Revisiting TFHE security against hybrid lattice attacks

No Author Given

No Institute Given

Abstract. Fast Fully Homomorphic Encryption scheme over the Torus (TFHE) is one of the fastest homomorphic encryption scheme based on the Learning with Errors (LWE) problem. In this paper, we investigate the actual security level of TFHE. To do so, we improve the dual lattice attack used in the original security estimate of the scheme. More precisely, we use the dual attack together with a continuous relaxation of lazy modulus switching technique on a projected sublattice, allowing to generate instances of the LWE problem of slightly bigger noise which correspond to a fraction of the secret key. Then, we search for this remaining part of the secret key by computing the corresponding noise for each candidate using the constructed LWE samples. As the TFHE keys are binary vectors, we can perform the search step very efficiently by exploiting the recursive structure of this search space. This approach offers a trade-off between the cost of lattice reduction and the complexity of the search part which allows to speed up the attack.
We provide an estimate of the complexity of our method for various parameters under three different cost models for lattice reduction and show that security level of the TFHE scheme should be re-evaluated according to the proposed improvement. Our estimates show that the current security level of the TFHE scheme should be reduced by 27 to 49 bits, depending on the model used.

1 Introduction

Fully homomorphic encryption (FHE) allows performing arbitrary operations on encrypted data without decrypting it. The first fully homomorphic encryption scheme was introduced by Gentry in 2009 [25]. Since that time many FHE schemes were proposed, each offering new improvements (e.g. [10,16,20,21,24]). In this paper, we are interested in evaluating bit-security of the Fast Fully Homomorphic Encryption scheme over the Torus [17,18,19], which is currently the FHE scheme with the fastest implementation that we are aware of.

Security of the TFHE scheme relies on the hardness of the Learning with Errors (LWE) problem, which was first introduced by Regev [33] in 2005. LWE is provably as hard as certain lattice approximation problems in the worst-case [11]. Since its introduction, the LWE problem has been a rich source of cryptographic constructions. The original Regev's work presents an LWE-based public-key encryption scheme, but besides public key encryption, this problem allows building

feature-rich constructions like identity-based encryption [23] and fully homomorphic encryption [12].

The security of a cryptosystem, of course, depends on the complexity of the most efficient known attack against it. In particular, to estimate the security of an LWE-based construction, it is important to know which attack is the best for the parameters used in the construction. It can be a difficult issue; indeed, the survey of existing attacks against LWE given in [6] shows that no known attack would be the best for all sets of LWE parameters. In the case of THFE, the authors adapted and used the dual distinguishing lattice attack from [2],to evaluate the security of their scheme. As it turns out, this leads to an overestimate of the security of their THFE parameters.

1.1 Our contribution.

In this work, we generalize the dual lattice attack which is currently used to evaluate the security of the TFHE scheme. First, we present a complete and detailed analysis of the standard dual lattice attack[1] on LWE from [19]. Then, we show that applying the dual attack to a projected sublattice and combining it with the search for a fraction of the key can yield a more efficient attack.

More precisely, our attack starts by applying lattice reduction to a projected sublattice in the same way it is applied to the whole lattice in the dual attack with lazy modulus switching. This way we generate LWE instances with bigger noise but in smaller dimension, corresponding to a part of the secret key. Then, the freshly obtained instances are used to recover the remaining fraction of the secret key. For each candidate for this missing part, we compute the noise vector corresponding to the LWE instances obtained at the previous step. This allows performing a majority voting to discriminate the most likely candidates. As the TFHE scheme uses binary vectors for keys, this step boils down to computing a product of a matrix composed of the LWE samples with the matrix composed of all binary strings of length equal to the dimension of the part of the secret key that we are searching for. We show that this computation can be performed efficiently thanks to the recursive structure of the corresponding search space. The number of bits of the secret key that the attack aims to guess is an additional parameter for tuning the complexity of the attack. Hence, this hybrid approach offers a trade-off between the quality of lattice reduction in the dual attack part and the time spent in the exhaustive search part. Together with the efficient computation of the noise for each candidate, the optimal parameters for this trade-off gives an asymptotic improvement of the whole complexity.

We estimate the complexity of both attacks for a large set of various LWE parameters and particularly for the parameters of TFHE. In order to evaluate the cost of the lattice part of the attacks, we use three different models of behavior of lattice reduction algorithms. See Section 2.3 for the description of the models

[1] We shall remark that this attack is slightly more subtle than the classical dual lattice attack, as it encompasses a continuous relaxation of the *lazy modulus switching* technique of [2].

used. For all the models our attack significantly outperforms the attack used before. Table 1 presents the results of our estimates for the parameters of TFHE.

Table 1: Security of the parameters of TFHE scheme. λ denotes bit-security, "switching key" and "bootstrapping key" denote two different sets of parameters used in TFHE.

BKZ model	**switching key**		**bootstrapping key**	
delta-squared	attack	λ	attack	λ
	dual	169	dual	204
	our attack	119	our attack	159
sieving	dual	135	dual	144
	our attack	114	our attack	132
enumeration	dual	195	dual	230
	our attack	136	our attack	179

1.2 Related work.

The survey [6] outlines three strategies for attacks against LWE: exhaustive search, BKW algorithm [3,9] and lattice reduction. Lattice attacks against LWE can be separated into three categories depending on the lattice used: distinguishing dual attacks [2], decoding (primal) attacks [30, 31], and solving LWE by reducing it to unique Shortest Vector Problem [5].

The idea of hybrid lattice reduction attack was introduced by Howgrave–Graham in [27]. He proposed to combine a meet-in-the-middle attack with lattice reduction to attack NTRUEncrypt. Then, Buchmann et al. adapted Howgrave–Graham's attack to the settings of LWE with binary error [13] and showed that the hybrid attack outperforms existing algorithms for some sets of parameters. This attack uses the decoding (primal) strategy for the lattice reduction part. Following these two works, Wunderer have provided an improved analysis of the hybrid decoding lattice attack and meet-in-the-middle attack and re-estimated security of several LWE and NTRU based cryptosystems in [36]. Also, very recently, a similar combination of primal lattice attack and meet-in-the-middle attack was applied to LWE with ternary and sparse secret [35]. They show that the hybrid attack can also outperform other attacks in case of ternary and sparse secrets for parameters typical for FHE schemes.

Outline. This paper is organized as follows. In Section 2, we provide background. In Section 3, we revisit the dual lattice attack which is currently used to estimate the actual security level of TFHE. In Section 4, we describe our

hybrid dual lattice attack. In Section 5, we compare the complexities of two attacks, revisit security estimate of TFHE scheme and provide some experimental evidence.

2 Background

We use column notation for vectors. We use bold lower-case letters to denote vectors (e.g. $\mathbf{x}$) and bold upper-case letters to denote matrices (e.g. $\mathbf{A}$). For a vector $\mathbf{x}$, $\mathbf{x}^t$ denotes the transpose of $\mathbf{x}$. Base-2 logarithm is denoted as log, natural logarithm is denoted as ln. We denote the set of real numbers modulo 1 as the torus $\mathbb{T}$. For a finite set S, we denote by $\mathcal{U}(S)$ the discrete uniform distribution on S. For any compact set $S \subset \mathbb{R}^n$, uniform distribution over S is denoted as $\mathcal{U}(S)$. When S is not specified, $\mathcal{U}$ denotes uniform distribution over $(-0.5; 0.5)$.

2.1 LWE problem.

Abstractly, all operations of the TFHE scheme are defined on the real torus $\mathbb{T}$, and to estimate the security of the scheme it is convenient to consider a scale-invariant version of LWE problem.

Definition 1 (Learning with Errors, [11, Definition 2.11]). *Let $n \geqslant 1$, $\mathbf{s} \in \mathbb{Z}^n$, ξ be a distribution over $\mathbb{R}$ and $\mathcal{S}$ be distribution over $\mathbb{Z}^n$. We define the $LWE_{\mathbf{s},\xi}$ distribution as the distribution over $\mathbb{T}^n \times \mathbb{T}$ obtained by sampling $\mathbf{a}$ from $\mathcal{U}(\mathbb{T}^n)$, sampling e from ξ and returning $(\mathbf{a}, \mathbf{a}^t\mathbf{s} + e)$.*

Decision-LWE: distinguish, given arbitrarily many samples, between $\mathcal{U}(\mathbb{T}^n \times \mathbb{T})$ and $LWE_{\mathbf{s},\xi}$ distribution for a fixed $\mathbf{s}$ sampled from $\mathcal{S}$.

Search-LWE: given arbitrarily many samples from $LWE_{\mathbf{s},\xi}$ distribution with fixed $\mathbf{s} \leftarrow \mathcal{S}$, recover $\mathbf{s}$.

To complete the description of the LWE problem we need to choose error distribution ξ and distribution of the secret key $\mathcal{S}$. Following the description of the TFHE scheme, we choose $\mathcal{S}$ to be $\mathcal{U}(\{0,1\}^n)$ and ξ to be centered continuous Gaussian distribution, that is, we consider binary LWE problem. In [11] it is shown that LWE with binary secret remains hard. Further $\text{LWE}_{\mathbf{s},\sigma}$ distribution denotes $\text{LWE}_{\mathbf{s},\xi}$ distribution for some fixed $\mathbf{s} \leftarrow \{0,1\}^n$ and where ξ is the Gaussian distribution with mean 0 and standard deviation σ.

2.2 Lattices

A *lattice* Λ is a discrete subgroup of $\mathbb{R}^d$. A lattice Λ can be generated by integer linear combinations of vectors of its *basis* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\} \subset \mathbb{R}^d$:

$$\Lambda = \mathcal{L}(\mathbf{B}) := \mathbb{Z}\mathbf{b}_1 + \ldots \mathbb{Z}\mathbf{b}_d.$$

Bases are not unique, one lattice basis may be transformed into another one by applying an arbitrary unimodular transformation. The *volume of the lattice* $\text{vol}(\Lambda)$ is equal to the square root of the determinant of the Gram matrix

$\mathbf{B}^t\mathbf{B}$: $\mathrm{vol}(\Lambda) = \sqrt{\det(\mathbf{B}^t\mathbf{B})}$. For every lattice Λ we denote the length of its shortest non-zero vector as $\lambda_1(\Lambda)$. Minkowski's theorem states that $\lambda_1(\Lambda) \leqslant \sqrt{\gamma_n} \cdot \mathrm{vol}(\Lambda)^{1/n}$ for any d-dimensional lattice Λ, where $\gamma_d < d$ is d-dimensional Hermite's constant. The problem of finding the shortest non-zero lattice vector is called the *Shortest Vector Problem*(SVP). It is known to be NP-hard under randomized reduction [1].

2.3 Lattice reduction

A lattice reduction algorithm is an algorithm which, given as input some basis of the lattice, finds a basis that consists of relatively short and nearly orthogonal vectors. The quality of the basis produced by lattice reduction algorithms is usually measured by the Hermite factor $\delta = \dfrac{\|\mathbf{b}_1\|}{\det(\Lambda)^{1/d}}$, where $\mathbf{b}_1$ is the first vector of the outputted basis. Hermite factors bigger than $\left(\frac{4}{3}\right)^{n/4}$ can be reached in polynomial time by the so-called LLL algorithm [29]. In order to obtain smaller Hermite factors, blockwise lattice reduction algorithms, like BKZ-2.0 [15] or D-BKZ [32], are used. The BKZ algorithm takes as input a basis of dimension d and proceeds by solving SVP on lattices of dimension $\beta < d$ using sieving [8] or enumeration [22]. The quality of the output of BKZ depends on the blocksize β. In [26] it is shown that after a polynomial number of calls to SVP oracle, the BKZ algorithm with blocksize β produces a basis $\mathbf{B}$ that achieves the following bound:

$$\|\mathbf{b}_1\| \leqslant 2\gamma_\beta^{\frac{d-1}{2(\beta-1)}+\frac{3}{2}} \cdot \mathrm{vol}(\mathbf{B})^{1/d}.$$

However, up to our knowledge, there is no closed formula that tightly connects the quality and complexity of the BKZ algorithm. In this work, we use experimentally obtained models from [3, 4] in order to estimate the running time and quality of the output of lattice reduction. They are based on the following two assumptions on the quality and shape of the output of BKZ. The first assumption states that the BKZ algorithm outputs vectors with balanced coordinates, while the second assumption connects Hermite factor δ and chosen blocksize β.

Assumption 1 *Given as input basis B of d-dimensional lattice Λ , BKZ outputs a vector of norm close to $\delta^d \cdot \det(\Lambda)^{1/d}$ with balanced coordinates. Each coordinate of produced vector follows the distribution that can be approximated by Gaussian with mean 0 and standard deviation $\delta^d \det(\Lambda)^{1/d}/\sqrt{d}$.*

Assumption 2 *BKZ with blocksize β achieves Hermite factor*

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

This assumption is experimentally verified in [14].

BKZ cost models. To estimate the running time of BKZ we use three different models. The first model is an extrapolation by Albrecht [3] et al. of the Liu–Nguyen datasets [31]. According to that model the logarithm of the running time of BKZ-2.0 (expressed in bit operations) is a quadratic function of $\log(\delta)^{-1}$:

$$\log(T(\mathrm{BKZ}_\delta)) = \frac{0.009}{\log(\delta)^2} - 27.$$

We further refer to this model as the delta-squared model. The model was used in [19] to estimate the security of TFHE.

Another cost model [4] assumes that the running time of BKZ with blocksize β for d-dimensional basis is $T(\mathrm{BKZ}_{\beta,d}) = 8d \cdot T(\mathrm{SVP}_\beta)$, where $T(\mathrm{SVP}_\beta)$ is the running time of SVP oracle in dimension β. For SVP oracle we use the following two widely used models:

$$\begin{gathered}\text{Sieving model: } T(\mathrm{SVP}_\beta) = 2^{0.292\beta+16.4},\\ \text{Enumeration model: } T(\mathrm{SVP}_\beta) = 2^{0.187\beta\log(\beta)-1.019\beta+16.1}.\end{gathered}$$

2.4 Modular Gaussian distribution [19, Section 2.1].

Let $\sigma > 0$. For all $x \in \mathbb{R}$ the density of centered Gaussian distribution of standard deviation σ is defined as $\rho_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)$. We define the distribution that is obtained by sampling a centered Gaussian distribution of standard deviation σ and reducing it modulo 1 as the *modular Gaussian distribution* of parameter σ and denote it as $\mathcal{G}_\sigma$.

Its support is $\left(-\frac{1}{2};\frac{1}{2}\right)$ and the probability density function is given by the absolutely convergent series:

$$g_\sigma(x) = \sum_{k\in\mathbb{Z}} \rho_\sigma(x+k).$$

For big values of σ, the sum that defines the density of modular Gaussian can be closely approximated.

Lemma 1. *As $\sigma \to \infty$, $g_\sigma(x) = 1 + 2e^{-2\pi^2\sigma^2}\cos(2\pi x) + O(e^{-8\pi^2\sigma^2})$.*

Proof. The Fourier transform of the Gaussian function $\rho_{\sigma,m}(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x+m)^2}{2\sigma^2}}$ is given by $\widehat{\rho}_{\sigma,m}(y) = e^{-2\pi^2\sigma^2m^2+2\pi imx}$. Then, using the Poisson summation formula, we obtain:

$$\begin{aligned} g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma}\sum_{k\in\mathbb{Z}} e^{-\frac{(k+x)^2}{2\sigma^2}} = 1 + 2\sum_{k>0} e^{-2\pi^2\sigma^2k^2}\cos(2\pi kx) = \\ 1 + 2e^{-2\pi^2\sigma^2}\cos(2\pi x) + O(e^{-8\pi^2\sigma^2}). \end{aligned} \tag{1}$$

2.5 Berry-Esseen inequality.

The Berry-Esseen inequality shows how closely the distribution of the sum of independent random variables can be approximated by a Gaussian distribution.

Theorem 1. *Let $X_1, \ldots, X_n$ be independent random variables such that for all $i \in \{1, \ldots, n\}$ $\mathbb{E}\{X_i\} = 0$, $\mathbb{E}\{X_i^2\} = \sigma_i^2 > 0$, and $\mathbb{E}\{|X_i|^3\} = \rho_i < \infty$. Denote the normalized sum*

$$\frac{\sum_{i=1}^{n} X_i}{\sqrt{\sum_{i=1}^{n} \sigma_i^2}}$$

as S_n. Also denote F_n the cumulative distribution function of S_n, and denote Φ the cumulative distribution function of standard normal distribution. Then there exists a constant C_0 such that

$$\sup_{x \in \mathbb{R}} |F_n(x) - \Phi(x)| \leqslant C_0 \frac{\sum_{i=1}^{n} \rho_i}{\left(\sum_{i=1}^{n} \sigma_i^2\right)^{3/2}}.$$

We use the Berry-Esseen inequality in order to estimate how closely the distribution that we obtain after the lattice reduction step of the dual attack can be approximated by the discrete Gaussian distribution (see Lemma 5). The Berry-Esseen inequality requires a finite third absolute moment of the random variables. In the proof of Lemma 5, we need the third absolute moment of a Gaussian distribution.

Lemma 2. *Let $\sigma > 0$. Let X be a random variable of a Gaussian distribution with mean 0 and standard deviation σ^2. Then, $\mathbb{E}\{|X|^3\} = 2\sqrt{\frac{2}{\pi}}\sigma^3$.*

Proof. Classically we have:

$$\mathbb{E}\{|X|^3\} = 2 \cdot \frac{1}{\sqrt{2\pi\sigma}} \int_0^\infty x^3 e^{-\frac{x^2}{2\sigma^2}} dx = 2\sqrt{\frac{2}{\pi}}\sigma^3.$$

2.6 Distinguishing distributions.

Let P_0 and P_1 denote two probability distributions with density functions p_0 and p_1 respectively. A *distinguisher* D is a deterministic algorithm that, given as input a single point x from P_0 or P_1, outputs a guess of the underlying distribution of x. Let t be some constant. The following distinguisher is called the *likelihood-ratio distinguisher* for P_0 and P_1:

$$D_t(x) = \begin{cases} 1, & \text{if } \dfrac{p_1(x)}{p_0(x)} \geqslant t, \\ 0, & \text{otherwise;} \end{cases}$$

t is called the threshold of D_t. The following lemma describes the optimal distinguisher for two probability distributions.

Lemma 3 ([28, Problem 3.10]). *Let $t > 0$ be some fixed constant. Let D_t be a likelihood-ratio distinguisher for the distributions P_0 and P_1. Assume that the probabilities that the distinguisher gets as input a point from P_0 or P_1 are equal to π_0 and $\pi_1 = 1 - \pi_0$ respectively. Then the probability of an error of the distinguisher is*

$$p_{error} = \pi_0 \cdot \mathbb{P}\Big\{\frac{p_1(x)}{p_0(x)} \geqslant t \,\Big|\, x \sim P_0\Big\} + \pi_1 \cdot \mathbb{P}\Big\{\frac{p_1(x)}{p_0(x)} < t \,\Big|\, x \sim P_1\Big\}.$$

Probability p_{error} is minimized when $t = \pi_0/\pi_1$.

In this paper, we are interested in the following problem. Suppose that we are given a sample of N points sampled independently from either uniform or modular Gaussian distribution. The goal is to guess the distribution. The optimal way to do it is to use Lemma 3 with $P_0 = \mathcal{U}^N$ and $P_1 = \mathcal{G}_\sigma^N$. But finding the threshold for the optimal distinguisher requires computing the likelihood function for $\mathcal{G}_\sigma^N$, which is rather cumbersome. Instead, for distinguishing the distributions we use the following slightly less efficient but much more simple procedure. First, we construct a distinguisher that, given one point from either uniform or modular Gaussian distribution, guesses the distribution with symmetric error probability. That is, the probability of error is the same independently of the distribution of the input point. Then, we apply the symmetric distinguisher to each point from the sample. The answer given by the distinguisher at the majority of points is our final guess for the distribution.

In the following paragraph, we recall how the probability of the correct guess of the majority voting depends on the size of the sample.

Amplification of distinguisher's bias through voting. Let $b \in \{0, 1\}$ be a secret bit and let $\varepsilon \in (0; 1/2)$ be a fixed constant. Assume that we have an access to an oracle that outputs a correct guess for the secret bit with probability $\frac{1}{2} + \varepsilon$. The goal is to guess the value of the bit b with high probability. That can be done by querying the oracle k times and choosing the value that occurs at the majority of oracle's answers. The following lemma says that taking $k = O(\varepsilon^{-2})$ is enough to make the probability of successful guess be arbitrarily close to 1.

Lemma 4. *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables each having probability $\frac{1}{2} + \varepsilon$ of being equal to 1. Then*

$$\mathbb{P}\Big\{\sum_{i=1}^{k} X_i \geqslant \frac{k}{2}\Big\} \geqslant 1 - \frac{1}{2}(1 - 4\varepsilon^2)^{k/2}.$$

Proof of Lemma 4 can be found in [34].

Remark. Even though the distinguisher that we use is suboptimal, for the case of uniform and modular Gaussian distributions, it is quite close to the optimal distinguisher. In Section 3.2 we show that the distinguishing advantage of our symmetric one point distinguisher is asymptotically close to the distinguishing advantage of the optimal one (see Lemma 7).

3 Dual distinguishing attack against LWE.

In this section, we recall the distinguishing dual attack against LWE described in [19]. Let $\mathbf{s} \in \{0,1\}^n$ be a secret vector and let $\alpha > 0$ be a fixed constant. The attack takes as input m samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{T}^{n+1} \times \mathbb{T}$ which are either all from $\mathrm{LWE}_{\mathbf{s},\alpha}$ distribution or all from $\mathcal{U}(\mathbb{T}^n \times \mathbb{T})$, and guesses the input distribution.

We can write input samples in a matrix form:

$$\mathbf{A} := (\mathbf{a}_1, \ldots, \mathbf{a}_m) \in \mathbb{T}^{n \times m}, \quad \mathbf{b} = (b_1, \ldots, b_m)^t \in \mathbb{T}^m,$$

if input samples are from $\mathrm{LWE}_{\mathbf{s},\alpha}$ distribution:

$$\mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e} \mod 1.$$

In order to distinguish two distributions, the attack searches for a short vector $\mathbf{v} = (v_1, \ldots, v_m)^t \in \mathbb{Z}^m$ such that linear combination of the left parts of the inputs samples defined by $\mathbf{v}$

$$\mathbf{x} := \sum_{i=1}^{m} v_i \mathbf{a}_i = \mathbf{A}\mathbf{v} \mod 1.$$

is also a short vector. If the input was from LWE distribution, then the corresponding linear combination of the right parts of the input samples is also small as a sum of two relatively small numbers:

$$\mathbf{v}^t\mathbf{b} = \mathbf{v}^t(\mathbf{A}^t\mathbf{s} + \mathbf{e}) = \mathbf{x}^t\mathbf{s} + \mathbf{v}^t\mathbf{e} \mod 1. \tag{2}$$

On the other hand, if the input is uniformly distributed, then independently of the choice of the non zero vector $\mathbf{v}$, the product $\mathbf{v} \cdot \mathbf{b} \mod 1$ has uniform distribution on $(-1/2; 1/2)$. Recovering a suitable $\mathbf{v}$ turns decisional-LWE problem into an easier problem of distinguishing two distributions on $\mathbb{T}$.

This section is organized in the following way. First, in Section 3.1 we describe how a suitable vector $\mathbf{v}$ can be recovered by lattice reduction and analyze the distribution of $\mathbf{v}^t\mathbf{b}$. Then, in Section 3.2, we estimate the complexity of distinguishing two distributions on $\mathbb{T}$ that we obtain after the lattice reduction part. In Section 3.3, we show how the time complexity of the attack can be estimated.

3.1 Lattice reduction part

Finding a vector $\mathbf{v}$ such that both parts of the sum (2) are small when the input has LWE distribution is equivalent to finding a short vector in the following $(m+n)$-dimensional lattice:

$$\mathcal{L}(\mathbf{A}) = \left\{ \begin{pmatrix} \mathbf{A}\mathbf{v} \mod 1 \\ \mathbf{v} \end{pmatrix} \in \mathbb{R}^{m+n} \,\middle|\, \forall\, \mathbf{v} \in \mathbb{Z}^m \right\}.$$

The lattice $\mathcal{L}(\mathbf{A})$ can be generated by the columns of the following matrix:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_n & \mathbf{A} \\ \mathbf{0}^{m\times n} & \mathbf{I}_m \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$$

A short vector in $\mathcal{L}(\mathbf{A})$ can be found by applying a lattice reduction algorithm to basis $\mathbf{B}$. We assume that lattice reduction produces a vector $\mathbf{w} = (\mathbf{x}||\mathbf{v})^t \in \mathbb{Z}^{n+m}$ with equidistributed coordinates. The goal is to minimize the product $\mathbf{v}^t\mathbf{b} = \mathbf{x}^t\mathbf{s} + \mathbf{v}^t\mathbf{e}$. The vectors $\mathbf{e}$ and $\mathbf{s}$ come from different distributions and have different expected norms. For the TFHE scheme, the variance of $\mathbf{e}$ is much smaller than the variance of $\mathbf{s}$. To take it into the account, the attack introduces an additional rescaling parameter $q \in \mathbb{R}_{>0}$. The first n rows of the matrix $\mathbf{B}$ are multiplied by q, the last m rows are multiplied by $q^{-n/m}$. This transformation doesn't change the determinant of the matrix. A basis $\mathbf{B}_q$ of the transformed lattice is given by

$$\mathbf{B}_q = \begin{pmatrix} q\mathbf{I}_n & q\mathbf{A} \\ \mathbf{0}^{m\times n} & q^{-n/m}\mathbf{I}_m \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}.$$

We apply a lattice reduction algorithm to $\mathbf{B}_q$. Denote the first vector of the reduced basis as $\mathbf{w}_q$. By taking last m coordinates of $\mathbf{w}_q$ and multiplying them by $q^{n/m}$ we recover the desired vector $\mathbf{v}$. That part of the attack is summarized in Algorithm 1.

Algorithm 1: Transform m LWE samples to one sample from modular Gaussian distribution

input : $\mathbf{A} \in \mathbb{T}^{n\times m}$, $\mathbf{b} \in \mathbb{T}^m$, $S > 0$, $\alpha > 0$, $\delta \in (1; 1.1)$
output: $x \in \mathbb{T}$

`computeV(`$\mathbf{A}$, S, α, δ`):`
- $q := \left(\frac{S}{\alpha}\right)^{\frac{m}{n+m}}$
- $\mathbf{B}_q := \begin{pmatrix} q\mathbf{I}_n & q\mathbf{A} \\ \mathbf{0}^{m\times n} & q^{-n/m}\mathbf{I}_m \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$
- $\mathbf{w}_q \leftarrow \mathrm{BKZ}_\delta(\mathbf{B}_q)$
- $\mathbf{v} := q^{n/m} \cdot (w_{q_{n+1}}, \dots w_{q_{n+m}})^t$
- **return** $(\mathbf{v})$

`LWEtoModGaussian(`$\mathbf{A}$, $\mathbf{b}$, S, α, δ`):`
- $\mathbf{v} \leftarrow \textsc{computeV}(\mathbf{A}, S, \alpha, \delta)$
- **return** $\mathbf{v}^t\mathbf{b} \mod 1$

The following lemma describes the distribution of the output of Algorithm 1 under Assumption 1 that BKZ outputs vectors with balanced coordinates.

Lemma 5 (see [19, Section 7]). *Let $\alpha > 0$ and $S \in (0; 1)$ be fixed constants, $n \in \mathbb{Z}_{>0}$. Let $\mathbf{s} \in \{0, 1\}^n$ be a binary vector such that all bits of $\mathbf{s}$ are sampled independently from Bernoulli distribution with parameter S^2: for all $i \in \{1, \dots, n\}$:*

$\mathbb{P}\{s_i = 1\} = S^2$, $\mathbb{P}\{s_i = 0\} = 1 - S^2$. *Suppose that Assumption 1 holds and let* $\delta > 0$ *be the quality of the output of BKZ algorithm. Then, given as input* $m = \sqrt{n \cdot \frac{\ln(S/\alpha)}{\ln(\delta)}} - n$ *samples from* $LWE_{\mathbf{s},\alpha}$ *distribution, Algorithm 1 outputs a random variable* x *with distribution that can be approximated by a Gaussian distribution with mean 0 and standard deviation* σ

$$\sigma = \alpha \cdot \exp\Big(2\sqrt{n \ln(S/\alpha) \ln(\delta)}\Big).$$

Denote as F_x *the cumulative distribution function of* x *and denote as* Φ_σ *the cumulative distribution function of the Gaussian distribution with mean 0 and standard deviation* σ*. Then, the distance between the two distributions can be bounded:*

$$\sup_{t \in \mathbb{R}} |F_x(t) - \Phi_\sigma(t)| = O\Big(\frac{1}{\sqrt{S^2(m+n)}}\Big),$$

as $n \to \infty$*.*

Lemma 5 can be proved using the Berry-Esseen theorem. We give a proof in Appendix A for completeness.

3.2 Distinguishing modular Gaussian and uniform distributions

In this section, we are estimating the complexity of distinguishing uniform and modular Gaussian distributions. We assume that we are given as input N points that are sampled independently all from the same distribution, and the goal is to guess the underlying distribution. We use the following procedure, first, we construct a distinguisher that, given as input one point from one of the two distributions, outputs a correct guess of the distribution with probability $p > \frac{1}{2}$, then we run the distinguisher many times and output the result given at the majority of trials.

One point distinguisher We start by constructing a single point distinguisher for uniform and modular Gaussian distributions. We require that the probability of error of the distinguisher is the same independently of the underlying distribution of the input point.

Lemma 6. *Let* $\sigma > 0$ *be a fixed constant and let* $t \in (0; 0.5)$ *be the solution of the following equation*

$$\int_0^t g_\sigma(x)dx = \frac{1}{2} - t. \tag{3}$$

Assume that Algorithm 2 is given as input one point which has uniform distribution $\mathcal{U}$ *or modular Gaussian distribution* $\mathcal{G}_\sigma$*. Then, Algorithm 2 guesses the distribution correctly with probability* $1 - 2t$ *independently of the distribution of the input. The time complexity of the algorithm is linear in the size of the input.*

Algorithm 2: Distinguish $\mathcal{U}$ and $\mathcal{G}_\sigma$

input : $x \in (-\frac{1}{2}; \frac{1}{2})$, $t \in (0; \frac{1}{2})$
output: G or U — guess for the distribution of the input.

```
onePointDistinguisher(x, t):
    if (|x| ⩽ t) then
        return G
    else
        return U
```

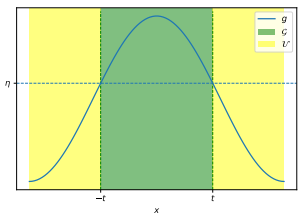


Fig. 1: Distinguish $\mathcal{U}$ and $\mathcal{G}_\sigma$

Fig. 2: Distinguisher D_η for the uniform and the modular Gaussian distributions. In Figure 1, the blue line denotes the density of the modular Gaussian distribution, the dashed blue line denotes the threshold η of the distinguisher. The green region corresponds to the values of x for which D_η decides that the distribution of the input is the modular Gaussian, the yellow region corresponds to the values of x for which D_η decides that the distribution is uniform.

Proof. Let x be an input of Algorithm 2. The ratio of density functions for modular Gaussian and uniform distributions is $\frac{g_\sigma(x)}{1} = g_\sigma(x)$, so the likelihood-ratio distinguisher in that case just checks whether $g_\sigma(x)$ is bigger then some threshold η. Because of the shape of density function g_σ, it is equivalent to choosing $\mathcal{G}_\sigma$ when $|x| \leqslant t = g_\sigma^{-1}(\eta)$ and choosing $\mathcal{U}$ otherwise (see Figure 1). The value t is chosen to obtain an equal probability of error independently of the distribution of the input, that is

$$\mathbb{P}\{|x| > t \,|\, x \sim \mathcal{G}_\sigma\} = \mathbb{P}\{|x| \leqslant t \,|\, x \sim \mathcal{U}\}.$$

This is equivalent to finding t that satisfies Equation (3). The left part of Equation (3) is an increasing function on the interval $(0; 1/2)$ that has range $(0; 1/2)$. Since the right part of Equation (3) is a decreasing linear function on $(0; 1/2)$ with the same range as the right part, Equation (3) has a unique solution on the interval $(0; 1/2)$. It can be found numerically using the bisection method (see [7, Chapter 2.1]).

The probability that the distinguisher outputs a wrong guess is

$$\mathbb{P}\{|x| \leqslant t \,|\, x \sim \mathcal{U}\} = 2t.$$

The threshold t for the distinguisher described in Algorithm 2 can be computed numerically by solving Equation (3). For large values of the parameter σ of the modular Gaussian distribution, the computation of t can be simplified by replacing the density of the modular Gaussian distribution by its approximation.

Lemma 7. *Let $\sigma > 0$ be the parameter of modular Gaussian distribution and let $t(\sigma)$ be the value of the threshold for the distinguisher described in Algorithm 2.*

Then, as $\sigma \to \infty$,

$$t(\sigma) = \frac{1}{4} - \frac{e^{-2\pi^2\sigma^2}}{2\pi} + O(e^{-6\pi^2\sigma^2}).$$

Proof. By Lemma 6, the optimal value of t is the solution of Equation (3) which belongs to an interval $(0; 0.5)$. For big values of σ, the integral of the density of a modular Gaussian distribution $\mathcal{G}_\sigma$ can be approximated using Lemma 1:

$$\begin{aligned}\int_0^t g_\sigma(x)dx &= \int_0^t \left(1 + 2e^{-2\pi^2\sigma^2\cos(2\pi x)} + O(e^{-8\pi^2\sigma^2})\right)dx \\ &= t + \frac{1}{\pi}\cdot e^{-2\pi^2\sigma^2}\sin(2\pi t) + t\cdot O(e^{-8\pi^2\sigma^2}).\end{aligned}$$

After applying this approximation to Equation (3) we obtain:

$$t = \frac{1}{4} - \frac{1}{2\pi}\cdot e^{-2\pi^2\sigma^2}\sin(2\pi t) + O(e^{-8\pi^2\sigma^2}). \tag{4}$$

From this equation it follows that $t = \frac{1}{4} + O(e^{-2\pi^2\sigma^2})$. Then $\sin(2\pi t)$ can also be approximated:

$$\sin(2\pi t) = \sin\left(\frac{\pi}{2} + O(e^{-2\pi^2\sigma^2})\right) = \cos(O(e^{-2\pi^2\sigma^2})) = 1 - O(e^{-4\pi^2\sigma^2}).$$

By combining the approximation for $\sin(2\pi t)$ and Equation (4) we obtain an approximation for the optimal value of t.

From Lemma 7 it follows that the distinguishing advantage of the symmetric distinguisher is $O(e^{-2\pi^2\sigma^2})$ when $\sigma \to \infty$, which, ignoring constants, coincides with the advantage of the distinguisher used in [19] (see [19, Section 7, Equation (6)]).

By querying the one point distinguisher described in Algorithm 2 many times we can amplify the probability of successful guess of the distribution.

The complexity of distinguishing uniform and modular Gaussian distributions is summarized in the following lemma.

Lemma 8. *Let* $p \in (0; 1)$ *be a fixed constant and let* $\sigma > 0$ *be a parameter of the modular Gaussian distribution* $\mathcal{G}_\sigma$*. Let* $\varepsilon = \frac{1}{2} - 2t$*, where* $t \in (0; 0.5)$ *is the solution of Equation (3) and let* N *be an integer number greater or equal than* $-\frac{\ln(2(1-p))}{2}\cdot\frac{1}{\varepsilon^2}$*. Then, Algorithm 3, given a sample* $X = (x_1, \ldots, x_N)$ *either from distribution* $\mathcal{U}$ *or from* $\mathcal{G}_\sigma$*, guesses the distribution correctly with probability at least* p *in time* $O(N)$*.*

Proof. Algorithm 2 outputs the correct answer if one point distinguisher chooses the correct distribution at least at $\frac{N}{2}$ trials out of N. By Lemma 6, one point distinguisher guesses the distribution correctly with probability $1 - 2t = \frac{1}{2} + \varepsilon$.

Algorithm 3: Distinguish uniform and modular Gaussian distributions

input : $X = (x_1, \ldots, x_N) \in (-1/2; 1/2)^N$, $\sigma > 0$
output: G or U — guess for the distribution of the input.

```
DistinguishGU(X, σ):
    t := solve(t + ∫_0^t g_σ(x)dx = 1/2), S := 0
    for x ∈ X do
        if onePointDistinguisher(x, t) = G then
            S := S + 1
    if (S ⩾ N/2) then
        return G
    else
        return U
```

Then, by Lemma 4, the probability that one point distinguisher gives the right answer at least $\frac{N}{2}$ times is at least

$$1 - \frac{1}{2}(1 - 4\varepsilon^2)^{N/2} = 1 - \frac{1}{2}(1 - 4\varepsilon^2)^{\frac{1}{4\varepsilon^2}\cdot\left(-\ln(2(1-p))\right)} \xrightarrow[\varepsilon\to 0]{} p.$$

The algorithm performs N queries to the one point distinguisher. The time complexity of the one point distinguisher is linear in size of the input, so the time complexity of Algorithm 3 is $O(N)$.

3.3 Complexity of the attack

The distinguishing attack is summarized in Algorithm 4. It takes as input $m \times N$ samples from an unknown distribution, then transforms them into N samples which have uniform distribution if the input of the attack was uniform and into modular Gaussian distribution if the input was from LWE distribution. Then attack guesses the distribution of obtained N samples using Algorithm 3 and outputs the corresponding answer.

The following theorem states that the cost of the distinguishing attack can be estimated by solving an optimization problem.

Theorem 2 (see [19, Section 7]). *Let $\alpha > 0$ and $S \in (0; 1)$ be some fixed constants, $n \in \mathbb{Z}_{>0}$. Let $\mathbf{s} \in \{0, 1\}^n$ be a binary vector such that all bits of $\mathbf{s}$ are sampled independently from a Bernoulli distribution with parameter S^2. For any $\sigma > 0$ let $\varepsilon(\sigma) = \frac{1}{2} - 2t$, where $t \in (0; 0.5)$ is the solution of Equation (3). Suppose that Assumption 1 holds. Then the time complexity of solving decision-LWE$_{\mathbf{s},\alpha}$ with probability of success p by distinguishing attack described in Algorithm 4 is*

$$T_{TFHEattack} = \min_{\delta}\Big(-\frac{\ln(2(1-p))}{2}\cdot\frac{1}{\varepsilon(\sigma)^2}\cdot T(BKZ_\delta)\Big), \tag{5}$$

where $\sigma = \alpha \cdot \exp\left(2\sqrt{n\ln(S/\alpha)\ln(\delta)}\right)$.

Algorithm 4: Dual distinguishing attack (adapted from [19, Section 7])

input : $\{(\mathbf{A}_i, \mathbf{b}_i)\}_{i=1}^{N}$, where $\forall i\ \mathbf{A}_i \in \mathbb{T}^{n\times m}$, $\mathbf{b}_i \in \mathbb{T}^m$, $\alpha > 0$, $S > 0$, $\delta \in (1; 1.1)$
output: guess for the distribution of the input: Uniform or LWE distribution

```
DistinguishingAttack({A_i, b_i}_{i=0}^{N}, α, S, δ):
    X := ∅
    σ := α · exp(2·sqrt(n ln(S/α) ln(δ)))
    for i ∈ {1, ..., N} do
        x ← LWEtoModGaussian(A_i, b_i, S, α, δ)
        X ← X ∪ x
    if (DistinguishGU(X, σ) = G) then
        return LWE distribution
    else
        return Uniform
```

Proof. The cost of the attack is the cost of lattice reduction multiplied by the number of samples N needed to distinguish uniform and modular Gaussian distributions with parameter σ:

$$T = N \cdot T(\mathrm{BKZ}_\delta). \tag{6}$$

By Lemma 8, the required number of samples $N = -\frac{\ln(2(1-p))}{2} \cdot \frac{1}{\varepsilon^2}$. The parameter of the discrete Gaussian distribution as a function of δ can be estimated using Lemma 5. Then, the time complexity can be obtained by optimizing expression (6) as a function of δ.

4 Hybrid key recovery attack

In this section, we show how the dual distinguishing attack recalled in Section 3 can be hybridized with exhaustive search on a fraction of the secret vector to obtain a continuum of more efficient key recovery attacks on the underlying LWE problem. Let $\mathbf{s} \in \{0,1\}^n$ be a secret vector and let $\alpha > 0$ be a fixed constant. Our attack takes as input samples from LWE distribution of form

$$(\mathbf{A}, \mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e} \mod 1) \in (\mathbb{T}^{n\times m}, \mathbb{T}^m), \tag{7}$$

where $\mathbf{e} \in \mathbb{R}^m$ has centered Gaussian distribution with standard deviation α. The attack divides the secret vector into two parts:

$$\mathbf{s} = (\mathbf{s}_1 || \mathbf{s}_2)^t, \quad \mathbf{s}_1 \in \{0,1\}^{n_1}, \quad \mathbf{s}_2 \in \{0,1\}^{n_2}, \quad n = n_1 + n_2.$$

The matrix $\mathbf{A}$ is also separated into two parts correspondingly to the separation of the secret $\mathbf{s}$:

$$\mathbf{A} = \left(\begin{array}{ccc} a_{1,1} & \dots & a_{1,m} \\ \vdots & & \vdots \\ a_{n_1,1} & \dots & a_{n_1,m} \\ \hline a_{n_1+1,1} & \dots & a_{n_1+1,m} \\ \vdots & \dots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{array}\right) = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \tag{8}$$

Then, Equation (7) can be rewritten as

$$\mathbf{A}_1^t \mathbf{s}_1 + \mathbf{A}_2^t \mathbf{s}_2 + \mathbf{e} = \mathbf{b} \mod 1.$$

By applying lattice reduction to matrix $\mathbf{A}_1$ as described in Algorithm 1, we recover a vector $\mathbf{v}$ such that $\mathbf{v}^t(\mathbf{A}_1^t \mathbf{s}_1 + \mathbf{e})$ is small and it allows us to transforms m input LWE samples $(\mathbf{A}, \mathbf{b}) \in (\mathbb{T}^{n \times m}, \mathbb{T}^m)$ into one new LWE sample $(\widehat{\mathbf{a}}, \widehat{b}) \in (\mathbb{T}^{n_2}, \mathbb{T})$ of smaller dimension and bigger noise:

$$\underbrace{\mathbf{v}^t \mathbf{A}_2^t}_{\widehat{\mathbf{a}}} \mathbf{s}_2 + \underbrace{\mathbf{v}^t(\mathbf{A}_1^t \mathbf{s}_1 + \mathbf{e})}_{\widehat{e}} = \underbrace{\mathbf{v}^t \mathbf{b}}_{\widehat{b}} \mod 1. \tag{9}$$

The resulting LWE sample in smaller dimension can be used to find $\mathbf{s}_2$. Let $\mathbf{x} \in \{0,1\}_2^n$ be a guess for $\mathbf{s}_2$. If the guess is correct, then the difference

$$\widehat{b} - \widehat{\mathbf{a}}^t \mathbf{x} = \widehat{b} - \widehat{\mathbf{a}}^t \mathbf{s}_2 = (\widehat{e} \mod 1) \sim \mathcal{G}_\sigma \tag{10}$$

is small.

If the guess is not correct and $\mathbf{x} \neq \mathbf{s}_2$, then there exist some $\mathbf{y} \neq \mathbf{0}$ such that $\mathbf{x} = \mathbf{s}_2 - \mathbf{y}$. Then, we rewrite $\widehat{b} - \widehat{\mathbf{a}}^t \mathbf{x}$ in the following way:

$$\widehat{b} - \widehat{\mathbf{a}}^t \mathbf{x} = (\widehat{b} - \widehat{\mathbf{a}}^t \mathbf{s}_2) + \widehat{\mathbf{a}}^t \mathbf{y} = \widehat{\mathbf{a}}^t \mathbf{y} + \widehat{e}.$$

We can consider $(\widehat{\mathbf{a}}, \widehat{\mathbf{a}}^t \mathbf{y} + \widehat{e})$ as a sample from the LWE distribution that corresponds to the secret $\mathbf{y}$. Therefore, we may assume that if $\mathbf{x} \neq \mathbf{s}_2$, the distribution of $\widehat{b} - \widehat{\mathbf{a}}^t \mathbf{x} \mod 1$ is close to uniform, unless the decision-LWE is easy to solve.

In order to recover $\mathbf{s}_2$, the attack generates many LWE samples with reduced dimension. Denote the number of generated samples as R and the generated samples in a matrix form as $(\widehat{\mathbf{A}}, \widehat{\mathbf{b}}) \in \mathbb{T}^{R \times d} \times \mathbb{T}^R$. There are 2^{n_2} possible candidates for $\mathbf{s}_2$. For each candidate $\mathbf{x} \in \{0,1\}^{n_2}$, the attack computes an R-dimensional vector $\mathbf{e}_\mathbf{x} = \mathbf{b} - \mathbf{A}^t \mathbf{s}$. The complexity of this computation for all the candidates is essentially the complexity of multiplication of matrices $\widehat{\mathbf{A}}$ and $\mathbf{S}_2$, where $\mathbf{S}_2$ is a matrix whose columns are all binary vectors of dimension n_2. Naively the matrix multiplication requires performing $O(n \cdot 2^{n_2} \cdot R)$ operations, but, by exploiting the recursive structure of $\mathbf{S}_2$, we can do it faster in time $O(R \cdot 2^{n_2})$.

Then, for each candidate $\mathbf{x}$ for $\mathbf{s}_2$ the attack guesses whether the corresponding vector $\mathbf{e}_\mathbf{x}$ is uniform or concentrated around zero distribution. The attack

returns the only candidate $\mathbf{x}$ whose corresponding vector $\mathbf{e_x}$ has concentrated around zero distribution.

The rest of this section is organized as follows. First, we describe the auxiliary algorithm for multiplying any matrix by the matrix of all binary vectors that let us speed up the search for the part of the secret key, then, we evaluate the complexity of our attack.

4.1 Algorithm for computing a product of any matrix with a matrix of all binary vectors.

For any $d \in \mathbb{Z}_{>0}$, define a function $\mathbf{bin}_d : \mathbb{Z} \cap [0; 2^d] \to \{0,1\}^d$ such that for any positive integer $k \leqslant 2^d$, $\mathbf{bin}_d(k)$ is a d-dimensional binary vector, such that for all $i \in \{1, \ldots, d\}$ the i-th coordinate of $\mathrm{bin}_d(k)$ is equal to the i-th bit of the binary representation of k.

For any positive integer d denote as $\mathbf{S}_{(d)}$ a matrix of all binary vectors of dimension d written in the lexicographical order, that is, the i-th column of $\mathbf{S}_{(d)}$ is equal to $\mathbf{bin}_d(i)$. These matrices can be constructed recursively. For $d = 1$ it is given by $\mathbf{S}_{(1)} = \begin{pmatrix} 0 & 1 \end{pmatrix}$, and for any $d > 1$ the matrix $\mathbf{S}_{(d)}$ can be constructed by concatenating the two matrices $\mathbf{S}_{(d-1)}$ and adding a row which consists of 2^{d-1} zeros followed by 2^{d-1} ones as the first row to the resulting matrix:

$$\mathbf{S}_{(d)} = \begin{pmatrix} 0 \ldots 0 & 1 \ldots 1 \\ \mathbf{S}_{(d-1)} & \mathbf{S}_{(d-1)} \end{pmatrix}. \tag{11}$$

Let $\mathbf{a} = (a_1, \ldots, a_d)^t$ be a d-dimensional vector. Our goal is to compute the scalar products of $\mathbf{a}$ with each column of $\mathbf{S}_{(d)}$. We can do it by using the recursive structure of $\mathbf{S}_{(d)}$. Assume that we know the desired scalar products for $\mathbf{a}_{(d-1)} = (a_2, \ldots, a_d)^t$ and $\mathbf{S}_{(d-1)}$ Then, using Equation (11), we get

$$\mathbf{a}^t \mathbf{S}_{(d)} = \begin{pmatrix} a_1 & \mathbf{a}_{(d-1)}^t \end{pmatrix} \cdot \begin{pmatrix} 0 \ldots 0 & 1 \ldots 1 \\ \mathbf{S}_{(d-1)} & \mathbf{S}_{(d-1)} \end{pmatrix} = \begin{pmatrix} \mathbf{a}_{(d-1)}^t \mathbf{S}_{(d-1)} \\ \begin{pmatrix} a_1 \ldots a_1 \end{pmatrix}^t + \mathbf{a}_{(d-1)}^t \mathbf{S}_{(d-1)} \end{pmatrix}, \tag{12}$$

that is, the resulting vector is the sum of the vector $\mathbf{a}_{(d-1)}^t \mathbf{S}_{(d-1)}$ concatenated with itself with the vector whose first 2^{d-1} coordinates are zeros and the last 2^{d-1} coordinates are all equal to a_1. The approach is summarized in Algorithm 5.

Lemma 9. *Let d be a positive integer number. Algorithm 5, given as input a d-dimensional vector $\mathbf{a}$, outputs the vector $\mathbf{x}$ of dimension 2^d such that for all $i \in \{1, \ldots, 2^d\}$ $x_i = \mathbf{a}^t \boldsymbol{bin}_d(i)$. The time complexity of the algorithm is $O(2^d)$.*

Proof. The correctness of the algorithm follows from the recursive structure of the matrix $\mathbf{S}_{(d)}$ (see Equations (11) and (12)). The algorithm performs only additions of some coordinates of the vector $\mathbf{a}$. At the i-th iteration of the cycle (3-8) the algorithm performs 2^{d-i} additions. Number of iterations is $(d-1)$. The overall number of additions is $2 + 2^2 + \cdots + 2^{d-1} = 2^d - 2$.

Algorithm 5: Compute a scalar product of a vector with all binary vectors

input : $\mathbf{a} = (a_1, \ldots, a_d)^t$

output: $\mathbf{a}^t \mathbf{S}_{(d)}$, where $\mathbf{S}_{(d)} \in \{0,1\}^{2^d \times d}$ is the matrix whose columns are all binary vectors of dimension d written in the lexicographical order

```
computeScalarProductWithBinaryVectors(a):
    x ← (0, a_d)^t
    for i ∈ {d − 1, ..., 1} do
        y ← x
        for j ∈ {1, ..., 2^{d−i}} do
            y_j ← y_j + a_i
        x' ← x ∪ y
        x ← x'
    return x
```

Corollary 1. *Let* $\mathbf{A}$ *be a matrix with* R *rows and* d *columns. The product of* $\mathbf{A}$ *and* $\mathbf{S}_{(d)}$ *can be computed in time* $O(R \cdot 2^d)$.

Proof. In order to compute $\mathbf{A} \cdot \mathbf{S}_{(d)}$ we need to compute the product of each of the R rows of A with $\mathbf{S}_d$. By Lemma 9 it can be done in $O(2^d)$ time. Then the overall complexity of multiplying the matrices is $O(R \cdot 2^d)$.

4.2 Complexity of the attack

The pseudo-code corresponding to the full attack is given in Algorithm 6.

Theorem 3. *Let* $\alpha > 0$, $p \in (0;1)$, $S \in (0;1)$, *and* $n \in \mathbb{Z}_{>0}$ *be fixed constants. Let* $\mathbf{s} \in \{0,1\}^n$, $\sigma > 0$, *and* $\varepsilon : \mathbb{R}_{>0} \to (0;1/2)$ *be as defined in Theorem 2. Suppose that Assumption 1 holds. Then, the time complexity of solving the search-LWE*$_{\mathbf{s},\alpha}$ *problem with probability of success* p *by the attack described in Algorithm 6 is*

$$T_{attack} = \min_{\delta, n_2} \left(\frac{1}{\varepsilon(\sigma)^2} \cdot \left(2^{n_2} + T(BKZ_\delta)\right) \left((n_2 - 1)\ln(2) - \ln(\ln(p^{-1}))\right) \right). \quad (13)$$

Proof. The attack can be divided into two steps: lattice reduction step and search for the fraction of the secret key. At the first step attack takes $R \times m$ $\mathrm{LWE}_{\mathbf{s},\alpha}$ samples and transforms them into R $\mathrm{LWE}_{\mathbf{s}_2,\sigma}$ samples such that $\mathbf{s}_2$ is a part of the secret key $\mathbf{s}$ and the noise parameter σ is bigger than the noise parameter α of the input. It requires $R \cdot T(\mathrm{BKZ}_\delta)$ time. Denote the matrix form of obtained LWE samples as $(\widehat{\mathbf{A}}, \widehat{\mathbf{b}}) \in (\mathbb{T}^{n_2 \times R}, \mathbb{T}^R)$.

At the search step, the goal is to recover $\mathbf{s}_2$ using the obtained LWE samples. For each of the candidates for $\mathbf{s}_2$ the attack computes the error vector that

Algorithm 6: Hybrid key recovery attack

input : $\{(\mathbf{A}_i, \mathbf{b}_i)\}_{i=1}^{R}$, where $\forall i\ \mathbf{A}_i \in \mathbb{T}^{n\times m}$, $\mathbf{b}_i \in \mathbb{T}^m$, $\alpha > 0$, $S > 0$, $\delta > 1$, $n_1 \in \{2, \ldots, n-1\}$

output: $\mathbf{s}_2 \in \{0,1\}^{n-n_1}$

recoverS$(\{(\mathbf{A}_i, \mathbf{b}_i)\}_{i=1}^{R}, \alpha, S, \delta, n_1)$**:**

- $n_2 \leftarrow (n - n_1)$
- $\sigma \leftarrow \alpha \cdot \exp\left(2\sqrt{n_1 \ln(S/\alpha)\ln(\delta)}\right)$
- $\widehat{\mathbf{A}} \leftarrow \emptyset$, $\widehat{\mathbf{b}} \leftarrow \emptyset$
- /* lattice reduction part */
- **for** $i \in \{1, \ldots, R\}$ **do**
  - $\mathbf{A} \leftarrow \mathbf{A}_i$, $\mathbf{b} \leftarrow \mathbf{b}_i$
  - $(\mathbf{A}_1, \mathbf{A}_2) \leftarrow$ splitMatrix$(\mathbf{A}, n_1)$ ▷ see Equation (8)
  - $\mathbf{v} \leftarrow$ computeV$(\mathbf{A}_1, S, \alpha, \delta)$ ▷ Algorithm 1
  - $\widehat{\mathbf{A}} \leftarrow \widehat{\mathbf{A}} \cup \{\mathbf{A}_2\mathbf{v}\}$, $\widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} \cup \{\mathbf{v}^t\mathbf{b}\}$
- /* search for $\mathbf{s}_2$ */
- $\mathbf{S}_{(n_2)} \leftarrow$ matrix of all binary vectors of dimension n_2 in lexicographical order
- $\widehat{\mathbf{B}} \leftarrow (\widehat{\mathbf{b}}, \ldots, \widehat{\mathbf{b}}) \in \mathbb{T}^{R\times 2^{n_2}}$
- $\widehat{\mathbf{E}} \leftarrow \widehat{\mathbf{B}} - \widehat{\mathbf{A}}^t\mathbf{S}_{(n_2)} \mod 1$ ▷ see Corollary 1 and Algorithm 5
- **for** $i \in \{1, \ldots, 2^{n_2}\}$ **do**
  - $\widehat{\mathbf{e}} \leftarrow \widehat{\mathbf{E}}[i]$
  - /* guess the distribution of e (see Algorithm 3) */
  - **if** (distinguishGU$(\widehat{\mathbf{e}}, \sigma) = \mathcal{G}$) **then**
    - **return** $\mathbf{S}_{(n_2)}[i]$

corresponds to R LWE samples obtained at the previous step. It is equivalent to computing the following matrix expression:

$$\widehat{\mathbf{E}} = \widehat{\mathbf{B}} - \widehat{\mathbf{A}}^t\mathbf{S}_{(n_2)} \mod 1,$$

where $\mathbf{S}_{(n_2)}$ is a matrix composed of all binary vectors of length n_2 written in lexicographical order and $\widehat{\mathbf{B}} \in \mathbb{T}^{R\times 2^{n_2}}$ is a matrix composed of vector $\widehat{\mathbf{b}}$ repeated 2^{n_2} times. The complexity of computing that expression is dominated by the complexity of computing the product of $\widehat{\mathbf{A}}^t \in \mathbb{T}^{R\times 2^{n_2}}$ and $\mathbf{S}_{(n_2)}$. By Corollary 1, it can be computed in $O(R \cdot 2^{n_2})$ operations. Once the attack obtain an error vector for each of the candidates, it guesses the distribution of each error vector using Algorithm 3 and returns the candidate whose error vector has concentrated around zero modular Gaussian distribution.

The time complexity of the attack is the sum of complexities of the two steps:

$$T_{\text{attack}} = R \cdot \left(2^{n_2} + T(\text{BKZ}_\delta)\right). \tag{14}$$

Now the goal is to evaluate the number of samples R needed to recover $\mathbf{s}_2$ with probability p. First, we compute the probability of the correct guess for one candidate needed to achieve the probability p of the correct guess for all candidates. Assume that the probability of the correct guess for one candidate

is $1-\mu$ for some $\mu \in (0;1)$ Then, the probability that the attack successfully guesses $\mathbf{s}_2$ is at least $(1-\mu)^{2^{n_2}}$. Let's take $\mu = \frac{c}{2^{n_2}}$ for some $c > 0$. Then

$$\lim_{n_2 \to \infty} (1-\mu)^{2^{n_2}} = e^{-c}$$

and to achieve success probability p we should take $\mu = \frac{-\ln(p)}{2^{n_2}}$.

Then, we estimate the sample size needed to achieve the probability $(1-\mu)$ for a single candidate using Lemma 8. In order to achieve this probability of success for one candidate, the size of the sample should be at least

$$R = -\frac{\ln(2\mu)}{2\varepsilon(\sigma)^2} = \frac{\ln(2)(n_2-1) - \ln(-\ln(p))}{2\varepsilon(\sigma)^2}, \tag{15}$$

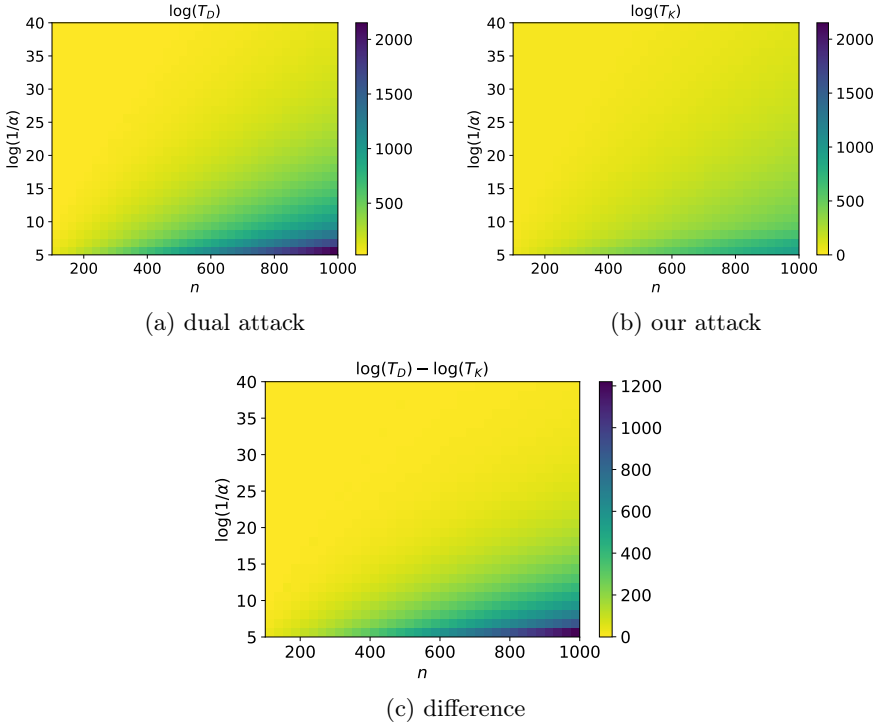
where ε is the same as described in Lemma 8. By combining Equations (14) and (15) we obtain the time complexity of the attack.

5 Bit-security estimation and experimental verification

We implement a python script that, given parameters of an LWE problem and a BKZ cost model as an input, finds optimal parameters for the dual attack (see Section 3) and for our attack (see Section 4). Using this script we evaluate the computational cost of the dual attack and our attacks for a wide range of LWE parameters and in particular for the parameters used in the TFHE scheme. In this section, we report the results of our numerical estimation and show that the security level of the TFHE scheme should be updated with regard to the hybrid attack. We support our argument by an implementation working on a small example.

5.1 Bit-security of LWE parameters

We numerically estimate the cost of solving LWE problem by the dual attack and by our attack for all pairs of parameters (n, α) from the following set: $(n, -\log(\alpha)) \in \{100, 125, \ldots, 975\} \times \{5, 6.25, \ldots, 38.5\}$. In all cases, we take $S^2 = 1/2$, which corresponds to choosing the secret key uniformly random from $\{0,1\}^n$ as it is done in the TFHE scheme. For each attack, we consider three BKZ cost models. For each case, we create a heatmap representing the cost of the attack as a function of parameters n and α. The results obtained by using the enumeration BKZ cost model are presented in Figure 3. The left heatmap in Figure 3 represents the logarithm of the time complexity of the dual attack, the right heatmap represents the logarithm of the time complexity of our attack. Figure 3 shows that for the same sets of parameters the cost of our attack is always less then or equal to the cost of the dual distinguishing attack and that the difference between the costs of the attacks grows with the hardness of the problem. We obtain similar pictures for the two other considered models. For completeness, we present the heatmaps for the other models in Appendix B.



(a) dual attack

(b) our attack

(c) difference

Fig. 3: Comparison of costs of the attacks under the enumeration BKZ cost model. n and α denote the dimension and the standard deviation of the noise of LWE samples, T_D denotes the time complexity of the dual distinguishing attack, T_K denotes the time complexity of our key recovery attack.

5.2 Application to TFHE scheme

The TFHE scheme uses the following two sets of parameters [19]: $n = 500, \alpha = 2.43 \cdot 10^{-5}$ for switching key and $n = 1024, \alpha = 3.73 \cdot 10^{-9}$ for bootstrapping key. The security of the scheme is defined by the security of the switching key, which is the weaker part. In Table 2 we present estimates of bit-security of switching and bootstrapping keys according to both attacks under three different models of the cost of BKZ. We also provide optimal parameters for the attacks.

In all cases, the cost of our attack is lower than the cost of the dual attack. In addition, the lattice reduction part is always easier for our attack than for the dual attack, because the required quality parameter of lattice reduction δ is always bigger for our attack than for the dual attack. However, the difference of the costs depends on the choice of the model: it is bigger for models that predict higher complexity of BKZ. For example, for parameters of the switching key, the

Table 2: Security of TFHE scheme. λ denotes security in bits, δ and n_1 are optimal parameters for the attacks. "-" means that the distinguishing attack doesn't have parameter n_1.

BKZ model	**switching key**				**bootstrapping key**			
	attack	λ	δ	n_1	attack	λ	δ	n_1
delta-squared	dual	169	1.0052	-	dual	204	1.0046	-
	our attack	119	1.0059	406	our attack	159	1.0051	889
sieving	dual	135	1.0047	-	dual	144	1.0043	-
	our attack	114	1.0056	405	our attack	132	1.0048	810
enumeration	dual	195	1.0051	-	dual	230	1.0045	-
	our attack	136	1.0062	389	our attack	179	1.0052	868

difference under the sieving model is 11 bits while under enumeration model it is 59 bits.

In Figure 8 we present estimation of bit-security of LWE parameters revisited according to the combination of our attack and collision attack of time complexity $2^{n/2}$. That is, Figure 8 presents the function $\min(T_{\text{ourAttack}}(n,\alpha), 2^{n/2})$, where $T_{\text{ourAttack}}(n,\alpha)$ is the cost of our attack for parameters n and α. Figure 8 is obtained under the enumeration BKZ cost model. See Appendix B for other models.

5.3 Experimental verification

In order to verify the correctness of our attack, we have implemented it on small examples. Our implementation recovers 5 bits of a secret key for LWE problems with the following two sets of parameters: $(n,\alpha) = (30, 2^{-8})$ and $(n,\alpha) = (50, 2^{-8})$.

For implementation purposes, we rescaled all the elements defined over torus $\mathbb{T}$ to integers modulo 2^{32}. For both examples, we use BKZ with blocksize 20, which yields the quality of the lattice reduction around $\delta \lesssim 1.013$. We compute the values of parameters of the attack required to guess correctly 5 bits of the key with probability 0.99 assuming that quality of the output of BKZ. The required parameters for both experiments are summarized in Table 3.

The first experiment was repeated 20 times, the second – 10 times. For both experiments, the last five bits of the key were successfully recovered at all attempts.

The correctness of both attacks rely on assumptions made in Lemma 5 for approximating the distribution of $\mathbf{v}^t(\mathbf{A}^t\mathbf{s}+\mathbf{e}) \mod 1$ by modular Gaussian distribution $\mathcal{G}_\sigma$. In order to verify these assumptions, while running both experiments we have collected samples to check the distribution: each time when the attack found correctly the last bits of the secret key $\mathbf{s}_2$, we collected the corresponding $\widetilde{e} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{a}}^t\mathbf{s}_2 = \mathbf{v}^t(\mathbf{A}^t\mathbf{s}_1 + \mathbf{e})$. For the first experiment the size of the

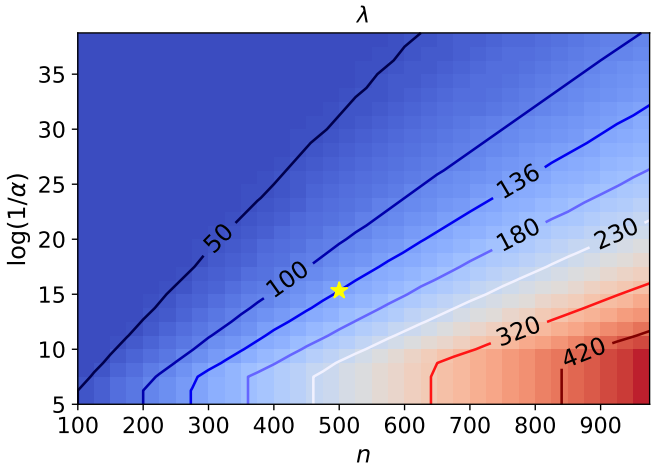


Fig. 4: Bit-security as a function of LWE parameters n and α assuming the enumeration BKZ cost model. Here n denotes the dimension, α denotes the standard deviation of the noise. The picture represents the security level λ of LWE samples, $\lambda = \log(\min(T_{\text{ourAttack}}(n, \alpha), 2^{n/2}))$. Numbered lines on the picture represent security levels. Star denotes current parameters of key switching in the TFHE scheme.

Table 3: Parameters required for guessing 5 bits of the key with $\delta = 1.013$. m is the number of samples needed for one lattice reduction (19), σ is the parameter of modular Gaussian distribution $\mathcal{G}_\sigma$ (Lemma 5), R is the number of samples needed to distinguish distributions $\mathcal{G}_\sigma$ and $\mathcal{U}$ (15).

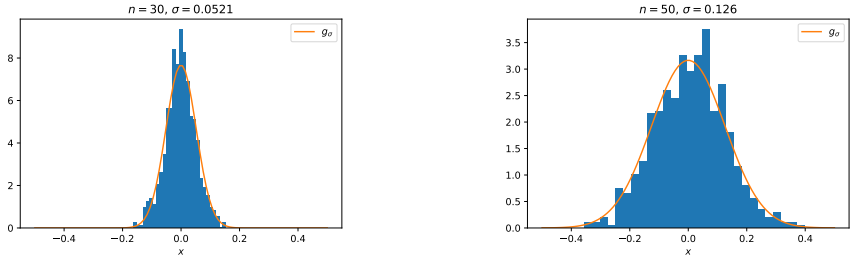
$(n, -\log(\alpha))$	m	σ	R
(30,8)	76	0.0521	32
(50,8)	90	0.126	74

collected sample is $20 \times R_1 = 640$, for the second experiment: $10 \times R_2 = 740$. The collected data is presented in Figure 5.

In Table 4 we compare theoretical predictions and estimations obtained from the experiments for the parameters of modular Gaussian distribution $\mathcal{G}_\sigma$. Experimental estimations of mean and variance in both cases match closely theoretical predictions.

6 Conclusion

In this work, we demonstrated that the dual lattice attack used to estimate the security of the TFHE scheme can be improved by applying a hybrid approach



(a) (b)

Fig. 5: Distribution of $\widetilde{e} = \mathbf{v}^t(\mathbf{A}^t\mathbf{s}_1 + \mathbf{e}) \mod 1$.
Figure 3a represents data from the experiment with parameters $(n, \alpha) = (30, 2^{-8})$, figure 3b – from the experiment with parameters $(n, \alpha) = (50, 2^{-8})$. Blue histograms denote observed data, orange lines – theoretical predictions of the distribution.

Table 4: Estimated mean and variance. σ is the parameter of the modular Gaussian distribution $\mathcal{G}_\sigma$, $\mathrm{Var}(\mathcal{G}_\sigma)$ is variance of $\mathcal{G}$

(n, α)	sample size	σ	$\mathrm{Var}(\mathcal{G}_\sigma)$	estimated variance	average of sample
$(30, 2^{-8})$	640	0.0521	0.002714	0.002619	-0.00207
$(50, 2^{-8})$	740	0.126	0.1587	0.14515	0.0064

consisting in a dual attack on a projected sublattice, lazy modulus switching, and an efficient batch computation of the leaves of the enumeration tree, performed using fast matrix multiplication to exploit the recursive structure of the space we are searching in. This techniques offer an asymptotic speedup and allow to re-evaluate the actual security level of the TFHE scheme. We estimate the complexity of the proposed attack under several widely used BKZ cost models. Even if it is still an open question to determine which model gives the most accurate predictions of the behavior of lattice reduction, our results show that the security claim of TFHE is largely overestimated, in *any* cost model for lattice reduction. Therefore, the key size of the TFHE scheme should be significantly increased, which would lead to non-negligible slowdowns.

References

1. Ajtai, M.: The shortest vector problem in l 2 is np-hard for randomized reductions. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 10–19. ACM (1998)
2. Albrecht, M.R.: On dual lattice attacks against small-secret lwe and parameter choices in HElib and SEAL. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 103–129. Springer (2017)

3. Albrecht, M.R., Cid, C., Faugere, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Designs, Codes and Cryptography **74**(2), 325–354 (2015)
4. Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the {LWE, NTRU} schemes! In: International Conference on Security and Cryptography for Networks. pp. 351–367. Springer (2018)
5. Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the efficacy of solving lwe by reduction to unique-SVP. In: International Conference on Information Security and Cryptology. pp. 293–310. Springer (2013)
6. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)
7. Atkinson, K.E.: An introduction to numerical analysis. John Wiley & Sons (2008)
8. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms. pp. 10–24. Society for Industrial and Applied Mathematics (2016)
9. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM (JACM) **50**(4), 506–519 (2003)
10. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 13 (2014)
11. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 575–584. ACM (2013)
12. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Annual cryptology conference. pp. 505–524. Springer (2011)
13. Buchmann, J., Göpfert, F., Player, R., Wunderer, T.: On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In: International Conference on Cryptology in Africa. pp. 24–43. Springer (2016)
14. Chen, Y.: Réduction de réseau et sécurité concrete du chiffrement completement homomorphe. Ph.D. thesis, Paris 7 (2013)
15. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 1–20. Springer (2011)
16. Cheon, J.H., Stehlé, D.: Fully homomophic encryption over the integers revisited. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 513–536. Springer (2015)
17. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 3–33. Springer (2016)
18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 377–408. Springer (2017)
19. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology pp. 1–58 (2018)

20. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 617–640. Springer (2015)
21. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive **2012**, 144 (2012)
22. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 257–278. Springer (2010)
23. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)
24. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference. pp. 75–92. Springer (2013)
25. Gentry, C., et al.: Fully homomorphic encryption using ideal lattices. In: Stoc. vol. 9, pp. 169–178 (2009)
26. Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Annual Cryptology Conference. pp. 447–464. Springer (2011)
27. Howgrave-Graham, N.: A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In: Annual International Cryptology Conference. pp. 150–169. Springer (2007)
28. Lehmann, E.L., Romano, J.P.: Testing statistical hypotheses. Springer Science & Business Media (2006)
29. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)
30. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Cryptographers' Track at the RSA Conference. pp. 319–339. Springer (2011)
31. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Cryptographers' Track at the RSA Conference. pp. 293–309. Springer (2013)
32. Micciancio, D., Walter, M.: Practical, predictable lattice basis reduction. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 820–849. Springer (2016)
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography, 2005. In: STOC. pp. 84–93. ACM (2005)
34. Sarma, J.: IITM-CS6840: Advanced Complexity Theory, Lecture Notes, Lecture 11 (2012), uRL: https://www.cse.iitm.ac.in/~jayalal/teaching/CS6840/2012/lecture11.pdf. Last modified on 2017-10-24
35. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse and ternary secret LWE. IACR Cryptology ePrint Archive **2019**, 1019 (2019)
36. Wunderer, T.: Revisiting the hybrid attack: Improved analysis and refined security estimates. IACR Cryptology ePrint Archive **2016**, 733 (2016)

A Proof of Lemma 5.

Proof. Under Assumption 1, the coordinates of $\mathbf{w}_q$ are independent and distributed according to the Gaussian distribution with expectation 0 and standard deviation $\delta^{n+m}/\sqrt{n+m}$. Since $\mathbf{w}_q = (q \cdot \mathbf{x} \,||\, q^{-n/m} \cdot \mathbf{v})^t$, the coordinates

of vectors $\mathbf{x}$ and $\mathbf{v}$ also have centered Gaussian distribution, but with different standard deviations. Let

$$\sigma_{\mathbf{x}} = \frac{1}{q} \cdot \frac{\delta^{m+n}}{\sqrt{m+n}} \quad \text{and} \quad \sigma_{\mathbf{v}} = q^{n/m} \cdot \frac{\delta^{m+n}}{\sqrt{m+n}}$$

be the standard deviation of coordinates of $\mathbf{x}$ and of $\mathbf{v}$ correspondingly. Consider the distribution of

$$\mathbf{v}^t\mathbf{b} = \mathbf{x}^t\mathbf{s} + \mathbf{v}^t\mathbf{e} = \sum_{i=1}^{n} x_i \cdot s_i + \sum_{i=1}^{m} v_i \cdot e_i.$$

$\mathbf{v}^t\mathbf{b}$ is a sum of $m+n$ independent random variables and, therefore, its distribution can be approximated by a Gaussian distribution according to the Central Limit Theorem. In order to learn the parameters of the Gaussian, we need to obtain expectation and variance of $x_1 \cdot s_1$ and $v_1 \cdot e_1$.

First, consider the distribution of $x_1 \cdot s_1$. As s_1 has a Bernoulli distribution with parameter S^2, $x_1 s_1$ is a random variable from the distribution that can be obtained by sampling 0 with probability S^2 and sampling from a Gaussian distribution with mean 0 and variance $\sigma_{\mathbf{x}}^2$ with probability $1 - S^2$. Therefore, $\mathbb{E}(x_1 \cdot s_1) = 0$ and $\mathrm{Var}(x_i \cdot s_i) = S^2\sigma_{\mathbf{x}}^2$.

Then, consider $v_1 e_1$. As $\mathbf{v}$ and $\mathbf{e}$ are independent and $\mathbb{E}(v_1) = \mathbb{E}(e_1) = 0$, $\mathbb{E}(v_1 e_1) = \mathbb{E}(v_1)\mathbb{E}(e_1) = 0$ and $\mathrm{Var}(v_1 e_1) = \mathrm{Var}(v_1) \cdot \mathrm{Var}(e_1) = \alpha^2\sigma_{\mathbf{v}}^2$.

Then the distribution of $\mathbf{v}^t\mathbf{b}$ is close to the Gaussian distribution with expectation 0 and variance

$$\sigma^2 = n\,\mathrm{Var}(x_1 s_1) + m\,\mathrm{Var}(v_1 e_1) = nS^2\sigma_{\mathbf{x}}^2 + m\alpha^2\sigma_{\mathbf{v}}^2 = \frac{\delta^{2(m+n)}}{m+n}\Big(\frac{nS^2}{q^2} + m\alpha^2 q^{2n/m}\Big). \tag{16}$$

Our goal is to obtain a distribution that is as concentrated around zero as possible. Hence we choose parameters m and q in order to minimize variance of $\mathbf{v}^t\mathbf{b}$.

First, we find the optimal value of q by differentiation of Equation (16) :

$$\frac{\partial\sigma^2}{\partial q} = \frac{\delta^{2(m+n)}}{m+n} \cdot \Big(-\frac{2nS^2}{q^3} + \frac{2n}{m} \cdot m\alpha^2 q^{\frac{2n}{m}-1}\Big) = 0 \quad \rightarrow \quad q_{\mathrm{opt}} = \Big(\frac{S}{\alpha}\Big)^{\frac{m}{m+n}}.$$

After replacing q by q_{opt} in Equation (16) we obtain:

$$\sigma^2 = \left(S\delta^{m+n}\Big(\frac{\alpha}{S}\Big)^{\frac{m}{m+n}}\right)^2. \tag{17}$$

Also, for $\sigma_{\mathbf{x}}$ and $\sigma_{\mathbf{v}}$ we obtain the following relation

$$\frac{\sigma_{\mathbf{x}}}{\sigma_{\mathbf{v}}} = \frac{q^{-n/m}}{q} = \frac{\alpha}{S}. \tag{18}$$

Then, we find the optimal value of m by differentiating $\ln(\sigma)$:

$$\frac{\partial \ln(\sigma)}{\partial m} = \ln(\delta) + n \ln\left(\frac{\alpha}{S}\right) \cdot \frac{1}{(m+n)^2} = 0 \quad \rightarrow \quad m_{\text{opt}} = \sqrt{n \cdot \frac{\ln(S/\alpha)}{\ln(\delta)}} - n \tag{19}$$

Now we replace m by m_{opt} in Equation (17):

$$\sigma(\delta, n, S, \alpha) = \sigma(\widehat{m}, \delta, n, S, \alpha) = \alpha \cdot \exp\left(2\sqrt{n \ln(S/\alpha) \ln(\delta)}\right).$$

The distance between the distribution of $\mathbf{v}^t\mathbf{b}$ and the Gaussian distribution with mean 0 and variance σ^2 can be estimated by the Berry-Esseen inequality (see Theorem 1). To use this inequality we need to compute the third absolute moments of $x_1 s_1$ and $v_1 e_1$.

We start with $x_1 s_1$. As x_1 and s_1 are independent,

$$\mathbb{E}\{|x_1 s_1|^3\} = \mathbb{E}\{|x_1|^3\}\mathbb{E}\{|s_1|^3\}.$$

By Lemma 2, $\mathbb{E}\{|x_1|^3\} = 2\sqrt{2/\pi}\sigma_x^3$. As s_1 has the Bernoulli distribution with parameter S^2, $\mathbb{E}\{|s_1|^3\} = \mathbb{E}\{s_1\} = S^2$. Putting two parts together, we get

$$\rho_{x_1 s_1} = \mathbb{E}\{|x_1 s_1|^3\} = 2\sqrt{2/\pi} S^2 \sigma_{\mathbf{x}}^3. \tag{20}$$

In the same way we obtain

$$\rho_{v_1 e_1} \mathbb{E}\{|v_1 e_1|^3\} = \frac{8}{\pi} \alpha^3 \sigma_{\mathbf{v}}^3. \tag{21}$$

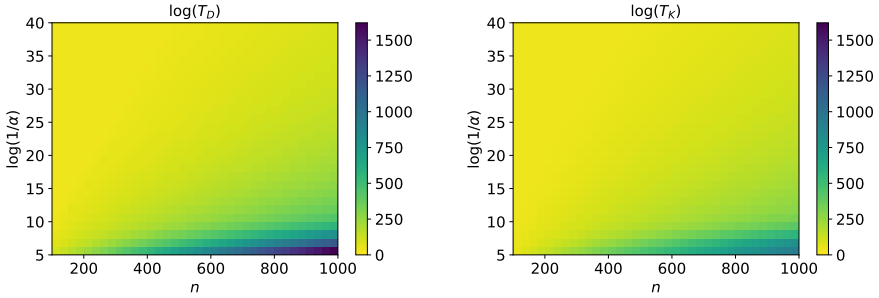
Denote the cumulative distribution function of $\mathbf{v}^t\mathbf{b}$ by $F_{\mathbf{v}^t\mathbf{b}}$, and denote the cumulative distribution function of the Gaussian distribution with mean 0 and variance σ^2 by Φ_σ. By the Berry-Esseen inequality, there exists a constant C_0 such that

$$\sup_{x \in \mathbb{R}} |F_{\mathbf{v}^t\mathbf{b}}(x) - \Phi_\sigma(x)| \leqslant C_0 \cdot \frac{n\rho_{x_1 s_1} + m\rho_{v_1 e_1}}{(nS^2\sigma_{\mathbf{x}}^2 + m\alpha^2\sigma_{\mathbf{v}}^2)^{3/2}}. \tag{22}$$

Then, using Equations (18) and (20) to (22), for the distance between the distributions we get:

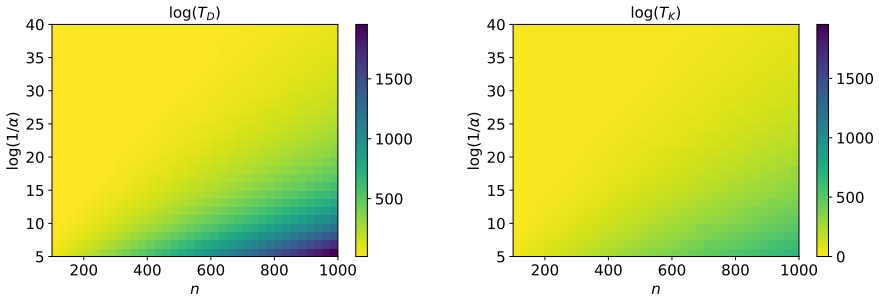
$$\sup_{x \in \mathbb{R}} |F_{\mathbf{v}^t\mathbf{b}}(x) - \Phi_\sigma(x)| \leqslant C_0 \sqrt{\frac{8}{S^2\pi}} \cdot \frac{n + mS\sqrt{8/\pi}}{(m+n)^{3/2}} \leqslant C_0 \sqrt{\frac{8}{S^2\pi}} \cdot \frac{1}{\sqrt{m+n}}. \tag{23}$$

B Heatmaps for sieving and delta-squared BKZ cost models



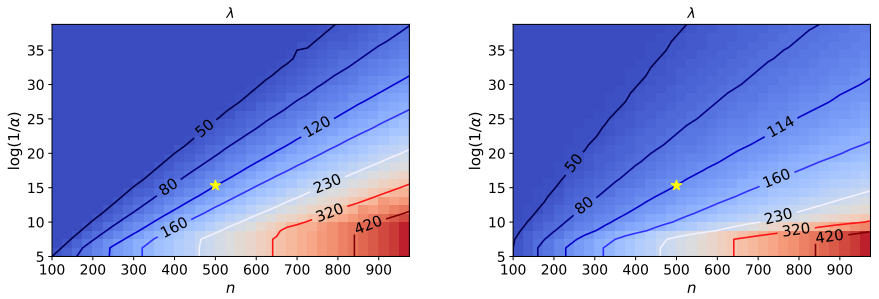
(a) dual attack (b) our attack

Fig. 6: Comparison of costs of the attacks under the sieving BKZ cost model.



(a) dual attack (b) our attack

Fig. 7: Comparison of costs of the attacks under the delta-squared BKZ cost model.



(a) delta-squared model (b) sieving

Fig. 8: Bit-security as a function of LWE parameters n and α assuming sieving and delta-squared BKZ cost models.