How to Meet Ternary LWE Keys on Babai's Nearest Plane

Minki Hhan* Jiseung Kim† Changmin Lee‡ Yongha Son§

October 27, 2022

Abstract

A cryptographic primitive based on the Learning With Errors (LWE) problem with its variants is a promising candidate for the efficient quantum-resistant public key cryptosystem. The recent schemes use the LWE problem with a small-norm or sparse secret key for better efficiency. Such constraints, however, lead to more tailor-made attacks and thus are a trade-off between efficiency and security. Improving the algorithm for the LWE problem with the constraints thus has a significant consequence in the concrete security of schemes.

In this paper, we present a new hybrid attack on the LWE problem. This new attack combines the primal lattice attack and an improved MitM attack called Meet-LWE, answering an open problem posed by May [Crypto'21].

According to our estimation, the new hybrid attack performs better than the previous attacks for the LWE problems with a sparse ternary secret key, which plays the significant role for the efficiency of fully homomorphic encryption schemes.

In terms of the technical part, we generalize the Meet-LWE algorithm to be compatible with Babai's nearest plane algorithm. As a side contribution, we remove the error guessing step in Meet-LWE, resolving another open question.

Keywords: LWE, Hybrid lattice attack, Meet-in-the-middle algorithm

1 Introduction

The (search) learning with errors (LWE) problem [49] asks to find the secret key s given $(A, b) \in \mathbb{Z}_q^{m\times(n+1)}$ such that $b = As + e \bmod q$, where e is a small noisy integer vector. The LWE problem and its variants have been used as the security ground for lattice-based cryptography with plenty of applications [7,19,30,31,44]. The hardness of the LWE problem is well established by the reduction from the worst-case lattice problems. This hardness provides strong confidence in the security of LWE-based cryptosystems. However, this so-called provable security is only guaranteed on a somewhat restricted parameter range, where the schemes have poor efficiency. Most practical LWE-based schemes have used the other parameters for efficiency, far from the provable range, and concrete security is measured against known attacks.

Many LWE-based cryptosystems use the binary/ternary secret/error vectors [21, 27, 32] or even sparse vectors [12] for the efficient implementation, or for the (perfect) correctness of decryption [28]. Furthermore, fully homomorphic encryption (FHE) schemes take advantage of the small norm of secret keys for efficiency. Indeed, most FHE libraries use or support the sparse secret keys [1,2,4,5]; in particular, HEAAN and Lattigo explicitly use the Hamming weight 64 and 192, respectively.

Of course, using a sparse small secret vector opens new possibilities in the adversary's view: The number of candidates of such a sparse small vector is much smaller than that of small or uniform secret keys.

*minkihhan@kias.re.kr. KIAS
†jiseungkim@jbnu.ac.kr. Jeonbuk National University
‡changminlee@kias.re.kr. KIAS
§yongha.son@samsung.com. Samsung SDS

The combinatorial attacks have exploit exactly this feature by directly guessing the secret key space of size S. Beside the exhaustively searching secret key that takes (roughly) S time, the Meet-in-the-middle (MitM) strategy due to Odlyzko [36] is shown to find the secret key in time $S^{0.5}$, and had remained the best algorithm up to recently.

Last year, a new combinatorial attack called Meet-LWE is presented by May [46], based on the representation technique in the subset sum-type problems [15,17,38]. The Meet-LWE attack takes asymptotically S^c time to solve the LWE problem for $c < 0.5$, heuristically $c \approx 0.25$. However, the current practical LWE parameters use a high dimensional vector as secret key so that the search space size S is extremely large. It turns out that the combinatorial algorithms, even including Meet-LWE, themselves cannot beat the lattice attacks (e.g., [10,43]) in the practical parameter setting.

The hybrid attack framework has leveraged lattice reduction algorithms to take advantage of both sides. This attack first reduces the dimension of search space as the first step; then, the combinatorial strategies are applied to solve the remaining part with a much smaller search space. This line of works has been initiated by the seminal work of Howgrave-Graham [37], which can be understood as a combination of the lattice reduction and the MitM algorithm.

The hybrid strategies [37, 47, 50, 52] are presumably the best known attack for the LWE problem with sparse ternary keys. As Meet-LWE improves over Odlyzko's MitM, the hybrid attack that combines the lattice reduction algorithm and Meet-LWE is apparently expected to have some implications on the hardness of LWE. This combination already was considered in the original paper, but it appears that the direct application of Meet-LWE failed [46, Section 10]. This lattice and Meet-LWE hybrid attack remains as a central open problem in this landscape of LWE attacks.

1.1 Our Contribution

We present a new hybrid attack against the LWE problem that combines the primal lattice attack and Meet-LWE, resolving the open question posed in [46]. To this end, we generalize the Meet-LWE algorithm to be compatible with Babai's nearest plane algorithm. As a side contribution, we remove the error guessing step in Meet-LWE, resolving another open problem [46].[1] It requires several technically non-trivial considerations, whose overview is presented in Section 2.

We also estimate the concrete complexity of our attack based on lattice-estimator [3] with some modifications. According to the estimation, our new attack is stronger than the previous attacks for the sparse small key LWE parameter, and has an actual impact against currently deployed FHE parameters in recent works [18, 41, 42], by showing that the parameters in those papers are below the claimed security. Our attack also lowers the security of some PQC schemes [6, 13] which takes highly sparse secret keys. However, this sparse choice is considered somewhat deprecated now, and the attack has little impact on currently deployed PQC schemes, especially on the finalists of NIST PQC standardization. We refer to Table 1 and Section 6.1 for a numerical estimation for the readers.

1.2 Discussions

Necessity of sparse ternary/binary secrets. As several attacks against sparse keys have been continuously reported [23,24] (including ours), one may wonder the benefit of sparse keys; if the benefit is not substantial, it is tempting to exclude sparse keys for LWE-based scheme parameterizations for preventing the potential future stronger attacks exploiting this feature. We stress that, at least in most FHE schemes, sparse keys still play a significant role on the performance of so-called *bootstrapping* operations, despite worse parameterization due to the sparse key tailored attacks. Although a line of works [41,42] has reported the improvements in the efficiency of bootstrapping even for the non-sparse key [18], the sparse-key bootstrapping procedures still perform much better in practice and have better asymptotic complexity.

[1] Strictly speaking, Kirshanova and May independently suggested a similar approach, but we argue their algorithm is problematic. For more detailed discussions, see Appendix A.

Dual hybrid attacks. There is another line of hybrid attack strategy called *dual hybrid* [8, 23, 33, 45]. It also comes from the combination of the lattice reduction and searching strategy of a fraction of the secret key, but with a large difference in a way to construct the target lattice. This difference enables dual strategy to leverage another searching strategy. In particular, the most recent dual hybrid attacks [33, 45] combine the dual lattice strategy with a novel brute-force search using the Fourier distinguisher [29] on the secret key fraction. For the dimension d of the searching partial secret key space and the number S of partial secret keys, the dual-Fourier hybrid attack [33] takes $O(\min(2^d, |S|))$ time complexity. Since our attack takes about S^c time for $c \approx 0.25$ to execute Meet-LWE part, our attack would be better as long as $|S|^c < 2^d$. While the accurate comparison is extremely tough due to several other issues, this inequality justifies that our algorithm outperforms when a sparse secret key is employed.

Further related works. Wunderer [52] and Nguyen [47] pointed out several issues in literature's primal-MitM hybrid attack [37] estimation, and provided refined analysis. Our analysis follows their analysis when required, and also reflects most issues pointed out by them. In fact, the main issue raised by [52] was the ignorance of *admissible* probability in the attack complexity estimation. We see the similar situation when analyzing our algorithm, and we faithfully reflect the probability in estimation. We use the torus LSH suggested in [47] with a new analysis for our setting.

Concurrently, a hybrid of dual and Meet-LWE attack suggested by [16]. This attack is a rather straightforward combination of two attacks, as the dual lattice attack can be used as a dimension-error trade-off of LWE [8, 23]. The combination of dual lattice reduction and Meet-LWE in [16] still requires a huge overhead in the exhaustive error enumeration step.[2] We believe that applying our near-collision finding algorithm in Section 4 to the dual and Meet-LWE hybrid attack will improve the efficiency, let alone the dual-MitM hybrid attack [23].

Future directions. The above discussion naturally motivates the possibility of other combinations of different algorithms. For example, can we construct the primal-Fourier hybrid? Constructing an efficient hybrid algorithm of combinatorial attacks is also an interesting question, such as the Fourier-MitM distinguisher. Any such hybrid attack will be competitive for some LWE instances if it appears possible. The other direction is to explore the quantum version of hybrid attacks and examine their costs for the post-quantum parameters.

Providing a more robust analysis can be the opposite direction of research. In fact, the hybrid attacks [37, 47, 50, 52] including ours, rely on several heuristic assumptions, and so do representation technique-based algorithms [15, 17, 26, 38, 46]. While we suggest some explanations for our assumptions, validating them rigorously as in [47, 52] or finding the minimal assumption for the algorithms would be an interesting problem.

2 Technical Overview

We provide an overview of our attack along with the lattice background in this section. In Section 2.1 presents preliminaries, and Section 2.2 gives a high-level description of Meet-LWE. The overview of our generalization and technical contributions are summarized in Section 2.3.

2.1 Preliminaries

Notations. For a positive integer n, $[n]$ denotes the set $\{1, ..., n\}$. The elements of $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ are identified with the representatives in $\mathbb{Z} \cap [-q/2, q/2)$. We write the uniform distribution over a set S by $\mathcal{U}(S)$, and the Gaussian distribution of standard deviation σ by $\mathcal{G}_\sigma$. For a (ordered) basis $B = (b_1, ..., b_n)$ of vector space (or lattice) with column vectors b_i for $i \in [n]$, we identify the basis B with the matrix $(b_1|b_2|...|b_n)$ and vice versa. We let $B^* = (b_1^*, ..., b_n^*)$ be the Gram-Schmidt orthogornalized basis of B and $\tilde{B}^* = (\tilde{b_1^*}, ..., \tilde{b_n^*})$ be the

[2] In other words, the current dual and Meet-LWE hybrid attack is simply prevented by slightly increasing the standard deviation of error from $\sigma = 3.2$ to, say, $\sigma = 6.4$.

normalized Gram-Schmidt basis of B, i.e. $\tilde{b_i^*} = b_i^*/\|b_i^*\|$. For a matrix $B = (b_1, ..., b_n)$ and a vector v, we write the coordinates of v with respect to $\tilde{B^*}$ by $\tau_B(v) = (\tilde{B^*})^{-1} \cdot v$ to denote the coordinates of vector v with respect to the basis $\tilde{B^*}$. For a matrix $B \in \mathbb{R}^{n\times n}$, the fundamental parallelepiped $\mathcal{P}(B)$ is defined by $\{B \cdot x \ : x \in [-\frac{1}{2}, \frac{1}{2})^n\}$.

Definition 1 ((Search) Learning with error problem)**.** *For a secret key s sampled from a distribution $\mathcal{D}_s$, the LWE problem asks to find the secret key s given a tuple $(A, b = As + e \bmod q) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ where the uniform random matrix A and an error vector e sampled from $\mathcal{D}_e$.*

This paper focuses on the LWE problem with sparse ternary secret key. More precisely, we denote the set of n-dimensional ternary vectors with weight $2w$ splitting evenly in ± 1-entries by

$$\mathcal{T}^n(w) = \{s \in \{\pm 1, 0\}^n \mid s \text{ has } w \ (\pm 1)\text{-nonzero entries respectively}\},$$

and we assume that $\mathcal{D}_s = \mathcal{U}(\mathcal{T}^n(w))$ for some $w > 0$ without special mention. We also consider that the error distribution is sampled from the discrete Gaussian distribution over $\mathbb{Z}$ with the standard deviation σ, denoted by $\mathcal{G}_\sigma$. When an m-dimensional vector is sampled from $\mathcal{G}_\sigma^m$, we sometimes omit the superscript m if it is clear from the context. The discrete Gaussian distribution is occasionally approximated by the *continuous* Gaussian distribution with the same standard deviation.

Lattice backgrounds. A lattice is a discrete subgroup of $\mathbb{R}^n$. For a full-rank matrix $B = (b_1, ..., b_n) \in \mathbb{R}^{n\times n}$, the sets $\mathcal{L}(B)$ denote the lattice defined by $\{v = \sum_{i=1}^n a_i b_i \mid a_i \in \mathbb{Z}\}$. The lattice reduction algorithm takes a basis of lattice $\mathcal{L}$ of rank n as input and returns a short basis of $\mathcal{L}$. Typically, BKZ algorithm is exploited for lattice reduction, which takes a block size β, and outputs a reduced basis B. As a widely used heuristic that represents the degree of basis reduction, the Geometric Series Assumption (GSA) predicts that the Gram-Schmidt norms $\|b_i^*\|$ of the basis B reduced by BKZ-β is estimated as follow for $i = 1, ..., n$,

$$\|b_i^*\| = \delta_0^{-2(i-1)+n} \cdot \det(B)^{\frac{1}{n}},$$

where $\delta_0 = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{1/(2(\beta-1))}$.

2.2 Overview of Meet-LWE

We start with a brief review of original Meet-LWE [46]. Suppose that we are given a matrix M and integer q as inputs, and want to find $s \in \mathcal{T}^n(w^{(0)})$ for some $w^{(0)}$ such that $Ms = e \bmod q$ for a unknown small error vector e.

Core idea of Meet-LWE. Meet-LWE starts from the fact that we may have quite many pairs $(s_1^{(1)}, s_2^{(1)}) \in \mathcal{T}^n(w^{(1)})$ for $w^{(1)} \geq w^{(0)}/2$ such that $s = s_1^{(1)} - s_2^{(1)}$, which we call *representations* of s. In order to exploit the redundancy of representations, it considers the projection map $\pi_r : \mathbb{Z}^m \to \mathbb{Z}^r$ on the first r coordinates to define the following sets with special *constraints* of the form $\pi_r(Ms^{(1)} \bmod q) = e$, precisely

$$\widetilde{\mathcal{T}}_1^n(w^{(1)}) = \{s^{(1)} \in \mathcal{T}^n(w^{(1)}) \mid \pi_r(Ms^{(1)} \bmod q) = e_1\}$$
$$\widetilde{\mathcal{T}}_2^n(w^{(1)}) = \{s^{(1)} \in \mathcal{T}^n(w^{(1)}) \mid \pi_r(Ms^{(1)} \bmod q) = e_2\},$$

where e_1, e_2 are proper guesses for $\pi_r(e)$ such that $\pi_r(e) = e_1 + e_2$. Assuming e_1, e_2 are correctly guessed, an (exact) collision search over $\widetilde{\mathcal{T}}_1^n(w^{(1)}) \times \widetilde{\mathcal{T}}_2^n(w^{(1)})$ suffices to find s, provided that there exists a representation $(s_1^{(1)}, s_2^{(1)})$ such that

$$\pi_r(Ms_1^{(1)} \bmod q) = e_1 \ \textit{and} \ \pi_r(Ms_2^{(1)} \bmod q) = e_2.$$

In fact, the projection dimension r is chosen so that at least one representation $(s_1^{(1)}, s_2^{(1)})$ survives in $\widetilde{\mathcal{T}}_1^n(w^{(1)}) \times \widetilde{\mathcal{T}}_2^n(w^{(1)})$ with high probability.

An extension to higher level. The Meet-LWE algorithm comes from a recursive application this idea, where the depth of recursion is called by *level*. As the underlying idea of recursion, it utilizes the fact that each element of $s^{(1)} \in \widetilde{\mathcal{T}}^{(1)}$ also has several representations $s^{(1)} = s_1^{(2)} - s_2^{(2)}$ for $s_1^{(2)}, s_2^{(2)} \in \mathcal{T}^n(w^{(2)})$ where $w^{(2)} \geq w^{(1)}/2$. Then the above constraint idea can be applied again to define a smaller lists $\widetilde{\mathcal{T}}^n(w^{(2)})$, so that one can recover $\widetilde{\mathcal{T}}^n(w^{(1)})$. In words, the Meet-LWE algorithm is a recursive application of *list reconstruction* using the standard MitM approach.

Error guessing by enumeration. Remark that this strategy succeeds only when the error guessing e_1, e_2 for the level-1 lists are correct, that is, when it holds that $\pi_r(e) = e_1 + e_2$. Regarding this, the original Meet-LWE simply enumerates all possible error candidates e_i. As the projection dimension r is set to be a sub-linear number $O(n/\log n)$, this enumeration is quite affordable and can be asymptotically neglected. In practice, however, this error guessing step takes a huge overhead, and removing this step is another open question posed in [46].

2.3 Generalization of Meet-LWE

A conventional primal hybrid attack for LWE is composed of two stages, where $\mathsf{NP}_B(\cdot)$ denotes the output of Babai's nearest plane (NP) algorithm with respect to a basis B:

1. Convert given LWE instances $(A, b = As' + e' \bmod q)$ into another problem that, given M and B, asks to find s such that $\mathsf{NP}_B(Ms) = e$ for some small e, where s is a part of the LWE secret key s'.
2. Recover the solution of s of $\mathsf{NP}_B(Ms) = e$ using combinatorial approaches.

Our main technical contribution in this formulation is solving the second stage $\mathsf{NP}_B(Ms) = e$ via the Meet-LWE algorithm [46], which is the main target on this overview. For the first conversion, we follow the almost similar procedure to the previous hybrid attacks, whose details are presented in Section 3.

Babai's NP algorithm and matrix modulus. As a preparation, we recall Babai's nearest plane (NP) algorithm and introduce a matrix modulus notation. Writing the Gram-Schmidt basis of B by B^* and the fundamental parallelepiped of B^* by $\mathcal{P}(B^*)$, Babai's NP algorithm takes a target vector t and a basis matrix B of lattice $\mathcal{L}$ as inputs, and outputs $\mathsf{NP}_B(t) \in \mathcal{P}(B^*)$ such that $t - \mathsf{NP}(t)_B \in \mathcal{L}$. Regarding this, we introduce a notation of matrix modulus by

$$t \bmod B := \mathsf{NP}_B(t),$$

which indeed satisfies similar properties to $\bmod q$. See Section 3.1 and Lemma 2 for formal descriptions.

We again note that the primal hybrid attacks [37, 52] are to transform the given problem instance to another problem, which we usually refer the matrix modulus case (of LWE), of the form

$$Ms = e \bmod B \tag{1}$$

for some small e, and to find s using the combinatorial strategy such as the exhaustive search or MitM.

As the previous primal hybrid attacks already developed technical tools for dealing with the search problem over the Babai's nearest plane, one may think that to apply Meet-LWE on the same setting can be analogously done. However, it turns out that there are (at least) two technical problems for (primal,Meet-LWE)-hybrid: The first one is the incompatibility between two different bases when considering the constraints, and the other is the error guessing step. We describe these difficulties below.

Projection for matrix modulus. The constraint of original Meet-LWE is based on the projection map π_r, and implicitly uses the standard basis to take r coordinates. However, we should work with the output of NP algorithm for solving Eq. (1), and it appears that the projection to standard basis and map $\bmod B$ are not compatible well in general. Precisely, it would happen that

$$\pi_r\left(v \bmod B\right) \neq \pi_r(v) \bmod \pi_r(B),$$

where $\pi_r(B)$ is defined by column-wise projection.

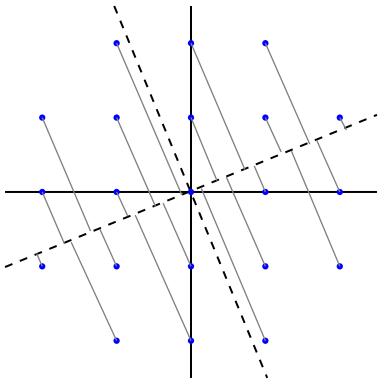

Figure 1: The blue points denote the integer coordinate points with respect to the standard basis. The dotted line represents the axis from B^*, and the gray lines show the projection $\tau_{B,r}$. Different to the projection over standard basis, the number of points remains the same even after projection.

To resolve this issue, we come up with another basis and corresponding projection map that is compatible with Babai's NP algorithm. More precisely, we consider the *Gram-Schmidt* orthogonalized basis B^* and a projection map $\tau_{B,r}$ that takes the *last* r coordinates with respect to B^*-basis coordinate representation. Under this representation, we prove that $\tau_{B,r}$ is well-compatible with the matrix modulus as follows

$$\tau_{B,r}\left(v \bmod B\right) = \tau_{B,r}(v) \bmod \tau_{B,r}(B).$$

We note that the choice of last r coordinates is important, as the above compatibility does not hold for the other choices. For a more detailed description, we refer to Lemma 5.

Approximate constraint. Applying the new projection map on Eq. (1) gives

$$\tau_{B,r}(Ms \bmod B) = \tau_{B,r}(e) \bmod \tau_{B,r}(B).$$

We further can check that $\tau_{B,r}(Ms \bmod B) = \tau_{B,r}(M)s \bmod \tau_{B,r}(B)$ holds. Unfortunately, this equation itself is not sufficient to complete Meet-LWE over matrix modulus because of the error guessing step.

Recall that the original Meet-LWE algorithm performs a brute-force guess on the noise vector $\pi_r(e)$ to find out the solution s. This strategy does not work for the matrix modulus case. Due to the non-orthogonality between the standard basis and B^*, the projection map $\tau_{B,r}$ provide no reduction on the number of $\tau_{B,r}(e)$ candidates, as Figure 1 shows. Therefore, guessing all possible $\tau_{B,r}(e)$ results in an extremely inefficient algorithm.

To circumvent the difficulty from error guessing, we consider an *approximate* constraint for defining the lists as $\tau_{B,r}(Ms^{(1)} \bmod B) \approx 0$. To be precise, we consider a small neighborhood A of origin, and define a level-1 list by

$$\widetilde{\mathcal{T}}^n(w^{(1)}) = \{s^{(1)} \in \mathcal{T}^n(w^{(1)}) \mid \tau_{B,r}(M)s^{(1)} \bmod \tau_{B,r}(B) \in A\}. \tag{2}$$

Similarly to the original Meet-LWE, we choose the projection dimension r and the neighborhood A so that there exists one representation $(s_1^{(1)}, s_2^{(1)})$ of s on expectation such that

$$\tau_{B,r}(M)s_1^{(1)} \in A, \quad \text{and} \quad \tau_{B,r}(M)s_2^{(1)} \in A. \tag{3}$$

Here and below, we omitted $\bmod\ \tau_{B,r}(B)$ for brevity. Our goal is then to find a representation $(s_1^{(1)}, s_2^{(1)})$ that satisfies Eq. (3). At this point, two technical problems remain: 1) The choice of r and A to ensure the existence of such $(s_1^{(1)}, s_2^{(1)})$ in the list with approximate constraint, which requires a more delicate argument

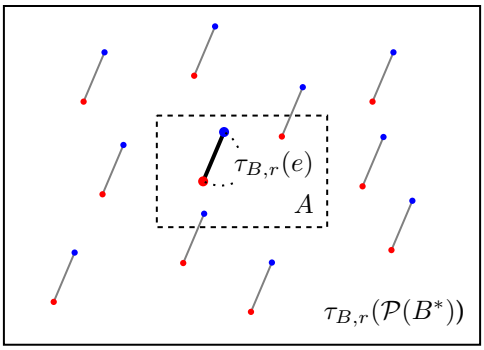

Figure 2: The blue and red points represent $\tau_{B,r}(M)s_1^{(1)}$ and $\tau_{B,r}(M)s_2^{(2)}$ respectively, which are at distance $\tau_{B,r}(e)$. We need a representation $(s_1^{(1)}, s_2^{(2)})$ that makes both points are in A, emphasized by a thick line. Note that it may happens that one point is in the area A, but the other lies outside due to the error $\tau_{B,r}(e)$

than the original Meet-LWE, and 2) an alternative algorithm to find such a pair, provided that the existence, from the list defined in an approximate manner.

The choice of parameters. In the original Meet-LWE, the event that $s_1^{(1)}$ is included in the level-1 list directly implies that $s_2^{(2)}$ is also included in the (opposite) list. However, this is not the case for approximate constraints; for example, for $y = \tau_{B,r}(M)s_2^{(1)} \in A$ and $z = \tau_{B,r}(e)$, their addition $y + z = \tau_{B,r}(M)s_1^{(1)}$ (modulo $\tau_{B,r}(B)$) may not be included in A as Figure 2 shows.[3] Because of this possibility, we should carefully analyze the relation between two events $y \in A$ and $y + z \in A$. This situation is in fact highly similar to the *admissible* property of the context of previous primal hybrids [37,52], especially for z follows some Gaussian distribution. We adapt some parts of corresponding analysis therein.

List reconstruction using locality sensitive hashing. With the appropriate parameter choice of r and A, we expect that there is a representation $(s_1^{(1)}, s_2^{(1)})$ satisfying Eq. (3) in the list $\widetilde{\mathcal{T}}^n(w^{(1)})$. The main observation for finding such a pair without error guessing is that for any representation $s = s_1^{(1)} - s_2^{(1)}$ naturally gives a near-collision pair $Ms_1^{(1)}$ and $Ms_2^{(1)}$ modulo B, that is, the following equation holds

$$Ms_1^{(1)} = Ms_2^{(1)} + e \bmod B. \tag{4}$$

In other words, the problem we are given can be understood as a near-collision search problem from the list, regardless of constraints. This problem is a variant of well-studied nearest neighbor search (NNS) problem, and the original primal-MitM hybrid attack [37] already took this approach. We solve the near-collision finding problem using the locality-sensitive hashing technique adapting the similar result for NNS [35], but with some modifications for our purpose as elaborated in Remark 1. This also can be seen as a generalization of the original approach of Howgrave-Graham [37].

An extension to higher levels. Interestingly, approximate constraints allow us to extend this strategy to higher levels, despite it was originally proposed to remove error guessing. To describe the extension, we recast the problem to find s satisfying Equation (1) as to construct the set

$$\widetilde{\mathcal{T}}^n(w^{(0)}) = \{s \in \mathcal{T}^n(w^{(0)}) \mid Ms \bmod B \approx 0\}.$$

Hence, the aforementioned argument reduces the original problem into constructing

$$\widetilde{\mathcal{T}}^n(w^{(1)}) = \{s^{(1)} \in \mathcal{T}^n(w^{(1)}) \mid \tau_{B,r}(M)s^{(1)} \bmod \tau_{B,r}(B) \approx 0\}$$

[3] In fact, this is related to the flaw of [40], which is described in Appendix A.

for appropriately chosen parameters. By setting $M^{(1)} := \tau_{B,r}(M)$ and $B^{(1)} := \tau_{B,r}(B)$, the two lists are essentially the same, except that 1) the list $\widetilde{\mathcal{T}}^n(w^{(1)})$ has many elements while $\widetilde{\mathcal{T}}^n(w^{(0)})$ is a singleton set, and 2) the error term hides in $\approx$ of $\widetilde{\mathcal{T}}^n(w^{(1)})$ is determined by A. It turns out that the LSH-based near-collision finding algorithm works well for higher levels even though these differences. In this perspective, we can extend this strategy for higher level lists, for example by considering

$$\widetilde{\mathcal{T}}^n(w^{(2)}) = \{s^{(2)} \in \mathcal{T}^n(w^{(2)}) \mid \tau_{B,r^{(2)}}(M)s^{(2)} \bmod \tau_{B,r^{(2)}}(B) \approx 0\}$$

for $w^{(2)} \approx w^{(1)}/2$ and appropriate $r^{(2)}$ and so on forth.

In summary, we can find the solution s of $Ms = e \bmod B$ given the list with approximate constraint in Eq. (2) using the LSH-based near-collision finding algorithm, which is again reconstructed by using the higher level lists with approximate constraint. The actual algorithm will do this iteration in a top-down manner; first construct a list without constraint as a top level, and recover the lower level lists in order.

3 Previous Primal Hybrid Attacks

In this section, we review the primal hybrid attacks suggested in literature, e.g., in [20, 37, 50, 52]. These attacks share the similar structures; the combination of lattice reduction and meet-in-the-middle attack exploiting Babai's nearest plane algorithm, with some differences in targets, analysis, and optimizations.

We recall Babai's nearest plane algorithm in Section 3.1 and then summarize the framework of the primal hybrid attacks in Section 3.2.

3.1 Babai's Nearest Plane Algorithm and Matrix Modulus

Let $\mathcal{L} = \mathcal{L}(B)$ be a lattice with the basis B. Babai's nearest plane algorithm on inputs B and a target vector t is given by Algorithm 1.

Algorithm 1: Babai's Nearest Plane Algorithm

Input: Basis matrix $B = \{b_1, \ldots, b_n\}$
a target vector $t \in span(B)$
Output: $t - v \in \mathcal{P}(B^*)$ with $v \in \mathcal{L}(B)$

Compute Gram-Schmidt basis $B^* = \{b_1^*, \ldots, b_n^*\}$
Set $t_n = t$
for $i = n$ *down to* 1 **do**
  Set $l_i = \langle t_i, b_i^* \rangle / \langle b_i^*, b_i^* \rangle$, the i-th coordinate of $\tau_B(t_i)$.
  $y_i = \lfloor l_i \rceil \cdot b_i$
  Set $t_{i-1} = t_i - y_i$
end
return $t - v$ with $v = \sum_{i=1}^{n} y_i \in \mathcal{L}(B)$ such that $t - v \in \mathcal{P}(B^*)$

The classic fact on the output of Babai's algorithm is as follows.

Lemma 2. *Given an input vector t and basis $B = (b_1, ..., b_n)$, the output s of Algorithm 1 lies in the parallelepiped $\mathcal{P}(B^*)$. Further, the vector s is the unique vector in $\mathcal{P}(B^*)$ such that $t - s$ is in $\mathcal{L}(B)$.*

Note that we define the output of the algorithm by $t - v$ for convenience, while the standard Babai's nearest plane algorithm aims at finding a lattice point $v \in \mathcal{L}(B)$ such that $t - v \in \mathcal{P}(B^*)$. Given basis B and target v, we denote the output of Babai's algorithm by $\mathsf{NP}_B(v)$, and the B^*-basis representation of $\mathsf{NP}_B(v)$ by $[v]_B$, equivalently

$$[v]_B := \tau_B\left(\mathsf{NP}_B(v)\right).$$

Moreover, we introduce a *matrix modulus* notation by

$$a = b \bmod B \iff \mathsf{NP}_B(a) = \mathsf{NP}_B(b).$$

The following equation justifies the use of the matrix modulus notation $\bmod B$, and can be proved by Lemma 2.

$$\mathsf{NP}_B(a+b) = \mathsf{NP}_B(\mathsf{NP}_B(a) + \mathsf{NP}_B(b))$$

3.2 Primal Hybrid Attacks for LWE

We proceed to the primal hybrid attack framework, mainly adapted from the previous works [20,50,52].

Consider an LWE instance $(A, b = As' + e' \bmod q) \in \mathbb{Z}_q^{m' \times n} \times \mathbb{Z}_q^{m'}$ with a secret key $s' \in \mathcal{T}^n(w)$ and an error vector e' sampled from a certain distribution, e.g., the discrete Gaussian distribution $\mathcal{G}_\sigma^{m'}$. For ease of representation, we consider an augmented matrix $M := (-A||b)$ and $s = (s', 1) \in \mathcal{T}^{n+1}(w+1)$ so that the LWE equation $b = As' + e' \bmod q$ is equivalent to $M \cdot s = e' \bmod q$. The primal attack strategy considers the lattice $\mathcal{L}_P$ defined by the following basis matrix P.

$$P = \begin{pmatrix} qI_{m'} & M \\ & I_{n+1} \end{pmatrix} \in \mathbb{Z}^{(m'+n+1)\times(m'+n+1)}$$

Note that $\mathcal{L}_P$ contains a short vector $(e', s) = P \cdot (x, s)$ for a vector $x \in \mathbb{Z}^{m'}$.

We divide the matrix and secret key into

$$M = (M_1, M_2) \in \mathbb{Z}_q^{m' \times (n-d+1)} \times \mathbb{Z}_q^{m' \times d} \text{ and } s = (s_1, s_2) \in \mathcal{T}^{n-d+1} \times \mathcal{T}^d$$

for a guessing dimension parameter $0 \le d \le n$, and represent the basis matrix by

$$P = \begin{pmatrix} B & M_2 \\ & I_d \end{pmatrix} \text{ where } B = \begin{pmatrix} qI_{m'} & M_1 \\ & I_{n+1-d} \end{pmatrix}. \tag{5}$$

From this representation, we have the relation

$$Bk + M_2 s_2 = e \tag{6}$$

where $e = (e', s_1)$ and $k = (x, s_1)$. It implies $M_2 s_2 - e \in \mathcal{L}(B)$, or equivalently

$$M_2 s_2 = e \bmod \mathcal{L}(B).$$

The primal hybrid attacks [20,37,50,52] can be abstracted using the above relation as follows.

Primal hybrid attack proceeds as follows.

1. Sufficiently reduce the basis matrix B using lattice reduction algorithms so that e is included in the parallelepiped $\mathcal{P}(B^*)$, that is, the following equation holds
$$\mathsf{NP}_B(e) = e. \tag{7}$$
2. Using some combinatorial guessing strategies, finding the partial secret key s_2 such that
$$\mathsf{NP}_B(M_2 s_2) = e$$

Success probability. It should be remarked that the event $\mathsf{NP}_B(e) = e$ of Eq. (7) does not always happen, even if the basis B is highly reduced. Thus the probability of the event, denoted by p_{NP}, should be taken

into account when estimating the attack complexity. Regarding this, the previous works [50, 52] provided the detailed analysis for p_{NP}, especially when e follows (continuous) Gaussian $\mathcal{G}_\sigma$:

$$p_{NP} = \Pr_{e \leftarrow \mathcal{G}_\sigma}[\mathsf{NP}_B(e) = e] = \prod_{i=1}^{m+n+1-d} \mathrm{erf}\left(\frac{\|b_i^*\|\sqrt{2}}{\sigma}\right),$$

where $\|b_i^*\|$ is the length of i-th length of B^*.

Normalizing the new error vector. It remains as an issue that the error vector $e = (e', s_1)$ is quite far from Gaussian: e' follows a discrete Gaussian $\mathcal{D}_\sigma$ and s_1 is a sparse ternary vector. We found no previous works explicitly deal with this, and suggest two explanations regarding this. One is theoretically correct, and the other is somewhat heuristic but leads to a more efficient attack.

The first method is to replace the partial secret key by a part of the error vector using the transformation proposed in [11]. This transform changes the distribution of some secret key entries to follow the error distribution. We can apply this to a part of secret key s_1, which makes $e = (e', s_1)$ to exactly follow $\mathcal{G}_\sigma$.

The other method is to introduce the scaling factor as in [8, 14]. More precisely, we choose a new parameter $\nu > 0$ and consider an error vector $e_\nu = (e', \nu s_1)$ in order to balance the norm of both parts. For the exact choice of ν, we follow the choice of previous works: $\nu \approx \frac{\sigma}{\sqrt{2\pi}} \cdot \sqrt{\frac{n-d}{w}}$. Note that the scaled error e_ν can be easily reflected in the attack scenario by considering $B_\nu = \begin{pmatrix} qI_m & M_2 \\ & \nu I_{n+1-d} \end{pmatrix}$ instead of B in Eq. (5). However, even with the scaling factor ν, the latter part of e_ν obviously does not follow the Gaussian distribution. At this point, we establish a heuristic that each coordinate of e_ν with respect to the Gram-Schmidt basis B^* behaves as Gaussian sampled from $\mathcal{G}_\sigma$, whose justification is provided in Appendix B.

4 Finding Near-collisions

This section describes the near-collision finding algorithm of the list L using locality sensitive hashing (LSH), which will be used in Section 5 as a subroutine.

We first clarify the problem setting. Let $\mathcal{B} := \prod_{i=1}^{r}[-q_i/2, q_i/2)$ be an r-dimensional box. A ℓ-near-collision pair $(y_1, y_2) \in \mathcal{B} \times \mathcal{B}$ is defined by

$$|y_1 - y_2|_\infty \le \ell, \tag{8}$$

which can be efficiently verified given two elements y_1, y_2.

In the near-collision finding problem, we are given a list $L \subset \mathcal{B}$ having several ℓ-near-collision pairs in $L \times L$, and our goal is to find out the near-collision pairs. We assume that the list L consists of a small number of near-collision pairs and many additional random points in $\mathcal{B}$. In particular, each near-collision pair (y_1, y_2) are sampled by a certain error distribution $\mathcal{D}$ as follows.[4]

$$y_1 \leftarrow \mathcal{U}(\mathcal{B}), e \leftarrow \mathcal{D}, y_2 := y_1 + e, \text{ reject if } y_2 \notin \mathcal{B}. \tag{9}$$

We further treat the error e as a random variable in the analysis, which we call by a *$\mathcal{D}$-random/near model*. A similar models have been (at least implicitly) used in the previous works on the LWE cryptanalysis [40, 47, 50, 52] as well as in the context of representation techniques such as [26, 38].

Proposition 3 below states the main result of this section. We need some parameters and notations. First of all, we consider a block-length parameter b, and define the following probability p:

$$p(\ell, b) = \Pr\left[b + e \in [0, b) \mid b \leftarrow \mathcal{U}[0, b), e \leftarrow \mathcal{U}[-\ell, \ell)\right].$$

[4]The reason behind the rejection (rather than to define modulo $\mathcal{B}$) is to be consistent with the setting of the Meet-LWE for matrix modulus in Section 5.

This can be easily computed by

$$p(\ell, b) = \begin{cases} 1 - \frac{\ell}{2b} & \text{if } \ell \le b/2, \\ \frac{1}{4} + \frac{b}{4\ell} & \text{if } b/2 \le \ell < b. \end{cases} \tag{10}$$

We then write $n_i := \lceil q_i/b \rceil$ and $b_i := q_i/n_i$ for each $1 \le i \le r$.

Proposition 3. *Let $\mathcal{B} = \prod_{i=1}^{r}[-q_i/2, q_i/2)$ an r-dimensional box. Let $L \subset \mathcal{B}$ be a list of size $|L| = N$, which contains some ℓ-near-collision pairs, under the $\mathcal{D}$-random/near model. There is a randomized algorithm that outputs a list P of pairs such that a random near-collision pair in L is included in P with probability at least 0.95, and takes*

1. *$O(N/p_{good})$ hash evaluations,*
2. *$O(N^2 p_{bad}/p_{good})$ near-collision checks on expectation, and*
3. *$O(N)$ words of space for hash evaluation,*

where p_{good}, p_{bad} are defined as follows

1. *$p_{good} \ge \prod_{1 \le i \le r} p(\ell, b_i) \ge p(\ell, b)^r$ for $\mathcal{D} = \mathcal{U}([-\ell, \ell]^r)$,*
2. *$p_{good} = \left(\mathrm{erf}(x) + \frac{e^{-x^2}-1}{\sqrt{\pi}x}\right)^r$ for $\mathcal{D} = \mathcal{G}_\sigma^r$, where $x = \frac{b}{\sqrt{2}\sigma}$, and*
3. *$p_{bad} = \prod_{1 \le i \le r}(1/n_i)$ for any $\mathcal{D}$.*

Remark 1. *This proposition is indeed an adaptation of [35], but we made some modifications and considerations for our purpose: 1) We require an average-case algorithm, 2) we need to find almost all near-collision pairs while the original result tries to find a single solution, and 3) we cannot sample sufficiently many independent locality sensitive hash functions for amplifying $p_{bad} \approx 1/N$ in many cases, which is assumed in the original paper. We refer a similar discussion in [39] for LSH-like approaches in lattice sieving.*

4.1 LSH-based Near-collision Finding Algorithm

We describe the procedure to find the near-collision pairs using the locality sensitive hashing technique, and analyze the performance of this algorithm.

Hash functions. We start with the torus LSH functions, adapting [47]. Recall that for a block-length parameter b, we define $n_i := \lceil q_i/b \rceil$, and $b_i := q_i/n_i$ for each $1 < i \le r$. The family of torus LSH functions $\mathcal{H}(\mathcal{B}; b)$ with the block-length parameter b is defined over the domain $\mathcal{B}$ as follows.

$$\left\{ h_c\left((y_i)_{i\in[r]}\right) = \left(\left\lfloor \frac{y_i + c_i}{b_i} \right\rfloor \bmod n_i \right)_{1 \le i \le r} \;\middle|\; c \in \pi_r(\mathcal{B}) \right\}$$

The algorithm. Now we describe the near-collision finding algorithm. The algorithm do: randomly select a hash function h_c from $\mathcal{H}(\mathcal{B}; b)$, and label each element y in L with $h_c(y)$. Then, check if the elements with the same label are the near-collision pair or not. This procedure is repeated R times to collect almost all near-collisions. Formally, the algorithm proceeds as in Algorithm 2. The number of repetition R is to be determined below.

Analysis. We discuss the correctness and efficiency of Algorithm 2 here. We say that a pair (y_1, y_2) is hash-collision if $h_c(y_1) = h_c(y_2)$ for a hash function h_c used in the algorithm. The following lemma shows that the quantities p_{good} and p_{bad} in Proposition 3 are precisely the probabilities that the near-collision pair and two random points are hash-collision in a single iteration of the outer for-loop (line 4).

Lemma 4. *In the $\mathcal{D}$-random/near model, for a ℓ-near-collision pair (y_1, y_2), it holds that over the random choice of $h_c \in \mathcal{H}$ and $\mathcal{D}$,*

Algorithm 2: LSH-based near-collision finding algorithm

Input: A domain $\mathcal{B}$, a list $L \subset \mathcal{B}$, a block parameter b, and a distribution $\mathcal{D}$
Output: A set S of near-collision pairs in L

```
1  Compute p_good using Proposition 3, and set repetition numbers R = ⌈3/p_good⌉
2  Set a family of torus LSH functions H = H(B; b)
3  If D = G_σ, set the near-collision bound ℓ = 6σ
4  Set S ← ∅
5  for i = 1 to R do
6      Sample a function h_c ← H uniformly at random
7      for each address addr in Range(h_c) do
8          Initialize an empty bin T(addr)
9      end
10     for each y ∈ L do
11         Store y in the table T(h_c(y))
12     end
13     for each address addr in Range(h_c) do
14         for each pair (y_1, y_2) in T(addr) × T(addr) do
15             if |π_r(y_1 − y_2)|_∞ ≤ ℓ then
16                 S ← S ∪ {(y_1, y_2)}
17             end
18         end
19     end
20 end
21 return S
```

1. *for* $\mathcal{D} = \mathcal{U}([-\ell, \ell)^r)$, $\Pr[h_c(y_1) = h_c(y_2)] \geq \prod_{1 \leq i \leq r} p(\ell, b_i) \geq p(\ell, b)^r$ *and*
2. *for* $\mathcal{D} = \mathcal{G}_\sigma^r$, $\Pr[h_c(y_1) = h_c(y_2)] = \left(\mathrm{erf}(x) + \frac{e^{-x^2}-1}{\sqrt{\pi}x}\right)^r$ *where* $x = \frac{b}{\sqrt{2}\sigma}$.

If at least one of (y_1, y_2) *is random, then*

3. $\Pr[h_c(y_1) = h_c(y_2)] = \prod_{1 \leq i \leq r} 1/n_i$ *over the random choice of* $h_c \in \mathcal{H}$.

We assume Lemma 4 for now and analyze the main algorithm first.

Proof of Proposition 3. Consider each iteration of line 4. By the first two claims of Lemma 4, the iteration finds a near-collision pair with probability p_{good}. By repeating this procedure $R = 3/p_{good}$ times with the different randomness, the algorithm finds the near-collision pair with probability at least

$$1 - (1 - p_{good})^{3/p_{good}} \geq 1 - 1/e^3 \geq 0.95.$$

For the two points that are not collide, the model says that at least one of them is randomly sampled from $\mathcal{D}$. The third claim of Lemma 4 asserts that they collide by a single iteration with probability p_{bad}. Therefore the expected number of hash-collisions is $\binom{N}{2} p_{bad} = O(N^2 p_{bad})$ for each single iteration, and the full algorithm is expected to check whether $O(N^2 p_{bad}/p_{good})$ hash-collision pairs are near-collision or not.

Finally, the space complexity is clear since each iteration only takes the $O(N)$ hash evaluations. □

It remains to verify Lemma 4. The second case of $\mathcal{D} = \mathcal{G}_\sigma$ is essentially the same to [50, Lemma 4.2], and the third case is obvious, because a random point is uniformly distributed over $\mathcal{B}$.

The case of $\mathcal{D} = \mathcal{U}(\mathcal{B})$ is slightly involved. Consider the one-dimensional case $r = 1$ with $\mathcal{B} = [-q/2, q/2)$ and fix $y \in \mathcal{B}$. If y is not in the boundary, meaning that $y + e$ for $e \leftarrow [-\ell, \ell]$ never lies outside of $\mathcal{B}$, the probability that $y, y + e$ have the same image under a random hash function in $\mathcal{H}$ is exactly $p(\ell, b)$. If y is in the boundary, the condition that $y + e \in \mathcal{B}$ only increases the probability, thus $p(\ell, b)$ is a lower bound of the

near-collision finding probability over the one-dimensional space $[-q/2, q/2)$. Multiplying these bounds over each axis, we obtain the first item.

Remark 2. *In the actual estimation, we sometimes use better bounds for peculiar cases. For example, if $b \geq q_i$ holds for some i, then we have $b_i = q_i$ and the probabilities for such coordinates by 1, instead of using the function $p(\ell, b)$. We also note that when $\mathcal{D} = \mathcal{G}_\sigma$, we define the corresponding ℓ-near collision pair so that the sampled pair $(y, y + e)$ in the model becomes ℓ-near collision with high probability, e.g. $\ell = 6\sigma$.*

Remark 3. *The previous version of this paper includes another hash function family called Hamming LSH, which is now removed because it has essentially a negligible effect on the attack.*

5 Meet-LWE for Matrix Modulus

In this section, we describe how to generalize Meet-LWE attack to the matrix modulus setting. First, we start with a key observation that enables the projection concept to matrix modulus Section 5.1. Then, we will introduce a main technical contribution in Section 5.2 and Section 5.3. The final description of attack is given in Section 5.4.

5.1 Projection of Matrix Modulus

We first define the projection map with respect to matrix modulus that is compatible with Babai's nearest plane algorithm in some sense. Recall that we write $v \bmod B$ to denote $\mathsf{NP}_B(v)$, the normalized Gram-Schmidt orthogonalized matrix $\widetilde{B^*}$ and the coordinate of vector v with respect to $\widetilde{B^*}$ written by $\tau_B(v) = (\tau_B(v)_1, ..., \tau_B(v)_n)$.

The projected coordinate map $\tau_{B,r}(v)$ for an n-dimensional vector v is defined by the *last* r coordinates of $\tau_B(v)$, that is, $\tau_{B,r}(v) := (\tau_B(v)_{n-r+1}, ..., \tau_B(v)_n)$. Note that when we project the matrix $B = [b_1, \cdots, b_n]$ using $\tau_{B,r}$, we can neglect the first $n - r$ vectors $\tau_{B,r}(b_i)$ for $i \in [n - r]$ as they become all zeroes. In this regard, we now only consider the last r vectors and define $\tau_{B,r}(B) := (\tau_{B,r}(b_{n-r+1}), ..., \tau_{B,r}(b_n))$, which is the $r \times r$ lower-right submatrix of $R_B = \tau_B(B)$.

The following lemma states the commutativity of the projection map and Babai's algorithm. Intuitively, this lemma holds because of the top-down iteration of Babai's algorithm.

Lemma 5. *For any basis $B \in \mathbb{Z}_q^{n \times n}$ and any vector $v \in span(B)$, it holds that*

$$\tau_{B,r}\left(\mathsf{NP}_B(v)\right) = \mathsf{NP}_{\tau_{B,r}(B)}\left(\tau_{B,r}(v)\right).$$

In particular, it holds that $\tau_B\left(\mathsf{NP}_B(v)\right) = \mathsf{NP}_{R_B}(\tau_B(v))$ by choosing $r = n$.

Proof. We first write the matrix B as $B = \widetilde{B^*} \cdot R_B$ for an upper triangular matrix R_B using QR-decomposition.

Let us consider the case $r = n$ first. Note that $\tau_B(v) = (\widetilde{B^*})^{-1} \cdot v$ and $R_B = (\widetilde{B^*})^{-1} \cdot B$. Thus the left hand side is the Babai's algorithm with the inputs that are transformed by $v \mapsto (\widetilde{B^*})^{-1} \cdot v$. Since the inner products are all the same and the resulting vector is also transformed by the same map, the identity for $r = n$ presents the result of Babai's algorithm with different basis.

For an arbitrary r, the left hand side is the projection of $\tau_B\left(\mathsf{NP}_B(v)\right)$ to the last r coordinates. To analyze the right hand side, we define the other projection map p_r into $span(b^*_{n-r+1}, ..., b^*_n)$, and write $B_r := p_r(b_{n-r+1}, ..., b_n)$ and $v_r := p_r(v)$. With these notations, we have $\tau_{B_r}(B_r) = \tau_{B,r}(B)$ and $\tau_{B,r}(v) = \tau_{B_r}(v)$. The right hand side is equal to

$$\mathsf{NP}_{\tau_{B,r}(B)}\left(\tau_{B,r}(v)\right) = \mathsf{NP}_{\tau_{B_r}(B_r)}(\tau_{B_r}(v_r)) = \tau_{B_r}(\mathsf{NP}_{B_r}(v_r))$$

where the last equality is derived from the above special case by substituting B and v by B_r and v_r, respectively. In other words, the right hand side is the last r coordinates with basis $\widetilde{B^*}$ of the output of Algorithm 1

with projected inputs by p_r. This is the same to the left hand side because the last r coordinates with basis $\widetilde{B^*}$ is only changed by the first r iterations; in the last $n-r$ iterations, the algorithm only subtracts b_i for $1 \le i \le n-r$, which satisfies $\tau_{B,r}(b_i) = 0$. □

With this lemma, we can define the projected matrix modulus reduction by

$$[v]_{B,r} := \tau_{B,r}(\mathsf{NP}_B(v)) = \mathsf{NP}_{\tau_{B,r}(B)}(\tau_{B,r}(v)) = \tau_{B,r}(v) \bmod \tau_{B,r}(B).$$

5.2 Meet-LWE for Matrix Modulus: Overview

In this section, we describe the overview of meet-LWE attack for matrix modulus. This section conveys an intuition of our attack, and the necessary details are filled in Section 5.3 separately.

Recall that our goal is to find $s \in \mathcal{T}^d(w^{(0)})$ given (M, B) such that $\mathsf{NP}_B(Ms) = e$ where $M \in \mathbb{Z}^{m\times d}$, $B \in \mathbb{Z}^{m\times m}$, and e is sampled from the Gaussian distribution $\mathcal{G}_\sigma^m$ for some standard deviation σ. For ease of explanation, we (informally) rephrase this problem as the other problem to find the level-0 list by suppressing e to ≈ 0,

$$L^{(0)} = \{s^{(0)} \in \mathcal{T}^n(w^{(0)}) \mid Ms^{(0)} \bmod B \approx 0\},$$

which would be a singleton set $\{s\}$ with high probability.

In what follows, we will define level-i lists $L^{(i)}$ inductively up to top level t. Once the lists are correctly defined, the attack proceeds by constructing from the top level list $L^{(t)}$ down to the level-0 list $L^{(0)}$. In particular, each i-th level element can be recovered from the near-collision in the $(i+1)$-level list, and these near-collision pairs will be found by the near-collision finding algorithm of Section 4 for $L^{(i+1)} \times L^{(i+1)}$. In other words, the list $L^{(i)}$ is constructed by invoking the near-collision finding algorithm to the above level $L^{(i+1)}$, except the top level list $L^{(t)}$ that should be directly computed.

List definitions. We consider a level-1 weight $w^{(1)} \approx w^{(0)}/2$, and a pair $(s_1^{(1)}, s_2^{(1)}) \in \mathcal{T}^d(w^{(1)}) \times \mathcal{T}^d(w^{(1)})$ such that $s = s_1^{(1)} - s_2^{(1)}$, called a level-1 *representation* of s. For any level-1 representation $(s_1^{(1)}, s_2^{(1)})$ of s, it holds that $Ms_1^{(1)} = Ms_2^{(1)} + e \bmod B$. That is, we have $Ms_1^{(1)}$ and $Ms_2^{(1)}$ is at distance e modulo B so that they become near-collision.

The main idea of [46] starts by observing that the number of representation is fairly large so that we may focus on the specific representation and avoid the redundant representations by using constraints. Similarly, we restrict our attention to a special representation such that $[Ms_1^{(1)}]_{B,r^{(1)}}$ and $[Ms_2^{(1)}]_{B,r^{(1)}}$ are both small. More precisely, we consider a representation $(s_1^{(1)}, s_2^{(1)})$ such that $[Ms_1^{(1)}]_{B,r^{(1)}}$ and $[Ms_2^{(1)}]_{B,r^{(1)}}$ are both contained in some neighborhood of origin, say $A^{(1)}$. By taking proper dimension $r^{(1)}$ and area of $A^{(1)}$ whose details are presented in Section 5.3, we expect at least one representation satisfy that condition with high probability. Based on this intuition, we define a *level-1* list by

$$L^{(1)} = \{(s^{(1)}, [Ms^{(1)}]_B) \mid s^{(1)} \in \mathcal{T}^d(w^{(1)}) \wedge [Ms^{(1)}]_{B,r^{(1)}} \in A^{(1)}\}.$$

Then, thanks to our choice of $r^{(1)}$ and $A^{(1)}$, at least one representation $(s_1^{(1)}, s_2^{(2)})$ of s survives in $L^{(1)} \times L^{(1)}$ on expectation. In particular, we will choose the length of each edge of $A^{(1)}$ to be larger than the size of e (or σ), $[Ms_1^{(1)}]_{B,r^{(1)}}$ and $[Ms_2^{(1)}]_{B,r^{(1)}}$ becomes a near-collision pair in $L^{(1)}$. In other words, we can recover s given $L^{(1)}$ using near-collision finding on $L^{(1)} \times L^{(1)}$.

We then reduce the construction of $L^{(1)}$ to another level-2 list $L^{(2)}$ using a similar strategy. Note that each element $s^{(1)} \in L^{(1)}$ also has several *level-2* representations $(s_1^{(2)}, s_2^{(2)}) \in \mathcal{T}^d(w^{(2)}) \times \mathcal{T}^d(w^{(2)})$ for some $w^{(2)} \approx w^{(1)}/2$. Then with a proper choice of $r^{(2)}$ and $A^{(2)}$, we define the level-2 list

$$L^{(2)} = \{(s^{(2)}, [Ms^{(2)}]_B) \mid s^{(2)} \in \mathcal{T}^d(w^{(2)}) \wedge [Ms^{(2)}]_{B,r^{(2)}} \in A^{(2)}\}$$

so that at least one representation of $s^{(1)}$ survives in $L^{(2)} \times L^{(2)}$ with high probability.

Remark 4. *It would be confused that we have to consider different list $L^{(2)}$ for each target $s^{(1)} \in L^{(1)}$, but this is not the case. Note that the definition of $L^{(2)}$ is independent to target $s^{(1)} \in L^{(1)}$. It implies that one near-collision search over $L^{(2)} \times L^{(2)}$ can find each $s^{(1)} \in L^{(1)}$ with the same probability.*

Optimization for the top level. The above recursive definitions of list can be continued toward an arbitrary level, but we slightly tweak the top level list construction as an optimization. For the ease of explanation, let the top level $t = 3$. In the top level, we have to directly construct the list instead of reducing it to higher level. The direct construction requires an expensive exhaustively searching for every $s^{(3)} \in \mathcal{T}^d(w^{(3)})$ that takes $O(|\mathcal{T}^d(w^{(3)})|)$ time, even if the top level list may have some constraint, say $[Ms^{(3)}]_{B,r^{(3)}} \in A^{(3)}$. In other words, the constraint idea provides no benefit on the list construction timing cost in the top level.

Instead, we exploit the redundancy of representations by defining $L^{(3)}$ as a *random* subset of $\mathcal{T}^d(w^{(3)})$

$$L^{(3)} \subset \{(s^{(3)}, [Ms^{(3)}]_B) \mid s^{(3)} \in \mathcal{T}^d(w^{(3)})\},$$

where the size of $L^{(3)}$ is chosen so that at least one level-3 representation in $L^{(3)} \times L^{(3)}$ for each $s^{(2)} \in L^{(2)}$ with high probability, whose detailed choice is analyzed in Section 5.3.

Remark 5 (The primal-MitM hybrid)**.** *The primal MitM attack of [37] can be understood as the top level $t = 1$ case of the attack described here. Roughly speaking, the original description of [37] repeatedly samples random element $s^{(1)}$ from $\mathcal{T}^d(w^{(1)})$ and store it in a sort of LSH, until it finds one representation $(s_1^{(1)}, s_2^{(1)})$ of s. Our attack with the top level $t = 1$ case is slightly different since it considers a random subset of $\mathcal{T}^d(w^{(1)})$ in advance, but the core idea that aims to find only one representation is exactly same.*

5.3 Filling Details of Meet-LWE for Matrix Modulus

We first recall the following lemma from [46] that computes the number of representations.

Lemma 6. *For weight parameters $w^{(i)}$ and $w^{(i-1)}$ such that $w^{(i)} \geq w^{(i-1)}/2$, define $\epsilon^{(i)} = w^{(i)} - w^{(i-1)}/2$. Then the number of level-i representations $R^{(i)}$ is*

$$R^{(i)} = \binom{w^{(i-1)}}{w^{(i-1)}/2}^2 \cdot \binom{d - 2w^{(i-1)}}{\epsilon^{(i)}, \epsilon^{(i)}, \cdot}.$$

where $\binom{n}{a_1,\ldots,a_k,\cdot} = \binom{n}{a_1,\ldots,a_k,n-a_1-\ldots-a_k} = \frac{n!}{a_1!a_2!\ldots a_k!(n-a_1-\ldots-a_k)!}$

Concrete choice of small area. We specify the area $A^{(i)}$ by a 0-centered box of dimension $r^{(i)}$ of each edge length $2\ell^{(i)}$, equivalently $[-\ell^{(i)}, \ell^{(i)})^{r^{(i)}}$. We write the level-$i$ area by $A_{r^{(i)},\ell^{(i)}}$ to clarify the dimension $r^{(i)}$ and the length $\ell^{(i)}$. As mentioned in Section 5.2, the parameter choice should allow us to expect that for each element $s^{(i-1)} \in L^{(i-1)}$, there is one representation $(s_1^{(i)}, s_2^{(i)})$ of $s^{(i-1)}$ such that

$$[Ms_1^{(i)}]_{B,r^{(i)}}, [Ms_2^{(i)}]_{B,r^{(i)}} \in A_{r^{(i)},\ell^{(i)}}. \tag{11}$$

among $R^{(i)}$ of i-level representation pairs.

Note that since $s^{(i-1)} \in L^{(i-1)}$, we know $e_{i-1} := [Ms^{(i-1)}]_{B,r^{(i-1)}}$ is also a relatively small vector contained in $A_{r^{(i-1)},\ell^{(i-1)}}$, i.e., $e_{i-1} \in A_{r^{(i-1)},\ell^{(i-1)}}$. The equation $s^{(i-1)} = s_1^{(i)} - s_2^{(i)}$ implies that

$$[Ms_2^{(i)}]_{B,r^{(i)}} = [Ms_1^{(i)}]_{B,r^{(i)}} + [e_{i-1}]_{B,r^{(i)}} \bmod \tau_{B,r^{(i)}}(B)$$

Thus, we can re-write the condition Eq. (11) as follows:

$$[Ms_1^{(i)}]_{B,r^{(i)}}, [Ms_1^{(i)}]_{B,r^{(i)}} + [e_{i-1}]_{B,r^{(i)}} \in A_{r^{(i)},\ell^{(i)}} \text{ where } e_{i-1} \in A_{r^{(i)},\ell^{(i-1)}}. \tag{12}$$

By writing $p_{rep}^{(i)}$ the probability that the condition Eq. (12) holds for a randomly chosen pair (s_1, s_2) among $R^{(i)}$ representations, we expect $R^{(i)} \cdot p_{rep}^{(i)}$ number of level-i pairs satisfying Eq. (11). We set hypercube parameters $r^{(i)}$ and $\ell^{(i)}$ so that $R^{(i)} \cdot p_{rep}^{(i)} \gtrsim 1$, and achieve our desired goal: Now we can expect one representation $(s_1^{(i)}, s_2^{(i)})$ survives in $L^{(i)}$.

It remains to specify the probability $p_{rep}^{(i)}$ in terms of other parameters. For that, we assume that $[Ms_1^{(i)}]_B$ uniformly distributes over $\mathcal{P}(B^*)$, which was already used implicitly in the previous work [37,40,46] to analyze the performance of each algorithm. We also remark that it may happens that the hypercube length $\ell^{(i)}$ is larger than Gram-Schmidt norm on some last coordinates. Thus, in fact, the k-th coordinate of hypercube length is set by $\min(\ell^{(i)}, \|b_{m-k}^*\|)$.

Lemma 7. *For a level i lower than the top level, let $p_{rep}^{(i)}$ be the probability of Eq. (11) over the random choice of level-i representation $(s_1^{(i)}, s_2^{(i)})$ of $s \in L^{(i-1)}$. It holds that,*

- *for $i = 1$, $p_{rep}^{(1)} = \frac{(2\ell^{(1)})^{r^{(i)}}}{\prod_{k=1}^{r^{(i)}} \|b_{m-k+1}^*\|} \cdot \left(\mathrm{erf}(x) + \frac{e^{-x^2}-1}{\sqrt{\pi}x}\right)^{r^{(i)}}$ where $x = \ell^{(1)}/\sqrt{2\sigma}$*
- *for $i > 1$, $p_{rep}^{(i)} = \frac{(2\ell^{(1)})^{r^{(i)}}}{\prod_{k=1}^{r^{(i)}} \|b_{m-k+1}^*\|} \cdot \left(p(\ell^{(i-1)}, \ell^{(i)})\right)^{r^{(i)}}$ where $p(\ell, b)$ is defined in Section 4, and $\|b_k^*\|$ is the norm of k-th Gram-Schmidt basis.*

Proof. (Sketch) First, we compute the probability of first part of representation, $s_1^{(i)}$, satisfies $[Ms_1^{(i)}]_{B,r^{(i)}} \in A_{r^{(i)},\ell^{(i-1)}}$. This can be easily computed considering volume ratio between the projected parallelepiped and the area $A_{r^{(i)},\ell^{(i-1)}}$, which is the first term of the results. We then have to consider the other part $s_2^{(i)}$ also satisfies $[Ms_1^{(i)}]_{B,r^{(i)}} \in A_{r^{(i)},\ell^{(i-1)}}$, conditioned on the above. Note that this is almost similar to LSH probability of Lemma 4: We have some (uniform) random point x in some area, and want to compute the probability of the random point remains in the hypercube after adding some error. One can obtain the statement by appropriately substituting block-length and errors with the probability in Lemma 4. □

Top level list size. Recall that the top level list $L^{(t)}$ is defined as a random subset without constraint. We compute the explicit set size of $L^{(t)}$, which is essentially identical to the analysis of previous hybrid attacks [37, 52]. For that, we consider a representation of $s^{(t-1)} \in L^{(t-1)}$ by $s^{(t-1)} = s_1^{(t)} - s_2^{(t)}$. If the following condition holds

$$[Ms_1^{(t)}]_{B,r^{(t-1)}} = [Ms_2^{(t)}]_{B,r^{(t-1)}} + [Ms^{(t-1)}] \bmod \tau_{B,r^{(t-1)}}(B), \tag{13}$$

a near-collision finding algorithm on $L^{(t)}$ can recover the representation $(s_1^{(t)}, s_2^{(t)})$. Writing the probability of Eq. (13) by $p_{adm}^{(t)}$ over a randomly chosen representation, we expect $\sqrt{R^{(t)} \cdot p_{adm}^{(t)}}$ number of representations satisfying Eq. (13). For any random subset $L^{(t)}$ of $\mathcal{T}^d(w^{(t)})$, each element of $\mathcal{T}^d(w^{(t)})$ is included in $L^{(t)}$ with probability $|L^{(t)}|/|\mathcal{T}^d(w^{(t)})|$. Therefore, by taking

$$|L^{(t)}| \approx |\mathcal{T}^d(w^{(t)})| \,/\, \sqrt{R^{(t)} \cdot p_{adm}^{(t)}}. \tag{14}$$

we can expect one representation included in $L^{(t)}$. Finally, p_{adm} is computed by the following lemma.

Lemma 8. *Let $p_{adm}^{(t)}$ be the probability of Eq. (13) over the random choice of the top level t representation $(s_1^{(t)}, s_2^{(t)})$ of $s \in L^{(t-1)}$. It holds that $p_{adm}^{(t)} = \prod_{k=1}^{r^{(t)}} p(\ell^{(t-1)}, \|b_{m-k+1}^*\|)$ where $p(\ell, b)$ is defined in Section 4, and $\|b_k^*\|$ is the norm of k-th Gram-Schmidt basis.*

Proof. (Sketch) The proof can be done similarly Lemma 7, by understanding $L^{(t)}$ uses the entire hypercube $\tau_{r^{(t-1)}}(B^*)$ as a constraint area. □

5.4 Full Description and Analysis

Recall our target matrix modulus equation $Ms = e \bmod B$ with $s \in \mathcal{T}^d(w^{(0)})$. Our Meet-LWE algorithm with top level t is parameterized by

- Weight parameters $w^{(i)}$ for $1 \le i \le t$,
- Hypercube length parameters $\ell^{(i)}$ for $1 \le i < t$, and
- Torus LSH length parameters $b^{(i)}$ for $0 \le i < t$.

Given the above parameters, the number of representations $R^{(i)}$ from $w^{(i)}$ and $w^{(i-1)}$ are determined by Lemma 6. Then, the projection dimension $r^{(i)}$, that totally determines the constraint area $A_{r^{(i)},\ell^{(i)}}$ along with the other parameters, are iteratively determined as follows: Take the minimal $r^{(1)}$ so that $R^{(1)} \cdot p_{rep}^{(1)} \ge 1$ using Lemma 7 with the bottom level error $\mathcal{G}_\sigma$. Higher levels except the top level projection dimension $r^{(i+1)}$ is chosen so that $R^{(i+1)} \cdot p_{rep}^{(i+1)} \ge 1$ also using Lemma 7 with the lower level error; uniform distribution bounded by $\ell^{(i)}$. Finally, we compute the top level list size $|L^{(t)}| \approx |\mathcal{T}^d(w^{(t)})|/\sqrt{R^{(t)} \cdot p_{adm}^{(t)}}$ (Eq. (14)), where $p_{adm}^{(t)}$ is computed using Lemma 8. This completes the parameter setting.

After determining all parameters, the actual Meet-LWE for matrix modulus algorithm proceeds by performing the actual list construction. First, it directly constructs the top level list $L^{(t)}$, and repeatedly constructs the lower level lists down to $L^{(0)}$. Except the top level list $L^{(t)}$, each level-i list construction requires an investigation of $L^{(i+1)} \times L^{(i+1)}$. We make use of the near-collision algorithm of Algorithm 2, on input list $L^{(i+1)}$ and near-collision condition derived from $L^{(i)}$. As a summary, Algorithm 3 shows the detailed procedures for the top level $t = 3$ case.

Algorithm 3: Meet-LWE for matrix modulus with the top level $t = 3$

Input: Matrices $M \in \mathbb{Z}^{m \times d}$ and $B \in \mathbb{Z}^{m \times m}$ such that $\mathsf{NP}_B(Ms) = e$,
a standard deviation σ for the error distribution $\mathcal{G}_\sigma$,
the solution weight $w^{(0)}$ of $s \in \mathcal{T}^d(w^{(0)})$

Parameters : The weight parameters $w^{(1)}, w^{(2)}, w^{(3)}$,
the hypercube length parameters $\ell^{(1)}, \ell^{(2)}$,
the torus LSH length parameters $b^{(0)}, b^{(1)}, b^{(2)}$

Output: The solution s such that $\mathsf{NP}_B(Ms) = e$.

1 Compute the number of representations $R^{(1)}, R^{(2)}, R^{(3)}$ using Lemma 6
2 Determine the projection dimensions $r^{(1)}, r^{(2)}$ so that $R^{(i)} \cdot p_{rep}^{(i)} \ge 1$ where $p_{rep}^{(i)}$ is computed by Lemma 7.
3 Construct the list $L^{(3)}$ of size $|\mathcal{T}^d(w^{(3)})|/\sqrt{R^{(3)} \cdot p_{adm}^{(3)}}$ by randomly sampling elements from $\mathcal{T}^d(w^{(3)})$ where $p_{adm}^{(3)}$ is computed by Lemma 8.
4 **for** $i = 3$ down to $i = 1$ **do**
5 Construct $L^{(i-1)}$ from Algorithm 2 on inputs $\mathcal{B}^{(i)}$, $L^{(i)}$, $b^{(i-1)}$, $\mathcal{D}^{(i-1)}$ where $\mathcal{B}^{(i)}$ is defined by Eq. (16), and $\mathcal{D}^{(i-1)}$ by Eq. (15).
6 **end**
7 **return** $s \in L^{(0)}$

Success probability. Recall that the constraint hypercube $A_{r^{(i)},\ell^{(i)}}$ is chosen so that one representation of $s^{(i-1)} \in L^{(i-1)}$ survives in $L^{(i)} \times L^{(i)}$ *on expectation*. In other words, each element $s^{(i-1)}$ is recovered with some constant probability, and it implies that only a subset of $L^{(i-1)}$ will be recovered during the attack process. Regarding this, one may think that the attack succeeds only when each list $L^{(i)}$ are fully constructed, whose probability is devastating. However, note that our attack implicitly fixes a target representation for each level i when the lists $L^{(i)}$ are defined. Therefore, the success of our attack only depends on whether each list $L^{(i)}$ contains that fixed representation, which happens with constant probability.

Time complexity analysis. By denoting the cost of list construction by $T_{build}(\cdot)$ and the cost of near-collision finding algorithm by $T_{LSH}(\cdot)$, the total time complexity is

$$T_{guess} = T_{build}(L^{(t)}) + T_{LSH}(L^{(t)}) + T_{LSH}(L^{(t-1)}) + \cdots + T_{LSH}(L^{(1)}).$$

Note that this cost estimation assumes that each list $L^{(i)}$ is fully constructed. However, as said in the above paragraph, the attack only recovers a subset of $L^{(i)}$, and this is rather an overestimation of attack complexity.

The list construction cost T_{build} is rather immediate, as the size of the top level list can be computed by $|L^{(t)}| \approx |S^{(t)}|/\sqrt{R^{(t)} \cdot p_{adm}^{(t)}}$. Note that it takes one $\mathsf{NP}_B(\cdot)$ call to construct one element in $L^{(t)}$. We assume that Babai's NP call for $m \times m$ basis B takes $O(m)$ time by following [52], which concludes that $T_{build}(L^{(t)}) = O(m \cdot |L^{(t)}|)$.

We proceed to the details of $T_{LSH}(L^{(i)})$, where we want to build $L^{(i-1)}$ using Algorithm 2 from the upper level list $L^{(i)}$. We use the LSH block length $b^{(i-1)}$ given as parameter, but need further clarifications on the domain $\mathcal{B}^{(i)}$ and the distribution $\mathcal{D}^{(i)}$ in order to compute time complexity $T_{LSH}(L^{(i)})$ using Proposition 3. First, the distribution $\mathcal{D}^{(i)}$ for near-collision is totally determined by the constraint of $L^{(i-1)}$. To explicitly represent, we have

$$\mathcal{D}^{(i)} = \begin{cases} \mathcal{U}(A_{r^{(i-1)},\ell^{(i-1)}}) & \text{if } i > 1, \\ \mathcal{G}_\sigma^m & \text{if } i = 1. \end{cases} \tag{15}$$

Then it is clear that the domain $\mathcal{B}^{(i)}$ should be a subset of the last $r^{(i-1)}$ coordinates of $\mathcal{P}(B^*)$. As a critical detail, note that the last $r^{(i)}$ coordinates of each element $L^{(i)}$ are restricted by i-th constraint hypercube $A_{r^{(i)},\ell^{(i)}}$, except the top level list $L^{(t)}$. Considering this, the domain $\mathcal{B}^{(i)}$ is explicitly determined as

$$\mathcal{B}^{(i)} = \begin{cases} \prod_{k=1}^{r^{(t)}} [-\|b^*_{m-k+1}\|/2, \|b^*_{m-k}\|/2) & \text{if } i = t, \\ \prod_{k=r^{(i)}+1}^{r^{(i-1)}} [-\|b^*_{m-k+1}\|/2, \|b^*_{m-k+1}\|/2) \times [-\ell^{(i)}, \ell^{(i)})^{r^{(i)}} & \text{if } i < t \end{cases} \tag{16}$$

where $\|b^*_k\|$ denotes the norm of k-th Gram-Schmidt basis. With the above clarifications on $\mathcal{D}^{(i)}$ and $\mathcal{B}^{(i)}$, we can compute p_{good} and p_{bad} using Proposition 3, which finally determines $T_{LSH}(L^{(i)}) = O(|L^{(i)}|^2 \cdot p_{bad}/p_{good})$.

Remark 6. *Rigorously speaking, each element of $L^{(i)}$ is of the form $(s, [Ms]_B)$. We then actually consider a list*

$$\hat{L}^{(i)} := \{[Ms]_{B,r^{(i)}} \mid (s, [Ms]_B) \in L^{(i)}\},$$

and feed it to Algorithm 2. For each near-collision $([Ms_1]_{B,r^{(i)}}, [Ms_2]_{B,r^{(i)}})$ in $\hat{L}^{(i)}$, we take the corresponding pair $(s_1, [Ms_1]_B)$ and $(s_2, [Ms_2]_B)$ in $L^{(i)}$ and insert $(s_1 - s_2, [Ms_1]_B - [Ms_2]_B)$ in $L^{(i-1)}$. In the main algorithm description of Algorithm 3, we do not bother to clarify this and simply state to run Algorithm 2 with a list $L^{(i)}$.

6 Primal-Meet-LWE

Finally, we propose a new attack algorithm, Primal-Meet-LWE, for the standard LWE instance $b = As + e \bmod q$; the lattice reduction phase that converts the given LWE instance into $Ms = e \bmod B$, and the guessing phase that mounts Meet-LWE. The formal description is given in Algorithm 4.

Time complexity analysis. The time complexity would be $T_{lat} + T_{guess}$ modulo the repetition of the main loop, for the success probability amplification. T_{lat} is clearly dominated by BKZ-β execution time, and T_{guess} is analyzed in Section 5.4. Note that the permutation in each repetition acts as a shuffle on the secret key $s' \in \mathcal{T}^n(w)$, producing a new LWE instance.

Algorithm 4: Primal-Meet-LWE

Input: LWE instance $(A, b) \in \mathbb{Z}_q^{m' \times (n+1)}$ and the standard deviation σ
Parameters : For defining matrix modulus equation $Ms = e \bmod B$
m: the rank of basis matrix B
β: the blocksize for the BKZ reduction
d: the guessing dimension
Parameters : For Meet-LWE for matrix modulus
w_g: the (expected) number of ± 1 in guessing dimension d
$w^{(i)}$: the weight parameters
$\ell^{(i)}$: the hypercube length parameters
$b^{(i)}$: the torus LSH length parameters
Output: The secret key $s' \in \mathcal{T}^n(w)$ such that $b = As' + e'$

repeat
  Randomly permute the columns of A
  Using BKZ-β and (A, b), define the problem that solves $[Ms]_B = e$ given $M \in \mathbb{Z}^{m \times d}$, $B \in \mathbb{Z}^{m \times m}$, as described in Section 3.2.
  Recover s by solving matrix modulus equation $[Ms]_B = e$ from Algorithm 3 with input $M, B, \sigma, w_g = w^{(0)}$, and optimization parameters.
until $s \in \mathcal{T}^d(w_g)$
Recover the remaining s_r from $B^{-1}(Ms_2 - e) = (x, s_r)$
return (s_r, s)

It remains to discuss the number of repetitions, which is estimated by $1/p_{suc}$ for attack success probability p_{suc}. There are two events to consider for attack success. First we assume that $\mathsf{NP}(e) = e$, whose probability p_{NP} is already analyzed in Section 3.2. Second, as we restrict the searching space for Meet-LWE by $\mathcal{T}^d(w_g)$, the attack succeeds only when $s_2 \in \mathcal{T}^d(w_g)$, whose probability is computed by $p_{HW} = \binom{n-2w}{d-2w_g}\binom{w}{w_g}^2 / \binom{n}{d}$. Because Algorithm 3 succeeds with a constant probability, we conclude that Algorithm 4 succeeds with probability $p_{suc} = p_{NP} \cdot p_{HW}$. The final attack complexity is estimated by $\frac{1}{p_{suc}}(T_{lat} + T_{guess})$.

6.1 Concrete Estimations

Given an input LWE parameters such as n, q, σ, w, there would be the optimal parameterizations of Algorithm 4 giving the minimal complexity. For a quick search of optimal parameters, we search the parameterization space by the binary search for each parameters in a plausible range. For example, we take the hypercube lengths by $\ell^{(1)} = 2\sigma$ and $\ell^{(i+1)} = 2\ell^{(i)}$ for $i \geq 1$, and LSH block size $b^{(i)} = \ell^{(i)}$. For someone's interest, this opens some possibility to further reduce the attack complexity by parameter fine-tuning. We implement a python script for this procedure, which can be found in the supplementary material.

Finally, we estimate the concrete attack complexity of several LWE parameters, from homomorphic encryption to post-quantum cryptography regime via our code. To compare with previous attacks, we take the cost estimations of other attacks from lattice-estimator [3], which continuously updates the advances in lattice-based attacks [23, 25, 33, 48, 50, 52]. To ensure the reliability of our estimation, we rightly check that our script and lattice-estimator are consistent, in the sense that they output almost similar attack costs for the previous attack strategy, such as Primal hybrid MitM [37]. In particular, we use the default BKZ cost model of lattice-estimator [3][5].

The fully homomorphic encryption regimes. The fully homomorphic encryption schemes employ huge LWE parameters such as $n \geq 2^{15}$ and $q \geq 2^{500}$. Many recent schemes and implementations choose the sparse secret key for supporting efficient bootstrapping procedures [18, 22, 34, 42]. Homomorphic encryption libraries set the Hamming weight of the secret key by 64 or 192 [1, 4], arguing the 128-attack complexity

[5] Precisely, it assumes BKZ with block-size β and basis dimension n takes $5.46n \cdot 2^{0.296\beta + 20.388}$ [45]

of schemes. Table 1 shows our estimation results on the LWE problem for homomorphic encryptions. The estimation shows that our attack algorithm has the best performance on the homomorphic encryption parameter regime.

Lattice	Search	$(15, 699, 128)$	$(15, 768, 192)$	$(16, 1553, 192)$	$(16, 1450, 64)$
Primal	**Meet-LWE**$_{t=2}$	**126.0**	**126.0**	**127.5**	**104.4**
	Meet-LWE$_{t=3}$	**125.3**	**125.5**	**126.5**	**104.0**
Primal	None [10]	160.5	145.0	144.6	155.4
	Brute-force [9]	160.3	137.6	-	127.3
	MitM [37, 50]	133.4	**132.1**	-	112.1
Dual	None [8]	161.6	146.4	145.6	156.5
	FFT [33]	144.4	140.0	140.1	125.6
	MitM [23]	**133.3**	133.1	**135.9**	**108.4**

Table 1: Complexity estimations of LWE attacks on FHE regime parameters found from [18, 34, 41, 42]. Our proposals correspond to the combination of Primal and Meet-LWE. The costs of our attacks and the best of previous attacks are marked by bold. The parameters are of the form $(\log n, \log q, w)$, where n is the dimension of secret key and q is the ciphertext modulus, and w is the hamming weight of secret key. The standard deviation of error σ is fixed by 3.2. The dash (-) denotes the case that the estimator failed to find the optimal cost because of out of memory.

The post-quantum cryptography regimes. We also applied our estimation algorithm to the LWE parameters for the post-quantum cryptography schemes. We mainly tested the schemes with (sparse) ternary secret keys [12, 13, 21, 36].

The concrete security of the schemes with a highly sparse secret key is fairly reduced by our attack. For example, our attack reduces the attack complexity of IEEE standard NTRU [6] parameter $(n, q, w, \sigma) = (659, 2048, 76)$ from 131.2 (Dual FFT) to 114.6 (Lv 2), and Round2 [13] parameter $(n, q, w, \sigma) = (500, 2^{14}, 74, 2.29)$ from 104.2 (Dual FFT) to 92.5 (Lv 2).

However, these extremely sparse secret keys are now considered deprecated, and the recent PQC schemes with sparse secret keys, such as Round5 [12] (an updated version of Round2 scheme) and NTRU [21], have much larger Hamming weights. Our estimation predicts that the LWE problems for such schemes are less susceptible to our new hybrid attack. This is a predictable result because the previous primal hybrid attacks have much higher attack costs than the other attacks; the difference in attack costs between the primal MitM hybrid and the other attacks, say > 20-bit, is beyond the improvement from our attack. For example, the attack cost of the NTRU parameter setting $(n, q, w) = (509, 2048, 254)$ is 131.4 against the non-hybrid primal, while the primal hybrid attack cost is about 183.9.

Acknowledgements. MH is supported by a KIAS Individual Grant QP089801. CL is supported by a KIAS Individual Grant CG080601.

References

[1] HEAAN. https://github.com/snucrypto/HEAAN. Accessed: 2022-10-07.

[2] HElib. https://github.com/homenc/HElib. Accessed: 2022-10-07.

[3] Lattice-Estimator. https://github.com/malb/lattice-estimator. Accessed: 2022-10-07.

[4] Lattigo. https://github.com/tuneinsight/lattigo. Accessed: 2022-10-07.

[5] PALISADE. https://gitlab.com/palisade/palisade-release. Accessed: 2022-10-07.

[6] IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. IEEE Std 1363.1-2008, 2008. Accessed: 2022-05-27.

[7] S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference*, pages 333–362. Springer, 2016.

[8] M. R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 103–129. Springer, 2017.

[9] M. R. Albrecht, B. R. Curtis, and T. Wunderer. Exploring trade-offs in batch bounded distance decoding. In *SAC*, pages 467–491. Springer, 2019.

[10] M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 297–322. Springer, 2017.

[11] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

[12] H. Baan, S. Bhattacharya, S. Fluhrer, O. Garcia-Morchon, T. Laarhoven, R. Rietman, M.-J. O. Saarinen, L. Tolhuizen, and Z. Zhang. Round5: Compact and fast post-quantum public-key encryption. In *International Conference on Post-Quantum Cryptography*, pages 83–102. Springer, 2019.

[13] H. Baan, S. Bhattacharya, O. Garcia-Morchon, R. Rietman, L. Tolhuizen, J.-L. Torre-Arce, and Z. Zhang. Round2: KEM and PKE based on GLWR. *Cryptology ePrint Archive*, 2017. Technical Report.

[14] S. Bai and S. D. Galbraith. Lattice decoding attacks on binary LWE. In *Australasian Conference on Information Security and Privacy*, pages 322–337. Springer, 2014.

[15] A. Becker, J.-S. Coron, and A. Joux. Improved generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 364–385. Springer, 2011.

[16] L. Bi, X. Lu, J. Luo, and K. Wang. Hybrid dual and meet-lwe attack. In *To appear at Australasian Conference on Information Security and Privacy (ACISP) 2022*, 2022.

[17] X. Bonnetain, R. Bricout, A. Schrottenloher, and Y. Shen. Improved classical and quantum algorithms for subset-sum. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 633–666. Springer, 2020.

[18] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 587–617. Springer, 2021.

[19] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325, 2012.

[20] J. Buchmann, F. Göpfert, R. Player, and T. Wunderer. On the hardness of lwe with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In *International Conference on Cryptology in Africa*, pages 24–43. Springer, 2016.

[21] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, T. Saito, P. Schwabe, W. Whyte, K. Xagawa, T. Yamakawa, and Z. Zhang. PQC round-3 candidate: NTRU. Technical report, 2020. Accessed: 2022-05-20.

[22] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer, 2018.

[23] J. H. Cheon, M. Hhan, S. Hong, and Y. Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE. *IEEE Access*, 7:89497–89506, 2019.

[24] B. R. Curtis and R. Player. On the feasibility and impact of standardising sparse-secret LWE parameter sets for homomorphic encryption. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–10, 2019.

[25] D. Dachman-Soled, L. Ducas, H. Gong, and M. Rossi. LWE with side information: attacks and concrete security estimation. In *Annual International Cryptology Conference*, pages 329–358. Springer, 2020.

[26] C. Delaplace, A. Esser, and A. May. Improved low-memory subset sum and lpn algorithms via multiple collisions. In *IMA International Conference on Cryptography and Coding*, pages 178–199. Springer, 2019.

[27] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In *Annual Cryptology Conference*, pages 40–56. Springer, 2013.

[28] J.-P. D'Anvers, M. Rossi, and F. 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.

[29] T. Espitau, A. Joux, and N. Kharchenko. On a dual/hybrid approach to small secret LWE. In *International Conference on Cryptology in India*, pages 440–462. Springer, 2020.

[30] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[31] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 469–477, 2015.

[32] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 530–547. Springer, 2012.

[33] Q. Guo and T. Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 33–62. Springer, 2021.

[34] K. Han and D. Ki. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference*, pages 364–390. Springer, 2020.

[35] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

[36] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[37] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Annual International Cryptology Conference*, pages 150–169. Springer, 2007.

[38] N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–256. Springer, 2010.

[39] E. Kirshanova and T. Laarhoven. Lower bounds on lattice sieving and information set decoding. In *Annual International Cryptology Conference*, pages 791–820. Springer, 2021.

[40] E. Kirshanova and A. May. How to find ternary LWE keys using locality sensitive hashing. In *IMA International Conference on Cryptography and Coding*, pages 247–264. Springer, 2021.

[41] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 618–647. Springer, 2021.

[42] Y. Lee, J.-W. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and H. Kang. High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. Cryptology ePrint Archive, Paper 2020/1549, 2022. To appear at Eurocrypt 2022.

[43] M. Liu and P. Q. Nguyen. Solving bdd by enumeration: An update. In *Cryptographers' Track at the RSA Conference*, pages 293–309. Springer, 2013.

[44] V. Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[45] MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack. https://zenodo.org/record/6493704, 2022. Accessed: 2022-10-07.

[46] A. May. How to meet ternary LWE keys. In *Annual International Cryptology Conference*, pages 701–731. Springer, 2021.

[47] P. Q. Nguyen. Boosting the hybrid attack on ntru: Torus LSH, Permuted HNF and Boxed Sphere. *NIST Third PQC Standardization Conference*, 2021.

[48] E. W. Postlethwaite and F. Virdia. On the success probability of solving unique SVP via BKZ. In *IACR International Conference on Public-Key Cryptography*, pages 68–98. Springer, 2021.

[49] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[50] Y. Son and J. H. Cheon. Revisiting the hybrid attack on sparse secret LWE and application to HE parameters. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 11–20, 2019.

[51] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[52] T. 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 Flaw in the analysis of [40]

As noted in the introduction, Kirshanova and May suggested the improved Meet-LWE algorithm based on the locality sensitive hashing, called LSH-Meet-LWE [40]. We briefly describe some steps of the LSH-Meet-LWE algorithm and the flaw in the analysis, which was confirmed by the personal conversation with the authors.

Suppose that we are given a simplified LWE instance $As = e \bmod q$ with the secret key $s \in \mathcal{T}^n(w)$ that has w (± 1)-entries each[6] and the ternary error vector e. The following sets are the level-1 lists in the LSH-Meet-LWE algorithm

$$L_1^{(1)} = \left\{ s_1^{(1)} \in \mathcal{T}^n(w/2) : As_1^{(1)} \in \mathbb{Z}_q^{n-r} \times \{0\}^{r/2} \times \{\pm 1, 0\}^{r/2} \right\}$$
$$L_2^{(1)} = \left\{ s_2^{(1)} \in \mathcal{T}^n(w/2) : As_2^{(1)} \in \mathbb{Z}_q^{n-r} \times \{\pm 1, 0\}^{r/2} \times \{0\}^{r/2} \right\}.$$

These two lists have the different constraints on the last r coordinates; half of the r coordinates are fixed to 0, and the other half is bounded by 1.

The number of representations $(s_1^{(1)}, s_2^{(1)}) \in \mathcal{T}^n(w/2) \times \mathcal{T}^n(w/2)$ such that $s_1^{(1)} + s_2^{(1)} = s$ is $R = \binom{w}{w/2}^2$. The authors claim that the choice of r such that

$$R \approx q^{r/2} \cdot (q/3)^{r/2} \tag{17}$$

ensures that there is a representation $(s_1^{(1)}, s_2^{(1)}) \in L_1^{(1)} \times L_2^{(1)}$ on expectation. However, this claim turns out to be false.

Let us clarify the details of the problem. Since the list $L_1^{(1)}$ has $3^{r/2}$ possible entries in the last r coordinates, the choice of Eq. (17) indeed ensures that there is a representation $(s_1^{(1)}, s_2^{(1)})$ such that $s_1^{(1)} \in L_1^{(1)}$ on expectation. In the original Meet-LWE attack (with the corresponding list definitions therein), $s_1^{(1)} \in L_1^{(1)}$ automatically implies the event $s_2^{(1)} \in L_2^{(1)}$ [46, Section 5]. This is because the algorithm essentially enumerates all the possible r coordinates of the error vector.

This automatic implication is not the case for LSH-Meet-LWE, which does not enumerate the partial error vector. The condition $s_1^{(1)} \in L_1^{(1)}$ says that

$$\pi_r(As_1^{(1)}) \in \{0\}^{r/2} \times \{\pm 1, 0\}^{r/2}.$$

From the (simplified) LWE identity $As = e \bmod q$, we have

$$\pi_r(As_2^{(1)}) = \pi_r(e) - \pi_r(As_1^{(1)}).$$

To make $\pi_r(As_2^{(1)}) \in \{\pm 1, 0\}^{r/2} \times \{0\}^{r/2}$ to be true, it must hold that

$$\pi_{r/2}(As_1^{(1)}) = \pi_{r/2}(e)$$

which only happens with probability $1/3^{r/2}$ (assuming e is sampled from the uniform random ternary vector.) This probability is neglected in the original analysis, and the algorithm becomes impractical if this analysis is included.

We remark that the paper [40] suggested an alternative algorithm in the appendix, which does not use the locality sensitive hashing technique.

B Heuristic on Scaling Factor Normalization

We provide a more explanation for the heuristic described in the end of Section 3.2. First, the lattice reduction algorithm in the first step randomizes the Gram-Schmidt basis of B, so we can regard B^* as a random orthogonal basis independent of e_ν. The distribution of the coordinate vector of any vector with respect to the random orthogonal vector follows the spherical distribution of the same norm. In a high dimensional space, an n-dimensional Gaussian distribution of variance σ is highly concentrated in a thin spherical shell of width $O(1)$ around the sphere of radius $\sqrt{n}\sigma$ (See e.g., [51, Chapter 3]). In our case, the distribution of e_ν is very similar to the same dimensional Gaussian distribution with an appropriate variance, assuming more mild assumptions. Note that a similar assumption was used in e.g., [52].

[6] Note that the original paper used $\mathcal{T}^n(w/2)$ as the secret key of LWE, while we use $\mathcal{T}^n(w)$.