On the Hardness of LWE with Binary Error: Revisiting the Hybrid Lattice-Reduction and Meet-in-the-Middle Attack

Johannes Buchmann[1], Florian Göpfert[1], Rachel Player[2], and Thomas Wunderer[1]

[1] Technische Universität Darmstadt, Germany
{buchmann, fgoepfert, twunderer}@cdc.informatik.tu-darmstadt.de
[2] Information Security Group, Royal Holloway, University of London, UK
Rachel.Player.2013@live.rhul.ac.uk

Abstract. The security of many cryptographic schemes has been based on special instances of the Learning with Errors (LWE) problem, e.g., Ring-LWE, LWE with binary secret, or LWE with ternary error. However, recent results show that some subclasses are weaker than expected. In this work we show that *LWE with binary error*, introduced by Micciancio and Peikert, is one such subclass. We achieve this by applying the Howgrave-Graham attack on NTRU, which is a combination of lattice techniques and a Meet-in-the-Middle approach, to this setting. We show that the attack outperforms all other currently existing algorithms for several natural parameter sets. For instance, for the parameter set $n = 256$, $m = 512$, $q = 256$, this attack on LWE with binary error only requires 2^{103} operations, while the previously best attack requires 2^{117} operations. We additionally present a complete and improved analysis of the attack, using analytic techniques. Finally, based on the attack, we give concrete hardness estimations that can be used to select secure parameters for schemes based on LWE with binary error.

Keywords: Learning with Errors, Lattice-based Cryptography, Cryptanalysis, NTRU, Hybrid Attack

1 Introduction

The Learning with Errors problem (LWE) is one of the most important problems in lattice-based cryptography. A huge variety of schemes, ranging from basic primitives like signature [18] and encryption schemes [32] to highly advanced schemes like group signatures [30] and fully homomorphic encryption [12], base their security on the LWE assumption. Understanding the concrete hardness of LWE is therefore important for selecting parameters.

Many cryptographic schemes are based on the hardness of special LWE instances like Ring-LWE [34], or LWE with ternary error [22]. Understanding the hardness of subclasses of the LWE problem and identifying those that are easy to

solve is therefore an important task. In fact, several recent results [19,20,14,29] show that some subclasses are easier than expected.

We show that the subclass LWE with binary error, which has been considered before in several papers [35,1], fits into this category. To show that LWE with binary error is considerably easier than expected, we modify the hybrid lattice-reduction and meet-in-the-middle attack by Howgrave-Graham [25] (refered to as hybrid attack in the following), apply it to this setting, and analyze its complexity. In order to compare our approach to existing ones, we apply known attacks on LWE to the binary error setting and analyze their complexities in this case. Our comparison shows that the hybrid attack outperforms existing methods such as the enumeration attack [32,33], or the embedding approach [4] for several natural parameter sets. Figure 1 illustrates our improvement, by comparing the runtime of the best previously known attack with the hybrid attack, where $m = 2n$ samples from an LWE distribution with binary error are given and n is the dimension of the secret vector. For example, in the case of $n = 256$ and $q = 256$, the hardness of the problem drops from 117 to 103 bits, which is quite a noticeable improvement. A detailed comparison between the hybrid attack and previous approaches is given in Table 1 in Section 4.

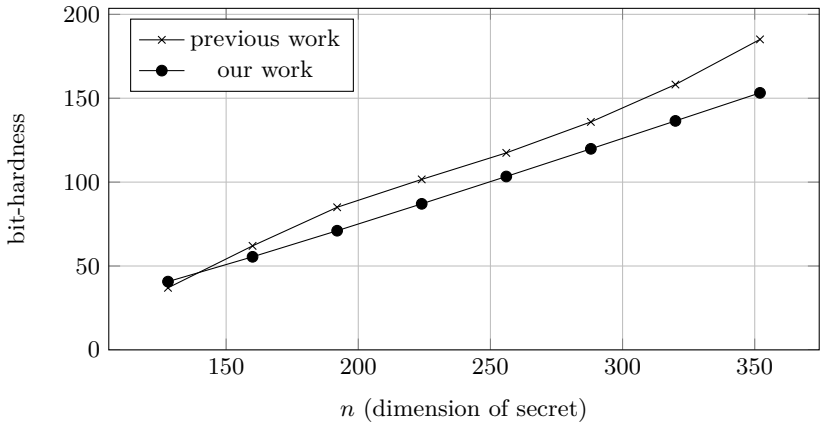

Fig. 1. Hardness of LWE instances with number of samples $m = 2n$ and modulus $q = 256$ before and after this work

The hybrid attack can also be seen as an improvement of an idea sketched by Bai and Galbraith [9]. However, Bai and Galbraith did not provide an analysis of their suggestion, and the analysis of Howgrave-Graham is partly based on experiments. A theoretical analysis of the hybrid attack that is not based on experimental results has been presented by Hirschhorn et al. in [24]. However, their analysis requires an additional assumption.

In this work we present a complete and improved analysis based on the same assumptions used in [25] without the additional assumption of [24], that does not require experimental support. For this reason, we introduce new analytic tech-

niques. Our new analysis can also be applied to the Howgrave-Graham attack, as well as to the attack mentioned by Bai and Galbraith (see [9]). In addition, we show how to use our techniques to analyze the decoding attack on LWE with binary error.

Related work. A number of recent works have highlighted the importance of considering the hardness of variants of LWE. For example, certain choices of rings lead to weak instances of the Ring-LWE problem [19,20,14]. Additionally, Laine and Lauter [29] provide a polynomial time attack for LWE instances with an exponentially large modulus q and a sufficiently narrow Gaussian error. The existence of such weak instances shows the necessity of studying the hardness of special instances of the LWE problem separately.

The hardness of LWE with binary error has been considered in some detail. So far, there are known attacks that require access to superlinearly many samples (i.e., $m > \mathcal{O}(n)$), and hardness results when the crypanalyst is given a sublinear number of additional samples (i.e., $m = n + \mathcal{O}(n/\log(n))$), where n is the dimension of the secret vector. More precisely, the problem can be solved in polynomial time using the algorithm of Arora and Ge [6], when the number of samples is $m = \mathcal{O}(n^2)$ (see, e.g., [1]). Furthermore, Albrecht et al. [1] showed that LWE with binary error can be solved in subexponential time using an improved version of the Arora-Ge attack, if the attacker has access to a quasi-linear number of samples, e.g., $m = \mathcal{O}(n \log \log n)$. On the other hand, Micciancio and Peikert [35] proved that LWE with binary error reduces to worst-case lattice problems when the number of samples is restricted to $n + \mathcal{O}(n/\log(n))$. We close the margin between these hardness results on the one side and the weakness results on the other side by presenting an attack that runs with only n additional samples.

The idea of Bai and Galbraith which we build upon is to guess the first r components of the secret vector and apply a lattice attack on the remaining problem [9]. As noted in [5], this strategy enables the transformation of any algorithm for solving LWE into another one whose complexity is bounded by the cost of exhaustive search. Howgrave-Graham's algorithm [25], which we apply here, involves a Meet-in-the-Middle component to speed up this guessing: this was not considered in either of [9,5]. The existence of a Meet-in-the-Middle approach for solving LWE (without combining with any another algorithm) was mentioned in [9] and such an algorithm was presented in [5]. In Section 4 we show that it is much more efficient to combine a Meet-in-the-Middle approach with a decoding attack than to solve LWE with binary error entirely by a Meet-in-the-Middle approach.

Structure. In Section 2 we give some notation and required preliminaries. In Section 3 we describe how to apply the hybrid attack to LWE with binary error and analyze its complexity. In Section 4 we apply other possible attacks on LWE to the binary error case, analyze their complexities, and compare the results to the hybrid attack.

2 Notation and preliminaries

Notation. In this work vectors are denoted in bold lowercase letters, e.g., $\mathbf{a}$, and matrices in bold uppercase letters, e.g., $\mathbf{A}$. For a vector $\mathbf{v} \in \mathbb{R}^n$ we write $\mathbf{v} \mod q$ for its unique representative modulo q in $[-\lfloor \frac{q}{2} \rfloor, \frac{q}{2})^n$. Logarithms are base two unless stated otherwise, and $\ln(x)$ denotes the natural logarithm of x.

Learning with Errors. The Learning with Errors (LWE) problem, introduced by Regev [41], is a computational problem, whose presumed hardness is the basis for several cryptographic constructions, e.g., [41,39,40]. In this work, we consider the variant *LWE with binary error*.

Problem Statement 1 (LWE with binary error) *Let n, q be positive integers, $\mathcal{U}$ be the uniform distribution on $\{0,1\}$ and $\mathbf{s} \xleftarrow{\$} \mathcal{U}^n$ be a secret vector in $\{0,1\}^n$. We denote by $L_{\mathbf{s},\mathcal{U}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \xleftarrow{\$} \mathcal{U}$ and returning $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*
LWE with binary error is the problem of recovering $\mathbf{s}$ from m samples $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\mathcal{U}}$, with $i \in \{1, \ldots, m\}$.

Note that Regev defined LWE with a secret vector $\mathbf{s}$ chosen uniformly at random from the whole of $\mathbb{Z}_q^n$. However, it is well-known that LWE with arbitrarily distributed secret can be transformed to LWE with secret distributed according to the error distribution. Consequently, most cryptographic constructions are based on LWE where secret and error are identically distributed, and we focus on this case in this work.

Lattices and bases. A lattice is a discrete additive subgroup of $\mathbb{R}^m$. A set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \subset \mathbb{R}^m$ is called a basis of a lattice Λ, if $\Lambda = \Lambda(\mathbf{B})$, where

$$\Lambda(\mathbf{B}) = \{\mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{b}_i \text{ for } \alpha_i \in \mathbb{Z}\}.$$

The dimension of a lattice Λ is defined as the cardinality of some (equivalently any) basis of Λ. For the rest of this work we restrict our studies to lattices in $\mathbb{R}^m$ whose dimension is maximal, e.g., m, which are called full-ranked lattices. The fundamental parallelepiped of a lattice basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_m\} \subset \mathbb{R}^m$ is given by

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \text{ for } -1/2 \leq \alpha_i < 1/2\}.$$

The determinant of a lattice $\Lambda(\mathbf{B})$ for a basis $\mathbf{B}$ is defined as the m dimensional volume of its fundamental parallelepiped. Note that the determinant of the lattice is independent of the choice of the basis.

Every lattice of dimension $m \geq 2$ has infinitely many different bases. A measure for the quality of a basis is provided by the Hermite delta. A lattice basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_m\}$ has Hermite delta δ if $\|\mathbf{b}_1\| = \delta^m \det(\Lambda)^{1/m}$.

Differing estimates exist in the literature for the number of operations of a basis reduction necessary to achieve a certain Hermite delta δ (see for example [32,15,33,5,37]). Throughout this work, whenever an estimate for the runtime of a basis reduction algorithm to achieve a given Hermite delta δ is required, we will use the estimate given by Lindner and Peikert [32]. This is that the number of operations needed to achieve a certain Hermite delta δ is around

$$\text{ops}_{\text{BKZ}}(\delta) = 2^{1.8/\log_2(\delta)-110} \cdot 2.3 \cdot 10^9. \tag{1}$$

A lattice Λ satisfying $q \cdot \mathbb{Z}^m \subset \Lambda \subset \mathbb{R}^m$ is a q-ary lattice. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, we define the q-ary lattice

$$\Lambda_q(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{w} \in \mathbb{Z}^n : \mathbf{A}\mathbf{w} = \mathbf{v} \mod q\}.$$

With high probability, we have $\det(\Lambda_q(\mathbf{A})) = q^{m-n}$.

The closest vector problem is the problem of recovering the lattice vector closest to a given target vector, given also a basis of the lattice. One can consider a relaxation, namely a close vector problem, where the inputs are the same (a basis and a target vector), and the task is to recover a lattice vector which is sufficiently close to the target. We refer to close vector problem as CVP throughout this work.

Babai's nearest plane. The hybrid attack uses Babai's nearest plane algorithm [7] (denoted by NP in the following) as subroutine. It gets a lattice basis $\mathbf{B} \subset \mathbb{Z}^m$ and a target vector $\mathbf{t} \in \mathbb{R}^m$ as input and outputs a vector $\mathbf{e} \in \mathbb{R}^m$ such that $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$, which we denote by $\text{NP}_{\mathbf{B}}(\mathbf{t}) = \mathbf{e}$. If the used lattice basis is clear from the context, we omit it in the notation and simply write $\text{NP}(\mathbf{t})$. A detailed explanation of nearest plane can be found in Babai's original work [7] and Lindner and Peikert's follow up work [32]. The output of nearest plane plays an important role in the analysis of the hybrid attack and can be understood without knowing details about the algorithm itself. It depends on the Gram-Schmidt basis of the input basis $\mathbf{B}$, which is defined as $\overline{\mathbf{B}} = \{\overline{\mathbf{b}_1}, \ldots, \overline{\mathbf{b}_n}\}$ with

$$\overline{\mathbf{b}_i} = \mathbf{b}_i - \sum_{j=1}^{i-1} \frac{\langle \overline{\mathbf{b}_j}, \mathbf{b}_i \rangle}{\langle \overline{\mathbf{b}_j}, \overline{\mathbf{b}_j} \rangle} \overline{\mathbf{b}_j},$$

where $\overline{\mathbf{b}_1} = \mathbf{b}_1$. We will use the following result from [8].

Lemma 1. *For a lattice basis* $\mathbf{B}$ *with Gram-Schmidt basis* $\overline{\mathbf{B}}$ *and a target vector* $\mathbf{t}$ *as input, the nearest plane algorithm returns the unique vector* $\mathbf{e} \in \mathcal{P}(\overline{\mathbf{B}})$ *that satisfies* $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$.

Lemma 1 shows that analyzing the output of the nearest plane algorithm requires to estimate the lengths of the basis vectors of the corresponding Gram-Schmidt basis. The established way to do this is via the the following heuristic (see Lindner and Peikert [32] for more details).

Heuristic 1 (Geometric Series Assumption) *Let $\{\mathbf{b}_1 \dots \mathbf{b}_m\} \subset \mathbb{Z}^m$ be a reduced basis with Hermite delta δ of an m-dimensional lattice with determinant D. Also let $\overline{\mathbf{b}_i}$ denote the basis vectors of the corresponding Gram-Schmidt basis. Then the length of $\overline{\mathbf{b}_i}$ is approximated by*

$$\left\|\overline{\mathbf{b}_i}\right\| \approx \delta^{-2(i-1)+m} D^{\frac{1}{m}}.$$

3 The attack

In this section we present and analyze the hybrid attack on LWE with binary error. The attack is described in Algorithm 1 of Section 3.1. In Theorem 1 of Section 3.2 we analyze the expected runtime of the hybrid attack. Section 3.3 shows how to optimize the attack parameters and perform a trade-off between precomputation and the actual attack in order to minimize the runtime of the attack.

3.1 The hybrid attack

In the following we describe the hybrid attack on LWE with binary error. The attack is presented in Algorithm 1.

Let $m, n, q \in \mathbb{N}$ and let

$$(\mathbf{A}, \mathbf{b} = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e} \mod q) \tag{2}$$

with $\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{b} \in \mathbb{Z}_q^m, \tilde{\mathbf{s}} \in \{0,1\}^n$ and $\mathbf{e} \in \{0,1\}^m$ be an LWE instance with binary error $\mathbf{e}$ and binary secret $\tilde{\mathbf{s}}$. In order to obtain a smaller error vector we can subtract the vector $(1/2) \cdot \mathbf{1}$ consisting of all $1/2$ entries from Equation (2). This yields a new LWE instance $(\mathbf{A}, \mathbf{b}' = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}' \mod q)$, where $\mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1}$ and $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1}$. The new error vector $\mathbf{e}'$ now has norm $\sqrt{m/4}$ instead of the expected norm $\sqrt{m/2}$ of the original error vector $\mathbf{e}$. For $r \in \{1, \dots, n-1\}$, we can split the secret $\tilde{\mathbf{s}} = \begin{pmatrix} \mathbf{v} \\ \mathbf{s} \end{pmatrix}$ and the matrix $\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$ into two parts and rewrite this LWE instance as

$$\mathbf{b}' = (\mathbf{A}_1 | \mathbf{A}_2) \begin{pmatrix} \mathbf{v} \\ \mathbf{s} \end{pmatrix} + \mathbf{e}' = \mathbf{A}_1 \mathbf{v} + \mathbf{A}_2 \mathbf{s} + \mathbf{e}' \mod q, \tag{3}$$

where $\mathbf{v} \in \{0,1\}^r, \mathbf{s} \in \{0,1\}^{n-r}, \mathbf{A}_1 \in \mathbb{Z}_q^{m \times r}, \mathbf{A}_2 \in \mathbb{Z}_q^{m \times (n-r)}, \mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1} \in \mathbb{Q}^m$, and $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1} \in \{-1/2, 1/2\}^m$.

The main idea of the attack is to guess $\mathbf{v}$ and solve the remaining LWE instance $(\mathbf{A}_2, \tilde{\mathbf{b}} = \mathbf{b}' - \mathbf{A}_1 \mathbf{v} = \mathbf{A}_2 \mathbf{s} + \mathbf{e}' \mod q)$, which has binary secret $\mathbf{s}$ and error $\mathbf{e}' \in \{-1/2, 1/2\}^m$. The new LWE instance obtained in this way turns out to be considerably easier to solve, since the determinant $\det(\Lambda_q(\mathbf{A}_2)) = q^{m-n+r}$ of the new lattice is significantly bigger than the determinant $\det(\Lambda_q(\mathbf{A})) = q^{m-n}$ of the original lattice (see Section 6.1 of [9]). The newly obtained LWE instance is solved by solving a close vector problem in the lattice $\Lambda_q(\mathbf{A}_2)$. In

Algorithm 1: The Hybrid Attack

Input : $q, r \in \mathbb{Z}$
$\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$, where $\mathbf{A}_1 \in \mathbb{Z}_q^{m \times r}$, $\mathbf{A}_2 \in \mathbb{Z}_q^{m \times (n-r)}$
$\mathbf{b} \in \mathbb{Z}_q^m$
$\mathbf{B}$, a lattice basis of $\Lambda_q(\mathbf{A}_2)$

calculate $c = \lfloor r/4 \rceil$;
calculate $\mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1}$;
while *true* **do**
  guess a binary vector $\mathbf{v}_1 \in \{0,1\}^r$ with c ones ;
  calculate $\mathbf{x}_1 = -\operatorname{NP}_{\mathbf{B}}(-\mathbf{A}_1\mathbf{v}_1) \in \mathbb{R}^m$;
  calculate $\mathbf{x}_2 = \operatorname{NP}_{\mathbf{B}}(\mathbf{b}' - \mathbf{A}_1\mathbf{v}_1) \in \mathbb{R}^m$;
  store $\mathbf{v}_1$ in all the boxes addressed by $\mathcal{A}_{\mathbf{x}_1}^{(r)} \cup \mathcal{A}_{\mathbf{x}_2}^{(r)}$;
  for *all* $\mathbf{v}_2 \neq \mathbf{v}_1$ *in all the boxes addressed by* $\mathcal{A}_{\mathbf{x}_1}^{(r)} \cup \mathcal{A}_{\mathbf{x}_2}^{(r)}$ **do**
    Set $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$ and calculate $\mathbf{x} = (1/2) \cdot \mathbf{1} + \operatorname{NP}_{\mathbf{B}}(\mathbf{b}' - \mathbf{A}_1\mathbf{v}) \in \mathbb{R}^m$;
    if $\mathbf{x} \in \{0,1\}^m$ *and* $\exists \tilde{\mathbf{s}} \in \{0,1\}^n : \mathbf{b} = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{x} \mod q$ **then**
      return $\mathbf{x}$;

more detail, $\tilde{\mathbf{b}} = \mathbf{A}_2\mathbf{s} + q\mathbf{w} + \mathbf{e}'$ for some vector $\mathbf{w} \in \mathbb{Z}^m$ is close to the lattice vector $\mathbf{A}_2\mathbf{s} + q\mathbf{w} \in \Lambda_q(\mathbf{A}_2)$ since $\mathbf{e}'$ is small. Hence one can hope to find $\mathbf{e}'$ by running the nearest plane algorithm in combination with a sufficient basis reduction as a precomputation (see [32]).

The guessing of $\mathbf{v}$ is sped up by a Meet-in-the-Middle approach, i.e., guessing binary vectors $\mathbf{v}_1 \in \{0,1\}^r$ and $\mathbf{v}_2 \in \{0,1\}^r$ such that $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$. In order to recognize matching guesses $\mathbf{v}_1$ and $\mathbf{v}_2$ that sum up to $\mathbf{v}$, one searches for collisions in (hash) boxes. The addresses of these boxes are determined in the following way.

Definition 1. *Let $m \in \mathbb{N}$. For a vector $\mathbf{x} \in \mathbb{R}^m$ the set $\mathcal{A}_{\mathbf{x}}^{(m)} \subset \{0,1\}^m$ is defined as*

$$\mathcal{A}_{\mathbf{x}}^{(m)} = \left\{ \mathbf{z} \in \{0,1\}^m \middle| \begin{array}{l} (\mathbf{z})_i = 1 \text{ for all } i \in \{1,\ldots,m\} \text{ with } (\mathbf{x})_i > -1/2, \text{ and} \\ (\mathbf{z})_i = 0 \text{ for all } i \in \{1,\ldots,m\} \text{ with } (\mathbf{x})_i < -1/2 \end{array} \right\}.$$

Intuitively, for $\mathbf{x}_2$ obtained during Algorithm 1, the set $\mathcal{A}_{\mathbf{x}_2}^{(m)}$ captures all the possible sign vectors of $\mathbf{x}_2$ added up with a vector in $\{-1/2, 1/2\}^m$ (where 1 represents a non-negative and 0 a negative sign). For $\mathbf{x}_1$ obtained during Algorithm 1, the set $\mathcal{A}_{\mathbf{x}_1}^{(m)}$ consists only of the sign vector of $\mathbf{x}_1$. This is due to the fact that $\mathbf{x}_2 \in \mathbb{Z}^m + \{1/2\}^m$, whereas $\mathbf{x}_1 \in \mathbb{Z}^m$. This leads to the desired collisions, as can be seen in the upcoming Lemma 3.

3.2 Runtime analysis

In this section we analyze the runtime and success probability of the attack presented in Algorithm 1. We start by presenting our main result.

Theorem 1. *Let $n, m, q, c \in \mathbb{N}$, and $1 \leq \delta \in \mathbb{R}$ be fixed. Consider the following input distribution of $(q, r, \mathbf{A}, \mathbf{b}, \mathbf{B})$ for Algorithm 1. The modulus q and the attack parameter $r = 4c$ are fixed, $\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$, where $\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{m \times r}$, $\mathbf{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{m \times (n-r)}$, $\mathbf{b} = \mathbf{A} \begin{pmatrix} \mathbf{v} \\ \mathbf{s} \end{pmatrix} + \mathbf{e} \mod q$, where $\mathbf{v} \xleftarrow{\$} \{0,1\}^r$, $\mathbf{s} \xleftarrow{\$} \{0,1\}^{n-r}$, $\mathbf{e} \xleftarrow{\$} \{0,1\}^m$, and $\mathbf{B}$ is some lattice basis of $\Lambda_q(\mathbf{A}_2)$ with Hermite delta δ. Let all notations be as in the above description of the input distribution. Assume that the approximations given in Heuristic 2, Heuristic 3 and Heuristic 5 are in fact equations. Then, if Algorithm 1 terminates, it finds a valid binary error vector of the LWE with binary error instance $(\mathbf{A}, \mathbf{b})$. The probability that Algorithm 1 terminates is*

$$p_0 = 2^{-r} \binom{r}{2c} \prod_{i=1}^{m} \left(1 - \frac{2}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i, -1)} (1-t^2)^{\frac{m-3}{2}} dt \right),$$

where $B(\cdot, \cdot)$ denotes the Euler beta function (see [38]) and

$$r_i = \frac{\delta^{-2(i-1)+m} q^{\frac{m-n+r}{m}}}{2\sqrt{m/4}}.$$

In case that Algorithm 1 terminates, the expected number of operations is

$$2^{16} \binom{r}{c} \left(p \binom{2c}{c} \right)^{-1/2},$$

where

$$p = \prod_{i=1}^{m} \left(1 - \frac{1}{r_i B(\frac{m-1}{2}, \frac{1}{2})} J(r_i, m) \right),$$

and

$$J(r_i, m) = \begin{cases} \int_{-r_i - 1}^{r_i - 1} \int_{-1}^{z + r_i} (1-y^2)^{\frac{m-3}{2}} dy dz \\ \quad + \int_{r_i - 1}^{-r_i} \int_{z - r_i}^{z + r_i} (1-y^2)^{\frac{m-3}{2}} dy dz & \text{for } r_i < \frac{1}{2} \\ \int_{-r_i - 1}^{-r_i} \int_{-1}^{z + r_i} (1-y^2)^{\frac{m-3}{2}} dy dz & \text{for } r_i \geq \frac{1}{2}, \end{cases}$$

Remark 1 *Algorithm 1 gets some basis $\mathbf{B}$ as input. This basis has a certain quality, given by the Hermite delta δ. In practice, we can improve the attack by providing a basis with better, i.e., smaller, Hermite delta. We achieve this by running a basis reduction (e.g., BKZ) on $\mathbf{B}$ in a precomputation step. More details about the time necessary to achieve a certain Hermite delta and the trade-off between the runtime of the precomputation and the hybrid attack is given in Section 3.3.*

We postpone the proof of Theorem 1 to the end of this subsection, since we first need to develop some necessary tools. We start by giving the following definition, which is crucial to our analysis as the notion will be used frequently throughout this section.

Definition 2. *Let $m \in \mathbb{N}$. A vector $\mathbf{x} \in \mathbb{Z}^m$ is called $\mathbf{y}$-admissible for some vector $\mathbf{y} \in \mathbb{Z}^m$ if* $\mathrm{NP}(\mathbf{x}) = \mathrm{NP}(\mathbf{x} - \mathbf{y}) + \mathbf{y}$.

Intuitively, $\mathbf{x}$ being $\mathbf{y}$-admissible means that running the nearest plane algorithm on $\mathbf{x}$ and running it on $\mathbf{x} - \mathbf{y}$ yields the same lattice vector, since then we have $\mathbf{x} - \mathrm{NP}(\mathbf{x}) = (\mathbf{x} - \mathbf{y}) - \mathrm{NP}(\mathbf{x} - \mathbf{y})$.

We provide the following useful result about Definition 2.

Lemma 2. *Let $\mathbf{t}_1 \in \mathbb{R}^m, \mathbf{t}_2 \in \mathbb{R}^m$ be two arbitrary target vectors. Then the following are equivalent.*

1. $\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$.
2. $\mathbf{t}_1$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*
3. $\mathbf{t}_2$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*

Proof: Let $\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_2$ and $\mathbf{y} = \mathrm{NP}(\mathbf{t})$. By symmetry it suffices to show

$$\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathbf{y} \quad \Leftrightarrow \quad \mathrm{NP}(\mathbf{t}_1) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) + \mathbf{y}.$$

By definition, $\mathbf{t} - \mathbf{y}$ is a lattice vector and therefore $\mathrm{NP}(\mathbf{x} - (\mathbf{t} - \mathbf{y})) = \mathrm{NP}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^m$. This leads to

$$\mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y} - (\mathbf{t} - \mathbf{y})) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{t}) = \mathrm{NP}(-\mathbf{t}_2) = -\mathrm{NP}(\mathbf{t}_2).$$

Using this, one direction of the equivalence follows from

$$\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) + \mathbf{y} + \mathrm{NP}(\mathbf{t}_2) = -\mathrm{NP}(\mathbf{t}_2) + \mathbf{y} + \mathrm{NP}(\mathbf{t}_2) = \mathbf{y},$$

and the other from

$$\mathrm{NP}(\mathbf{t}_1) = \mathbf{y} - \mathrm{NP}(\mathbf{t}_2) = \mathbf{y} + \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}).$$

■

As we will see in our analysis, the expected runtime heavily depends on the following probability. Let all notations be as in Theorem 1 and $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1}$. For

$$W = \{\mathbf{w} \in \{0,1\}^r : \text{exactly } c \text{ entries of } \mathbf{w} \text{ are } 1\} \tag{4}$$

we define

$$p := \begin{cases} \Pr_{\mathbf{v}_1 \leftarrow W}[-\mathbf{A}_1\mathbf{v}_1 \text{is } \mathbf{e}'\text{-admissible} | \mathbf{v} - \mathbf{v}_1 \in W] & \text{if } \Pr_{\mathbf{v}_1 \leftarrow W}[\mathbf{v} - \mathbf{v}_1 \in W] > 0 \\ 0 & \text{else.} \end{cases} \tag{5}$$

Note that the analysis of the attack on the NTRU encryption proposed by Howgrave-Graham [25] also requires to calculate the probability p. In the original work, this is done experimentally. Replacing this probability estimation with the analytic methodology presented in the following removes the dependency on experimental support in the analysis of the hybrid attack. A first mathematical calculation of the probability p has already been presented by Hirschhorn et al. in [24]. However, their analysis requires an additional assumption that we no longer need.

Success probability

In this subsection we determine the probability that Algorithm 1 terminates. We start by giving a sufficient condition for this event.

Lemma 3. *Let all notations be as in Theorem 1 and let* $\mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1}$ *and* $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1}$*. Assume that* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *are guessed in separate loops of Algorithm 1 and satisfy* $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$*. Also let* $\mathbf{t}_1 = -\mathbf{A}_1\mathbf{v}_1$ *and* $\mathbf{t}_2 = \mathbf{b}' - \mathbf{A}_1\mathbf{v}_2$ *and assume* $\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2) = \mathbf{e}'$ *holds. Then* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *collide in at least one box chosen during Algorithm 1 and the algorithm outputs the error vector* $\mathbf{e}$ *of the given LWE instance.*

Proof: According to the notation used in Algorithm 1, let $\mathbf{x}_1 = -\mathrm{NP}(\mathbf{t}_1)$ correspond to $\mathbf{v}_1$ and $\mathbf{x}_2 = \mathrm{NP}(\mathbf{t}_2)$ correspond to $\mathbf{v}_2$. By assumption we have $\mathbf{x}_1 = \mathbf{x}_2 - \mathbf{e}'$. Using the definition it is easy to verify that $\mathbf{x}_1$ and $\mathbf{x}_2$ share at least one common address, since $\mathbf{e}' \in \{-1/2, 1/2\}^m$. Therefore $\mathbf{v}_1$ and $\mathbf{v}_2$ collide in at least one box. Again by assumption, we obtain $\mathbf{x} = \mathrm{NP}(\mathbf{b}' - \mathbf{A}_1\mathbf{v}) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2) = \mathbf{e}'$. Hence the algorithm outputs the error vector $\mathbf{e}$. ■

In the following lemma we give a lower bound on the probability that Algorithm 1 terminates in case $\mathrm{NP}(\mathbf{b}' - \mathbf{A_1}\mathbf{v}) = \mathbf{e}'$. This condition is necessary for the algorithm to terminate. We then determine the probability that this condition is fulfilled and combine both probabilities to the overall success probability.

Lemma 4. *Let all notations be as in Theorem 1 and let* $\mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1}$ *and* $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1}$*. Assume that if* $\mathbf{v}$ *has exactly* $2c$ *one-entries, then* $p > 0$*, where* p *is as defined in Equation (5). If* $\mathrm{NP}(\mathbf{b}' - \mathbf{A_1}\mathbf{v}) = \mathbf{e}'$*, then Algorithm 1 terminates with probability at least*

$$\tilde{p_0} = 2^{-r} \binom{r}{2c}.$$

Proof: We show that Algorithm 1 terminates if $\mathbf{v}$ consists of exactly $2c$ one-entries. The probability of this happening is exactly $\tilde{p_0}$, since there are 2^r binary vectors of length r, and $\binom{r}{2c}$ of them have exactly $2c$ one-entries. Assume that v consists of exactly $2c$ one-entries. The claim follows directly from Lemma 2 and Lemma 3. Since $p > 0$ there exist binary vectors $\mathbf{v}_1, \mathbf{v}_2 \in \{0, 1\}^r$, each containing exactly c one-entries, such that $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$ and $-\mathbf{A_1}\mathbf{v_1}$ is $\mathbf{e}'$-admissible. These vectors will eventually be guessed during Algorithm 1 if it does not terminate before. By Lemma 2 they satisfy

$$\mathrm{NP}(-\mathbf{A}_1\mathbf{v}_1) + \mathrm{NP}(\mathbf{b}' - \mathbf{A}_1\mathbf{v}_2) = \mathrm{NP}(\mathbf{b}' - \mathbf{A}_1\mathbf{v}) = \mathbf{e}'.$$

Lemmas 3 now guarantees that Algorithm 1 then outputs the error vector $\mathbf{e}$. ■

It remains to calculate the probability that in fact it holds that $\mathrm{NP}(\mathbf{b}' - \mathbf{A_1}\mathbf{v}) = \mathbf{e}'$, which is equivalent to $\mathbf{e}' \in \mathcal{P}(\overline{\mathbf{B}})$. The probability that this is the case is calculated later, see Equation 7. Using Equation 7 and assuming independence we obtain the following estimate.

Heuristic 2 *Let all notations be as in Theorem 1. The probability that Algorithm 1 terminates is approximately*

$$p_0 \approx 2^{-r} \binom{r}{2c} \prod_{i=1}^{m} \left(1 - \frac{2}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i,-1)} (1-t^2)^{\frac{m-3}{2}} dt\right).$$

Estimating the number of loops

The next step is to estimate the number of loops until the attack terminates.

Heuristic 3 *Let all notations be as in Theorem 1 and let* $\mathbf{b}' = \mathbf{b} - (1/2) \cdot \mathbf{1}$ *and* $\mathbf{e}' = \mathbf{e} - (1/2) \cdot \mathbf{1}$*. Assume that* $\mathrm{NP}(\mathbf{b}' - \mathbf{A_1 v}) = \mathbf{e}'$*, and that* $\mathbf{v}$ *consists of exactly* $2c$ *one-entries. Then the expected number of loops of Algorithm 1 is*

$$L \approx \binom{r}{c} \left(p \binom{2c}{c}\right)^{-1/2},$$

and the probability p*, as given in Equation (5), is*

$$p \approx \prod_{i=1}^{m} \left(1 - \frac{1}{r_i B(\frac{m-1}{2}, \frac{1}{2})} J(r_i, m)\right),$$

with $B(\cdot,\cdot)$*,* $J(\cdot,\cdot)$*, and* r_i *defined as in Theorem 1.*

In the following, we justify the heuristic. Assume that $\mathbf{v}$ consists of exactly $2c$ one-entries. In addition to W (see Equation (4)), define the set

$$V = \{\mathbf{v}_1 \in W : \mathbf{v} - \mathbf{v}_1 \in W \text{ and } -\mathbf{A}_1\mathbf{v}_1 \text{ is } \mathbf{e}'\text{-admissible}\}.$$

Note that W is the set from which Algorithm 1 samples the vectors $\mathbf{v}_1$. Lemma 3 shows that the attack succeeds if two vectors $\mathbf{v}_1, \mathbf{v}_2 \in V$ satisfying $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$ are sampled in different loops of Algorithm 1. Since otherwise the probability of success is close to zero, for simplicity we assume that the attack is only successful in this case. Therefore we need to estimate the necessary number of loops in Algorithm 1 until some $\mathbf{v}_1, \mathbf{v}_2 \in V$ with $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$ are found. Note that by Lemma 2 if $\mathbf{v}_1 \in V$, then also $\mathbf{v}_2 = \mathbf{v} - \mathbf{v}_1 \in V$.

We start by calculating the probability that a vector sampled during Algorithm 1 lies in V. By definition of p, this probability is given by

$$\Pr_{\mathbf{v}_1 \xleftarrow{\$} W}[\mathbf{v}_1 \in V] = p_1 p, \text{ where } p_1 := \Pr_{\mathbf{v}_1 \xleftarrow{\$} W}[\mathbf{v} - \mathbf{v}_1 \in W].$$

Therefore we expect to sample a vector $\mathbf{v}_1 \in V$ every $\frac{1}{p_1 p}$ loops in Algorithm 1. The above equation also implies $p_1 p = \frac{|V|}{|W|}$, which gives us

$$|V| = p_1 p |W| = p_1 p \binom{r}{c}.$$

The probability p_1 is given by $p_1 = \binom{2c}{c} / \binom{r}{c}$ as can be seen in the following lemma.

Lemma 5. *Let all notations be as in Theorem 1. Let* $\mathbf{v} \in \{0,1\}^r$ *be binary with exactly* $2c$ *one-entries and* W *be defined as in Equation (4). Then*

$$p_1 := \Pr_{\mathbf{v}_1 \xleftarrow{\$} W}[\mathbf{v} - \mathbf{v}_1 \in W] = \binom{2c}{c} / \binom{r}{c}.$$

Proof: There are exactly $\binom{r}{c}$ binary vectors of length r containing exactly c ones. For such a vector $\mathbf{v}_1$, the vector $\mathbf{v}_2 = \mathbf{v} - \mathbf{v}_1$ is binary and has exactly c one entries if and only if for all $i \in \{1, \ldots, r\}$ satisfying $(\mathbf{v}_1)_i = 1$, we also have $(\mathbf{v})_i = 1$. In other terms, the index set of one-entries of $\mathbf{v}_1$ has to be a subset of the index set of one-entries of $\mathbf{v}$, and there are exactly $\binom{2c}{c}$ such subsets. ■

Therefore by the birthday paradox, the expected number of loops in Algorithm 1 until some $\mathbf{v}_1, \mathbf{v}_2 \in V$ with $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$ are found can be estimated by

$$L \approx \frac{1}{p_1 p}\sqrt{|V|} = \frac{\sqrt{\binom{r}{c}}}{\sqrt{p_1 p}} = \binom{r}{c}\left(p\binom{2c}{c}\right)^{-1/2}.$$

It remains to approximate the probability p which we do in the following. Let $\mathbf{v}_1 \in \{0,1\}^r$ and $\mathbf{B}$ be some basis of $\Lambda_q(\mathbf{A}_2)$. By Lemma 1 there exist unique $\mathbf{u}_1, \mathbf{u}_2 \in \Lambda_q(\mathbf{A}_2)$ such that $\mathrm{NP}_{\mathbf{B}}(-\mathbf{A}_1\mathbf{v}_1) = -\mathbf{A}_1\mathbf{v}_1 - \mathbf{u}_1 \in \mathcal{P}(\overline{\mathbf{B}})$ and $\mathrm{NP}_{\mathbf{B}}(-\mathbf{A}_1\mathbf{v}_1 - \mathbf{e}') + \mathbf{e}' = -\mathbf{A}_1\mathbf{v}_1 - \mathbf{u}_2 \in \mathbf{e}' + \mathcal{P}(\overline{\mathbf{B}})$. Without loss of generality, in the following we assume $\mathbf{u}_1 = \mathbf{0}$, or equivalently $-\mathbf{A}_1\mathbf{v}_1 \in \mathcal{P}(\overline{\mathbf{B}})$. Now $-\mathbf{A}_1\mathbf{v}_1$ is $\mathbf{e}'$-admissible if and only if $\mathbf{u}_2 = \mathbf{u}_1 = \mathbf{0}$, which is equivalent to $\mathbf{e}' + \mathbf{A}_1\mathbf{v}_1 \in \mathcal{P}(\overline{\mathbf{B}})$. Therefore p is equal to the probability that $\mathbf{e}' + \mathbf{A}_1\mathbf{v}_1 \in \mathcal{P}(\overline{\mathbf{B}})$, which we determine in the following.

There exists some orthonormal transformation that aligns $\mathcal{P}(\overline{\mathbf{B}})$ along the standard axes of $\mathbb{R}^m$. By applying this transformation, we may therefore assume that $\mathcal{P}(\overline{\mathbf{B}})$ is aligned along the standard axes of $\mathbb{R}^m$ and that in consequence $\mathbf{e}'$ is a uniformly random vector of length $\sqrt{m/4}$. Because $\mathbf{A}_1$ is uniformly random in $\mathbb{Z}_q^{m \times r}$ we may further assume that $\mathbf{A}_1\mathbf{v}_1$ is uniformly random in $\mathcal{P}(\overline{\mathbf{B}})$, since without loss of generality we assume $\mathbf{A}_1\mathbf{v}_1 \in \mathcal{P}(\overline{\mathbf{B}})$. This gives rise to the following heuristic.

Heuristic 4 *The probability* p *as defined in Equation 5 (with respect to a reduced basis with Hermite delta* δ*) is*

$$p \approx \Pr_{\mathbf{t} \xleftarrow{\$} R,\, \mathbf{e}' \xleftarrow{\$} S_m(\sqrt{m/4})}[\mathbf{t} + \mathbf{e}' \in R],$$

where

$$S_m(\sqrt{m/4}) = \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x}\| = \sqrt{m/4}\}$$

is the surface of a sphere with radius $\sqrt{m/4}$ *centered around the origin and*

$$R = \{\mathbf{x} \in \mathbb{R}^m \mid \forall i \in \{1, \ldots, m\} : -R_i/2 \leq x_i < R_i/2\}$$

is the search rectangle with edge lengths

$$R_i = \delta^{-2(i-1)+m} q^{\frac{m-n+r}{m}}.$$

In the heuristic, the edge lengths are implied by the Geometric Series Assumption.

We continue calculating the approximation of p given in Heuristic 4. Let R and R_i be as defined in Heuristic 4. We can rewrite the approximation given in Heuristic 4 as

$$p \approx \Pr_{t_i \xleftarrow{\$} [-R_i/2, R_i/2], \mathbf{e}' \xleftarrow{\$} S_m(\sqrt{m/4})} [\forall i \in \{1, \ldots, m\} : t_i + e'_i \in [-R_i/2, R_i/2]].$$

Rescaling everything by a factor of $1/\sqrt{m/4}$ leads to

$$p \approx \Pr_{t_i \xleftarrow{\$} [-r_i, r_i], \mathbf{e}' \xleftarrow{\$} S_m(1)} [\forall i \in \{1, \ldots, m\} : t_i + e'_i \in [-r_i, r_i]],$$

where

$$r_i = \frac{R_i}{2\sqrt{m/4}} = \frac{\delta^{-2(i-1)+m} q^{\frac{m-n+r}{m}}}{2\sqrt{m/4}}. \tag{6}$$

Unfortunately, the distributions of the coordinates of $\mathbf{e}$ are not independent, which makes calculating p extremely complicated. In practice, however, the probability that $e_i \in [-R_i/2, R_i/2]$ is big for all but the last few indices i. This is due to the fact that by the Geometric Series Assumption typically only the last values R_i are small. Consequently, we expect the dependence of the remaining entries not to be strong. This assumption was already established by Howgrave-Graham [25] and appears to hold for all values of R_i appearing in practice.

It is therefore reasonable to assume that

$$p \approx \prod_{i=1}^{m} \Pr_{t_i \xleftarrow{\$} [-r_i, r_i], e'_i \xleftarrow{\$} D_m} [t_i + e'_i \in [-r_i, r_i]],$$

were D_m denotes the distribution on the interval $[-1, 1]$ obtained by the following experiment: sample a vector $\mathbf{w}$ uniformly at random on the unit sphere and then output the first (equivalently, any arbitrary but fixed) coordinate of $\mathbf{w}$.

Next we explore the density function of D_m. The probability that $e'_i \leq x$ for some $-1 < x < 0$, where $e'_i \xleftarrow{\$} D_m$, is given by the ratio of the surface area of a hyperspherical cap of the unit sphere in $\mathbb{R}^m$ with height $h = 1 + x$ and the surface area of the unit sphere. This is illustrated in Figure 2 for $m = 2$. The surface area of a hyperspherical cap of the unit sphere in $\mathbb{R}^m$ with height $h < 1$ is given by (see [31])

$$A_m(h) = \frac{1}{2} A_m I_{2h-h^2}\left(\frac{m-1}{2}, \frac{1}{2}\right),$$

Fig. 2. Two-dimensional hyperspherical cap

where $A_m = 2\pi^{m/2}/\Gamma(m/2)$ is the surface area of the unit sphere and

$$I_x(a,b) = \frac{\int_0^x t^{a-1}(1-t)^{b-1}dt}{B(a,b)}$$

is the regularized incomplete beta function (see [38]) and $B(a,b)$ is the Euler beta function.

Consequently, for $-1 < x < 0$, we have

$$\begin{aligned}
\Pr_{e_i' \xleftarrow{\$} D_m}[e_i' \le x] &= \frac{A_m(1+x)}{A_m} \\
&= \frac{1}{2} I_{2(1+x)-(1+x)^2}\left(\frac{m-1}{2}, \frac{1}{2}\right) \\
&= \frac{1}{2} I_{1-x^2}\left(\frac{m-1}{2}, \frac{1}{2}\right) \\
&= \frac{1}{2B(\frac{m-1}{2}, \frac{1}{2})} \int_0^{1-x^2} t^{\frac{m-3}{2}}(1-t)^{-1/2}dt \\
&= \frac{1}{2B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{m-3}{2}}(1-(1-t^2))^{-1/2}(-2t)dt \\
&= -\frac{1}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{m-3}{2}}|t|^{-1}tdt \\
&= \frac{1}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{m-3}{2}}dt. \qquad (7)
\end{aligned}$$

Together with

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i]}[t_i \le x] = \int_{-r_i}^{x} \frac{1}{2r_i}dy,$$

we can use a convolution to obtain

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], e_i' \xleftarrow{\$} D_m} [t_i + e_i' \leq x] = \frac{1}{2r_i B(\frac{k-1}{2}, \frac{1}{2})} \int_{-r_i-1}^{x} \int_{\max(-1, z-r_i)}^{\min(1, z+r_i)} (1-y^2)^{\frac{m-3}{2}} dy dz.$$

Since

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], e_i' \xleftarrow{\$} D_m} [t_i + e_i' \in [-r_i, r_i]] = 1 - 2 \left(\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], e_i' \xleftarrow{\$} D_m} [t_i + e_i' < -r_i] \right),$$

it suffices to calculate the integral

$$J(r_i, m) = \int_{-r_i-1}^{-r_i} \int_{\max(-1, z-r_i)}^{z+r_i} (1-y^2)^{\frac{m-3}{2}} dy dz \tag{8}$$

in order to calculate p, which we do in the following. For the lower end of the inner integral, we have to distinguish two cases. If $r_i < 1/2$, we can split it into

$$J(r_i, m) = \int_{-r_i-1}^{r_i-1} \int_{-1}^{z+r_i} (1-y^2)^{\frac{m-3}{2}} dy dz + \int_{r_i-1}^{-r_i} \int_{z-r_i}^{z+r_i} (1-y^2)^{\frac{m-3}{2}} dy dz,$$

while in the simpler case $r_i > 1/2$ we have

$$J(r_i, m) = \int_{-r_i-1}^{-r_i} \int_{-1}^{z+r_i} (1-y^2)^{\frac{m-3}{2}} dy dz.$$

This concludes our calculation of the probability p. All integrals can be calculated symbolically using sage [42], which allows an efficient calculation of p.

Time spend per loop cycle

With the estimation of the number of loops given, the remaining task is to estimate the time spend per loop cycle. Each cycle consists of four steps:

1. Guessing a binary vector.
2. Running the nearest plane algorithm (twice).
3. Calculating $\mathcal{A}^{(r)}_{\mathbf{x}_1} \cup \mathcal{A}^{(r)}_{\mathbf{x}'_1}$.
4. Dealing with collisions in the boxes.

We assume that the runtime of one inner loop of Algorithm 1 is dominated by the runtime of the nearest plane algorithm, as argued in the following. It is well known that sampling a binary vector is extremely fast. Furthermore, note that only very few of the 2^n addresses contain a vector, since filling a significant proportional would take exponential time. Consequently, collisions are extremely rare, and lines 8-11 of Algorithm 1 do not contribute much to the overall runtime.

An estimation by Howgrave-Graham [25] shows that for typical instances, the runtime of the nearest plane algorithm exceeds the time spent for storing the collision. We therefore omit the latter from our considerations.

Lindner and Peikert [32] estimated the time necessary to run the nearest plane algorithm to be about 2^{-16} seconds, which amounts to about 2^{15} bit operations on their machine. This leads to the following heuristic for the runtime of the attack.

Heuristic 5 *The average number of operations per inner loop in Algorithm 1 is* $N \approx 2^{16}$.

Total runtime

We are now able to prove our main theorem.

Proof (Theorem 1): By definition, every output of Algorithm 1 is a valid binary error vector of the given LWE with binary error instance. The rest follows directly from Heuristic 2, Heuristic 3, and Heuristic 5. ■

3.3 Minimizing the expected runtime

As previously mentioned in Remark 1, we can perform a basis reduction to obtain a lattice basis with smaller Hermite delta δ before running the actual attack in order to speed up the attack.

The Hermite delta δ determines the trade-off between the runtime of the precomputation and the actual attack. More precisely, choosing a smaller value for δ increases the runtime of the basis reduction, but at the same time decreases the runtime of the actual attack, since it increases the success probability and probability that a vector is $\mathbf{e}'$-admissible. From the attacker's perspective it is necessary to optimise the choice of δ and r, the Meet-in-the-Middle dimension.

The Meet-in-the-Middle dimension r balances the trade-off between the Meet-in-the-Middle and the lattice part of the attack. On the one hand, increasing r increases the complexity of Meet-in-the-Middle part, since more entries of the secret have to be guessed. On the other hand, it also increases the determinant of the lattice, making CVP easier and thereby increasing the probability that vectors are $\mathbf{e}'$-admissible. Finding the optimal values for r and δ is therefore, at first sight, non-trivial. We perform this task in the following way. For each r we find the optimal δ that minimizes the runtime. We then take the optimal r and the corresponding δ to determine the overall minimal runtime. Since there are only finitely many possible values for r, this can be performed numerically. Figure 3 shows the expected runtime for the attack depending on the Meet-in-the-Middle dimension r.

4 Comparison

In this section we consider other approaches to solve LWE with binary error and compare these algorithms to Algorithm 1. In particular we give upper bounds for the runtimes of the algorithms. A comparison of the most practical attacks, including the hybrid attack, is given in Table 1.

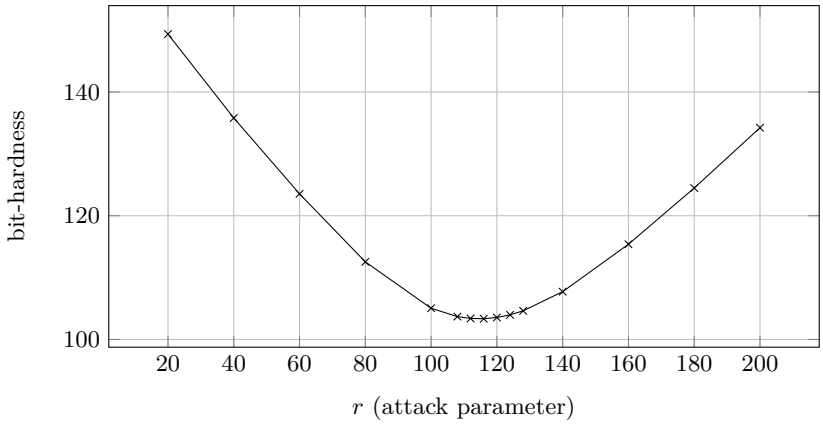

Fig. 3. Hardness of LWE instances with dimension of secret $n = 256$, number of samples $m = 512$ and modulus $q = 256$ for different values of r

Much of the analyses below are in a similar spirit to that given in the survey [5] for methods of solving standard LWE. However we are often able to specifically adapt the analysis for the binary error case. Note that to solve LWE with binary error, in addition to algorithms for standard LWE, one may also be able to apply algorithms for the related Inhomogeneous Short Integer Solution problem. A discussion of these algorithms is given in [10].

4.1 Number of samples

Some algorithms require a large number of LWE samples to be available in order to run. However it is well known (see, e.g., [16,35]) that if one has at least $m = \mathcal{O}(n^2)$ samples, the algorithm of Arora and Ge [6] solves LWE with binary error in polynomial time. Recall also that for reducing LWE with binary error to worst-case problems on lattices, one must restrict the number of samples to be $m = n\,(1 + \Omega(1/\log n))$ [35, Theorem 1.2]. On the other hand, with slightly more than linear samples, such as $m = \mathcal{O}(n \log\log n)$, the algorithm given in [1] is subexponential. Therefore if a scheme bases its security on the hardness of LWE with binary error, it is reasonable to expect that one has only access to at most linearly many samples. In the analysis below, we assume we the available number of samples is m, where m is linear in n. For concreteness, we fix $m = 2n$.

4.2 Algorithms for solving LWE

There are several approaches one could use to solve LWE or its variants (see the survey [5]). One may employ combinatorial algorithms such as the BKW [11,2] algorithm and its variants [3,17,23,27]. However, all these algorithms require far more samples than are available in the binary error case, and are therefore ruled out. In the comparison we also omit a Meet-in-the-Middle attack [5] or attacks based on the algorithm of Arora and Ge [6,1], as they will be slower than other methods, but nevertheless discuss them for completeness.

Distinguishing attack One can solve LWE via a distinguishing attack as described in [36,32]. The idea is to find a short vector $\|\mathbf{v}\|$ in the scaled dual lattice of $\mathbf{A}$, i.e. the lattice $\Lambda = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{wA} \equiv 0 \mod q\}$. Then, if the problem is to distinguish $(\mathbf{A}, \mathbf{b})$ where $\mathbf{b}$ is either formed as an LWE instance $\mathbf{b} = \mathbf{As} + \mathbf{e}$ or is uniformly random, one can use this short vector $\mathbf{v}$ as follows. Consider $\langle \mathbf{v}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle$ if $\mathbf{b}$ is from an LWE instance, which as the inner product of two short vectors, is small mod q. On the other hand, if $\mathbf{b}$ is uniform then $\langle \mathbf{v}, \mathbf{b} \rangle$ is uniform on $\mathbb{Z}_q$ so these cases can be distinguished if $\mathbf{v}$ is suitably small.

We determine how small a $\mathbf{v}$ which must be found as follows. Recall that our errors are chosen uniformly at random from $\{0, 1\}$. So they follow a Bernoulli distribution with parameter $1/2$, and have expectation $1/2$ and variance $1/4$. Consider the distribution of $\langle \mathbf{v}, \mathbf{e} \rangle$. Since the errors e_i are chosen independently, its expectation is $\frac{1}{2}\sum_{i=1}^{m} v_i$ and its variance is $\frac{1}{4}\sum_{i=1}^{m} v_i^2$. Since $\langle \mathbf{v}, \mathbf{e} \rangle$ is the sum of many independent random variables, asymptotically it follows a normal distribution with those parameters. Since the distinguishing attack success is determined by the variance and not the mean, and we can account for the mean, we assume it is zero. Then we can use the result of [32] to say that we can distinguish a Gaussian from uniform with advantage close to $\exp(-\pi(\|\mathbf{v}\| \cdot s/q)^2)$, where s is the width parameter of the Gaussian. In our case $s^2 = 2\pi \cdot \frac{1}{4}$ so we can distinguish with advantage close to $\epsilon = \exp(-\pi^2 \|\mathbf{v}\|^2 / 2q^2)$. Therefore to distinguish with advantage ϵ we require a vector $\mathbf{v}$ of length $\|\mathbf{v}\| = q \cdot \frac{\sqrt{2\ln(1/\epsilon)}}{\pi}$.

We calculate a basis of the scaled dual lattice Λ and find a short vector $\mathbf{v} \in \Lambda$ by lattice basis reduction. With high probability the lattice Λ has rank m and volume q^n [36,5]. By definition of the Hermite delta we therefore have $\|\mathbf{v}\| = \delta^m q^{n/m}$. So the Hermite delta we require to achieve for the attack to succeed with advantage ϵ is given by $\delta^m q^{n/m} = q \cdot \frac{\sqrt{2\ln(1/\epsilon)}}{\pi}$. Assuming that the number of samples m is large enough to use the 'optimal subdimension' $m = \sqrt{n \log(q)/\log(\delta)}$ [36], we rearrange to obtain

$$\log \delta = \frac{\left(\log(q) + \log\left(\frac{\sqrt{2\ln(1/\epsilon)}}{\pi}\right)\right)^2}{4n\log(q)}.$$

To establish the estimates for the runtime of this attack given in Table 1, we assume one has to run the algorithm about $1/\epsilon$ times to succeed, and consider δ as a function of ϵ. The overall running time is then given by $1/\epsilon$ multiplied the estimated time, according to Lindner and Peikert [32], to achieve $\delta(\epsilon)$. We pick the optimal ϵ such that this overall running time is minimized.

It is possible that we do not have enough samples to use the 'optimal subdimension'. We firstly calculate δ assuming we have as many samples as we need for the 'optimal subdimension', then check that the m this corresponds to is indeed less than or equal to $2n$. If so, we use the runtime estimate for that δ. If not, we take $m = 2n$ and derive a Hermite delta using $\delta^{2n} q^{n/2n} = q \cdot \frac{\sqrt{2\ln(1/\epsilon)}}{\pi}$.

The number of operation necessary to achieve this Hermite delta is estimated using Equation (1).

Reducing to uSVP One may solve LWE via Kannan's embedding technique [26], thus seeing an LWE instance as a unique shortest vector problem instance. This technique is used in [4,9]. We follow analogously the analysis in [4,5] for the LWE with binary error case and obtain that we require a Hermite delta of $\log(\delta) = \frac{[\log(q)-\log(2\tau\sqrt{\pi e})]^2}{4n\log(q)}$ for this attack to succeed. The number of operations necessary to achieve this Hermite delta is estimated using Equation (1). The full analysis is presented in the following.

In a nutshell, the idea of the attack is to add the vector $\mathbf{b}$ to the lattice

$$\Lambda_q(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{x} \in \mathbb{Z}^n : \mathbf{A}\mathbf{x} = \mathbf{v} \mod q\}.$$

Since $\mathbf{A}\mathbf{s}$ and $\mathbf{b}$ are lattice vectors, $\mathbf{e}$ is a very short lattice vector, and one can apply a solver for unique-SVP to recover $\mathbf{e}$ (the typical solver is BKZ2.0). Albrecht et al. [4] claimed that the attack succeeds with high probability if

$$\frac{\lambda_1(\Lambda_q(A))}{\|\mathbf{e}\|} \geq \tau\delta^m,$$

where $\lambda_1(\Lambda_q(A))$ is the shortest vector in $\Lambda_q(A)$, δ is the Hermite delta achieved by BKZ.0 and $\tau \approx 0.4$ is a constant depending on the unique-SVP solver used. Applying this attack on our LWE instances seems reasonable, since we have a very short error vector $\mathbf{e}$, which leads to a big gap $\frac{\lambda_1(\Lambda_q(A))}{\|\mathbf{e}\|}$. The common method for predicting $\lambda_1(\Lambda)$ for an arbitrary lattice is the Gaussian heuristic. It predicts the length of the shortest lattice vector in an n-dimensional lattice Λ via

$$\lambda_1(\Lambda) \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \det(\Lambda)^{1/m},$$

where the determinant $\det(\Lambda)$ is a constant of the lattice. Applying this heuristic shows that the attack succeeds if

$$\tau\delta^m \leq \frac{\frac{\Gamma(1+n/2)^{1/m}}{\sqrt{\pi}} \det(\Lambda)^{1/m}}{\|\mathbf{e}\|}.$$

We examine the right-hand side in more detail and see that

$$\frac{\frac{\Gamma(1+n/2)^{1/m}}{\sqrt{\pi}} \det(\Lambda)^{1/m}}{\|\mathbf{e}\|} \geq \frac{\frac{\Gamma(1+m/2)^{1/m}}{\sqrt{\pi}} q^{(m-n)/m}}{\sqrt{m}} \approx \frac{\frac{\sqrt{m/(2e)}}{\sqrt{\pi}} \cdot q^{1-n/m}}{\sqrt{m}} = \frac{q^{1-n/m}}{\sqrt{2e\pi}}.$$

The attack is therefore successful if

$$\frac{q^{1-n/m}}{\sqrt{2e\pi}\tau\delta^m} \geq 1. \tag{9}$$

Standard arguments show that the left-hand side of Inequality (9) is maximized for $m = \sqrt{n\log(q)/\log(\delta)}$ [36]. However, for most instances considered, this is requires more than the $2n$ samples provided. In this case, it is optimal to

set $m = 2n$ and use all samples. Consequently, the attack succeeds with high probability if

$$\delta \leq \sqrt[m]{\frac{q^{1-n/m}}{\sqrt{2\pi e}\tau}}.$$

If the optimal number of samples is smaller than $2n$, the optimal choice $m = \sqrt{n\log(q)/\log(\delta)}$ leads to $\delta = \exp(n\log(q)/m^2) = q^{n/m^2}$, which leads to

$$\frac{q^{1-n/m}}{\sqrt{2e\pi}\cdot\tau\delta^m} = \frac{q^{1-n/m}}{\sqrt{2e\pi}\cdot\tau}q^{-n/m} = \frac{q^{1-2n/m}}{\sqrt{2e\pi}\cdot\tau}.$$

An easy (but somehow exhausting) calculation shows that the smallest delta that leads to an attack dimension satisfying Inequality (9) is given by

$$\log(\delta) = \frac{[\log(q) - \log(2\tau\sqrt{\pi e})]^2}{4n\log(q)}.$$

Decoding The decoding approach for solving LWE was first described in [32] and is based on Babai's nearest plane algorithm [7]. The aim is to recover the error vector (so seeing LWE as a Bounded Distance Decoding instance). Recall (Lemma 1) that the error vector can be recovered using Babai's algorithm if it lies within the fundamental parallelepiped of the Gram-Schmidt basis. The idea of Lindner and Peikert in [32] is to widen the search parallelepiped to

$$\mathcal{P}_{\text{decoding}} = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x} = \sum_{i=1}^{n} \alpha_i d_i \overline{\mathbf{b}}_i \text{ for } -1/2 \leq \alpha_i < 1/2\},$$

where $d_1, ..., d_m$ are integers chosen by the attacker.

Following the analysis of Lindner and Peikert, we estimate that an attack on a reduced basis with Hermite delta δ requires about $2^{15} \cdot \prod_{i=1}^{m} d_i$ operations. However, the analysis of the success probability is more complicated. By definition of search parallelepiped, the attack succeeds if (and only if) the error $\mathbf{e}$ lies in the search rectangle $\mathcal{P}_{\text{decoding}}$. Under the same assumption as in Section 3.2 (and using the same error transformation), this probability can be estimated via

$$p_{\text{decoding}} \approx \prod_{i=1}^{m} \left(\Pr_{\mathbf{e}_i \xleftarrow{\$} D_m} [\mathbf{e}_i \in [-r_i, r_i]] \right)$$

where

$$r_i = d_i \frac{\delta^{-2(i-1)+m} q^{\frac{m-n}{m}}}{2\sqrt{m/4}}.$$

Together with Equation (7), this leads to

$$p_{\text{decoding}} \approx \prod_{i=1}^{m} \left(1 - \frac{2}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i,-1)} (1-t^2)^{\frac{m-3}{2}} dt \right).$$

A standard way to increase the runtime of the attack is to use basis reduction (like BKZ2.0) as precomputation. Predicting the runtime of BKZ2.0 according to Equation (1) leads to the runtime estimation

$$T_{\text{decoding}} \approx \frac{2^{1.8/\log_2(\delta)-110} \cdot 2.3 \cdot 10^9 + 2^{15} \sum_{i=1}^{m} d_i}{p_{\text{decoding}}}.$$

Using the same numeric optimization techniques as presented above to minimize the expected runtime leads to the complexity estimates given in Table 1.

Meet-in-the-Middle Attack We adapt the analysis in [5] to determine an upper bound on the complexity of a Meet-in-the-Middle attack on LWE with binary error. The proof follows [5] entirely analogously.

Theorem 2. *[5, Theorem 2] Let n, q parametrise an LWE instance with binary error. If there are m samples satisfying $m/q < 1/C$ for some constant $C > 1$ and $(2/q)^m \cdot 2^{n/2} = \text{poly}(n)$, then there is Meet-in-the-Middle algorithm which solves search LWE with binary error with non-negligible probability which runs in time $\mathcal{O}\left(2^{n/2}\left(\frac{n}{2}(5m+1) + (m+1)\log m\right)\right)$ and requires memory $m \cdot 2^{n/2}$.*

Proof: Given m samples $(\mathbf{a}_k, \langle \mathbf{a}_k, \mathbf{s}\rangle + e_k)$ split $\mathbf{a}_k = \mathbf{a}_k^l || \mathbf{a}_k^r$ in half and for each possibility $\mathbf{s}_i^l$ of the first half of $\mathbf{s}$ compute inner product of the first half of $\mathbf{a}_k$ and $\mathbf{s}_i^l$. Let the output of guess $\mathbf{s}_i^l$ for each of the m samples be $\mathbf{u}_{\mathbf{s}_i^l} = (\langle \mathbf{a}_1, \mathbf{s}_1^l \rangle, \ldots, \langle \mathbf{a}_m, \mathbf{s}_m^l \rangle)$. Store a table T whose entries map vectors $\mathbf{u}_{\mathbf{s}_i^l}$ to $\mathbf{s}_i^l$. Generating this table costs $m \cdot 2n \cdot 2^{n/2}$ operations since there are $2^{n/2}$ candidate secrets and for each we calculate m inner products. Sort the table into lexicographical ordering component-wise, which costs $\mathcal{O}\left(m \cdot 2^{n/2} \log\left(m \cdot 2^{n/2}\right)\right)$ operations. Now for each candidate $\mathbf{s}_j^r$ for the second half of the secret, and for each sample, compute $c_k - \langle \mathbf{a}_k^r, \mathbf{s}_j^r \rangle$, to form the vector $\mathbf{v}_{\mathbf{s}_j^r} = (c_1 - \langle \mathbf{a}_1^r, \mathbf{s}_j^r \rangle, \ldots, c_m - \langle \mathbf{a}_m^r, \mathbf{s}_j^r \rangle)$. Sort $\mathbf{v}_{\mathbf{s}_j^r}$ into T, which costs $\log |T| = \log\left(m \cdot 2^{n/2}\right) = \log m + n/2$ operations. Since there are $2^{n/2}$ possible second $\mathbf{s}_j^r$ the total cost of this step is $2^{n/2}(\log m + n/2)$. When $\mathbf{s}_j^r$ is sorted into the list, check which $\mathbf{u}_{\mathbf{s}_i^l}$ it is between. If $\mathbf{v}_{\mathbf{s}_j^r}$ and $\mathbf{u}_{\mathbf{s}_i^l}$ have a binary difference, return $\mathbf{s}_i^l$ and treat $\mathbf{s}_i^l || \mathbf{s}_j^r$ as a candidate secret, and check if it is correct. This procedure would then identify the correct secret, so long as there is not a wrap around mod q, since if $\mathbf{s}_{i'}^l || \mathbf{s}_{j'}^r$ is the correct secret then $\mathbf{v}_{\mathbf{s}_{j'}^r} - \mathbf{u}_{vecs_{i'}^l} = (e_1, \ldots, e_m) \mod q$ which is a binary vector. Let $b_k = \langle \mathbf{a}_k, \mathbf{s} \rangle \mod q$, then a wrap around error will not occur as long as $b_k \neq q - 1$. The probability that one component has $b_k = q - 1$ is $1/q$ so by the union bound the probability that at least one component has $b_k = q - 1$ is $\leq m/q$. We want to bound m so that this event happens only with probability at most $1/C$ for some constant C, i.e., $m/q < 1/C$. It remains to consider the chance of a false positive, that is, an incorrect candidate secret $\mathbf{s}_i^l$ being suggested for some $\mathbf{s}_j^r$. Since a_k is uniformly random, for any $\mathbf{s}_i^l$, the vector $\mathbf{u}_{\mathbf{s}_i^l}$ is also random with each component taking one of q values. A wrong $\mathbf{s}_j^r$ will produce a $\mathbf{v}_{\mathbf{s}_j^r}$

that matches to $\mathbf{u}_{\mathbf{s}_i^l}$ only if its difference is 0 or 1 on every component. Therefore the chance of a false positive is $(2/q)^m$. There are $2^{n/2} - 1$ wrong choices for $\mathbf{s}_i^l$, so we expect to test $(2/q)^m \cdot 2^{n/2}$ candidates per $\mathbf{s}_j^r$. Hence we require $(2/q)^m \cdot 2^{n/2} = \text{poly}(n)$. ■

Theorem 2 gives an upper bound on the complexity of a Meet-in-the-Middle attack, but note that it also takes at least $m \cdot 2n \cdot 2^{n/2}$ operations just to generate the table, excluding the costs of the other steps, e.g., sorting. Hence, we do not include estimates for the runtime of this approach in Table 1, as there is always a faster choice.

Arora-Ge algorithm The basic idea of the Arora-Ge algorithm [6] is setting up a system of nonlinear equations of which the secret is a root, and then solving the system. Solving may be via linearisation (as in [6]) or by Gröbner basis methods (as in [1]). The authors of [1] consider the complexity of their algorithm for solving LWE with binary error for various numbers of samples (see [1, Theorem 7]. In particular, if $m = 2n$ their algorithm solves LWE with binary error in time $\mathcal{O}(n^2 \cdot 2^{0.43\omega n})$ where $2 \leq \omega < 3$ is the linear algebra constant. Although this is an upper bound, it is significantly more than the cost of the other attacks. Therefore we do not expect that the actual runtime is smaller than for the other possible approaches, so we omit this algorithm from consideration in Table 1.

4.3 Comparison

Instance	n	q	$\log_2(T_{\text{Hybrid attack}})$	$\log_2(T_{\text{Decoding}})$	$\log_2(T_{\text{uSVP}})$	$\log_2(T_{\text{Distinguishing}})$
I	128	256	**41**	67	82	**37**
II	160	256	**55**	77	122	**62**
III	192	256	**71**	88	162	**85**
IV	224	256	**87**	**102**	165	109
V	256	256	**103**	**117**	203	132
VI	288	256	**120**	**136**	254	154
VII	320	256	**136**	**158**	327	176
VIII	352	256	**153**	**185**	443	198

Table 1. Comparison of attacks on LWE with binary error using at most $m = 2n$ samples. $\log_2(T_{\text{attack}})$ denotes the bit operations required to perform the algorithm described in 'attack' [36].

Table 1 shows the runtime of the hybrid attack compared with some of the possible attacks described above on at most $m = 2n$ samples of LWE with binary error. For algorithms requiring lattice reduction, we choose whichever is the fewer of $m = 2n$ or the 'optimal subdimension' $m = \sqrt{n \log(q)/\log(\delta)}$ [36].

Acknowledgements. Player was supported by an ACE-CSR PhD grant. This work has been co-funded by the DFG as part of project P1 within the CRC 1119 CROSSING. We thank Sean Murphy for useful discussions and comments.

References

1. M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret. Algebraic algorithms for LWE problems. *IACR Cryptology ePrint Archive*, 2014:1018, 2014. 2, 3, 17, 22
2. M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret. On the complexity of the BKW algorithm on LWE. *Des. Codes Cryptography*, 74(2):325–354, 2015. 17
3. M. R. Albrecht, J. Faugère, R. Fitzpatrick, and L. Perret. Lazy modulus switching for the BKW algorithm on LWE. In Krawczyk [28], pages 429–445. 17
4. M. R. Albrecht, R. Fitzpatrick, and F. Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In H. Lee and D. Han, editors, *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, volume 8565 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2013. 2, 19
5. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015. 3, 5, 17, 18, 19, 21
6. S. Arora and R. Ge. New algorithms for learning in presence of errors. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011. 3, 17, 22
7. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem (shortened version). In K. Mehlhorn, editor, *STACS '86*, volume 82 of *Lecture Notes in Computer Science*, pages 13–20. Springer, 1985. 5, 20
8. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. 5
9. S. Bai and S. D. Galbraith. Lattice decoding attacks on binary LWE. In W. Susilo and Y. Mu, editors, *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, volume 8544 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 2014. 2, 3, 6, 19
10. S. Bai, S. D. Galbraith, L. Li, and D. Sheffield. Improved exponential-time algorithms for inhomogeneous-sis. *IACR Cryptology ePrint Archive*, 2014:593, 2014. 17
11. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. 17
12. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011. 1
13. R. Canetti and J. A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013. 24, 25

14. H. Chen, K. E. Lauter, and K. E. Stange. Attacks on search RLWE. *IACR Cryptology ePrint Archive*, 2015:971, 2015. 2, 3
15. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In D. H. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011. 5
16. N. Döttling and J. Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2013. 17
17. A. Duc, F. Tramèr, and S. Vaudenay. Better algorithms for LWE and LWR. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 173–202. Springer, 2015. 17
18. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In Canetti and Garay [13], pages 40–56. 1
19. K. Eisenträger, S. Hallgren, and K. E. Lauter. Weak instances of PLWE. In A. Joux and A. M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2014. 2, 3
20. Y. Elias, K. E. Lauter, E. Ozman, and K. E. Stange. Provably weak instances of ring-lwe. In Gennaro and Robshaw [21], pages 63–92. 2, 3
21. R. Gennaro and M. Robshaw, editors. *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*. Springer, 2015. 24, 25
22. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012. 1
23. Q. Guo, T. Johansson, and P. Stankovski. Coded-bkw: Solving LWE using lattice codes. In Gennaro and Robshaw [21], pages 23–42. 17
24. P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing ntruencrypt parameters in light of combined lattice reduction and MITM approaches. In M. Abdalla, D. Pointcheval, P. Fouque, and D. Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 437–455, 2009. 2, 9
25. N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007. 2, 3, 9, 13, 15

26. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, Aug. 1987. 19
27. P. Kirchner and P. Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Gennaro and Robshaw [21], pages 43–62. 17
28. H. Krawczyk, editor. *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*. Springer, 2014. 23, 25
29. K. Laine and K. E. Lauter. Key recovery for LWE in polynomial time. *IACR Cryptology ePrint Archive*, 2015:176, 2015. 2, 3
30. A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-based group signature scheme with verifier-local revocation. In Krawczyk [28], pages 345–361. 1
31. S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011. 13
32. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In A. Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011. 1, 2, 5, 7, 16, 18, 20
33. M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In E. Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2013. 2, 5
34. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. 1
35. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In Canetti and Garay [13], pages 21–39. 2, 3, 17
36. D. Micciancio and O. Regev. Lattice-based cryptography. pages 147–191. Springer, Berlin, Heidelberg, New York, 2009. 18, 19, 22
37. D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. *IACR Cryptology ePrint Archive*, 2015:1123, 2015. 5
38. F. W. Olver. *NIST handbook of mathematical functions*. Cambridge University Press, 2010. 8, 14
39. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In M. Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009. 4
40. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011. 4
41. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. 4
42. W. Stein et al. *Sage Mathematics Software (Version 6.3)*. The Sage Development Team, 2014. `http://www.sagemath.org`. 15