



# Too Many Hints – When LLL Breaks LWE

Alexander May<sup>1</sup>  and Julian Nowakowski<sup>1</sup> 

Ruhr-University Bochum, Bochum, Germany  
{alex.may, julian.nowakowski}@rub.de

**Abstract.** All modern lattice-based schemes build on variants of the LWE problem. Information leakage of the LWE secret  $\mathbf{s} \in \mathbb{Z}_q^n$  is usually modeled via so-called hints, i.e., inner products of  $\mathbf{s}$  with some known vector.

At Crypto'20, Dachman-Soled, Ducas, Gong and Rossi (DDGR) defined among other so-called *perfect hints* and *modular hints*. The trailblazing DDGR framework allows to integrate and combine hints successively into lattices, and estimates the resulting LWE security loss.

We introduce a new methodology to integrate and combine an arbitrary number of perfect and modular in a single stroke. As opposed to DDGR's, our methodology is significantly more efficient in constructing lattice bases, and thus easily allows for a large number of hints up to cryptographic dimensions – a regime that is currently impractical in DDGR's implementation. The efficiency of our method defines a large LWE parameter regime, in which we can fully carry out attacks faster than DDGR can solely estimate them.

The benefits of our approach allow us to practically determine which number of hints is sufficient to efficiently break LWE-based lattice schemes in practice. E.g., for mod- $q$  hints, i.e., modular hints defined over  $\mathbb{Z}_q$ , we reconstruct KYBER-512 secret keys via LLL reduction (only!) with an amount of 449 hints.

Our results for perfect hints significantly improve over these numbers, requiring for LWE dimension  $n$  roughly  $n/2$  perfect hints. E.g., we reconstruct via LLL reduction KYBER-512 keys with merely 234 perfect hints. If we resort to stronger lattice reduction techniques like BKZ, we need even fewer hints.

For mod- $q$  hints our method is extremely efficient, e.g., taking total time for constructing our lattice bases and secret key recovery via LLL of around 20 mins for dimension 512. For perfect hints in dimension 512, we require around 3 hours.

Our results demonstrate that especially perfect hints are powerful in practice, and stress the necessity to properly protect lattice schemes against leakage.

**Keywords:** LWE with Hints, Partial Key Exposure, PQC Standards

## 1 Introduction

**History of lattice schemes.** Basing the (post-quantum) security of cryptographic schemes on the hardness of lattice problems has been a big success story in the last 25 years, resulting in the recent NIST standardization

of KYBER [BDK<sup>+</sup>18], DILITHIUM [DKL<sup>+</sup>18] and FALCON [FHK<sup>+</sup>18]. Moreover, Google [KMS22] currently chooses to secure its internal communication with NTRU [HPS98].

As a historical curiosity, back in the 80s and early 90s lattices were mainly considered a powerful cryptanalysis tool [JS98]. After the invention of the famous Lenstra-Lenstra-Lovász (LLL) lattice reduction algorithm [LLL82], many cryptosystems have been broken disastrously via lattice reduction. E.g., knapsack-based cryptosystems [Odl90], which can be seen as an early predecessor of modern lattice schemes, were successfully attacked via lattices [CLOS91].

This led to a common belief that lattice reduction behaves much better than theoretically predicted, and not few cryptographers thought that finding short lattice vectors is feasible in general. This misunderstanding came from the design of knapsack/lattice schemes in too small dimension, for which lattices do not reveal their hardness.

The situation changed with Ajtai’s [Ajt96] NP hardness proof of the shortest vector problem, and the construction of the Ajtai-Dwork cryptosystem [AD97] with its cryptographically desirable worst- to average-case reduction.

While Ajtai-Dwork focussed on the hardness guarantees of lattice-based crypto, the invention of the NTRU cryptosystem of Hoffstein, Pipher and Silverman [HPS98] with its compact lattice bases as public keys was a cornerstone for the practicality of lattice schemes.

Back on the provable security path, the introduction of the Learning with Errors (LWE) problem together with Regev’s encryption scheme [Reg05] paved the way to amazingly versatile lattice constructions in all areas of cryptography. Eventually, a combination of the strong LWE security guarantees with the practicality of the NTRU cryptosystem was achieved via formulating the Ring-LWE [SSTX09,LPR10,PRS17] and Module-LWE [BGV14,LS15] variants.

**LWE in Practice.** In summer 2022, NIST announced the standardization of KYBER [BDK<sup>+</sup>18] as a lattice-based encryption/key encapsulation mechanism, and DILITHIUM [DKL<sup>+</sup>18] and FALCON [FHK<sup>+</sup>18] as lattice-based signature schemes. KYBER can be considered a highly-optimized version of Regev encryption [Reg05], based on Module-LWE. KYBER encryption comes in a package, called CRYSTALS, with a corresponding signature scheme DILITHIUM [DKL<sup>+</sup>18], also based on Module-LWE. The signature scheme FALCON [FHK<sup>+</sup>18] is based on an NTRU-type security assumption.

**Motivation of Hints.** Given the importance of side-channel leakage in real-world cryptography, NIST especially focused before its standardization decision on vetting lattice candidates against secret key leakage.

An LWE public key  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  satisfies the LWE relation  $\mathbf{s}\mathbf{A} + \mathbf{e} \equiv \mathbf{b} \pmod{q}$  for some small-norm secret  $\mathbf{s} \in \mathbb{Z}_q^n$ , and some error  $\mathbf{e} \in \mathbb{Z}_q^m$ . NTRU can be considered a special case with  $\mathbf{b} = \mathbf{0}^m$ .

LWE-based encryption schemes like Regev [Reg05] and Kyber [BDK<sup>+</sup>18] only store the secret  $\mathbf{s}$ , but not the error  $\mathbf{e}$ . Decryption of a ciphertext  $\mathbf{c} \in \mathbb{Z}_q^n$

is realized via computing the inner product  $\langle \mathbf{c}, \mathbf{s} \rangle$ . This computation may leak information about  $\mathbf{s}$ .

LWE/NTRU-based signature schemes like DILITHIUM [DKL<sup>+</sup>18] and FALCON [FHK<sup>+</sup>18] usually store both  $(\mathbf{s}, \mathbf{e})$  as secret key. DILITHIUM computes for a salted hash  $\mathbf{h} := H(\mathbf{m}, \cdot)$  of message  $\mathbf{m}$  both inner products  $\langle \mathbf{h}, \mathbf{s} \rangle$  and  $\langle \mathbf{h}, \mathbf{e} \rangle$ . FALCON computes for a salted hash  $\mathbf{h} = H(\mathbf{m}, \cdot)$  a polynomial ring product  $\mathbf{h} \cdot \mathbf{s} \in \mathbb{Z}_q[X]/(X^n + 1)$ , and later a ring product involving  $(\mathbf{s}, \mathbf{e})$ . As a conclusion, all these computations may either leak information on  $\mathbf{s}$  or  $\mathbf{e}$  alone, or on  $(\mathbf{s}, \mathbf{e})$  together.

In order to model the effect of such a secret key leakage, Dachman-Soled, Ducas, Gong, and Rossi [DDGR20] proposed a general lattice framework that quantifies the LWE security loss when revealing a so-called hint  $(\mathbf{v}, \mathbf{w}, \ell) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^m \times \mathbb{Z}$  satisfying

$$\langle (\mathbf{v}, \mathbf{w}), (\mathbf{s}, \mathbf{e}) \rangle = \ell. \tag{1}$$

The inner product computation of Equation (1) is usually performed in  $\mathbb{Z}_q$ , which we call a *mod- $q$  hint*. However, the authors of [DDGR20] also point out that fast NTT-based schemes like KYBER, DILITHIUM and FALCON usually postpone the reduction modulo  $q$  to the end of the computation for efficiency reasons. Thus, a side-channel may as well leak the value of  $\ell$  in Equation (1) before mod  $q$  reduction, a so-called *perfect hint*.

The framework of [DDGR20] is more generally applicable, also allowing for *modular hints* other than mod- $q$  hints, and for so-called *approximate hints*. In this work, we solely focus on perfect and modular hints, since they are especially simple, and allow for tremendous speedups in practice. Additionally, we study mod- $q$  hints in more detail, as we consider them practically highly relevant for cryptographic systems with mod- $q$  arithmetic. It remains an interesting open problem to provide similar speedups for the technically more involved approximate hints in the DDGR-framework.

As opposed to DDGR, our approach addresses hints for the secret  $\mathbf{s}$  only, i.e., hints  $(\mathbf{v}, \ell)$  with  $\langle \mathbf{v}, \mathbf{s} \rangle = \ell$ . We show that (at least) for mod- $q$  hints this is no limitation, and actually provides benefits. Additionally, for perfect hints and general modular hints (i.e., not necessarily mod- $q$  hints), this significantly simplifies the analysis of the resulting lattice bases.

**Too Many Hints.** Any mod- $q$  hint  $(\mathbf{v}, \ell)$  can be considered an error-free LWE sample. If we obtain  $n$  hints  $(\mathbf{v}_1, \ell_1), \dots, (\mathbf{v}_n, \ell_n)$  with linearly independent  $\mathbf{v}_i$ , then we can solve for  $\mathbf{s}$  via Gaussian elimination, even without  $\mathbf{A}$ . Therefore, clearly an amount of  $n$  (linearly independent) mod- $q$  hints is sufficient to reach a *too many hints* regime, in which we can attack LWE in polynomial time. Since perfect hints can be reduced modulo  $q$ , the upper bound of  $n$  also trivially holds for perfect hints.

Our goal in this work is to explore and expand the *too many hints* regime as far as possible by determining the minimal number of hints that is required to break LWE and its various cryptographic instances in practice, efficiently. This includes the regime where we break LWE in polynomial time using LLL reduction

only, as well as the regime where breaking LWE is still feasible in practice with BKZ lattice reduction. In such a hint regime where BKZ reduction is feasible, our algorithm yields practical LWE attacks, as opposed to the implementation of the DDGR framework, in which the lattice construction (and therefore the whole attack) is currently infeasible.

While [DDGR20] provides pioneering work on LWE hints and their effects, the DDGR framework currently fails to let us explore the *too many hints* regime in a satisfactory way. First, the implementation of DDGR includes hints successively in a computationally intensive manner via the dual lattice, which in practice does not allow us for integrating a number of hints in the order of  $n$ . Second, since we cannot even construct the desired lattice bases, we especially cannot test lattice reduction on real-world instances of lattice schemes.

**Our results.** Our new approach resolves these issues, and provides us with real-world data on standardized schemes, rather than security estimates.

*Idea of our Method.* In comparison to DDGR, our method is less lattice-centric and more LWE-centric. That is, whereas DDGR starts with a basis, which is successively transformed by each hint, we first process all hints, and then integrate them into a lattice basis.

From the trivial upper bound  $n$  argument for the *too many hints* regime we already see that every mod- $q$  hint reduces the subspace of all possible  $\mathbf{s}$  by one dimension. Thus, one can view mod- $q$  hints as a dimension reduction method, as it was, e.g., used in the recent analysis of NTRU in the more restricted attack setting of secret key bit leakage [EMVW22]. Hence, we expect that  $k$  mod- $q$  hints leave us with the hardness of an  $(n - k)$ -dimensional LWE problem.

Since, e.g., LWE with KYBER-like parameters is solvable with LLL reduction in dimension around  $n = 63$ , we would expect that  $512 - 63 = 449$  modular hints are sufficient for extracting KYBER-512 secret keys. We show that this is indeed the case.

*Efficiency of Lattice Basis Construction.* For mod- $q$  hints we propose a simple, and extremely efficient linear algebra approach that in the presence of  $k$  hints reduces the LWE dimension from  $n$  to  $n - k$ . Even in cryptographic dimensions, our lattice basis construction takes only a matter of seconds.

For perfect hints our lattice basis construction is technically more involved. We first construct a matrix solely involving our hints, then use LLL for dimension reduction, and eventually integrate the reduced hints together with the LWE samples into a lattice basis. This construction is still efficient, but significantly slower than our mod- $q$  hint method. Lattice basis construction takes, e.g., 3 hours for LWE dimension  $n = 512$ , and up to one week for  $n = 1024$ .

The case of general *modular hints* is (essentially) reduced to the case of *perfect hints*, making their lattice basis construction as efficient as in the perfect hint case.

Our lattice basis construction methodology does not only allow to integrate different types of hints separately, but also to freely combine them.

*Quality of Lattice Basis.* It goes without saying, that a construction method for a lattice basis that incorporates hints should be efficient. But of even larger importance is the quality of the resulting basis, meaning from a cryptanalyst’s perspective that the information provided by the hints has been fully exploited, and the resulting lattice requires reduction methods as weak as possible to reveal the LWE secret.

We thoroughly analyze the characteristics of our constructed lattices, in terms of the three main criteria lattice dimension, lattice determinant, and length of the desired secret short vector. Our analysis shows that the quality of our lattice bases for all types of hints is identical to the quality achievable with the DDGR framework.

*Our Experimental Results.* A rough outline of our experiments is provided in Table 1.

	KYBER 512	FALCON 512	NTRU-HRSS 701	KYBER 768	DILITHIUM 1024
mod- $q$	449 (88%)	452 (88%)	622 (89%)	702 (91%)	876 (85%)
Time	20 mins	20 mins	45 mins	35 mins	10 hours
perfect	234 (46%)	233 (46%)	332 (47%)	390 (51%)	463 (45%)
Time	3 hours	3 hours	11 hours	1 day	7 days

**Table 1.** Minimal amount  $k$  of mod- $q$ /perfect hints required for solving instances with LLL. Time includes both lattice basis construction and LLL reduction.

In the case of mod- $q$  hints, we require for LWE dimension  $n$  roughly  $k \approx 0.9n$  hints to reach the *too many hints* regime, in which we can solve via LLL reduction, see Table 1. Recall that our mod- $q$  approach directly constructs an LWE problem in dimension  $n - k \approx 0.1n$ .

Notice that for KYBER-512, FALCON-512, and KYBER-768 we have  $60 \leq n - k \leq 66$ . NTRU-HRSS allows for larger LWE dimension  $n - k = 79$ , since it has larger  $q$  and smaller secret vector norm. As one would expect, Dilithium’s very large  $q$  enables LLL-only attacks for the largest LWE dimension  $n - k = 148$ .

We would like to stress that Table 1 only provides the number of hints, for which we can solve via simple LLL reduction. E.g. we also solved KYBER-512 instances with 440 mod- $q$  hints with stronger BKZ reduction of block-size 3 in less than an hour. Thus, Table 1 basically provides the number of hints for which we obtain minimal attack time. The attack time in the mod- $q$  scenario is almost exclusively spend on LLL reduction, since our lattice basis construction can be performed in a matter of seconds.

In the case of perfect hints, we require for LWE dimension  $n$  roughly  $k \approx n/2$  hints. We find such a small number of hints quite remarkable!

Interestingly, in contrast to the mod- $q$  setting, run time in the perfect hint setting is almost exclusively spend on lattice basis construction. Recall that our

construction process first uses the hints only, uses LLL reduction for dimension reduction, and eventually integrates the LWE samples. It seems that LLL reduction of the hints only already yields an overall well-reduced lattice basis. Hence, after our lattice basis construction step we could almost always directly read off the desired secret lattice vector, and therefore solve LWE.

*Our Software.* We provide a highly efficient open-source Python implementation of our framework. The source code is available together with an extensive documentation at

[https://github.com/juliannowakowski/lwe\\_with\\_hints](https://github.com/juliannowakowski/lwe_with_hints).

At the heart of our implementation lies the class `LWELattice`, which allows to easily construct lattice bases for attacking LWE – with or without hints. The class `LWELattice` also provides an implementation of the (progressive) BKZ algorithm, based on the `fpyll` library [dt21]. Our implementation also ships with key generation algorithms for KYBER, FALCON, NTRU-HPS, NTRU-HRSS and DILITHIUM, as well as for KYBER-like and FALCON-like toy instances in small dimensions.

**Lattice Attacks go Practice.** Classical public-key schemes like RSA encryption and DSA signatures have experienced extensive studies on hint vulnerabilities, starting with the seminal works of Boneh and Venkatesan [BV96] and Coppersmith [Cop97]. This led to critical security issues in real world application like [HDWH12]. We see our work as a step towards making hint vulnerabilities also practical in the lattice world.

**Organisation of the Paper.** In Section 2, we provide some background on lattices and LWE. Section 3 introduces our highly efficient LWE transformation for mod- $q$  hints. Section 4 is devoted to our technically more involved lattice basis construction for perfect hints. In Section 5, we show how to integrate general modular hints in the aforementioned lattice basis construction. Section 6 provides a runtime comparison of our method with DDGR, and demonstrates how significant we improve in efficiency. Our experiments are presented in Section 7.

**Acknowledgements.** We are grateful to Carl Richard Theodor Schneider and Martin R. Albrecht for help with and bug-fixing in `fpyll`. We thank the anonymous reviewers for their detailed comments, that helped to improve our work.

Both authors are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – grant 465120249. Alexander May is additionally supported by grant 390781972.

## 2 Preliminaries

### 2.1 Linear Algebra

Vectors are denoted by lower-case bold vectors, matrices are denoted by upper-case bold vectors. We use row notation for vectors and write  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$  for a matrix  $\mathbf{B}$  with rows  $\mathbf{b}_i$ . To denote a matrix  $\mathbf{B}$  with columns  $\mathbf{b}_i^T$ , where  $(\cdot)^T$  denotes the transpose, we write  $\mathbf{B} = (\mathbf{b}_1^T | \dots | \mathbf{b}_n^T)$ . The  $i$ -th vector of the standard basis of  $\mathbb{R}^n$  is denoted by  $\mathbf{e}_i$ , e.g.,  $\mathbf{e}_1 = (1, 0, \dots, 0)$ . The  $n$ -dimensional identity matrix is denoted by  $\mathbf{I}_n$ , all-zero  $(n \times m)$ -matrices are denoted by  $\mathbf{0}_{n \times m}$ , and the  $n$ -dimensional all-zero vector is denoted by  $\mathbf{0}^n$ . If the dimensions are clear from the context, we drop the indices from  $\mathbf{0}_{n \times m}$  and  $\mathbf{0}^n$ . The Euclidean norm and the Euclidean inner product are denoted by  $\|\cdot\|$  and  $\langle \cdot, \cdot \rangle$ , respectively.

**Lemma 2.1.** *Let  $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{R}^n$  be a vector, whose coordinates are i.i.d. random variables with zero mean and standard deviation  $\sigma < \infty$ . Then it holds that*

$$\mathbb{E}[\|\mathbf{v}\|] \leq \sigma\sqrt{n}.$$

*Asymptotically, the upper bound is sharp, i.e.,*

$$\mathbb{E}[\|\mathbf{v}\|] \sim \sigma\sqrt{n}$$

*as  $n \rightarrow \infty$ .*

A proof for Lemma 2.1 is given in the full version of the paper [MN23].

For  $\mathbf{v} \in \mathbb{R}^n$ , we denote by  $\mathbf{v}^\perp$  the subspace orthogonal to  $\mathbf{v}$ . More generally, for a linear subspace  $U \subseteq \mathbb{R}^n$ , we denote by  $U^\perp$  the orthogonal complement of  $U$ . The orthogonal projection of  $\mathbf{v}$  onto  $U$  is denoted by  $\pi_U(\mathbf{v})$ .

### 2.2 Lattices

For a matrix  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n] \in \mathbb{Q}^{n \times m}$ , we denote by

$$\mathcal{L}(\mathbf{B}) := \mathbb{Z}^n \cdot \mathbf{B} = \{\alpha_1 \mathbf{b}_1 + \dots + \alpha_n \mathbf{b}_n \mid \alpha_i \in \mathbb{Z}\}$$

the lattice generated by the rows of  $\mathbf{B}$ .<sup>1</sup> If the rows of  $\mathbf{B}$  are linearly independent, we call  $\mathbf{B}$  a *basis matrix* of  $\mathcal{L}(\mathbf{B})$ . Two bases  $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{R}^{n \times m}$  generate the same lattice if and only if there exists a unimodular matrix  $\mathbf{U} \in \mathbb{Z}^{n \times n}$  such that  $\mathbf{B}_1 = \mathbf{U} \cdot \mathbf{B}_2$ . The number of rows in any basis matrix of some lattice  $\Lambda$  is called the *dimension* of  $\Lambda$  and denoted by  $\dim \Lambda$ . Equivalently, the dimension of  $\Lambda$  is defined as the dimension of the linear subspace  $\text{span}_{\mathbb{R}}(\Lambda)$ . A lattice with quadratic basis matrix is called a *full-rank* lattice.

The *determinant* of a lattice  $\Lambda$  with basis matrix  $\mathbf{B}$  is defined as

$$\det \Lambda := \sqrt{\det(\mathbf{B}\mathbf{B}^T)}.$$

<sup>1</sup> We restrict ourselves to rational matrices, because for irrational  $\mathbf{B}$  with linearly dependent rows, the resulting set  $\mathcal{L}(\mathbf{B})$  might not be a lattice.

Notice that the determinant does not depend on the choice of basis. The  $i$ -th successive minimum of  $\Lambda$  is defined as

$$\lambda_i(\Lambda) := \min \{r > 0 \mid \Lambda \text{ contains } i \text{ linearly independent vectors of length } \leq r\}.$$

A lattice vector  $\mathbf{v} \in \Lambda$  of length  $\|\mathbf{v}\| = \lambda_1(\Lambda)$  is called a *shortest vector* of  $\Lambda$ .

**Heuristic 2.2 (Gaussian Heuristic).** *Let  $\Lambda$  be an  $n$ -dimensional lattice. The Gaussian heuristic predicts that  $\lambda_1(\Lambda)$  equals*

$$\text{gh}(\Lambda) := \sqrt{\frac{n}{2\pi e}} \det(\Lambda)^{1/n}.$$

A set of linearly independent lattice vectors  $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset \Lambda$  is called *primitive* (with respect to  $\Lambda$ ), if it can be extended to a basis of  $\Lambda$ . Equivalently, the set  $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$  is called primitive, if  $\Lambda \cap \text{span}_{\mathbb{R}}(\{\mathbf{v}_1, \dots, \mathbf{v}_k\}) = \mathcal{L}(\mathbf{v}_1, \dots, \mathbf{v}_k)$ .

For instance,  $\{2\mathbf{e}_1, \dots, 2\mathbf{e}_n\} \subset \mathbb{Z}^n$  is not primitive with respect to  $\mathbb{Z}^n$ , since  $\mathbb{Z}^n \cap \text{span}_{\mathbb{R}}(\{2\mathbf{e}_1, \dots, 2\mathbf{e}_n\}) = \mathbb{Z}^n$ , but  $\mathcal{L}(2\mathbf{e}_1, \dots, 2\mathbf{e}_n) = 2\mathbb{Z}^n$ .

**Lemma 2.3.** *Let  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_k] \in \mathbb{Z}^{k \times n}$ . The set  $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset \mathbb{Z}^n$  is primitive with respect to  $\mathbb{Z}^n$  if and only if  $\mathcal{L}(\mathbf{V}^T) = \mathbb{Z}^k$ .*

A proof for Lemma 2.3 is given in the full version of the paper [MN23].

**Lemma 2.4 (Adapted from [MRW11, Proposition 1]).** *Let  $1 \leq k < n$  be integers. Let  $\mathbf{B} \in \mathbb{Z}^{n \times k}$  be a matrix, whose entries are independent and uniformly distributed over  $\{-B, \dots, B-1\}$  for some  $B \in \mathbb{N}$ . Then it holds that*

$$\Pr[\mathcal{L}(\mathbf{B}) = \mathbb{Z}^k] > (1 - 2^{2+k-n}) + o(1),$$

as  $B \rightarrow \infty$ .

A proof for Lemma 2.4 is given in the full version of the paper [MN23].

The *dual* of a lattice  $\Lambda$  is defined as

$$\Lambda^* := \{\mathbf{w} \in \text{span}_{\mathbb{R}}(\Lambda) \mid \forall \mathbf{v} \in \Lambda : \langle \mathbf{w}, \mathbf{v} \rangle \equiv 0 \pmod{1}\}.$$

For every lattice  $\Lambda$  it holds that  $(\Lambda^*)^* = \Lambda$ . If  $\Lambda \neq \{\mathbf{0}\}$ , then it holds that

$$\det(\Lambda) \cdot \det(\Lambda^*) = 1. \tag{2}$$

The integer lattice  $\mathbb{Z}^n$  is self-dual, i.e.,  $(\mathbb{Z}^n)^* = \mathbb{Z}^n$ .

**Lemma 2.5 ([Mar13, Proposition 1.3.4]).** *Let  $\Lambda \subset \mathbb{R}^n$  be a full-rank lattice and let  $U \subseteq \mathbb{R}^n$  be a linear subspace. Then it holds that  $\Lambda \cap U = (\pi_U(\Lambda^*))^*$ .*

**Lemma 2.6 ([Mar13, Proposition 1.2.9]).** *Let  $\Lambda \subset \mathbb{R}^n$  be a full-rank lattice and let  $U \subset \mathbb{R}^n$  be a  $d$ -dimensional linear subspace with  $0 < d < n$ , such that  $\Lambda \cap U$  is a  $d$ -dimensional lattice. Then it holds that*

$$\det(\pi_{U^\perp}(\Lambda)) = \frac{\det \Lambda}{\det(\Lambda \cap U)}.$$



### 2.3 LWE

**Definition 2.7 (LWE).** Let  $n$ ,  $m$  and  $q$  be positive integers, and let  $\chi$  be a distribution over  $\mathbb{Z}$ . The LWE problem or more precisely the LWE problem with short secrets for parameters  $(n, m, q, \chi)$  is defined as follows: Given

- a uniformly random matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ , and
- a vector  $\mathbf{b} \equiv \mathbf{s}\mathbf{A} + \mathbf{e} \pmod{q}$ , where  $\mathbf{s} \leftarrow \chi^n$ ,  $\mathbf{e} \leftarrow \chi^m$ ,

find  $\mathbf{s} \in \mathbb{Z}_q^n$ . The vector  $\mathbf{s}$  is called the secret,  $\mathbf{e}$  is called the error. The tuple  $(\mathbf{A}, \mathbf{b}, q)$  is called an LWE instance. A tuple  $(\mathbf{a}_i^T, b_i)$ , where  $\mathbf{a}_i^T$  is the  $i$ -th column of  $\mathbf{A}$  and  $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$  is the  $i$ -th coordinate of  $\mathbf{b}$ , is called an LWE sample.

We note that in Regev’s original definition of LWE [Reg05], the coordinates of the secret and the error do not follow the same distribution. Most practical LWE-based schemes, however, use the *short secret* variant from Definition 2.7. We further note that for efficiency purposes most practical LWE-based schemes do not sample the matrix  $\mathbf{A}$  uniformly at random, but instead use  $\mathbf{A}$ ’s, which can be stored more compactly. Most importantly, in so-called *Ring-LWE* and *Module-LWE* variants one encodes ring-/module structure into  $\mathbf{A}$ . This allows to store  $\mathbf{A}$  with only  $k \cdot n$  elements from  $\mathbb{Z}_q$ , where  $k \ll n$  is a small integer, typically  $1 \leq k < 10$ . The *NTRU problem* [HPS98] can be considered as a special variant of (Ring-)LWE, where the vector  $\mathbf{b}$  is fixed to  $\mathbf{0}$ .

An overview of parameters used in practice is given in Table 2.

Scheme	$n$	$m$	$q$	$\sigma \approx \frac{\mathbb{E}[\ \mathbf{e}, \mathbf{s}\ ]}{\sqrt{m+n}}$	Variant
KYBER-512	512	512	3329	1.22	Module-LWE
KYBER-768	768	768	3329	1.00	Module-LWE
KYBER-1024	1024	1024	3329	1.00	Module-LWE
DILITHIUM-1024	1024	1024	8380417	1.41	Module-LWE
DILITHIUM-1280	1280	1536	8380417	2.58	Module-LWE
DILITHIUM-1792	1792	2048	8380417	1.41	Module-LWE
FALCON-512	512	512	12289	4.05	NTRU
FALCON-1024	1024	1024	12289	2.87	NTRU
NTRU-HPS-509	509	509	2048	0.76	NTRU
NTRU-HPS-677	677	677	2048	0.72	NTRU
NTRU-HPS-821	677	677	4096	0.80	NTRU
NTRU-HRSS	701	701	8192	0.99	NTRU

**Table 2.** Parameters of practical LWE-based schemes.

*Remark 2.8.* For KYBER and DILITHIUM we calculated the standard deviation of the coordinates of  $(\mathbf{e}, \mathbf{s})$ , and then used Lemma 2.1 to approximate  $\frac{\mathbb{E}[\|(\mathbf{e}, \mathbf{s})\|]}{\sqrt{m+n}}$  in Table 2. For NTRU, we could not apply Lemma 2.1, because NTRU-HRSS and NTRU-HPS keys do not meet the requirements of the lemma. Instead, we determined the value of  $\mathbb{E}[\|(\mathbf{e}, \mathbf{s})\|]$  experimentally by calculating the average over 100 random keys each. FALCON keys, on the other hand, follow a discrete Gaussian distribution with standard deviation  $\sigma = 1.17\sqrt{\frac{q}{m+n}}$ , allowing us to compute  $\mathbb{E}[\|(\mathbf{e}, \mathbf{s})\|] = 1.17\sqrt{q}$  exactly.

## 2.4 The Primal Lattice Reduction Attack

**Definition 2.9 (LWE Lattice).** For an LWE instance  $(\mathbf{A}, \mathbf{b}, q)$ , where  $\mathbf{A} \in \mathbb{Z}^{n \times m}$ , we define the corresponding LWE lattice  $\Lambda^{\text{LWE}}$  as the lattice generated by the rows of the following basis matrix

$$\mathbf{B}^{\text{LWE}} := \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{b} & \mathbf{0} & 1 \end{pmatrix}. \quad (3)$$

Equivalently,  $\Lambda^{\text{LWE}}$  is defined as

$$\Lambda^{\text{LWE}} := \left\{ (\mathbf{x}, \mathbf{y}, t) \in \mathbb{Z}^m \times \mathbb{Z}^n \times \mathbb{Z} \mid \mathbf{x} \equiv \mathbf{y}\mathbf{A} + t\mathbf{b} \pmod{q} \right\}.$$

One can easily verify that the LWE lattice contains the vector

$$\mathbf{t} := (-\mathbf{e}, \mathbf{s}, -1) \in \Lambda^{\text{LWE}}. \quad (4)$$

Since the coordinates of  $\mathbf{s}$  and  $\mathbf{e}$  follow in practice a distribution with zero mean and small standard deviation  $\sigma$ , we have by Lemma 2.1

$$\mathbb{E}[\|(\mathbf{e}, \mathbf{s})\|] \leq \sigma\sqrt{m+n}.$$

For typical parameters (see Table 2), the expected norm of  $\mathbf{t}$  is therefore significantly shorter than what the Gaussian heuristic  $\text{gh}(\Lambda^{\text{LWE}})$  predicts for  $\lambda_1(\Lambda^{\text{LWE}})$ . Accordingly,  $\mathbf{t}$  is likely a shortest vector of  $\Lambda^{\text{LWE}}$ .

The *primal lattice reduction attack* solves the LWE problem by running the BKZ algorithm [Sch87] on  $\mathbf{B}^{\text{LWE}}$  to search for a shortest vector of  $\Lambda^{\text{LWE}}$ .

**Complexity.** The complexity of the primal lattice reduction attack is usually measured in the *Core-SVP* model, as introduced in [ADPS16]. In this model, one only estimates the so-called *BKZ-blocksize* at which BKZ will successfully recover  $\mathbf{t}$  from  $\Lambda^{\text{LWE}}$ . The blocksize is the most important parameter for the runtime of BKZ. Running the algorithm with blocksize  $\beta$  takes time at least

$$2^{0.292\beta + o(\beta)}.$$

Worth noting, for  $\beta = 2$  the BKZ algorithm is (essentially) identical to the famous LLL algorithm [LLL82].

Estimating the exact required blocksize is still an active area of research. The current state of the art is heavily based on heuristics and experimental observations. We refer the reader to the survey of Albrecht and Ducas [AD21] for a nice overview. The *Leaky-LWE estimator* from [DDGR20] currently provides the most accurate estimates for the required blocksize.

For our purposes, it suffices to know that the complexity of BKZ for finding a shortest vector  $\mathbf{v}$  in a lattice  $\Lambda$  mainly depends on the following two parameters:

1. the lattice dimension  $\dim \Lambda$ ,
2. the so-called *gap*  $\frac{\|\mathbf{v}\|}{\text{gh}(\Lambda)}$ .

The smaller the above two parameters get, the smaller is the necessary blocksize for BKZ to recover  $\mathbf{v}$  from  $\Lambda$ , i.e., BKZ performs the better, the smaller the dimension and the length of  $\mathbf{v}$  get, and the larger the determinant of  $\Lambda$  gets.

**The Embedding Factor.** In the typical setting, where both secret and error follow a distribution with zero mean and (known) standard deviation  $\sigma$ , one can slightly improve the lattice basis  $\mathbf{B}^{\text{LWE}}$  by replacing the so-called *embedding factor*, i.e., the 1 in the bottom right of  $\mathbf{B}^{\text{LWE}}$ , by  $\sigma$ . (This slightly decreases the gap of  $\Lambda^{\text{LWE}}$ .) Additionally, if the distribution has a non-zero mean  $\mu \neq 0$ , then the vectors  $\mathbf{b}$  and  $\mathbf{0}$  in the last row of  $\mathbf{B}^{\text{LWE}}$  should be replaced by  $\mathbf{b} - \boldsymbol{\mu}^m$  and  $\boldsymbol{\mu}^n$ , respectively, where  $\boldsymbol{\mu}^i := (\mu, \dots, \mu) \in \mathbb{Z}^i$ .

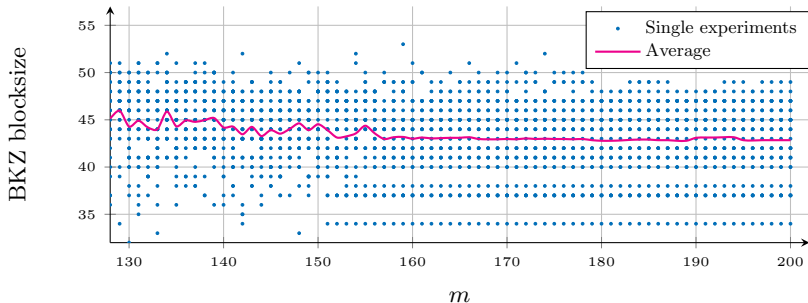
## 2.5 Ignoring LWE Samples

By removing columns from the LWE matrix  $\mathbf{A}$  and accordingly updating the lattice basis  $\mathbf{B}^{\text{LWE}}$ , we can easily decrease the dimension of  $\Lambda^{\text{LWE}}$ , while still keeping the secret  $\mathbf{s}$  in the lattice. In the literature, this technique is commonly known as *ignoring LWE samples*.

For typical parameters, the current estimators suggest that applying this technique decreases the required blocksize for the primal attack. For instance, for KYBER-768, the leaky LWE estimator suggests that ignoring 70 samples minimizes the required blocksize.

However, as discussed in [DDGR20, Remark 30], it is not the case that (manually) ignoring samples actually decreases the required blocksize in practice. In fact, when adding too many samples, the estimators simply start to overestimate the required blocksize, but the actual blocksize necessary in practice will not increase. (See Figure 1 for an illustration of this phenomenon.)

This is caused by the fact that the estimators currently do not capture the phenomenon that BKZ can ignore unnecessary samples *on its own*. (See again [DDGR20, Remark 30] for an explanation.) For simplicity, one can therefore always use all available LWE samples, i.e., keep  $\mathbf{A}$  unchanged, and let BKZ perform the optimization on its own.



**Fig. 1.** Required blocksize for the primal attack on 32 random KYBER-like LWE instances with  $n = 128$ ,  $q = 3329$ ,  $\|(\mathbf{e}, \mathbf{s})\| \approx 1.22 \cdot \sqrt{m+n}$  and varying  $m \in \{128, \dots, 200\}$ . For all  $m \geq 157$  we require an average blocksize of roughly 43.

## 2.6 LWE Hints

In this section we recall the definition of *LWE hints*, as first introduced by Dachman-Soled, Ducas, Gong and Rossi (DDGR) in [DDGR20].

**Definition 2.10 (LWE Hints).** Let  $\mathbf{s} \in \mathbb{Z}_q^n$  be an LWE secret. We define the following LWE hints for  $\mathbf{s}$ .

1. A tuple  $\bar{\mathbf{v}} = (\mathbf{v}, \ell) \in \mathbb{Z}^n \times \mathbb{Z}$  with

$$\langle \mathbf{v}, \mathbf{s} \rangle = \ell$$

is called a *perfect hint*.

2. A tuple  $\bar{\mathbf{v}} = (\mathbf{v}, \ell, m) \in \mathbb{Z}^n \times \mathbb{Z} \times \mathbb{N}$  with

$$\langle \mathbf{v}, \mathbf{s} \rangle \equiv \ell \pmod{m}$$

is called a *modular hint*. If  $m = q$ , we call  $\bar{\mathbf{v}}$  a *mod- $q$  hint*.

As discussed in the introduction, we slightly deviate from the original definition in the DDGR framework.

First, the DDGR framework defines hints more generally for both LWE error and secret. However, we restrict ourselves to *secret-only* hints. Second, DDGR also define a noisy variant of perfect hints, called *approximate hints*. It is an open problem to adapt our framework for this type of hints. Third, DDGR define a fourth type of hints, called *short vector hints*. However, short vector hints are of a very different nature than perfect, modular and approximate hints. In particular, as noted in [DDGR20, Section 4.5], these are not expected to be obtained via side channels, but rather *by design*. For integrating approximate and short vector hints we do not propose any new techniques.

### 3 Integrating Mod- $q$ Hints

Let us first restrict ourselves to modular hints  $\bar{\mathbf{v}} = (\mathbf{v}, \ell, q)$ , which we call mod- $q$  hints. The case of general modular hints is analyzed in Section 5.

Since all operations in LWE-based schemes are performed modulo  $q$ , we consider leakage of mod- $q$  hints practically highly relevant. Therefore, mod- $q$  hints deserve a more in-depth analysis.

The downside of the simple methodology introduced in this section is that it cannot easily be combined with the perfect hint framework from Sections 4. If one obtains mod- $q$  and perfect hints *together*, then one has to use the more powerful general modular hint approach from Section 5.

**Secret-only hints.** Recall that in our work we use *secret-only* hints  $(\mathbf{v}, \ell, q)$  satisfying  $\langle \mathbf{v}, \mathbf{s} \rangle \equiv \ell \pmod{q}$ . In contrast, [DDGR20] uses *secret-error* hints  $(\mathbf{v}, \mathbf{w}, \ell, q)$  satisfying  $\langle (\mathbf{v}, \mathbf{w}), (\mathbf{s}, \mathbf{e}) \rangle \equiv \ell \pmod{q}$ . In the full version of the paper [MN23], we show that the more general secret-error hints form in the mod- $q$  case equivalence classes with the following two properties.

- (1) Each equivalence class contains exactly one representative with  $\mathbf{w} = \mathbf{0}$ , i.e., a secret-only hint.
- (2) Integrating more than one hint from the same equivalence class does not improve the resulting lattice basis.

By Property (1), we may work in the mod- $q$  setting without loss of generality with secret-only hints. By Property (2), it is also advised to exclude secret-error hints in the mod- $q$  setting for avoiding useless hints.

**Transforming LWE.** Suppose we are given mod- $q$  hints  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i, q)$ ,  $i = 1, \dots, k$  for some LWE instance  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  with  $n$ -dimensional secret  $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$  and error  $\mathbf{e} \in \mathbb{Z}_q^m$ . Then we construct via linear algebra an LWE instance  $(\hat{\mathbf{A}}, \hat{\mathbf{b}}, q) \in \mathbb{Z}_q^{(n-k) \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  with

- $(n - k)$ -dimensional secret  $\hat{\mathbf{s}} = (s_{k+1}, \dots, s_n)$ ,
- and the same error  $\mathbf{e} \in \mathbb{Z}_q^m$ .

In particular, we decrease the dimension by  $k$ , while leaving the number of samples  $m$  unchanged, thereby increasing the sample/dimension ratio from  $m/n$  to  $m/(n - k)$ . Other works that addressed mod- $q$  hints to reduce the LWE-dimension either addressed the restrictive case of standard unit vectors (that directly provide coordinates of  $\mathbf{s}$  and therefore also can be considered as perfect hints) [EMVW22], or transformed into a large norm secret [WWX22], unsuited for lattice reduction.

In lattice language, our mod- $q$  hint transformation of the LWE instance improves the primal lattice reduction attack from Section 2.4 by

- (1) decreasing the dimension of  $\Lambda^{\text{LWE}}$  by  $k$ ,

- (2) decreasing the length of the secret vector  $\mathbf{t}$  from Equation (4),
- (3) while preserving the determinant of  $\Lambda^{\text{LWE}}$ .

*Remark 3.1.* As in the DDGR framework, we assume throughout this work that our hints  $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k$  are linearly independent. In particular, we assume  $k \leq n$ . We would like to stress that linear independence is a very natural restriction. If there was a framework that could improve the primal lattice reduction attack via linearly *dependent* hints, then LWE would not be hard, since after guessing one initial perfect/modular hint an attacker can easily generate arbitrarily many linearly dependent hints.

### 3.1 Mod- $q$ Hints Provide LWE Dimension Reduction

Throughout this section, we assume that  $q$  is prime, which is true for all LWE-based schemes addressed in this work, only NTRU uses a power-of-two  $q$ . At the end of the section, we discuss in Remark 3.4 the small necessary adaptation for NTRU.

Let us begin by defining some useful matrix notation.

**Definition 3.2 (Hint Matrix).** Let  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i) \in \mathbb{Z}^n \times \mathbb{Z}$ , where  $i = 1, \dots, k$ . We define the corresponding hint matrix as

$$\text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k) := \begin{pmatrix} | & & | \\ \mathbf{v}_1^T & \cdots & \mathbf{v}_k^T \\ | & & | \\ \ell_1 & \cdots & \ell_k \end{pmatrix} = \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\ell} \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times k}. \quad (5)$$

**Idea of Dimension Reduction.** The hint matrix from Definition 3.2 satisfies

$$(\mathbf{s}, -1) \cdot \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\ell} \end{pmatrix} \equiv \mathbf{0}^k \pmod{q}. \quad (6)$$

Since the hint vectors  $\mathbf{v}_1, \dots, \mathbf{v}_k$  are linearly independent, there exists a full rank  $k \times k$  submatrix of  $\mathbf{V}$ . For ease of notation, we assume that the first  $k$  rows of  $\mathbf{V}$  form a full rank matrix  $\mathbf{V}_1$ . This can always be achieved by row permutation of  $\mathbf{V}$ , where  $\mathbf{s}$  has to be permuted accordingly.

Let  $\mathbf{V}_1^{-1}$  be the inverse of  $\mathbf{V}_1$  in  $\mathbb{F}_q^{k \times k}$ . Multiplying Equation (6) by  $\mathbf{V}_1^{-1}$  gives

$$(\mathbf{s}, -1) \cdot \begin{pmatrix} \mathbf{I}_k \\ \mathbf{V}_2 \mathbf{V}_1^{-1} \\ \boldsymbol{\ell} \mathbf{V}_1^{-1} \end{pmatrix} \equiv (\mathbf{s}, -1) \cdot \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\ell} \end{pmatrix} \cdot \mathbf{V}_1^{-1} \equiv \mathbf{0}^k \pmod{q}. \quad (7)$$

Let  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  be an LWE instance with secret  $\mathbf{s}$  and error  $\mathbf{e}$ . Write

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \text{ with } \mathbf{A}_1 \in \mathbb{Z}_q^{k \times m}, \mathbf{A}_2 \in \mathbb{Z}_q^{(n-k) \times m}.$$

Then

$$(\mathbf{s}, -1) \cdot \left( \begin{array}{c|c} \mathbf{I}_k & \mathbf{A}_1 \\ \hline \mathbf{V}_2 \mathbf{V}_1^{-1} & \mathbf{A}_2 \\ \hline \ell \mathbf{V}_1^{-1} & \mathbf{b} \end{array} \right) \equiv (\mathbf{0}^k, -\mathbf{e}) \pmod{q}.$$

Using column operations, we now use the identity matrix  $\mathbf{I}_k$  to eliminate  $\mathbf{A}_1$ , i.e., we eliminate the first  $k$  rows of  $\mathbf{A}$ . Notice that since our  $k$  modular hints are error-free, this operation *does not increase* the error vector  $\mathbf{e}$ . We obtain

$$(\mathbf{s}, -1) \cdot \left( \begin{array}{c|c} \mathbf{I}_k & \mathbf{0} \\ \hline \mathbf{V}_2 \mathbf{V}_1^{-1} & \hat{\mathbf{A}} \\ \hline \ell \mathbf{V}_1^{-1} & \hat{\mathbf{b}} \end{array} \right) \equiv (\mathbf{0}^k, -\mathbf{e}) \pmod{q}.$$

Eventually,  $(\hat{\mathbf{A}}, \hat{\mathbf{b}}, q)$  is our new LWE instance with the  $(n - k)$ -dimensional secret  $\hat{\mathbf{s}} = (s_{k+1}, \dots, s_n)$ . Thus, we used our  $k$  mod- $q$  hints to eliminate the first  $k$  coordinates of  $\mathbf{s}$ .

**Reconstruction of  $\mathbf{s}$ .** Our transformation of  $\mathbf{s}$  to  $\hat{\mathbf{s}}$  eliminates the first  $k$  coordinates  $(s_1, \dots, s_k)$ . By Equation (7) we have

$$(s_1, \dots, s_k) \equiv -\hat{\mathbf{s}} \mathbf{V}_2 \mathbf{V}_1^{-1} + \ell \mathbf{V}_1^{-1} \pmod{q},$$

which allows us to easily reconstruct the remaining  $k$  coordinates when given  $\hat{\mathbf{s}}$ .

The following theorem details all required linear algebra transformations in our LWE dimension reduction.

**Theorem 3.3.** *Let  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  be an LWE instance with  $n$ -dimensional secret  $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$  and error  $\mathbf{e} \in \mathbb{Z}_q^m$ . Let  $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k$  be mod- $q$  hints with hint matrix  $\text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k) = [\mathbf{V}, \ell] \in \mathbb{Z}_q^{(n+1) \times k}$ . Let us denote  $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$ ,  $\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2]$  with  $\mathbf{A}_1 \in \mathbb{Z}_q^{k \times m}$ ,  $\mathbf{V}_1 \in \mathbb{Z}_q^{k \times k}$ . Then  $(\hat{\mathbf{A}}, \hat{\mathbf{b}}, q) \in \mathbb{Z}_q^{(n-k) \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  with*

$$\begin{aligned} \hat{\mathbf{A}} &\equiv \mathbf{A}_2 - \mathbf{V}_2 \mathbf{V}_1^{-1} \mathbf{A}_1 \pmod{q}, \\ \hat{\mathbf{b}} &\equiv \mathbf{b} - \ell \mathbf{V}_1^{-1} \mathbf{A}_1 \pmod{q} \end{aligned}$$

*is an LWE instance with secret  $\hat{\mathbf{s}} = (s_{k+1}, \dots, s_n) \in \mathbb{Z}_q^{n-k}$  and error  $\mathbf{e} \in \mathbb{Z}_q^m$ .*

*Proof.* Let  $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$  with  $\mathbf{s}_2 = \hat{\mathbf{s}} \in \mathbb{Z}_q^{n-k}$ . We have to show that  $\hat{\mathbf{s}}\hat{\mathbf{A}} \equiv \hat{\mathbf{b}} - \mathbf{e} \pmod{q}$ . Using the definition of  $\hat{\mathbf{A}}$  we obtain

$$\hat{\mathbf{s}}\hat{\mathbf{A}} \equiv \mathbf{s}_2\mathbf{A}_2 - \mathbf{s}_2\mathbf{V}_2\mathbf{V}_1^{-1}\mathbf{A}_1 \pmod{q}.$$

By Equation (7) we have  $\mathbf{s}\mathbf{V}\mathbf{V}_1^{-1} \equiv \ell\mathbf{V}_1^{-1}$ . Since also  $\mathbf{s}\mathbf{V}\mathbf{V}_1^{-1} \equiv \mathbf{s}_1 + \mathbf{s}_2\mathbf{V}_2\mathbf{V}_1^{-1}$ , we obtain  $\mathbf{s}_2\mathbf{V}_2\mathbf{V}_1^{-1} \equiv \ell\mathbf{V}_1^{-1} - \mathbf{s}_1$ . This implies

$$\begin{aligned} \hat{\mathbf{s}}\hat{\mathbf{A}} &\equiv \mathbf{s}_2\mathbf{A}_2 - (\ell\mathbf{V}_1^{-1} - \mathbf{s}_1)\mathbf{A}_1 \equiv \mathbf{s}_1\mathbf{A}_1 + \mathbf{s}_2\mathbf{A}_2 - \ell\mathbf{V}_1^{-1}\mathbf{A}_1 \pmod{q} \\ &\equiv \mathbf{s}\mathbf{A} + \hat{\mathbf{b}} - \mathbf{b} \equiv \hat{\mathbf{b}} - \mathbf{e} \pmod{q}. \quad \square \end{aligned}$$

*Remark 3.4.* For the NTRU case with power-of-two  $q$ , we require that some  $k \times k$  submatrix  $\mathbf{V}_1$  of  $\mathbf{V}$  is invertible over  $\mathbb{F}_2$ , which implies invertibility over  $\mathbb{Z}_q$ . For  $k \ll n$  this happens with overwhelming probability.

## 4 Integrating Perfect Hints

Suppose we are given  $k$  perfect hints  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i) \in \mathbb{Z}^n \times \mathbb{Z}$ ,  $i = 1, \dots, k$ . In this section, we introduce our new approach for incorporating perfect hints, that improves the primal lattice reduction attack by

- (1) decreasing the dimension of the LWE lattice  $\Lambda^{\text{LWE}}$  by  $k$  (Section 4.1),
- (2) increasing its determinant by a factor  $\det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$  (Section 4.2), while
- (3) preserving the length of the secret vector  $\mathbf{t}$  from Equation (4) (Section 4.1).

Additionally, we show that the effect of the integration of perfect hints is the exact same as in the original DDGR framework (Section 4.2). However, in contrast to DDGR's approach, our novel and simplified view allows for a highly efficient implementation (Section 4.3). We provide a run time comparison with DDGR in Section 6.

### 4.1 Decreasing the Dimension of $\Lambda^{\text{LWE}}$ , while Preserving $\|\mathbf{t}\|$

**Embedding Hints into  $\Lambda^{\text{LWE}}$ .** Let us first *embed* the perfect hints into our lattice basis. Let  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  be an LWE instance with secret  $\mathbf{s} \in \mathbb{Z}_q^n$ . The main idea behind our new approach is to view the perfect hints

$$\ell_i = \langle \mathbf{v}_i, \mathbf{s} \rangle$$

as *error-free* LWE samples *without* reduction modulo  $q$ . A very natural approach for embedding the perfect hints into our lattice is then to construct a hint matrix  $\mathbf{H} = \text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k) = (\mathbf{V}, \ell)$  (Definition 3.2) and to generalize the definition of the LWE lattice  $\Lambda^{\text{LWE}}$  (Definition 2.9) as follows.



**Definition 4.1 (Hint Lattice).** Let  $(\mathbf{A}, \mathbf{b}, q)$  be an LWE instance, where  $\mathbf{A} \in \mathbb{Z}^{n \times m}$ , and let  $\mathbf{H} = (\mathbf{V}, \ell) \in \mathbb{Z}^{(n+1) \times k}$  be a hint matrix. The corresponding hint lattice  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  is defined as the lattice generated by the following matrix:

$$\mathbf{B}_{\mathbf{H}}^{\text{LWE}} := \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{V} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{b} & \ell & \mathbf{0} & 1 \end{pmatrix} \in \mathbb{Z}^{(m+n+1) \times (m+k+n+1)}. \quad (8)$$

Notice that we did not change the lattice dimension yet: Even though the hint lattice  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  lies in the larger vector space  $\mathbb{R}^{m+k+n+1}$  (as opposed to  $\Lambda^{\text{LWE}}$  lying in  $\mathbb{R}^{m+n+1}$ ), the dimension of the lattice remains

$$\dim \Lambda_{\mathbf{H}}^{\text{LWE}} = \dim \Lambda^{\text{LWE}} = m + n + 1.$$

**Decreasing dimension of  $\Lambda_{\mathbf{H}}^{\text{LWE}}$ .** By definition of the hint matrix  $\mathbf{H} = (\mathbf{V}, \ell)$ , the LWE secret  $\mathbf{s}$  satisfies

$$(\mathbf{s}, -1) \cdot \begin{pmatrix} \mathbf{V} \\ \ell \end{pmatrix} = (\langle \mathbf{v}_1, \mathbf{s} \rangle - \ell_1, \dots, \langle \mathbf{v}_k, \mathbf{s} \rangle - \ell_k) = \mathbf{0}^k.$$

From that, it easily follows that the hint lattice  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  contains the short vector

$$\mathbf{t}_{\mathbf{H}} := (-\mathbf{e}, \mathbf{0}^k, \mathbf{s}, -1),$$

which has the same length as the original secret vector  $\mathbf{t}$ , defined in Equation (4).

To reduce the dimension of our lattice by  $k$ , we now simply use the fact that the coordinates of  $\mathbf{t}_{\mathbf{H}}$  at positions  $m+1$  to  $m+k$  are zero. Instead of searching for  $\mathbf{t}_{\mathbf{H}}$  in  $\Lambda_{\mathbf{H}}^{\text{LWE}}$ , we simply search in the  $(m+n+1-k)$ -dimensional<sup>2</sup> sublattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}} \subset \Lambda_{\mathbf{H}}^{\text{LWE}}$  as defined below:

$$\begin{aligned} \Lambda_{\mathbf{H},k}^{\text{LWE}} &:= \{(v_1, \dots, v_{n+m+k+1}) \in \Lambda_{\mathbf{H}}^{\text{LWE}} \mid v_{m+1} = \dots = v_{m+k} = 0\} \\ &= \Lambda_{\mathbf{H}}^{\text{LWE}} \cap \mathbf{e}_{m+1}^\perp \cap \dots \cap \mathbf{e}_{m+k}^\perp. \end{aligned}$$

## 4.2 Perfect Hints Increase $\det \Lambda^{\text{LWE}}$ by $\det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$

To integrate  $k$  perfect hints  $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k$ , DDGR suggest to intersect the LWE lattice  $\Lambda^{\text{LWE}}$  with the subspace orthogonal to all  $\bar{\mathbf{v}}_i$ 's, i.e., to work with the lattice

$$\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}} := \Lambda^{\text{LWE}} \cap (\mathbf{0}^m, \bar{\mathbf{v}}_1)^\perp \cap \dots \cap (\mathbf{0}^m, \bar{\mathbf{v}}_k)^\perp.$$

<sup>2</sup> Here we require the hints to be linearly independent. More generally, we have  $\dim \Lambda_{\mathbf{H},k}^{\text{LWE}} = n + m + 1 - \text{rank}_{\mathbb{R}}(\mathbf{H})$ .

As shown by DDGR, this reduces the dimension of the lattice by  $k$  and increases the determinant by a factor of roughly  $\|\bar{\mathbf{v}}_1\| \cdot \dots \cdot \|\bar{\mathbf{v}}_k\|$ .<sup>3</sup>

At first glance, the DDGR approach may seem complementary to our approach, where we first construct the hint lattice  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  (lying in a different vector space than  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$ ) and then intersect it with the subspace orthogonal to the standard basis vectors  $\mathbf{e}_{m+1}, \dots, \mathbf{e}_{m+k}$ .

While we already showed that our approach also reduces the lattice dimension by  $k$ , it is not so obvious, how the determinant of our lattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  compares with that of  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$ . In particular, it is unclear whether one lattice performs better than the other.

Interestingly, our Theorem 4.2 below shows, however, that our new lattice has the exact same determinant as DDGR's. In fact, Theorem 4.2 even shows something slightly stronger: The lattices  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$  and  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  are *isometric*, i.e., there is an isomorphism between them, that preserves their geometries. Hence, the lattices  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$  and  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  have the exact same *quality* from a cryptanalytic perspective.

More importantly, we show in Theorem 4.3 that our restriction to secret-only hints allows us to precisely estimate the determinant. We prove under a mild assumption that the determinant increases exactly by a factor of  $\det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$ , as opposed to DDGR's rough estimation of  $\|\bar{\mathbf{v}}_1\| \cdot \dots \cdot \|\bar{\mathbf{v}}_k\|$ .

**Theorem 4.2.** *Let  $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k$  be perfect hints with hint matrix  $\mathbf{H} = \text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$ . There exists an isometry from the hint sublattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  to  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$ . In particular,*

$$\begin{aligned} \dim \Lambda_{\mathbf{H},k}^{\text{LWE}} &= \dim \Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}, \text{ and} \\ \det \Lambda_{\mathbf{H},k}^{\text{LWE}} &= \det \Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}. \end{aligned}$$

*Proof.* Let  $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{Z}^m \times \mathbb{Z}^{n+1}$  and let

$$\begin{aligned} \mathbf{x} &:= \mathbf{u} \cdot \mathbf{B}^{\text{LWE}} \in \Lambda^{\text{LWE}}, \\ \mathbf{y} &:= \mathbf{u} \cdot \mathbf{B}_{\mathbf{H}}^{\text{LWE}} \in \Lambda_{\mathbf{H}}^{\text{LWE}}, \end{aligned}$$

where  $\mathbf{B}^{\text{LWE}}$  and  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$  are defined as in Equations (3) and (8), respectively. From the shapes of  $\mathbf{B}^{\text{LWE}}$  and  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$  it easily follows that

$$\mathbf{x} = (\mathbf{w}, \mathbf{u}_2), \tag{9}$$

$$\mathbf{y} = (\mathbf{w}, \mathbf{u}_2 \cdot \mathbf{H}, \mathbf{u}_2), \tag{10}$$

for some  $\mathbf{w} \in \mathbb{Z}^m$ . Comparing the definitions of  $\Lambda_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}}$  and  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  with Equations (9) and (10), we obtain the following chain of equivalences:

$$\begin{aligned} \mathbf{x} \in \Lambda^{\text{DDGR}} &\iff \langle \mathbf{u}_2, \bar{\mathbf{v}}_i \rangle = 0, \text{ for all } i = 1, \dots, k \\ &\iff \mathbf{u}_2 \cdot \mathbf{H} = \mathbf{0}^k \\ &\iff \mathbf{y} \in \Lambda_{\mathbf{H},k}^{\text{LWE}}. \end{aligned}$$

<sup>3</sup> The DDGR estimate is correct under some primitivity condition (see [DDGR20, Section 4.1]) and the assumption that the hints are not too far from orthogonal (see [DDGR20, Remark 25]).

This, in turn, implies that

$$\varphi : A_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}} \rightarrow A_{\mathbf{H}, k}^{\text{LWE}}, (x_1, \dots, x_{m+n+1}) \mapsto (x_1, \dots, x_m, \mathbf{0}^k, x_{m+1}, \dots, x_{m+n+1})$$

is an isometry, which proves the theorem.  $\square$

**Theorem 4.3.** *Let  $\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k$  be (secret-only) perfect hints with hint matrix  $\mathbf{H} = \text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$ . Suppose  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$ . Then it holds that*

$$\det A_{\mathbf{H}, k}^{\text{LWE}} = \det A_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}} = \det A^{\text{LWE}} \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k).$$

*Proof.* The proof uses the technique from [DDGR20, Lemma 12]. Let  $U := \bar{\mathbf{v}}_1^\perp \cap \dots \cap \bar{\mathbf{v}}_k^\perp$ . From the shape of the basis matrix  $\mathbf{B}^{\text{LWE}}$  (see Equation (3)), it easily follows that<sup>4</sup>

$$\det A_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}} = q^m \cdot \det(\mathbb{Z}^{n+1} \cap U) = \det(A^{\text{LWE}}) \cdot \det(\mathbb{Z}^{n+1} \cap U). \quad (11)$$

Using Lemma 2.5 and the fact that  $\mathbb{Z}^{n+1}$  is self-dual, we obtain

$$\mathbb{Z}^{n+1} \cap U = (\pi_U(\mathbb{Z}^{n+1}))^*,$$

which together with Equation (2) and Lemma 2.6 gives

$$\det(\mathbb{Z}^{n+1} \cap U) = \frac{1}{\det(\pi_U(\mathbb{Z}^{n+1}))} = \frac{\det(\mathbb{Z}^{n+1} \cap U^\perp)}{\det(\mathbb{Z}^{n+1})} = \det(\mathbb{Z}^{n+1} \cap U^\perp). \quad (12)$$

By assumption, the rows of  $\mathbf{H}$  span the integer lattice  $\mathbb{Z}^k$ . Together with Lemma 2.3 and the definition of  $\mathbf{H}$  (Definition 3.2) this implies that  $\{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k\}$  is primitive with respect to  $\mathbb{Z}^{n+1}$ , and thus

$$\mathbb{Z}^{n+1} \cap U^\perp = \mathbb{Z}^{n+1} \cap \text{span}_{\mathbb{R}}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k) = \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k). \quad (13)$$

Combining Equations (11), (12) and (13), we obtain

$$\det A_{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k}^{\text{DDGR}} = \det(A^{\text{LWE}}) \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k),$$

which together with Theorem 4.2 proves the theorem.  $\square$

**The condition  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$ .** Theorem 4.3 requires the hint matrix  $\mathbf{H}$  to generate the integer lattice  $\mathbb{Z}^k$ . In practice, we can expect that  $\mathbf{H}$  behaves like a random matrix. If  $k$  is significantly smaller than  $n$ , then Lemma 2.4 shows that  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$  holds with very high probability. Hence, we expect that Theorem 4.3 typically applies in practice.

In the case of a single perfect hint, i.e.,  $k = 1$ , the condition  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$  simply requires that the greatest common divisor of the entries of  $\bar{\mathbf{v}}_1$  equals 1. From the shape of the basis matrix  $\mathbf{B}^{\text{LWE}}$  (Equation (3)), it is easy to see that for a secret only hint this is equivalent to requiring that  $\bar{\mathbf{v}}_1$  is primitive with respect to the dual  $(A^{\text{LWE}})^*$ . Hence, for the case of  $k = 1$ , our new Theorem 4.3 boils down to DDGR's original result.

<sup>4</sup> Equation (11) would become false, if we would allow secret-error hints.

**Corollary 4.4** ([DDGR20, Lemma 12]). *Let  $\bar{\mathbf{v}}_1$  be a (secret-only) perfect hint with hint matrix  $\mathbf{H} = \text{Hint}(\bar{\mathbf{v}}_1)$ . Suppose  $\bar{\mathbf{v}}_1$  is primitive with respect to the dual lattice  $(\Lambda^{\text{LWE}})^*$ . Then it holds that*

$$\det \Lambda_{\mathbf{H},1}^{\text{LWE}} = \Lambda_{\bar{\mathbf{v}}_1}^{\text{DDGR}} = \det \Lambda^{\text{LWE}} \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1) = \det \Lambda^{\text{LWE}} \cdot \|\bar{\mathbf{v}}_1\|.$$

### 4.3 Computing a basis for $\Lambda_{\mathbf{H},k}^{\text{LWE}}$

To be able to search for the secret vector  $\mathbf{t}_{\mathbf{H}}$  in the sublattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}} \subset \Lambda_{\mathbf{H}}^{\text{LWE}}$ , we of course first have to compute a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ . To this end, we introduce our new algorithm CONSTRUCT-SUBLATTICE (Algorithm 1). The runtime of CONSTRUCT-SUBLATTICE is dominated by one call to LLL in dimension  $n+1$ , and by multiplying two matrices in dimensions  $(n+1) \times (n+1)$  and  $(n+1) \times m$  – making the algorithm highly practical.

---

#### Algorithm 1: CONSTRUCT-SUBLATTICE

---

**Input:** An LWE instance  $(\mathbf{A}, \mathbf{b}, q)$ , where  $\mathbf{A} \in \mathbb{Z}^{n \times m}$ , a hint matrix  $\mathbf{H} := \text{Hint}((\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)) \in \mathbb{Z}^{(n+1) \times k}$ , and a scaling parameter  $c > 0$ .

**Output:** A basis of  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  or FAIL.

- 1 Multiply the first  $k$  columns of  $\mathbf{H}$  by  $\lceil 2^{\frac{n}{2}} \cdot c \rceil$ . Denote the resulting matrix by  $\tilde{\mathbf{H}}$ .
- 2 Run the LLL algorithm on  $(\tilde{\mathbf{H}} \mid \mathbf{I}_{n+1})$  to obtain a reduced basis  $\mathbf{H}_{\text{LLL}} \in \mathbb{Z}^{(n+1) \times (n+k+1)}$  and a unimodular matrix  $\mathbf{U} \in \mathbb{Z}^{(n+1) \times (n+1)}$ , such that

$$\mathbf{H}_{\text{LLL}} = \mathbf{U} \cdot (\tilde{\mathbf{H}} \mid \mathbf{I}_{n+1}).$$

- 3 if the upper-left  $(n+1-k) \times k$  block of  $\mathbf{H}_{\text{LLL}}$  is non-zero then

4 |   **Return** FAIL.

5 else

- 6 |   Compute a matrix  $\tilde{\mathbf{A}}$  as follows:

$$\tilde{\mathbf{A}} := \mathbf{U} \cdot \begin{pmatrix} \mathbf{A} \\ \mathbf{b} \end{pmatrix}.$$

- 7 |   Construct the following matrix:

$$\mathbf{B} := \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} \\ \tilde{\mathbf{A}} & \mathbf{H}_{\text{LLL}} \end{pmatrix} \in \mathbb{Z}^{(m+n+1) \times (m+k+n+1)}.$$

- 8 |   Delete the last  $k$  rows of  $\mathbf{B}$ .

9 |   **Return** the resulting matrix.

---

The main idea behind our algorithm is to appropriately scale the basis matrix  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$  of  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  by some *scaling parameter*  $c$ , such that LLL can only find lattice

vectors in  $\Lambda_{\mathbf{H}}^{\text{LWE}}$ , which have zeros in the coordinates  $m+1$  to  $m+k$ . Additionally, we exploit the fact that the  $q$ -vectors (i.e., the first  $m$  rows of  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$ , as defined in Equation (5)) already belong to the sublattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ .

In Theorem 4.5 below, we prove a rigorous – yet impractical – bound on the scaling parameter, for which CONSTRUCT-SUBLATTICE provably returns a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ . Building on the theorem, we then derive a heuristic bound on the scaling parameter, that works well in practice.

**Theorem 4.5.** *Let  $\mathbf{H} := \text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$  be a hint matrix. Let  $U := \mathbf{e}_1^\perp \cap \dots \cap \mathbf{e}_m^\perp \subset \mathbb{R}^{m+k+n+1}$  be the subspace orthogonal to the first  $m$  standard basis vectors. If we call CONSTRUCT-SUBLATTICE with scaling parameter  $c \geq \lambda_{n+1-k}(\pi_U(\Lambda_{\mathbf{H},k}^{\text{LWE}}))$ , then the algorithm outputs a basis of  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ .*

*Proof.* Let us first show that on input  $c \geq \lambda_{n+1-k}(\pi_U(\Lambda_{\mathbf{H},k}^{\text{LWE}}))$ , the algorithm does not output FAIL. Let  $c' := \lceil 2^{\frac{n}{2}} \cdot c \rceil$ . By construction, every row  $\mathbf{h}_i$  of  $\mathbf{H}_{\text{LLL}}$  is of the form

$$\mathbf{h}_i = (c' \cdot h_{i,1}, \dots, c' \cdot h_{i,k}, h_{i,k+1}, \dots, h_{i,n+k+1}). \quad (14)$$

Since  $\mathbf{H}_{\text{LLL}}$  is LLL-reduced, it holds that

$$\|\mathbf{h}_i\| \leq 2^{\frac{n}{2}} \cdot \lambda_i(\mathcal{L}(\tilde{\mathbf{H}})), \quad (15)$$

see [LLL82, Proposition 1.12].

From the shape of the basis matrix  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$  (Equation (8)) and the definition of  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ , it easily follows that the  $(n+1-k)$ -dimensional lattice  $\pi_U(\Lambda_{\mathbf{H},k}^{\text{LWE}})$  is (isometric to) a sublattice of  $\mathcal{L}(\tilde{\mathbf{H}})$ . Together with Equation (15), this yields

$$\|\mathbf{h}_i\| \leq 2^{\frac{n}{2}} \cdot \lambda_i(\pi_U(\Lambda_{\mathbf{H},k}^{\text{LWE}})) \leq 2^{\frac{n}{2}} \cdot \lambda_{n+1-k}(\pi_U(\Lambda_{\mathbf{H},k}^{\text{LWE}})) \leq c', \quad (16)$$

for every  $i = 1, \dots, n+1-k$ .

Since by Equation (14), the first  $k$  coordinates of  $\mathbf{h}_i$  are multiples of  $c'$ , Equation (16) implies that these coordinates are, in fact, equal to zero. In particular, the upper-left  $(n+1-k) \times k$  block of  $\mathbf{H}_{\text{LLL}}$  is non-zero. Hence, the algorithm does not output FAIL.

It remains to show that the matrix returned in Step 9 indeed is a basis matrix for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ . Let  $\mathbf{U}$  be the unimodular matrix produced by Step 2. One can easily verify that the matrix  $\mathbf{B}$ , produced by Step 7 of the algorithm, is given by

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{U} \end{pmatrix} \cdot \mathbf{B}_{\mathbf{H}}^{\text{LWE}} \cdot \begin{pmatrix} \mathbf{I}_m & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & c' \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{n+1} \end{pmatrix}.$$

The matrix  $\mathbf{B}$  is thus obtained by scaling the columns  $m+1$  to  $m+k$  of a basis matrix of  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  by  $c'$ . Since by construction the first  $m+n+1-k$  rows are zero in the columns  $m+1$  to  $m+k$ , this shows that the matrix returned in Step 9 is a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ , as required.  $\square$

To use CONSTRUCT-SUBLATTICE in practice, Theorem 4.5 shows that we need to efficiently compute an upper bound  $c$  on the  $(n+1-k)$ -th successive minimum of  $\pi_U(A_{\mathbf{H},k}^{\text{LWE}})$ . Unfortunately, we can not hope to rigorously prove any useful upper bound on  $\lambda_{n+1-k}(\pi_U(A_{\mathbf{H},k}^{\text{LWE}}))$ .<sup>5</sup> However, we may heuristically assume that

$$\lambda_1 \approx \lambda_2 \approx \dots \approx \lambda_{n+1-k}$$

and then use the Gaussian heuristic<sup>6</sup>

$$\begin{aligned} \text{gh}(\pi_U(A_{\mathbf{H},k}^{\text{LWE}})) &= \sqrt{\frac{n+m+1-k}{2\pi e}} \cdot (\det \pi_U(A_{\mathbf{H},k}^{\text{LWE}}))^{1/(n+m+1-k)} \\ &= \sqrt{\frac{n+m+1-k}{2\pi e}} \cdot \left( \frac{\det A_{\mathbf{H},k}^{\text{LWE}}}{q^m} \right)^{1/(n+m+1-k)} \end{aligned} \quad (17)$$

as an upper bound on  $\lambda_{n+1-k}(\pi_U(A_{\mathbf{H},k}^{\text{LWE}}))$ . Making the additional assumption that  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$  (which is justified by Lemma 2.4), we obtain by Theorem 4.3

$$\det A_{\mathbf{H},k}^{\text{LWE}} = \det A^{\text{LWE}} \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k) = q^m \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k),$$

which then yields the following heuristic:

**Heuristic 4.6.** *Let  $\mathbf{H} := \text{Hint}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)$  be a hint matrix. If we call CONSTRUCT-SUBLATTICE with scaling parameter*

$$c = \sqrt{\frac{n+m+1-k}{2\pi e}} \cdot \det \mathcal{L}(\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_k)^{1/(n+m+1-k)},$$

*then the algorithm outputs a basis of  $A_{\mathbf{H},k}^{\text{LWE}}$ .*

We experimentally confirm correctness of Heuristic 4.6 in Section 7.

*Remark 4.7.* Instead of LLL-reducing  $(\tilde{\mathbf{H}} \mid \mathbf{I}_{n+1})$  in Step 2 of the algorithm, we could first reduce only  $\tilde{\mathbf{H}}$ , and after that apply the corresponding transformation matrix to the  $(n+1)$ -dimensional identity matrix. (Similarly, as we do with  $[\mathbf{A}, \mathbf{b}]$  in Step 6.) However, using  $(\tilde{\mathbf{H}} \mid \mathbf{I}_{n+1})$  has the benefit that the identity matrix forces LLL to take *small* linear combinations of the rows of  $\tilde{\mathbf{H}}$ . In particular, it increases the probability of LLL taking the particularly small linear combination  $(\mathbf{s}, -1)$  to create a zero in the first  $k$  coordinates. Whenever this happens, we can immediately read off the LWE secret from the basis. As we show in Section 7, in the regime of *too many hints*, this frequently occurs in practice.

*Remark 4.8.* More generally, given any lattice  $\Lambda \subset \mathbb{R}^d$  and a collection of standard basis vectors  $\{\mathbf{e}_i\}_{i \in I}$ ,  $I \subseteq \{1, \dots, d\}$ , the ideas behind CONSTRUCT-SUBLATTICE can easily be adapted to efficiently compute a basis of  $\Lambda \cap (\bigcap_{i \in I} \mathbf{e}_i^\perp)$ .

<sup>5</sup> E.g., we cannot hope to upper bound  $\lambda_{n+1-k}(\pi_U(A_{\mathbf{H},k}^{\text{LWE}}))$  in terms of the determinant of the lattice, since it is easy to construct examples, where  $\lambda_2$  is arbitrarily large, while the determinant is small.

<sup>6</sup> Equation (17) easily follows from the shape of  $\mathbf{B}_{\mathbf{H}}^{\text{LWE}}$  (see Equation (8)).

## 5 Integrating Modular Hints

Suppose we are given  $k$  modular hints  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i, m_i) \in \mathbb{Z}^n \times \mathbb{Z} \times \mathbb{N}$ , for  $i = 1, \dots, k$ . Our new algorithm for incorporating modular hints improves the primal lattice reduction attack by

- (1) increasing the determinant of  $\Lambda^{\text{LWE}}$  by a factor of  $\prod_{i=1}^k m_i$ ,
- (2) while leaving dimension of the lattice,
- (3) and norm of the secret vector  $\mathbf{t}$  from Equation (4) unchanged.

As in the perfect hint case, the effect of the integration of modular hints is thus the exact same as in the DDGR framework. However, since our approach uses our algorithm CONSTRUCT-SUBLATTICE (Algorithm 1) from Section 4.3, it is significantly more efficient than DDGR's. Yet, it is slightly less efficient than our approach for mod- $q$  hints from Section 3, which requires only elementary linear algebra.

As we discuss in Section 5.2, an advantage of our general modular hint approach over our mod- $q$  approach is, however, that it allows to easily combine *modular* hints with *perfect* hints, and to integrate both types of hints very efficiently in one stroke. We give a more in-depth comparison with the approach from Section 3 in Section 5.3.

### 5.1 Increasing $\det \Lambda^{\text{LWE}}$ , while Preserving $\dim \Lambda^{\text{LWE}}$ and $\|\mathbf{t}\|$

Let  $(\mathbf{A}, \mathbf{b}, q) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \times \mathbb{N}$  be an LWE instance with secret  $\mathbf{s} \in \mathbb{Z}_q^n$ , and let  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i, m_i) \in \mathbb{Z}^n \times \mathbb{Z} \times \mathbb{N}$  be modular hints such that

$$\langle \mathbf{v}_i, \mathbf{s} \rangle \equiv \ell_i \pmod{m_i}, \quad (18)$$

for  $i = 1, \dots, k$ .

Our main idea from Section 4 for integrating *perfect* hints is to view our hints as error-free LWE samples without reduction modulo  $q$ . For *modular hints* we now follow a very similar approach: We simply view the hints as error-free LWE samples with reduction modulo  $m_i$ . Apart from some minor modifications, our approach for modular hints then boils down to the perfect hint setting.

**Embedding Hints into  $\Lambda^{\text{LWE}}$ .** Let  $\bar{\mathbf{v}}'_i := (\mathbf{v}_i, \ell_i)$ . Similarly as in Section 4, we start by defining a hint matrix  $\mathbf{H} = \text{Hint}(\bar{\mathbf{v}}'_1, \dots, \bar{\mathbf{v}}'_k) = (\mathbf{V}, \boldsymbol{\ell}) \in \mathbb{Z}^{(n+1) \times k}$  (Definition 3.2). However, instead of using  $\mathbf{H}$  to directly construct the hint lattice  $\Lambda_{\mathbf{H}}^{\text{LWE}}$  from Definition 4.1 (as we would in the perfect hint setting), we first define an additional matrix

$$\mathbf{M} := \begin{pmatrix} m_1 & & \\ & \ddots & \\ & & m_k \end{pmatrix} \in \mathbb{Z}^{k \times k}. \quad (19)$$

Then, closely following the definition of the hint lattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ , we define the following matrix

$$\mathbf{B}_{\mathbf{M},\mathbf{H}}^{\text{LWE}} := \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{V} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{b} & \ell & \mathbf{0} & 1 \end{pmatrix} \in \mathbb{Z}^{(m+k+n+1) \times (m+k+n+1)}. \quad (20)$$

Notice that  $\mathbf{B}_{\mathbf{M},\mathbf{H}}^{\text{LWE}}$  naturally extends the definition of the original basis matrix  $\mathbf{B}^{\text{LWE}}$  from Equation (3). Indeed, the columns  $m+1$  to  $m+k$  of  $\mathbf{B}_{\mathbf{M},\mathbf{H}}^{\text{LWE}}$  simply correspond to additional LWE samples, defined over  $\mathbb{Z}_{m_i}$ , instead of  $\mathbb{Z}_q$ .

**Increased Determinant.** Let  $\Lambda_{\mathbf{M},\mathbf{H}}^{\text{LWE}} := \mathcal{L}(\mathbf{B}_{\mathbf{M},\mathbf{H}}^{\text{LWE}})$ . Since  $\mathbf{B}_{\mathbf{M},\mathbf{H}}^{\text{LWE}}$  is triangular and  $\mathbf{M}$  is diagonal, we have

$$\det \Lambda_{\mathbf{M},\mathbf{H}}^{\text{LWE}} = q^m \cdot \det \mathbf{M} = q^m \cdot \prod_{i=1}^k m_i = \det \Lambda^{\text{LWE}} \cdot \prod_{i=1}^k m_i.$$

Hence, we already increased the determinant of  $\Lambda^{\text{LWE}}$  by the desired factor.

Notice that the increase in determinant, though, comes at the cost of increasing the lattice dimension by  $k$ .<sup>7</sup> However, as we show below, the techniques, that allow us to *decrease* the lattice dimension in the perfect hint setting to  $n+m+1-k$ , now allow us to *preserve* our lattice dimension of  $n+m+1$  in the modular hint setting.

**Preserving  $\dim \Lambda^{\text{LWE}}$ .** Lifting Equation (18) to the integers, we obtain

$$\langle \mathbf{v}_i, \mathbf{s} \rangle = \ell_i - r_i m_i,$$

for some *unknown*  $r_i \in \mathbb{Z}$ . Let  $\mathbf{r} := (r_1, \dots, r_k)$ . By construction, it then holds that

$$(\mathbf{r}, \mathbf{s}, -1) \cdot \begin{pmatrix} \mathbf{M} \\ \mathbf{V} \\ \ell \end{pmatrix} = (r_1 m_1 + \langle \mathbf{v}_1, \mathbf{s} \rangle - \ell_1, \dots, r_k m_k + \langle \mathbf{v}_k, \mathbf{s} \rangle - \ell_k) = \mathbf{0}^k.$$

Hence, by Equation (20),  $\Lambda_{\mathbf{M},\mathbf{H}}^{\text{LWE}}$  contains the short vector

$$\mathbf{t}_{\mathbf{H}} := (-\mathbf{e}, \mathbf{0}^k, \mathbf{s}, -1),$$

which has the same length as the original secret vector  $\mathbf{t}$ , defined in Equation (4).

<sup>7</sup> This is in contrast to the perfect hint setting, where embedding the hints does not increase the lattice dimension.



Completely analogous to the perfect hint setting, we now simply suggest to search for  $\mathbf{t}_H$  in the following  $(m+n+1)$ -dimensional sublattice of  $\Lambda_{M,H}^{\text{LWE}}$ :

$$\begin{aligned} \Lambda_{M,H,k}^{\text{LWE}} &:= \{(v_1, \dots, v_{n+m+k+1}) \in \Lambda_{M,H}^{\text{LWE}} \mid v_{m+1} = \dots = v_{m+k} = 0\} \\ &= \Lambda_{M,H}^{\text{LWE}} \cap \mathbf{e}_{m+1}^\perp \cap \dots \cap \mathbf{e}_{m+k}^\perp, \end{aligned}$$

which has the same dimension as the original lattice  $\Lambda^{\text{LWE}}$ .

Making again the assumption that our hint matrix  $\mathbf{H}$  behaves like a random matrix (as we already did in Section 4.2), Lemma 2.4 then suggests that with high probability  $\mathbf{H}$  generates the integer lattice  $\mathbb{Z}^k$ . In that case, the sublattice  $\Lambda_{M,H,k}^{\text{LWE}}$  also has the required determinant, as we show in the following theorem.

**Theorem 5.1.** *Suppose  $\mathcal{L}(\mathbf{H}) = \mathbb{Z}^k$ . Then it holds that*

$$\det \Lambda_{M,H,k}^{\text{LWE}} = \det \Lambda_{M,H}^{\text{LWE}} = \det \Lambda^{\text{LWE}} \cdot \prod_{i=1}^k m_i.$$

*Proof.* Let  $U := \mathbf{e}_{m+1}^\perp \cap \dots \cap \mathbf{e}_{m+k}^\perp$ . By Lemma 2.6, we obtain

$$\det(\Lambda_{M,H,k}^{\text{LWE}}) = \det(\Lambda_{M,H}^{\text{LWE}} \cap U) = \frac{\det(\Lambda_{M,H}^{\text{LWE}})}{\det(\pi_{U^\perp}(\Lambda_{M,H}^{\text{LWE}}))}. \quad (21)$$

For any subset  $A \subseteq \mathbb{R}^k$ , let  $A^\sim := \{0\}^m \times A \times \{0\}^{n+1} \subset \mathbb{R}^{m+k+n+1}$ . Looking at the shape of the basis matrix  $\mathbf{B}_{M,H}^{\text{LWE}}$  in Equation (20) and using  $U^\perp = (\mathbb{R}^k)^\sim$ , it easily follows that

$$\pi_{U^\perp}(\Lambda_{M,H}^{\text{LWE}}) = \mathcal{L}([\mathbf{M}, \mathbf{H}])^\sim \supseteq \mathcal{L}(\mathbf{H})^\sim = (\mathbb{Z}^k)^\sim.$$

Together with  $\pi_{U^\perp}(\Lambda_{M,H}^{\text{LWE}}) \subseteq (\mathbb{Z}^k)^\sim$ , this yields  $\pi_{U^\perp}(\Lambda_{M,H}^{\text{LWE}}) = (\mathbb{Z}^k)^\sim$ , and thus

$$\det(\pi_{U^\perp}(\Lambda_{M,H}^{\text{LWE}})) = 1. \quad (22)$$

Plugging in Equation (22) into Equation (21), the theorem follows.  $\square$

We note that (as in the previous sections) we require our hints  $\bar{\mathbf{v}}_i'$  to be linearly independent, see also Remark 3.1. If the hints were linearly dependent, we would have  $\mathcal{L}(\mathbf{H}) \subsetneq \mathbb{Z}^k$ , in which case Theorem 5.1 no longer applies, and we would have  $\det \Lambda_{M,H,k}^{\text{LWE}} < \det \Lambda_{M,H}^{\text{LWE}}$ .

**Efficiently Computing a Basis for  $\Lambda_{M,H,k}^{\text{LWE}}$ .** As discussed above, our new lattice  $\Lambda_{M,H,k}^{\text{LWE}}$  has the exact same quality as the original lattice of DDGR. However, since our lattice  $\Lambda_{M,H,k}^{\text{LWE}}$  is obtained by intersecting  $\Lambda_{M,H}^{\text{LWE}}$  with standard basis vectors, we can compute a basis for our lattice much more efficiently than DDGR, by simply using our algorithm CONSTRUCT-SUBLATTICE (Algorithm 1), as discussed in Remark 4.8.

## 5.2 Combining Modular and Perfect Hints

In a scenario, where we are given both *modular* hints  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i, m_i)$ , for  $i = 1, \dots, k$ , as well as *perfect* hints  $\bar{\mathbf{w}}_i = (\mathbf{w}_i, \ell_i)$ , for  $i = k + 1, \dots, k + \ell$ , our approach has the additional advantage that we can easily integrate all hints in one stroke. To this end, we simply construct a second hint matrix  $\mathbf{H}' := \text{Hint}(\bar{\mathbf{w}}_{k+1}, \dots, \bar{\mathbf{w}}_{k+\ell}) = (\mathbf{W}, \ell')$ , along with the following lattice basis

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{V} & \mathbf{W} & \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{b} & \ell & \ell' & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix},$$

and then search for the LWE secret in the sublattice, that has zeros in the columns  $m + 1$  to  $m + k + \ell$ .

## 5.3 Comparison with Section 3

In our simple linear algebra approach from Section 3 for integrating  $k \bmod q$  hints, the hints eliminate  $k$  coordinates of the secret vector  $\mathbf{t}$ , and decrease the dimension of  $\Lambda^{\text{LWE}}$  by  $k$ , while leaving the determinant unchanged. At first glance, this seems complementary to our more involved approach from Section 5.1, where  $\|\mathbf{t}\|$  and dimension remain unchanged, while the determinant grows by a factor  $q^k$ .

Notice that after increasing the determinant by  $q^k$  we may ignore, however, up to  $k$  LWE samples, as explained in Section 2.5. Since every ignored LWE sample decreases the dimension of the lattice by one, eliminates one coordinate of  $\mathbf{t}$ , and decreases the determinant by a factor  $q$ , our more involved approach thus

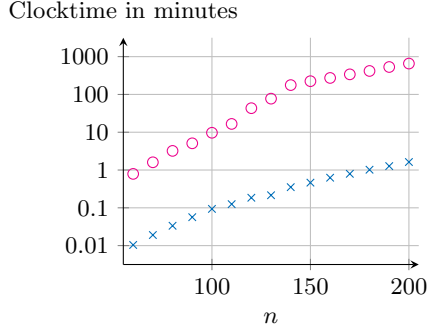
- (1) eliminates  $i$  coordinates of  $\mathbf{t}$ ,
- (2) decreases the dimension of  $\Lambda^{\text{LWE}}$  by  $i$ ,
- (3) and increases the determinant by  $q^{k-i}$ ,

for some freely choosable parameter  $0 \leq i \leq k$ . As discussed in Section 2.5, the BKZ algorithm can optimize the value of  $i$  on its own.

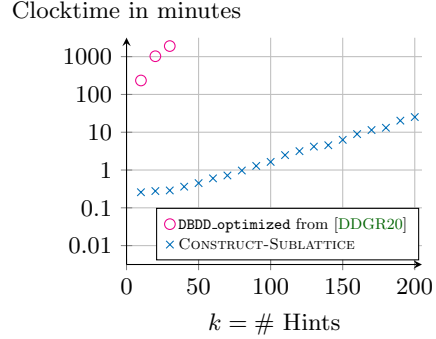
As one expects, this additional degree of freedom makes the more involved approach from Section 5.1 slightly better than the approach from Section 3, in the sense that it requires slightly smaller BKZ blocksizes to recover the secret. Worth noting, in the regime of *too many hints*, where mere basis construction dominates the runtime, the approach from Section 3 is, however, still preferable.

## 6 Runtime Comparison with DDGR

Instead of using CONSTRUCT-SUBLATTICE for constructing a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}} \subset \Lambda_{\mathbf{H}}^{\text{LWE}}$  (or  $\Lambda_{\mathbf{M},\mathbf{H},k}^{\text{LWE}} \subset \Lambda_{\mathbf{M},\mathbf{H}}^{\text{LWE}}$ ), we could also use the following slight modification



**Fig. 2.** Required runtime for integrating  $n/2$  random perfect hints into  $n$ -dimensional random KYBER-like LWE instances with  $m = n$ ,  $q = 3329$  and  $\|(\mathbf{e}, \mathbf{s})\| \approx 1.22 \cdot \sqrt{m+n}$ .



**Fig. 3.** Required runtime for integrating  $k$  random perfect hints into random KYBER-512 instances.

of [DDGR20] for integrating perfect hints. Using the algorithm from [DDGR20, Section 4.1] for computing a *lattice slice*, we first compute a basis for  $\Lambda_1 := \Lambda_{\mathbf{H},k}^{\text{LWE}} \cap \mathbf{e}_{m+1}^\perp$ , and then iteratively compute bases for  $\Lambda_i := \Lambda_{i-1} \cap \mathbf{e}_{m+i}^\perp$ , with  $i = 2, \dots, k$ , until we obtain a basis for  $\Lambda_k$ . Since

$$\Lambda_{\mathbf{H},k}^{\text{LWE}} = \Lambda_{\mathbf{H}}^{\text{LWE}} \cap \mathbf{e}_{m+1}^\perp \cap \dots \cap \mathbf{e}_{m+k}^\perp = \Lambda_k,$$

we obtain a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ .

While this approach runs in polynomial time, it is unfortunately too slow in practice, because it is sequential. This is, in fact, precisely the issue that renders DDGR's implementation impractical in cryptographic dimensions.

**Another inferior approach.** As another alternative to compute a basis for  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ , we could also use the following standard approach for computing the intersection of two lattices. Let  $d := m+k+n+1$ , define  $U := \mathbf{e}_{m+1}^\perp \cap \dots \cap \mathbf{e}_{m+k}^\perp \subset \mathbb{R}^d$ , and let  $\mathbf{B}_U \in \mathbb{Z}^{(d-k) \times d}$  be a basis matrix for the linear subspace  $U$ . (For instance,  $\mathbf{B}_U$  may be obtained by taking the identity matrix  $\mathbf{I}_d$ , and removing the  $(m+1)$ -th to  $(m+k)$ -th rows.) We construct the following lattice basis

$$\begin{pmatrix} \mathbf{B}_{\mathbf{H}}^{\text{LWE}} & \mathbf{B}_{\mathbf{H}}^{\text{LWE}} \\ \mathbf{B}_U & \mathbf{0} \end{pmatrix} \in \mathbb{Z}^{(2d-2k) \times 2d},$$

and compute its Hermite normal form (HNF). By a simple dimension counting argument, it is easy to see that the HNF then has the following shape

$$\left( \begin{array}{c|c} \mathbf{B}_1 & \mathbf{B}_2 \\ \hline \mathbf{0}_{d-k \times d} & \mathbf{B}_3 \end{array} \right),$$

where  $\mathbf{B}_1$  is a basis matrix of  $\mathcal{L}(\mathbf{B}_H^{\text{LWE}}) + \mathcal{L}(\mathbf{B}_U)$ , and – more importantly –  $\mathbf{B}_3$  is a basis matrix of  $\Lambda_{\mathbf{H}}^{\text{LWE}} \cap U = \Lambda_{\mathbf{H},k}^{\text{LWE}}$ .

However, this approach requires arithmetic on a  $2(n + m + 1)$ -dimensional lattice, whereas CONSTRUCT-SUBLATTICE mainly works on a  $(n+1)$ -dimensional lattice. Therefore, it is also much slower than our approach.

As Figures 2 and 3 show, our new algorithm greatly improves over the runtime of DDGR’s algorithm. For instance, to integrate 30 perfect hints into a KYBER-512 instance, the DDGR algorithm requires more than 31 hours, whereas ours requires less than 20 seconds.<sup>8</sup>

## 7 Experimental Results

We provide experimental data for our implementation of the mod- $q$  hint approach as described Section 3, and the implementation of CONSTRUCT-SUBLATTICE (Algorithm 1) from our perfect hint approach from Section 4.

**Setup.** In our experiments, we took hints  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i)$ , respectively  $\bar{\mathbf{v}}_i = (\mathbf{v}_i, \ell_i, q)$ , where  $\mathbf{v}_i$  is drawn uniformly at random from  $\{0, \dots, q\}^n$ . In the mod- $q$  hint setting, we generated 16 random keys per scheme. In the perfect hint setting, we generated 32 random keys per scheme (with the exception of DILITHIUM, where we used only 16 keys.)

Worth noting, we did not implement CONSTRUCT-SUBLATTICE exactly as in the pseudocode from Algorithm 1, but added a minor tweak: Instead of directly LLL-reducing the matrix  $(\tilde{\mathbf{H}} \mid \mathbf{I}_{n+1})$  (see Step 2 of the algorithm), we first removed for every perfect hint one column from the  $(n + 1)$ -dimensional identity matrix. (In other words, we projected the lattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  against some more standard basis vectors.) Curiously, we observed that this slightly worsens the gap of the lattice (the dimension remains unchanged), but BKZ finds the LWE secret at slightly smaller block sizes. Additionally, this decreases the practical runtime of LLL, since the lattice lies in a smaller vector space. We leave it as an interesting open question to further study this BKZ behavior.

**Hardware.** We performed all our experiments on an AMD EPYC 7763 with 1 TB of RAM, as well as on an AMD EPYC 7742 with 2 TB of RAM. Each EPYC is equipped with 128 physical cores that with parallelization give 256 threads. We used the high number of cores only to run multiple experiments in parallel, but we did *not* use parallelism to speed up any single experiment.

<sup>8</sup> We ran both the DDGR algorithm and CONSTRUCT-SUBLATTICE in Sage9.7, using the latest patch to speed up fpylll, see <https://github.com/fplll/fpylll/pull/239>.

**Results.** Our results are depicted in Figures 4 to 13. In the mod- $q$  setting, BKZ blocksize 2 denotes LLL reduction. We see, e.g., that for KYBER-512 with  $k \geq 449$  all instances could be solved via LLL, determining our *too many hints* regime. In the perfect hint setting, we denote by blocksize 0 that after running CONSTRUCT-SUBLATTICE we could already directly read off our secret vector, without further reduction of the resulting hint lattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$ . We see, e.g., that we are in the *too many hints* regime for perfect hints for KYBER-512 with  $k \geq 233$ .

We choose a different format for displaying our DILITHIUM perfect hint results, because we were unable to run the BKZ algorithm on the hint lattice  $\Lambda_{\mathbf{H},k}^{\text{LWE}}$  for DILITHIUM, since we always encountered the infamous **infinite loop in babai** error. Nevertheless, we still provide the data points, at which we could already read off the LWE secret from the output of CONSTRUCT-SUBLATTICE.

As expected, Heuristic 4.6 was valid in every experiment.

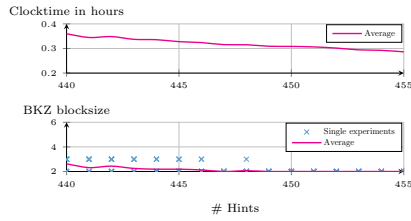


Fig. 4. [Kyber-512, mod- $q$  hints]

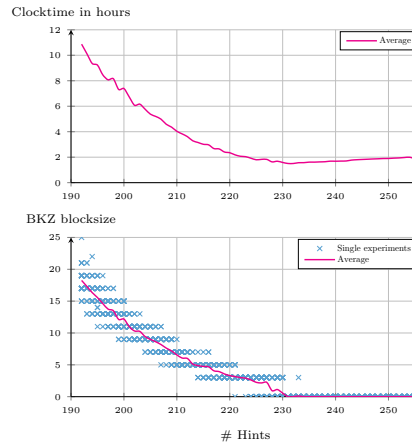


Fig. 5. [Kyber-512, perfect hints]

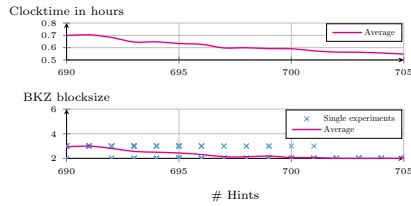


Fig. 6. [Kyber-768, mod- $q$  hints]

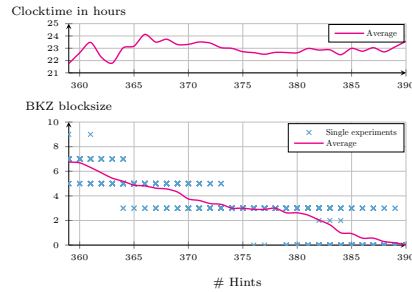


Fig. 7. [Kyber-768, perfect hints]

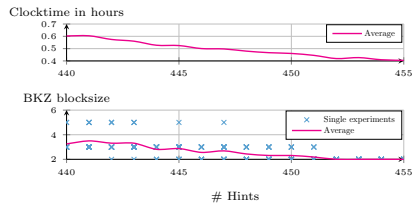


Fig. 8. [Falcon-512, mod- $q$  hints]

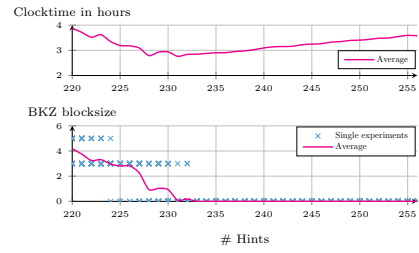


Fig. 9. [Falcon-512, perfect hints]

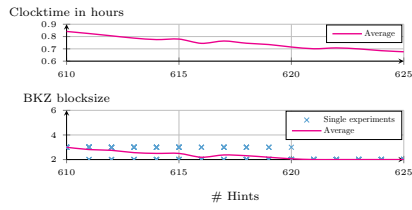


Fig. 10. [NTRU-701, mod- $q$  hints]

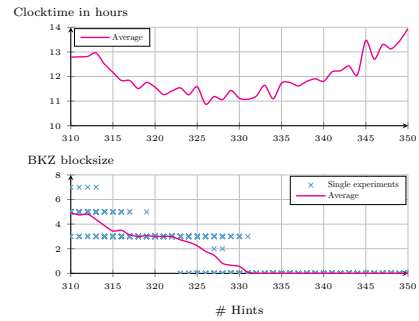


Fig. 11. [NTRU-701, perfect hints]

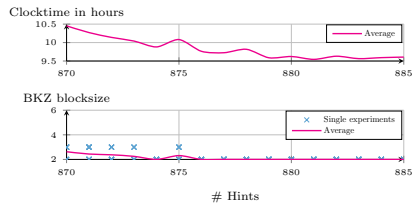


Fig. 12. [Dilithium-1024, mod- $q$  hints]

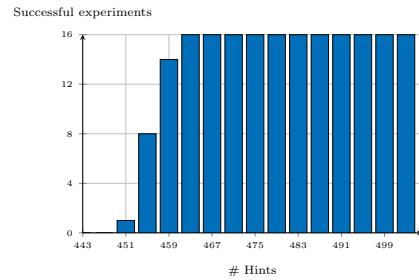


Fig. 13. [Dilithium-1024, perfect hints]

## References

- AD97. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293. ACM Press, May 1997.
- AD21. Martin R. Albrecht and Léo Ducas. *Lattice Attacks on NTRU and LWE: A History of Refinements*, page 15–40. 2021.
- ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 327–343. USENIX Association, August 2016.
- Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996.
- BDK<sup>+</sup>18. Joppe 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. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.
- BGV14. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
- BV96. Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO’96*, volume 1109 of *Lecture Notes in Computer Science*, pages 129–142. Springer, Heidelberg, August 1996.
- CLOS91. Matthijs J. Coster, Brian A. LaMacchia, Andrew M. Odlyzko, and Claus P. Schnorr. An improved low-density subset sum algorithm. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT’91*, volume 547 of *Lecture Notes in Computer Science*, pages 54–67. Springer, Heidelberg, April 1991.
- Cop97. Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *Journal of cryptology*, 10(4):233–260, 1997.
- DDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, Heidelberg, August 2020.
- DKL<sup>+</sup>18. Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.
- dt21. The FPLLL development team. fpylll, a Python wrapper for the fplll lattice reduction library, Version: 0.5.7. Available at <https://github.com/fplll/fpylll>, 2021.
- EMVW22. Andre Esser, Alexander May, Javier A. Verbel, and Weiqiang Wen. Partial key exposure attacks on BIKE, rainbow and NTRU. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 346–375. Springer, Heidelberg, August 2022.

- FHK<sup>+</sup>18. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 36(5), 2018.
- HDWH12. Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 205–220, 2012.
- HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 267–288, 1998.
- JS98. Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11:161–185, 1998.
- KMS22. Stefan Kölbl, Rafael Misoczki, and Sophie Schmieg. Securing tomorrow today: Why Google now protects its internal communications from quantum threats. <https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms?hl=en>, 2022.
- LLL82. Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.
- LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, Heidelberg, May / June 2010.
- LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.
- Mar13. Jacques Martinet. *Perfect lattices in Euclidean spaces*, volume 327. Springer Science & Business Media, 2013.
- MN23. Alexander May and Julian Nowakowski. Too many hints - when LLL breaks LWE. *Cryptology ePrint Archive*, Paper 2023/777, 2023. <https://eprint.iacr.org/2023/777>.
- MRW11. Gérard Maze, Joachim Rosenthal, and Urs Wagner. Natural density of rectangular unimodular integer matrices. *Linear algebra and its applications*, 434(5):1319–1324, 2011.
- Odl90. Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42(2), 1990.
- PRS17. Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, 2017.
- Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.
- Sch87. Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.



- SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, Heidelberg, December 2009.
- WWX22. Han Wu, Xiaoyun Wang, and Guangwu Xu. Reducing an lwe instance by modular hints and its applications to primal attack, dual attack and bkz attack. Cryptology ePrint Archive, Paper 2022/1404, 2022. <https://eprint.iacr.org/2022/1404>.