42

Table 2: Complexity comparison Odlyzko (Section 3) vs. Rep-0.

In Table 2 we computed Meet-LWE's complexity with our Rep-0 representations using the exact formula on the left hand side of Theorem 1 (and not the

$H(\cdot)$-approximation). In the notation of Section 5 we split MEET-LWE's complexity T in the cost T_ℓ of the inner loop for list construction, and T_g of the outer loop for guessing r coordinates. E.g. the NTRU-Encrypt instance $(509, 2048, 254)$ has total complexity $T = 323$ bit, which splits in $T_\ell = 287$ bit for list construction, and $T_g = 35$ bit for guessing. Since we round all complexities to the next integers, $T_\ell + T_g$ might deviate from T by one.

We observe that REP-0 already reduces Odlyzko's bit complexities by roughly 15%, resulting in complexity $\mathcal{S}^{0.43}$. Due to a large q, key guessing in BLISS and GLP can be performed much more efficiently, as can be observed by the small T_g-values of 17 and 23 bit.

7 Rep-1: Using Additional Ones

The idea of REP-1 is to represent a weight-w ternary $\mathbf{s} \in \mathcal{T}^n(w/2)$ as the sum of $\mathbf{s}_1^{(1)}, \mathbf{s}_2^{(1)} \in \mathcal{T}^n(w^{(1)})$, where $w^{(1)} > w/4$. In comparison to Section 6, this further increases the number R of representations. We build a complete binary search tree T of depth 4, see Figure 4. In the following, we describe the lists $L_i^{(j)}$ on level j.

7.1 Level-1 lists

Define $w^{(0)} = w/2$ and $w^{(1)} = w^{(0)}/2 + \epsilon^{(1)}$. Let $\mathbf{s}_1, \mathbf{s}_2 \in T^n(w^{(1)})$. Notice that $\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{T}^n(w^{(0)})$ iff in the sum $\epsilon^{(1)}$ many (-1)-coordinates cancel with 1-coordinates, and vice versa. In the terminology of representations, we additionally represent 0 as $(-1) + 1$ and $1 + (-1)$. Recall that in Section 6 we represented ± 1 as $\pm 1 + 0$ and 0 ± 1, but 0 was always represented solely as $0 = 0 + 0$.

Therefore, we increase the number of level-1 representations of $\mathbf{s} \in \mathcal{T}^n(w^{(0)})$ to

$$R^{(1)} = \binom{w^{(0)}}{\frac{w^{(0)}}{2}}^2 \cdot \binom{n - 2w^{(0)}}{\epsilon^{(1)}, \epsilon^{(1)}, \cdot},$$

where the second factor accounts for our additional representations. On the downside, an increased number $w^{(1)}$ of ± 1 also increases the search space size to

$$S^{(1)} = \binom{n}{w^{(1)}, w^{(1)}, \cdot}.$$

Our goal is to construct two lists $L_1^{(1)}, L_2^{(1)}$ satisfying Equation (7), both having size $L^{(1)} = S^{(1)}/R^{(1)}$. But as opposed to Section 6, due to the increased $S^{(1)}$ we have to construct these lists recursively, until the search space gets small enough to apply a simple MitM construction.

7.2 Level $2 \le j < d$ lists

For simplicity of exposition, we perform our analysis for trees of depth $d = 4$, such that the reader may easily follow via Figure 4. The analysis naturally generalizes to arbitrary d, see also Appendices A.2 and A.3.

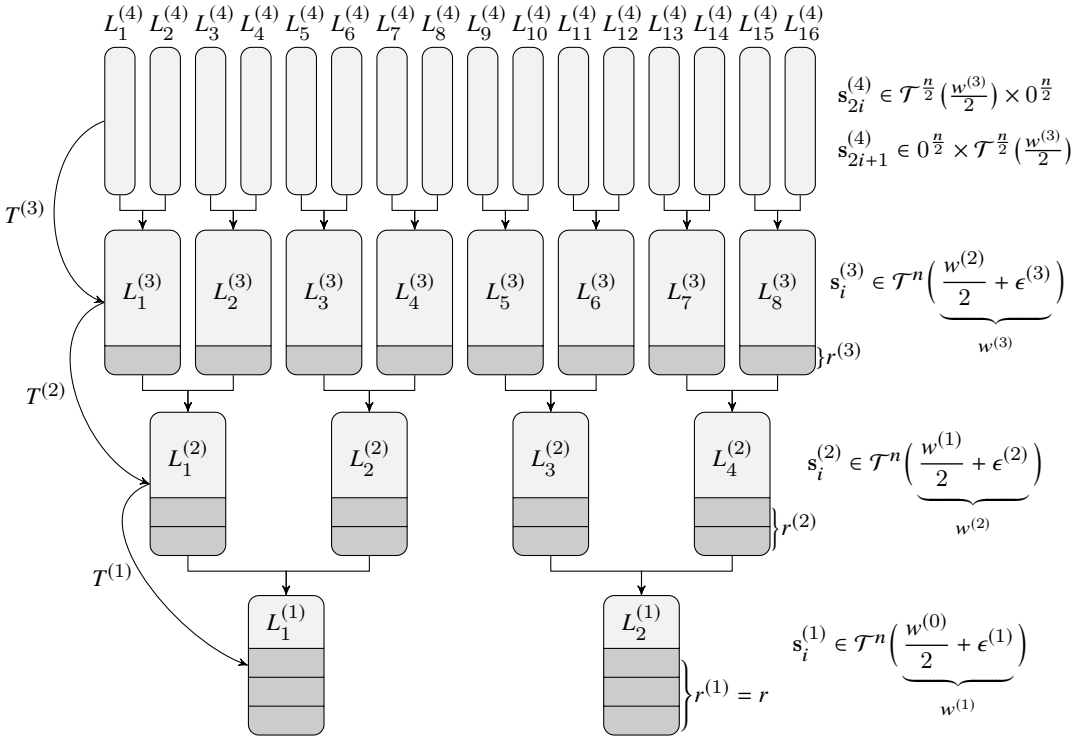

Fig. 4: Rep-1 instantiation of Meet-LWE with depth $d = 4$

On level $j \in \{1, 2, 3\}$ of our complete binary depth-4 search tree T we construct lists $L_1^{(j)}, \ldots, L_{2^j}^{(j)}$ with vectors $\mathbf{s}_1^{(j)}, \ldots, \mathbf{s}_{2^j}^{(j)}$ of weight $w^{(j)} = w^{(j-1)}/2 + \epsilon^{(j)}$.

This gives us representations and search space size

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

Let us now describe the level-2 lists more precisely, the level-3 lists are defined analogous. We rewrite Equation (1) as the following 4-sum for some binary unknowns $\mathbf{e}_1, \mathbf{e}_2 \in \{0, 1\}^n$:

$$A\mathbf{s}_1^{(2)} + A\mathbf{s}_2^{(2)} + \mathbf{e}_1 = \mathbf{b} - A\mathbf{s}_3^{(2)} - A\mathbf{s}_4^{(2)} + \mathbf{e}_2 \bmod q.$$

Let $r^{(1)} := r = \lfloor \log_q R^{(1)} \rfloor$ be the number of guessed coordinates in Meet-LWE (Algorithm 3), and let the fixed target be $\mathbf{t} = \pi_{r^{(1)}}(A\mathbf{s}_1^{(1)} + \mathbf{e}_1) \in \mathbb{Z}_q^{r^{(1)}}$. Let $r^{(2)} = \lfloor \log_q R^{(2)} \rfloor$ be the number of fixed coordinates on level-2, which is a subset of the $r^{(1)}$ fixed coordinates on level 1. Choose two random $\mathbf{t}_1^{(2)}, \mathbf{t}_2^{(2)} \in \mathbb{Z}_q^{r^{(2)}}$. Then

the level-2 lists are defined as

$$\begin{aligned}
L_1^{(2)} &= \{(\mathbf{s}_1^{(2)}, A\mathbf{s}_1^{(2)}) \mid \pi_{r^{(2)}}(A\mathbf{s}_1^{(2)}) = \mathbf{t}_1^{(2)} \bmod q\},\\
L_2^{(2)} &= \{(\mathbf{s}_2^{(2)}, A\mathbf{s}_2^{(2)}) \mid \pi_{r^{(2)}}(A\mathbf{s}_2^{(2)} + \mathbf{e}_1) = \pi_{r^{(2)}}(\mathbf{t}) - \mathbf{t}_1^{(2)} \bmod q\},\\
L_3^{(2)} &= \{(\mathbf{s}_3^{(2)}, \mathbf{b} - A\mathbf{s}_3^{(2)}) \mid \pi_{r^{(2)}}(\mathbf{b} - A\mathbf{s}_3^{(2)}) = \mathbf{t}_2^{(2)} \bmod q\},\\
L_4^{(2)} &= \{(\mathbf{s}_4^{(2)}, -A\mathbf{s}_4^{(2)}) \mid \pi_{r^{(2)}}(-A\mathbf{s}_4^{(2)} + \mathbf{e}_2) = \pi_{r^{(2)}}(\mathbf{t}) - \mathbf{t}_2^{(2)} \bmod q\}. \qquad (8)
\end{aligned}$$

Let $\mathbf{s}_1^{(1)} = \mathbf{s}_1^{(2)} + \mathbf{s}_2^{(2)}$. Notice that by definition in Equation (8) and the linearity of π we automatically have

$$\pi_{r^{(2)}}(A\mathbf{s}_1^{(1)}+\mathbf{e}_1) = \pi_{r^{(2)}}(A(\mathbf{s}_1^{(2)}+\mathbf{s}_2^{(2)})+\mathbf{e}_1) = \pi_{r^{(2)}}(A\mathbf{s}_1^{(2)})+\pi_{r^{(2)}}(A\mathbf{s}_2^{(2)}+\mathbf{e}_1) = \pi_{r^{(2)}}(\mathbf{t}) \bmod q.$$

Analogous, for $\mathbf{s}_2^{(1)} = \mathbf{s}_3^{(2)} + \mathbf{s}_4^{(2)}$ we obtain $\pi_{r^{(2)}}(\mathbf{b} - A\mathbf{s}_2^{(1)} + \mathbf{e}_2) = \pi_{r^{(2)}}(\mathbf{t}) \bmod q$.

That is, we automatically satisfy our target-$\mathbf{t}$ condition for the level-1 lists $L_1^{(1)}, L_2^{(1)}$ from Equation (7) on $r^{(2)}$ coordinates. It remains to match the elements in $L_1^{(2)} \times L_2^{(2)}$ and $L_3^{(2)} \times L_4^{(2)}$ on the remaining $r^{(1)} - r^{(2)}$ coordinates.

Eventually, the level-3 lists are constructed via a MitM approach out of level-4 lists, similar to Section 6.

7.3 Correctness

Notice that as opposed to the level-1 lists, we compute in (8) the value $A\mathbf{s}_i^{(2)}$ instead of the Odlyzko-hashed value $\ell(A\mathbf{s}_i^{(2)})$. So whereas on level-1, we compute an approximate matching of vectors via comparing hash values of some locality sensitive hash function, on all other levels $j > 1$ we compute an *exact* (non-hashed) matching of the projected vectors on $r^{(j)}$ coordinates.

On every level $j \geq 1$, our matching ensures that (on expectation) at least one representation of the solution satisfies all conditions. Thus, we expect that the approximate matching of level-1 lists $L_1^{(1)}$ and $L_2^{(1)}$ provides the desired solution in Meet-LWE.

Match-and-Filter. Notice that the sum of vector from $\mathcal{T}^n(w^{(j)})$, $1 \leq j < 4$, is in general not in the target distribution $\mathcal{T}^n(w^{(j-1)})$. We filter out all vector sums that do not have exactly $w^{(j-1)}$ of each (-1)- and 1-coordinates.

7.4 Run Time

The run time of Rep-1 is dominated by constructing all lists of size $L^{(j)}$ on all levels $j \geq 0$.

On level-4, we have sorted lists of size $L^{(4)} = \sqrt{S^{(3)}}$. Ignoring polynomial factors, the construction of each level-4 list costs time $T^{(3)} = L^{(3)}$. Since we match level-4 list elements on $r^{(3)}$ coordinates, the construction of level-3 lists costs on expectation time

$$T^{(3)} = \frac{(L^{(4)})^2}{q^{r^{(3)}}}.$$

Since level-3 list elements already sum to the target $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ on $r^{(3)}$ coordinates, for the construction of level-2 lists we have to match elements on the remaining $r^{(2)} - r^{(3)}$ coordinates. This can be done in expected time

$$T^{(2)} = \frac{(L^{(3)})^2}{q^{r^{(2)}-r^{(3)}}}.$$

Notice that $\frac{(L^{(3)})^2}{q^{r^{(2)}-r^{(3)}}} \geq L^{(2)}$, since we filter out matching level-3 vector sums that are not in $\mathcal{T}^n(w^{(2)})$.

Define $r^{(4)} = 0$. Then in general we can construct every level-j list for $4 > j > 0$ 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. Since 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

$$T = \max\{T^{(0)}, \ldots, T^{(4)}\} \text{ and } M = \max\{L^{(1)}, \ldots, L^{(4)}\}.$$

The following optimizations balance out the dominating terms $T^{(1)}, T^{(2)}, T^{(3)}$.

7.5 Optimization: Asymptotic and Non-Asymptotic

In Table 3 we optimized for different relative weights ω Meet-LWE's list construction cost T_ℓ, which depends on n and ω only (and not on q). Asymptotically, we can neglect the guessing cost T_g. We write $T_\ell = 2^{c(\omega)n(1+o(1))}$ for some constant $c(\omega)$ that we provide in Table 3, including the optimized additional ones that we add on level j, parametrized by $\bar{\epsilon}^{(j)} = \frac{\epsilon^{(j)}}{n}$.

ω	Odlyz.	Rep-0	Rep-1	$\log_{\mathcal{S}} T_\ell$	$\bar{\epsilon}^{(1)}$, $\bar{\epsilon}^{(2)}$, $\bar{\epsilon}^{(3)}$
0.3	0.591	0.469	0.298	**0.25**	0.054, 0.024, 0.005
0.375	0.665	0.523	0.323	**0.24**	0.056, 0.025, 0.005
0.441	0.716	0.561	0.340	**0.24**	0.061, 0.028, 0.007
0.5	0.750	0.588	0.356	**0.24**	0.062, 0.028, 0.007
0.62	0.790	0.625	0.389	**0.25**	0.069, 0.028, 0.006
0.667	0.793	0.634	0.407	**0.26**	0.068, 0.025, 0.006

Table 3: Asymptotics of Rep-1 compared to Odlylzko (Section 3) and Rep-0 (Section 6), where we also optimized the search tree depth for Rep-0 (see Remark 1).

(n, q, w)	Odly. REP-1 [bit]		params
NTRU-Encypt			
(509,2048,254)	377	$273 = 214 + 59$	3: 16,3
		$267 = 195 + 72$	4: 30,13,4
(677,2048,254)	445	$316 = 246 + 70$	3: 22,5,0
		$315 = 236 + 79$	4: 30,11,1
(821,4096,510)	643	$472 = 375 + 97$	3: 24,6
		$452 = 339 + 113$	4: 46,19,7
NTRU-Prime			
(653,4621,288)	463	$321 = 254 + 67$	3: 22,5
		$316 = 234 + 82$	4: 40,18,4
(761,4591,286)	502	$348 = 277 + 71$	3: 24,6
		$345 = 262 + 83$	4: 36,14,2
(857,5167,322)	565	$390 = 311 + 79$	3: 27,6
		$387 = 302 + 85$	4: 33,11,1
BLISS I+II			
(512,12289,154)	299	$208 = 171 + 37$	3: 15,3
GLP I			
(512,8383489,342)	401	$275 = 242 + 33$	3: 16,3
		$260 = 222 + 39$	4: 34,12,4

Table 4: Comparison Odlyzko vs. REP-1.

Since the REP-I exponent is roughly half of the Odlyzko MitM exponent, the list construction takes about $\mathcal{S}^{\frac{1}{4}}$ instead of $\mathcal{S}^{\frac{1}{2}}$.

The parameters $\bar{\epsilon}^{(j)}$ are useful starting points for the non-asymptotic analysis in Table 4. Column REP-1 is in a *bit complexity* format $T = T_\ell + T_g$, i.e. the total run time is expressed via T_ℓ for list construction and T_g for guessing. As an example take the first entry $273 = 214 + 59$. List construction takes time $T_\ell = 214$ bit and key guessing $T_g = 59$ bit for a total running time of $T = 273$ bit.

The params-column $4 : 30, 13, 4$ in Table 4 denotes that we construct a search tree of depth 4, where we add 30 additional ± 1 for every level-1 list, 13 additional ± 1 for every level-2 list, and 4 additional ± 1 for every level-3 list. Notice that the relative weight $\omega = 0.5$ in Table 3 has $(\bar{\epsilon}^{(1)}, \bar{\epsilon}^{(2)}, \bar{\epsilon}^{(3)}) = (0.062, 0.028, 0.007)$ and $509 \cdot (\bar{\epsilon}^{(1)}, \bar{\epsilon}^{(2)}, \bar{\epsilon}^{(3)}) \approx (31, 14, 4)$. Therefore, the optimal value $(30, 14, 3)$ is already well approximated by the asymptotic analysis.

We optimized every instance with depth-3 and depth-4 search trees. BLISS with its small weight was the only instance with an optimum achieved in depth 3, all others benefit from depth 4. Increasing to depth 5 did not give any further improvements, as predicted by our asymptotic analysis that was also optimal for depth-4 trees.

We observe from Table 4 that for depth-4 trees the list construction bit complexity of T_ℓ (and memory consumption) only—e.g. without T_g—is roughly half of Odlyzko's MitM, as predicted by Table 3. On the downside, in comparison to REP-0 in Table 2 the guessing complexity T_g increases quite significantly.

In a nutshell, MEET-LWE uses the additional representations to decrease T_ℓ at the cost of T_g. Since guessing is asymptotically cheaper than list construction, this tradeoff provides in total—already for practical size parameter settings—significant savings.

8 Rep-2: Extending the Digit Set with Two

In Section 7 we already saw that additional ones lead to a larger number $R^{(1)}$ of representations, thereby significantly improving run times. In this section, we extend the digit set with ±2, resulting in yet slight improvements.

The benefit of representing ternary $\mathbf{s}$ via $\mathbf{s}_1^{(1)}+\mathbf{s}_2^{(1)}$ with $\mathbf{s}_1^{(1)}, \mathbf{s}_2^{(1)} \in \{-2,-1,0,1,2\}^n$ is that we obtain additionally the following variety of representations for each coordinate of $\mathbf{s}$:

$$\begin{aligned}(-1) &= (-2) + 1 = (-1) + 0 = 0 + (-1) = 1 + (-2),\\ 1 &= (-1) + 2 = 0 + 1 = 1 + 0 = (-2) + 1,\\ 0 &= (-2) + 2 = (-1) + 1 = 0 + 0 = 1 + (-1) = 2 + (-2).\end{aligned} \tag{9}$$

Moreover, REP-2 also allows us for the first time to analyze LWE secrets $\mathbf{s} \in \mathbb{Z}_q^n \cap \{\pm 2, \pm 1, 0\}$, as they appear e.g. in some BLISS instances, see Section 8.3.

The benefits of REP-2 come at the price of a quite involved technical analysis, especially for counting the new representations via new optimization parameters. Since REP-2 is algorithmically close to REP-1 from Figure 4, in the following we only state the results. A detailed analysis including the semantics of our optimization parameters $\epsilon_{k\ell}^{(j)}$ is presented in Appendix A.

While we obtained signicant savings from REP-0 to REP-1, the savings from REP-2 are in comparison quite smallish. This demonstrates that our technique converges quite quickly, once we construct sufficiently many representations. Similar effects were already observed in the subset sum context [HJ10, BCJ11, BBSS20].

8.1 Optimization – Asymptotics

As in Section 7.5 we asymptotically neglect the guessing time T_g. The total run time T is dominated by the list construction $T_\ell = 2^{c(\omega)(n+o(n))}$ for some constant $c(\omega)$ that we provide in Table 5. We obtained optimal parameters for depth $d = 4$, further increasing the depth did not improve. As usual, we denote $\bar{\epsilon}_{k\ell}^{(j)} = \frac{\epsilon_{k\ell}^{(j)}}{n}$. In our optimization, we always had $\bar{\epsilon}_{20}^{(2)} = \bar{\epsilon}_{20}^{(3)} = \bar{\epsilon}_{21}^{(3)} = \bar{\epsilon}_{22}^{(3)} = 0$.

We see that in the range $\omega \in [\frac{3}{10}, \frac{2}{3}]$, our new run time exponent is smaller than half Odlyzko's run time exponent. Thus, for these ω the asymptotic complexity is less than $\mathcal{S}^{\frac{1}{4}}$ (see column $\log_{\mathcal{S}} T_\ell$).

ω	Odlyz.	Rep-1	Rep-2	$\log_{\mathcal{S}} T_\ell$	$\bar{\epsilon}_{10}^{(1)}, \bar{\epsilon}_{20}^{(1)}, \bar{\epsilon}_{21}^{(1)}, \bar{\epsilon}_{10}^{(2)}, \bar{\epsilon}_{21}^{(2)}, \bar{\epsilon}_{22}^{(2)}, \bar{\epsilon}_{10}^{(3)}$
0.3	0.591	0.298	0.295	**0.25**	50 , 0 , 1 , 26 , 0 , 0, 6 $[10^{-3}]$
0.375	0.665	0.323	0.318	**0.24**	44 , 1 , 3 , 24 , 1 , 1, 7 $[10^{-3}]$
0.441	0.716	0.340	0.334	**0.23**	41 , 1 , 4 , 25 , 1 , 1, 7 $[10^{-3}]$
0.5	0.750	0.356	0.348	**0.23**	40 , 1 , 4 , 25 , 1 , 1, 7 $[10^{-3}]$
0.62	0.790	0.389	0.371	**0.24**	35 , 1 , 5 , 26 , 1 , 1, 7 $[10^{-3}]$
0.667	0.793	0.407	0.379	**0.24**	33 , 0 , 6 , 26 , 1 , 1, 7 $[10^{-3}]$

Table 5: Asymptotics of Rep-2 in comparison to Odlyzko (Section 3) and Rep-1.

8.2 Optimization – Non-Asymptotic

In the non-asymptotic analysis from Table 6, we still get slight improvements from Rep-2 over Rep-1. Moreover, in comparison to Rep-1 we see a further emphasis on the key guessing complexity. We optimized our algorithm for every instance in depths $d = 3$ and $d = 4$. Take e.g. the NTRU-Encrypt $n = 509$ instance that has level-4 complexities $T_\ell = 190$ for list construction and $T_g = 74$ for guessing. In comparison to Rep-1 we have in level 4 complexities $T_\ell = 195$ and $T_r = 72$. So with Rep-2 we gain in list construction, while we only moderately lose in key guessing.

(n, q, w)	$\mathcal{S}$[bit]	Rep-1[bit]	Rep-2[bit]	params
NTRU-Encypt				
(509,2048,254)	754	273 = 214 + 59	271 = 211 + 60	3: 14,1,4
		267 = 195 + 72	**264** = 190 + 74	4: 28,1,15,3
(677,2048,254)	891	316 = 246 + 70	316 = 246 + 70	3: 22,0,5
		315 = 236 + 79	**315** = 232 + 82	4: 30,1,14,2
(821,4096,510)	1286	472 = 375 + 97	468 = 371 + 97	3: 20,1,6
		452 = 339 + 113	**438** = 320 + 118	4: 34,4,27,5
NTRU-Prime				
(653,4621,288)	925	321 = 254 + 67	320 = 254 + 66	3: 16,2,6
		316 = 234 + 82	**313** = 231 + 82	4: 36,1,19,4
(761,4591,286)	1003	348 = 277 + 71	348 = 277 + 72	3: 22,1,6
		345 = 262 + 83	**344** = 258 + 86	4: 36,1,17,2
(857,5167,322)	1131	390 = 311 + 79	390 = 312 + 78	3: 23,1,6
		387 = 302 + 85	**385** = 297 + 88	4: 33,1,14,2
BLISS I+II				
(512,12289,154)	597	**208** = 171 + 37	208 = 171 + 37	3: 15, 3, 0
GLP I				
(512,8383489,342)	802	275 = 242 + 33	273 = 240 + 33	3: 12,1,4
		260 = 222 + 39	**245** = 206 + 40	4: 22,3,19,4

Table 6: Comparison Rep-1 vs. Rep-2

The params-column with $3 : 14, 1, 4$ gives the parameters $\epsilon_{10}^{(1)}, \epsilon_{21}^{(1)}, \epsilon_{10}^{(2)}$. All other depth-3 parameters were always 0 in the optimization. In our example, we put on level 1 an amount of 14 additional ± 1 (corresponding to $\epsilon_{10}^{(1)}$), another amount of 1 additional ± 2 (corresponding to $\epsilon_{21}^{(1)}$), and on level 2 an amount of 4 additional ± 1 (corresponding to $\epsilon_{10}^{(2)}$). In the level-4 params column we provide parameters $\epsilon_{10}^{(1)}, \epsilon_{21}^{(1)}, \epsilon_{10}^{(2)}, \epsilon_{10}^{(3)}$, all others were 0.

As a function of the search space size $\mathcal{S}$ we get for all NTRU encryption schemes and BLISS non-asymptotic combinatorial attack complexities around $\mathcal{S}^{0.35}$, for GLP with its large q roughly $\mathcal{S}^{0.3}$. Notice that for all instances the list construction costs T_ℓ are roughly $\mathcal{S}^{\frac{1}{4}}$, as predicted by the asymptotic analysis.

8.3 BLISS with $\mathbf{s} \in \{0, \pm 1, \pm 2\}^n$

The BLISS IV parameter set suggests $(n, q) = (512, 12289)$, where the secret $\mathbf{s}$ has 230 (± 1)-entries and 30 (± 2)-entries. Asymptotically, such a weight distribution yields $\mathcal{S}^{0.21}$ with Rep-2 (adapted to $\mathbf{s} \in \{0, \pm 1, \pm 2\}^n$).

Instance	$\mathcal{S}$[bit]	Rep-2[bit]	$\epsilon_{10}^{(1)}, \epsilon_{12}^{(1)}, \epsilon_{10}^{(2)}, \epsilon_{10}^{(3)}$
BLISS IV (512,12289,230,30)	890	**281** = 212 + 69	4: 43, 3, 19, 4

Table 7: Rep-2 on secrets $\mathbf{s} \in \{0, \pm 1, \pm 2\}^n$

Non-asymptotically, we achieve $\mathcal{S}^{0.31}$ from Table 7, where we obtain list construction complexity $T_\ell = 212$ bit and guessing complexity $T_g = 69$ bit. This is not too far from the conservative 192 bit BLISS IV security estimate via the lattice Hybrid attack.

9 Small Memory Versions

Our new Meet-LWE attack has quite large memory consumption. For all instantiations of Meet-LWE in the previous sections the memory consumption is (almost) as large as the list construction time T_ℓ, i.e. roughly $\mathcal{S}^{\frac{1}{4}}$.

We show in this section that our representations Rep-0, Rep-1, Rep-2 used together with Howgrave-Graham's algorithm from Section 4 admit small memory versions and simple time-memory tradeoffs. Using Howgrave-Graham's original algorithm instead of Meet-LWE has the advantage that we do not require any key guessing.

Let us repeat the LWE identity from Equation (5)

$$A\mathbf{s}_1 + \mathbf{e}_1 = \mathbf{b} - A\mathbf{s}_2 + \mathbf{e}_2 \text{ for some binary } \mathbf{e}_1, \mathbf{e}_2 \in \{0, 1\}^n.$$

We follow the general van Oorschot and Wiener strategy [vW99] that turns a MitM attack memory-less. This strategy was already applied by van Vredendaal [vV16] to Odlyzko's MitM attack from Section 3.

Let D be the search space for $\mathbf{s}_1, \mathbf{s}_2$ with size $\mathcal{S}$. Let us define the two functions f_1, f_2 with domain D as

$$f_1 : \mathbf{s}_1 \mapsto \pi_k(\ell(A\mathbf{s}_1)) \text{ and } f_2 : \mathbf{s}_2 \mapsto \pi_k(\ell(\mathbf{b} - A\mathbf{s}_2)).$$

Here $\ell : \mathbb{Z}_q^n \to \{0,1,2\}^n$ is Odlyzko's hash function from Equation (2), changed to ternary labels in the canonical manner (by equipartitioning $\mathbb{Z}_q$), such that it has range size $3^n > \mathcal{S}$. Further, π_k is our projection function from Equation (6), where we choose $k = \lceil \log_3(\mathcal{S}) \rceil$. Thus, the range of f_1, f_2 is approximately of size $\mathcal{S}$.

Moreover we use an encoding function $h : \{0,1,2\}^k \to D$ that encodes the arguments $f_i(\cdot)$ back to D, such that we can iterate functions f_i.

With a cycle-finding algorithm we find in time roughly $\sqrt{S}$ a collision $\mathbf{s}_1, \mathbf{s}_2$ between f_1, f_2, i.e. $f_1(\mathbf{s}_1) = f_2(\mathbf{s}_2)$. This implies

$$\pi_k(\ell(A\mathbf{s}_1)) = \pi_k(\ell(\mathbf{b} - A\mathbf{s}_2)). \tag{10}$$

Let $\mathbf{s}$ be the desired LWE secret key, and let $(\mathbf{s}_1, \mathbf{s}_2)$ be a representation of $\mathbf{s}$, i.e. $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$. By definition of ℓ, any representation $(\mathbf{s}_1, \mathbf{s}_2)$ satisfies Equation (10). We call a collision $(\mathbf{s}_1, \mathbf{s}_2)$ *good* iff $(\mathbf{s}_1, \mathbf{s}_2)$ is a representation of $\mathbf{s}$. Let R be the number of representations of $\mathbf{s}$.

By A's randomness, the functions f_1, f_2 should behave like random functions. Therefore, we expect that there exist roughly $\mathcal{S}$ collisions in total between f_1 and f_2. This in turn implies that we obtain a good collision with probability $p = \frac{R}{\mathcal{S}}$. Since finding any collision takes time $\sqrt{\mathcal{S}}$ we expect overall running time

$$T = \sqrt{\mathcal{S}} p^{-1} = \sqrt{\mathcal{S}} \cdot \frac{\mathcal{S}}{R} = \frac{\mathcal{S}^{\frac{3}{2}}}{R}. \tag{11}$$

van Vredendaal [vV16] chooses Odlyzko's search space size $\mathcal{S} = \binom{n}{w/2, w/2, \cdot}$ with a unique representation of the solution. Thus, she obtains a polynomial memory algorithm with run time $\mathcal{S}^{\frac{3}{2}}$.

But from Equation (11) we minimize T by increasing the number R of representations, as long as $\mathcal{S}$ does not grow too fast. We show in the following that this tradeoff between $\mathcal{S}$ and R pays off for our representations Rep-0, Rep-1, Rep-2. To this end, we simply set $\mathcal{S} = \mathcal{S}^{(1)}, R = R^{(1)}$ using the expressions for $\mathcal{S}^{(1)}, R^{(1)}$ derived in Sections 6 to 8.

We obtain asymptotic runtimes $2^{c(\omega)(n+o(n))}$ for which we state the run time exponents $c(\omega)$ in Table 8. The optimization parameters for Rep-1, Rep-2 are given in the last two columns.

In comparison to the previously best results from [vV16], the runtime exponents drop by 10% to 15%.

We may also utilize the well-known time-memory tradeoff provided by Parallel Collision Search (PCS) [vW99]. Let $T_c = \sqrt{\mathcal{S}}$ be the time to find a single collision. Then with PCS using memory M we find M collisions in time $T = \sqrt{M} T_c$. Since

ω	Odl. [vV16]	Rep-0	Rep-1	Rep-2	$\bar{\epsilon}^{(1)}$	$\bar{\epsilon}^{(1)}_{10}$	$\bar{\epsilon}^{(1)}_{21}$
0.3	0.886	0.834	**0.772**	0.772	0.032	0.032	0.000
0.375	0.997	0.951	**0.858**	0.858	0.045	0.045	0.000
0.441	1.073	1.031	0.918	**0.918**	0.055	0.056	0.001
0.5	1.125	1.092	0.964	**0.962**	0.061	0.061	0.001
0.621	1.184	1.186	1.043	**1.038**	0.060	0.062	0.003
0.668	1.189	1.211	1.070	**1.064**	0.056	0.058	0.004

Table 8: Asymptotic complexities for polynomial memory LWE key search.

we need a total of $p^{-1} = \frac{\mathcal{S}}{R}$ collisions (for finding a good collision), using memory $M \leq p^{-1}$ we obtain the time-memory tradeoff

$$T = \frac{p^{-1}}{M}\sqrt{M}T_c = \frac{S^{\frac{3}{2}}}{\sqrt{M}R}.$$

E.g. if we use full memory $M = p^{-1} = \frac{\mathcal{S}}{R}$, we obtain run time $T = \frac{\mathcal{S}}{\sqrt{R}}$, which reproduces Howgrave-Graham's run time formula from Section 4, albeit with different (better) values of $\mathcal{S}$ and R.

10 Hybrid Attack

The currently best known attack on LWE-type cryptosystems with ternary keys is a combination of lattice reduction and MitM known as the *Hybrid attack*, due to Howgrave-Graham [How07]. Here, we give only a brief outline of the attack. We refer the reader to [How07, Wun19] for more details.

For ease of exposition we describe only the *Plain Hybrid* attack, a dimension reduction method [MS01] that combines lattice reduction with Brute-Force key guessing. Howgrave-Graham [How07] then showed that Brute-Force can be replaced by MitM at the cost of some (often quite smallish) success probability [Wun19].

We write $A = (A_1, A_2) \in \mathbb{Z}_q^{n\times k} \times \mathbb{Z}_q^{n\times(n-k)}$ for some $0 \leq k \leq n$, and the LWE key equation as $A_1\mathbf{s}_1 = \mathbf{b} - A_2\mathbf{s}_2 + \mathbf{e}$. In *Plain Hybrid* we enumerate all candidates $(\mathbf{s}_2, \mathbf{b} - A_2\mathbf{s}_2)$. Define the target $\mathbf{t}(\mathbf{s}_2) := \mathbf{b} - A_2\mathbf{s}_2$.

Further, we define a lattice L by the $(n+k)$-dimensional lattice basis

$$B = (\mathbf{b}_1|\mathbf{b}_2|\ldots|\mathbf{b}_{n+k}) = \begin{bmatrix} qI_n & A_1 \\ 0 & I_k \end{bmatrix}.$$

Let $\mathbf{s}_2$ be the correct key guess with target $\mathbf{t}(\mathbf{s}_2)$. L contains the vector $\mathbf{v} = B \cdot (\mathbf{k}, \mathbf{s}_1) = (\mathbf{t}(\mathbf{s}_2) + \mathbf{e}, \mathbf{s}_1)$ for some suitably chosen $\mathbf{k} \in \mathbb{Z}^n$. Thus $\mathbf{v}$ is close—in distance $(\mathbf{e}, \mathbf{s}_1)$—to the known target vector $(\mathbf{t}(\mathbf{s}_2), \mathbf{0})$. Hence $\mathbf{v}$ can be recovered by using Babai's *Nearest Plane* algorithm [Bab86], provided we have a sufficiently reduced basis B. Solving such a close vector instance is called Bounded Distance Decoding.

The *Plain Hybrid* approach now balances the cost for enumerating $\mathbf{s}_2$ with the lattice reduction cost for recovering $\mathbf{s}_1$.

Problems when Applying our MitM. It is tempting to replace the key search for $\mathbf{s}_2$ by our MitM approach. However, notice that we have to enumerate *all* vectors $\mathbf{s}_2$. This can surely be done by the MitM attacks from Sections 3 and 4.

Instead, the strength of our MitM attack is that we do not enumerate all potential keys, but only those that fulfill the LWE key equation on r coordinates. However, we cannot use the LWE key equation anymore for our projected subkey $\mathbf{s}_2$.

Work-around 1: One may introduce *additional* representations $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ for the lattice and guessing part by letting $\mathbf{s}_1, \mathbf{s}_2$ overlap. But our computations so far indicate that the cost of increasing $\mathbf{s}_1$'s dimension—and therefore of lattice reduction—is not compensated by the decrease to enumerate $\mathbf{s}_2$ with our improved MitM.

Work-around 2: One may define $\mathbf{s}_1, \mathbf{s}_2$ of full length n, but with *different weight*, again introducing additional representations. Especially, by choosing smaller weight for $\mathbf{s}_2$ one can balance the cost of lattice reduction and enumeration. Our computations so far show that such a *Weight Hybrid* is only slightly better than pure lattice reduction.

We leave it as an open problem whether list construction-type algorithms can improve lattice hybrid attacks. Another interesting question is whether we can omit key guessing in our Meet-LWE by using Nearest Neighbor techniques as in [Laa15, MO15].

References

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

ACF+14. Martin Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for lwe problems. 2014.

Bab86. László Babai. On lovász'lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

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, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

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, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

BCLvV. DJ Bernstein, C Chuengsatiansup, T Lange, and C van Vredendaal. Ntru prime: Round 2 specification (2019).

BCLvV17. Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. Ntru prime: reducing attack surface at low cost. In *International Conference on Selected Areas in Cryptography*, pages 235–260. Springer, 2017.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM.

BDK+17. Joppe W Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. *Cryptology ePrint Archive*, (20180716: 135545), 2017.

BG14. Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337, Wollongong, NSW, Australia, July 7–9, 2014. Springer, Heidelberg, Germany.

BGPW16. Johannes A. Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 24–43, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg, Germany.

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, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

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. 2019.

dBDJW18. Koen de Boer, Léo Ducas, Stacey Jeffery, and Ronald de Wolf. Attacks on the AJPS Mersenne-based cryptosystem. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 101–120, Fort Lauderdale, Florida, United States, April 9–11 2018. Springer, Heidelberg, Germany.

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, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

DN12. Léo Ducas and Phong Q. Nguyen. Faster Gaussian lattice sampling using lazy floating-point arithmetic. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 415–432, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

DRV20. Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. (one) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2020.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

GJS15. Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

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, Leuven, Belgium, September 9–12, 2012. Springer, Heidelberg, Germany.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

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

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, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

HKL18. Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. Speed-ups and time-memory trade-offs for tuple lattice sieving. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 407–436, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

HNP+03. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 226–246, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

How07. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

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

KF15. Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 43–62, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

Laa15. Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

MO15. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 203–228, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

MP13. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

MS01. Alexander May and Joseph H Silverman. Dimension reduction methods for convolution modular lattices. In *International Cryptography and Lattices Conference*, pages 110–125. Springer, 2001.

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

ntr08. 1363.1-2008 - ieee standard specification for public key cryptographic techniques based on hard problems over lattices, 2008.

NV08. Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

Reg03. Oded Regev. New lattice based cryptographic constructions. In *35th ACM STOC*, pages 407–416, San Diego, CA, USA, June 9–11, 2003. ACM Press.

SS11. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

vV16. Christine van Vredendaal. Reduced memory meet-in-the-middle attack against the ntru private key. *LMS Journal of Computation and Mathematics*, 19(A):43–57, 2016.

vW99. Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, January 1999.

WMM13. Hong Wang, Zhi Ma, and ChuanGui Ma. An efficient quantum meet-in-the-middle attack against ntru-2005. *Chinese Science Bulletin*, 58(28):3514–3518, 2013.

Wun19. Thomas Wunderer. A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack. *Journal of Mathematical Cryptology*, 13(1):1–26, 2019.

A Appendix: Detailed Rep-2 Analysis

A.1 Level-1 Search Space and Representations.

Let us introduce some notation. Let $n_0^{(j)}, n_1^{(j)}, n_2^{(j)}$ be the number of 0, ±1 and ±2 on level j, respectively. We denote by $\epsilon_{k\ell}^{(j)}$ the number of additional $\pm k$ on level j that represent $\pm\ell$, see also Table 9.

Parameter	$\epsilon_{10}^{(j)}$	$\epsilon_{21}^{(j)}$	$\epsilon_{20}^{(j)}$
Adds	±1	±2	±2
Represents	$(-1)+1$	$(-2)+1$	$(-2)+2$
	$1+(-1)$	$(-1)+2$	$2+(-2)$
		$1+(-2)$	
		$2+(-1)$	

Table 9: Semantics of optimization parameters

E.g. we denote by $\epsilon_{10}^{(j)}$ be the number of additional ±1 that we add on level j to realize the representations $(-1)+1$ and $1+(-1)$. Hence, $\epsilon_{10}^{(j)}$ plays the role of $\epsilon^{(j)}$ from Section 7. On level 1, we already have $n_1^{(0)}/2$ many ±1 by splitting the weight of $\mathbf{s}$ evenly, thus we obtain $n_1^{(1)} = n_1^{(0)}/2 + \epsilon_{10}^{(1)}$. Moreover, on level 1 we obtain an overall number of $n_2^{(1)} = \epsilon_{21}^{(1)} + \epsilon_{20}^{(1)}$ many ±2.

Let us explicitly construct $\mathbf{s} = \mathbf{s}_1^{(1)} + \mathbf{s}_2^{(1)} \in \mathcal{T}^n(w/2)$. We represent each level-0 coordinate in the following number of ways on level 1, as depicted in Table 10. Notice from Table 10 that we construct in total $n_1^{(0)} = 2\big(\epsilon_{21}^{(1)} + \frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}\big)$ many ±1 and $n_0^{(0)} = n - 2n_1^{(0)}$ many 0, the desired weight distribution of $\mathbf{s}$.

	$(-2)+1$	$(-1)+0$	$0+(-1)$	$1+(-2)$
(-1)	$\epsilon_{21}^{(1)}$	$\frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}$	$\frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}$	$\epsilon_{21}^{(1)}$

	$(-1)+2$	$0+1$	$1+0$	$2+(-1)$
1	$\epsilon_{21}^{(1)}$	$\frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}$	$\frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}$	$\epsilon_{21}^{(1)}$

	$(-2)+2$	$(-1)+1$	$0+0$	$1+(-1)$	$2+(-2)$
0	$\epsilon_{20}^{(1)}$	$\epsilon_{10}^{(1)}$	$n - 2n_1^{(0)} - 2\epsilon_{10}^{(1)} - 2\epsilon_{20}^{(1)}$	$\epsilon_{10}^{(1)}$	$\epsilon_{20}^{(1)}$

Table 10: Amount of different level-1 representations

From Table 10 we can also easily check the level-1 weight distribution of $\mathbf{s}_1^{(1)}, \mathbf{s}_2^{(1)}$. We have

$$\begin{aligned} n_2^{(1)} &= \epsilon_{21}^{(1)} + \epsilon_{20}^{(1)} \\ n_1^{(1)} &= \frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)} + \epsilon_{21}^{(1)} + \epsilon_{10}^{(1)} = \frac{n_1^{(0)}}{2} + \epsilon_{10}^{(1)} \\ n_0^{(1)} &= 2\left(\frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}\right) + n - 2n_1^{(0)} - 2\epsilon_{10}^{(1)} - 2\epsilon_{20}^{(1)} \\ &= n - n_1^{(0)} - 2(\epsilon_{10}^{(1)} + \epsilon_{21}^{(1)} + \epsilon_{20}^{(1)}) \end{aligned}$$

Notice that we get the correct terms for $n_2^{(1)}, n_1^{(1)}$ from above, and we verify that $2n_2^{(1)} + 2n_1^{(1)} + n_0^{(1)} = n$, as desired.

We obtain level-1 search space size

$$S^{(1)} = \binom{n}{n_2^{(1)}, n_2^{(1)}, n_1^{(1)}, n_1^{(1)}, n_0^{(1)}}.$$

Using Table 10, the number of level-1 representations is

$$R^{(1)} = \binom{n_1^{(0)}}{\epsilon_{21}^{(1)}, \epsilon_{21}^{(1)}, \frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}, \frac{n_1^{(0)}}{2} - \epsilon_{21}^{(1)}}^2 \cdot \binom{n - 2n_1^{(0)}}{\epsilon_{20}^{(1)}, \epsilon_{20}^{(1)}, \epsilon_{10}^{(1)}, \epsilon_{10}^{(1)}, \cdot},$$

where the first term represents the number of representations of ± 1, and the second term the number of representations of 0.

A.2 Level-j lists

Let d be the depth of our search tree T. For all levels $2 \le j \le d$ we represent vectors $\mathbf{s}_i^{(j-1)} \in \{\pm 2, \pm 1, 0\}^n$ as sums of vectors from $\{\pm, 2, \pm 1, 0\}^n$. In addition to the level-1 representations from Equation (9), we obtain the following representations of ± 2-coordinates in $\mathbf{s}_i^{(j-1)}$:

$$\begin{aligned} (-2) = (-2) + 0 = (-1) + (-1) = 0 + (-2) \\ 2 = 0 + 2 = 1 + 1 = 2 + 0. \end{aligned} \tag{12}$$

We denote by $\epsilon_{22}^{(j)}$ the additional ± 2 on level j for the representations $2 + 0$, $0 + 2$, $(-2) + 0$ and $0 + (-2)$. Thus, the level-1 representations from Table 10 generalize to those in Table 11.

We see in Table 11 that the additional ± 1 for the new representations $1 + 1$ and $(-1) + (-1)$ from Equation (12) are determined as $n_2^{(j-1)} - 2\epsilon_{22}^{(j)}$ (and thus do not require yet another optimization parameter).

$$
\begin{array}{c|ccc}
 & (-2)+0 & (-1)+(-1) & 0+(-2) \\
(-2) & \epsilon_{22}^{(j)} & n_2^{(j-1)} - 2\epsilon_{22}^{(j)} & \epsilon_{22}^{(j)}
\end{array}
$$

$$
\begin{array}{c|ccc}
 & 0+2 & 1+1 & 2+0 \\
2 & \epsilon_{22}^{(j)} & n_2^{(j-1)} - 2\epsilon_{22}^{(j)} & \epsilon_{22}^{(j)}
\end{array}
$$

$$
\begin{array}{c|cccc}
 & (-2)+1 & (-1)+0 & 0+(-1) & 1+(-2) \\
(-1) & \epsilon_{21}^{(j)} & \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)} & \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)} & \epsilon_{21}^{(j)}
\end{array}
$$

$$
\begin{array}{c|cccc}
 & (-1)+2 & 0+1 & 1+0 & 2+(-1) \\
1 & \epsilon_{21}^{(j)} & \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)} & \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)} & \epsilon_{21}^{(j)}
\end{array}
$$

$$
\begin{array}{c|ccccc}
 & (-2)+2 & (-1)+1 & 0+0 & 1+(-1) & 2+(-2) \\
0 & \epsilon_{20}^{(j)} & \epsilon_{10}^{(j)} & n - 2n_1^{(j-1)} - 2n_2^{(j-1)} - 2\epsilon_{10}^{(j)} - 2\epsilon_{20}^{(j)} & \epsilon_{10}^{(j)} & \epsilon_{20}^{(j)}
\end{array}
$$

Table 11: Amount of different level-j representations

From Table 11 we check the desired level-j weight distribution for all $\mathbf{s}_i^{(j)}$ as

$$
\begin{aligned}
n_2^{(j)} &= \epsilon_{22}^{(j)} + \epsilon_{21}^{(j)} + \epsilon_{20}^{(j)} \\
n_1^{(j)} &= n_2^{(j-1)} - 2\epsilon_{22}^{(j)} + \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)} + \epsilon_{21}^{(j)} + \epsilon_{10}^{(j)} \\
&= \frac{n_1^{(j-1)}}{2} + \epsilon_{10}^{(j)} + n_2^{(j-1)} - 2\epsilon_{22}^{(j)} \\
n_0^{(1)} &= 2\epsilon_{22}^{(j)} + 2\left(\frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)}\right) + n - 2n_1^{(j-1)} - 2n_2^{(j-1)} - 2\epsilon_{10}^{(j)} - 2\epsilon_{20}^{(j)} \\
&= n - n_1^{(j-1)} - 2n_2^{(j-1)} + 2\left(\epsilon_{22}^{(j)} - \epsilon_{10}^{(j)} - \epsilon_{21}^{(j)} - \epsilon_{20}^{(j)}\right). \qquad (13)
\end{aligned}
$$

We also verify that $2n_2^{(j)} + 2n_1^{(j)} + n_0^{(j)} = n$.
Using Equation (13) we obtain level-j search space size

$$
S^{(j)} = \binom{n}{n_2^{(j)}, n_2^{(j)}, n_1^{(j)}, n_1^{(j)}, n_0^{(j)}}.
$$

Moreover, from Table 11 we derive the number of level-j representations as

$$
R^{(j)} = \binom{n_2^{(j-1)}}{\epsilon_{22}^{(j)}, \epsilon_{22}^{(j)}, .}^2 \binom{n_1^{(j-1)}}{\epsilon_{21}^{(j)}, \epsilon_{21}^{(j)}, \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)}, \frac{n_1^{(j-1)}}{2} - \epsilon_{21}^{(j)}}^2 \cdot \binom{n_0^{(j-1)}}{\epsilon_{20}^{(j)}, \epsilon_{20}^{(j)}, \epsilon_{10}^{(j)}, \epsilon_{10}^{(j)}, .},
$$

where the first term represents the number of representations of ± 2, the second term the number of representations of ± 1, and the last term the number of representations of 0.

This results in level-j list sizes

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

The level-d lists are constructed in a Meet-in-the-Middle fashion and are therefore of size $L^{(d)} = \sqrt{S^{(d-1)}}$.

A.3 Run Time.

The construction of level-d lists takes time $T^{(d)} = L^{(d)} = \sqrt{S^{(d-1)}}$.

Analogous to the run time analysis in Section 7, we define $r^{(d)} = 0$ and construct every level-j list for $0 < j < d$ in time

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

As in Section 7, we have $T^{(0)} = \frac{(L^{(1)})^2}{2^{n-r^{(1)}}}$.

The total run time T and memory consumption M is then

$$T = \max_{i=0,\ldots,d}\{T^{(i)}\} \text{ and } M = \max_{i=1,\ldots,d}\{L^{(i)}\}.$$