Worst-Case to Average-Case Hardness of LWE: A Simple and Practical Perspective

Divesh Aggarwal[1], Leong Jin Ming[2], and Alexandra Veliche[3]

[1] National University of Singapore, Singapore
divesh@comp.nus.edu.sg
[2] Imperial College of London, London, U.K.
e0407679@u.nus.edu
[3] University of Michigan, Ann Arbor, MI, U.S.A.
aveliche@umich.edu

Abstract. In this work, we study the worst-case to average-case hardness of the Learning with Errors problem (LWE) under an alternative measure of hardness − the maximum success probability achievable by a probabilistic polynomial-time (PPT) algorithm. Previous works by Regev (STOC 2005), Peikert (STOC 2009), and Brakerski, Peikert, Langlois, Regev, Stehle (STOC 2013) give worst-case to average-case reductions from lattice problems, specifically the approximate decision variant of the Shortest Vector Problem (GapSVP) and the Bounded Distance Decoding (BDD) problem, to LWE. These reductions, however, are lossy in the sense that even the strongest assumption on the worst-case hardness of GapSVP or BDD implies only mild hardness of LWE. Our alternative perspective gives a much tighter reduction and strongly relates the hardness of LWE to that of BDD. In particular, we show that under a reasonable assumption about the success probability of solving BDD via a PPT algorithm, we obtain a nearly tight lower bound on the highest possible success probability for solving LWE via a PPT algorithm. Furthermore, we show a tight relationship between the best achievable success probability by any probabilistic polynomial-time algorithm for decision-LWE to that of search-LWE. Our results not only refine our understanding of the computational complexity of LWE, but also provide a useful framework for analyzing the practical security implications.

Keywords: LWE, BDD, success probability

1 Introduction

The Learning with Errors (LWE) problem has become one of the most important computational problems in post-quantum cryptography and computational complexity over the last two decades. Since Regev introduced this problem in 2005 [Reg09], the LWE problem has been used as the basis of a wide variety of cryptographic primitives, as well as a tool for proving hardness results in learning theory [Reg06]. Formally, the LWE problem is defined as follows: The input consists of a uniformly random matrix $\mathbf{A} \sim \mathbb{Z}_p^{m \times n}$ and a vector $\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_p^m$, where $\mathbf{s} \in \mathbb{Z}_p^n$ is a secret vector chosen uniformly at random from $\mathbb{Z}_p^n$ and $\mathbf{e} \in \mathbb{Z}_p^m$ is an error vector of small magnitude sampled according to a Gaussian distribution. The goal is to output $\mathbf{s}$. Here the positive integer p is called the modulus and n is the dimension. In his seminal work, Regev related LWE to worst-case lattice problems that form the foundation of lattice-based cryptography.

1.1 LWE and Computational Lattice Problems

Lattice Problems. A *lattice* $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$ that consists of all integer linear combinations of m linearly independent vectors $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_m\} \subset \mathbb{R}^n$. Formally, it is defined as

$$\mathcal{L}(\mathcal{B}) := \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i \mid \forall i \in \{1, ..., n\}, z_i \in \mathbb{Z} \right\}.$$

We call m the rank, n the dimension of the ambient space, and $\mathcal{B}$ a basis of the lattice. A lattice can have many possible bases.

The two most important computational lattice problems are the *Shortest Vector Problem* (SVP) and the *Closest Vector Problem* (CVP). In SVP, one is given a basis for a lattice and asked to output a shortest non-zero lattice vector. We denote the length of a shortest non-zero vector of a lattice $\mathcal{L}$ by $\lambda_1(\mathcal{L})$. In the approximation variant of SVP, denoted by γ-SVP for some $\gamma > 1$, the goal is to output a nonzero lattice vector whose length is at most $\gamma\lambda_1(\mathcal{L})$. In CVP, one is given a target vector and basis for a lattice and asked to output a closest lattice vector to the target vector. Similarly, in its approximation variant γ-CVP, the goal is to output a lattice vector whose distance from the target vector is at most γ times the minimum distance between the target vector and the lattice. There exists a polynomial-time reduction from SVP to CVP, which preserves the dimension, rank, and approximation factor [GMSS99].

A closely related problem to CVP is the *Bounded Distance Decoding* (BDD) problem, denoted by BDD_α for some $\alpha < \frac{1}{2}$. This is a promise problem in which the goal is to solve CVP under the promise that the distance of the target from the lattice is at most $\alpha\lambda_1(\mathcal{L})$. Note that, by the triangle inequality, this promise ensures that the closest vector to the target is unique. In our work, we only consider length and distance in the standard Euclidean (ℓ_2) norm.

Computational lattice problems are crucial because of their association with lattice-based cryptography. Specifically, the security of numerous cryptographic systems such as [Ajt96, MR04, Reg06, MR09, Reg09, Gen09, BV14] relies on the complexity of approximately solving lattice problems to within a polynomial factor. Aside from cryptosystem design, since the 1980s, solvers for lattice problems have found applications in cryptanalytic tools [Sha85, Bri83, LO85], algorithmic number theory [LLL82], and convex optimization [Kan87, FT87].

Algorithms for Lattice Problems. Algorithms for CVP and SVP have been designed and studied extensively for decades. Kannan proposed an enumeration algorithm [Kan87] for CVP and hence for all lattice problems, with a time complexity of $n^{O(n)}$ and space requirement of $\mathrm{poly}(n)$. Micciancio and Voulgaris introduced a deterministic algorithm for CVP with a time complexity of $2^{2n+o(n)}$ and space requirement of $2^{n+o(n)}$ [MV13]. A few years later, Aggarwal, Dadush, Regev, and Stephens-Davidowitz [ADRS14, ADRS15] presented the current fastest known algorithm for SVP and CVP, which has a time and space complexity of $2^{n+o(n)}$. The best-known and proven runtime for an approximation factor $\gamma = n^c$ is approximately $2^{n/(c+1)}$ for constant $c \geq 0$. For the current state of the art, we refer the reader to [ALS20].

Hardness of Lattice Problems. Both γ-SVP and γ-CVP are known to be NP-hard for nearly-polynomial approximation $n^{c/\log\log n}$ for some constant $c > 0$ [vEB81, DKRS03, Din02, Kho05, HR18]. Through a series of works, Aggarwal, Bennett, Golovnev, and Stephens-Davidowitz [BGS17, ASD18, ABGSD21], demonstrated that approximating CVP and SVP to a factor γ slightly greater than 1 is not achievable in time $2^{o(n)}$ under variants of the Exponential Time Hypothesis.

Worst-case to Average-case Reduction for LWE. The best known algorithm that solves LWE for dimension n and modulus p runs in time $p^{O(n/\log n)}$ [BKW00]. The decision variant of LWE is the one most directly related to the security of lattice-based cryptography. In decision-LWE, the goal is to distinguish between an LWE instance as described above and a uniform sample from $\mathbb{Z}_p^{m\times(n+1)}$. Regev [Reg09] gave a polynomial-time reduction from BDD to LWE. Additionally, Regev gave a quantum polynomial-time reduction from a decision variant of γ-SVP, known as GapSVP_γ, to BDD for γ polynomial in the dimension of the lattice. Peikert [Pei09] improved this result to a classical reduction from GapSVP_γ to LWE, albeit with the modulus p becoming exponential in the dimension n. Later, Brakerski et al. [BLP+13] gave a reduction from LWE with dimension n and modulus p exponential in n to LWE with dimension n^2 and modulus p polynomial in n, thus allowing the modulus to shrink from exponential to polynomial.

Overall, these results give us a polynomial-time reduction from lattice problems (GapSVP, BDD) in dimension n to LWE in dimension n^2 with modulus p polynomial in n. This means that even if we assume that the currently best known algorithms for BDD or GapSVP are the best possible, this reduction only says that LWE in dimension n cannot be solved faster than in $2^{\Omega(\sqrt{n})}$ time. This is a much worse lower bound than

one would expect based on the state-of-the-art algorithms for LWE [BKW00]. This leads us to the following natural question.

Question 1. Is there a tight reduction from worst-case lattice problems (such as BDD) to LWE that gives a tighter lower bound on the runtime for solving LWE?

1.2 A Novel Perspective on Computational Hardness

In cryptography, security models often assume that all possible adversaries are computationally bounded, based on the state-of-the-art capabilities of modern computers. Often, when we declare that a cryptographic scheme is 256-bit secure, we intuitively understand that the fastest algorithm for successfully breaking the cryptosystem runs in 2^{256} units of time. What we typically require, however, is that any algorithm that succeeds in attacking the cryptosystem with probability more than 2^{-256} cannot do so in a "reasonable" amount of time.

Unpredictability Entropy. Motivated by this discrepancy, Aggarwal and Maurer [AM11] proposed a different perspective on studying the complexity of a computational search problem. They introduced the concept of *unpredictability entropy* for a computational problem, defined as follows. If p is the maximum success probability of a probabilistic polynomial-time (PPT) algorithm that solves the problem, the unpredictability entropy of the problem is $\log_2(\frac{1}{p})$.

Two closely related properties of a search problem also studied in [AM11] are witness compression and oracle complexity. A search problem $\mathcal{P}$ is said to have *witness compression* w if there is a PPT reduction from $\mathcal{P}$ to another search problem $\mathcal{Q}$ such that the problem $\mathcal{Q}$ has a solution/witness of length w. The *oracle complexity* of the problem $\mathcal{P}$ is defined as the number of arbitrary YES/NO questions needed to get a solution to the search problem, i.e. find a witness.

It was shown in [AM11] that unpredictability entropy, witness compression, and oracle complexity of a computational problem are equal, up to lower order additive terms. Notice that these quantities are all indicative of the number of bits in which the hardness of a computational problem can be captured.

The authors of [AM11] also gave a straightforward polynomial-time algorithm for both SVP and CVP, that achieves a success probability of $2^{-n^2/4-o(n^2)}$, showing that both these problems have unpredictability entropy/witness compression/oracle complexity $n^2/4+o(n^2)$. These algorithms are straightforward adaptations of the LLL algorithm and Babai's nearest plane algorithm [LLL82, Bab86]. Despite the various algorithmic techniques available for solving lattice problems, none of these methods appear to yield meaningful results when constrained to PPT algorithms, even if we restrict our attention to approximation variants of the lattice problems with approximation factor γ polynomial in the lattice dimension n. Additionally, the close relationship between BDD_α and γ-SVP for $\frac{1}{\alpha}$ and γ both polynomial in n [LM09], suggests that it is unlikely for a polynomial-time algorithm for BDD to do much better than current algorithms. With this in mind, it is reasonable to conjecture the following.

Conjecture 1. For $\frac{1}{\alpha}$ and γ both polynomial in n, no PPT algorithm can solve BDD_α, γ-SVP, and γ-CVP with probability better than $2^{-\kappa n^2}$ for some universal constant $\kappa > 0$.

If we want to make a concrete conjecture, a choice of $\kappa \approx \frac{1}{10}$ seems reasonable. It is easy to see that any algorithm for solving LWE in polynomial time for modulus p and dimension n has success probability at least p^{-n}. This can be obtained by guessing the secret $\mathbf{s}$ uniformly at random and then checking whether $\mathbf{b} - \mathbf{As}$ is small. We ask the following natural question, which is a novel perspective on the worst-case to average-case reductions for LWE.

Question 2. Assuming Conjecture 1, is there a lower bound of $p^{-\Omega(n)}$ on the success probability of solving LWE via a polynomial-time algorithm?

In this work, we answer this question in the affirmative. Since the security of cryptosystems is based on the hardness of decision problems, we need to adapt the question above and formulate the measure of hardness of a decision problem in terms of the success probability of the best efficient (i.e. PPT) algorithm.

One-Sided Error PPT Algorithms. This question has been well studied particularly in the context of NP-hard problems, for which we expect any PPT algorithm to be able to distinguish only with a small advantage. Some previous works, (such as [PP10], and the references therein) have explored the realm of *one-sided error* probabilistic polynomial-time (OPP) algorithms for NP-hard decision problems. This relies on the assumption that when the algorithm is presented with a NO instance, it consistently outputs NO, while for a YES instance, the algorithm outputs YES with a small success probability α.

However, for decision problems like decision-LWE whose input is chosen according to a distribution, we cannot hope to output NO with probability 1 even given a NO instance. This is because a NO instance for this problem is just a random element from $\mathbb{Z}_q^{m\times(n+1)}$ that will look like a YES instance with non-zero probability. Thus, it is reasonable to adapt the notion of OPP algorithms to have two parameters α, β such that $\alpha \gg \beta$, and the algorithm outputs YES with probability at least α, when given a YES instance, and with probability at most β when given a NO instance. We call such an algorithm an (α, β)-*solver* for decision LWE. Using this notation, we ask the following natural question.

Question 3. Assuming that there is no PPT algorithm that succeeds in solving search-LWE with probability α, can we prove that there is no (α', β')-solver for the corresponding decision-LWE problem with $\alpha' \approx \alpha$ and $\beta' \ll \alpha'$.

We also answer this question in the affirmative.

1.3 Our Contributions

In this paper, we show that if no PPT algorithm can solve BDD on a lattice of rank n with success probability greater than $2^{-O(n^2)}$, then no PPT algorithm can solve search-LWE in n dimensions with success probability greater than $2^{-O(n^2)}$. Here n is the dimension of the lattice even if we restrict the secret to be a binary vector. Informally, our first main result is the following.

Theorem 1. *(informal) If no PPT algorithm can solve BDD_γ for gap $\gamma \in (0, \frac{1}{2})$ with success probability greater than $2^{-\kappa n^2-4}$ for some constant $\kappa > 0$, then no PPT algorithm can solve search-LWE for binary secret and modulus p polynomial in the dimension n with success probability $2^{-\kappa n^2}$.*

Note that the above statement can easily be extended to having the secret chosen uniformly at random from $\mathbb{Z}_p^n$ using a standard randomization of the secret. We show this explicitly in Section 3.3.

We emphasize here that while our reductions are adaptations of similar reductions in the literature, adapting these reductions to our setting required great care, since the number of oracle calls made in the reductions is crucial. In particular, for a reduction from problem $\mathcal{P}$ to problem $\mathcal{Q}$ that makes k calls to a solver for problem $\mathcal{Q}$, an upper bound of δ on the success probability for solving problem $\mathcal{P}$ in polynomial time would imply an upper bound of $\delta^{1/k}$ on the success probability for solving problem $\mathcal{Q}$ in polynomial time. So for our reductions, we needed to adapt known reductions, which make polynomially many oracle calls, into reductions that make only one call to the oracle and then guess successfully with a small probability. These reductions with a single oracle call are known as *one-shot reductions*. This approach enables us to obtain meaningful bounds on the success probability of polynomial-time reductions.

Our second main contribution is concretely relating the hardness of solving decision-LWE to that of solving search-LWE using our new framework. In particular, we show that if no algorithm can solve search-LWE on a lattice of rank n with modulus p in expected polynomial time with success probability close to α^2, then there is no PPT algorithm that can solve decision-LWE for the same dimension and modulus and answers correctly with probability close to α, outputs $\perp$ with probability $1-\alpha$, and answers incorrectly with the remaining tiny probability. This relies on the assumption that β is large and close to 1, which requires the oracle $\mathcal{B}$ to be correct with high probability when it does not output $\perp$. Intuitively, this means that $\mathcal{B}$ admits defeat by outputting $\perp$ far more often than it guesses the answer incorrectly. Using this framework, we can informally state our second main result as follows.

Theorem 2. *(informal) If no algorithm can solve search-LWE for modulus p polynomial in the dimension n with success probability α^2 in expected polynomial time, then no PPT algorithm can solve decision-LWE for the same modulus p and dimension n that outputs a correct answer with probability α and outputs $\perp$ with probability $1-\alpha$.*

To prove our second main result, we use a result by Levin [Lev12] This result is an improvement of the original Goldreich-Levin Theorem in [GL89] which gives a tight relationship between the success probability of finding a hard-core bit and that of inverting the corresponding one-way function. In our work, we rigorously prove Levin's result and generalise it from binary $\{0,1\}$ to $\mathbb{Z}_p$ for all but a few values of p. This required considerable care and can easily find applications elsewhere, so we consider it to be a contribution of independent interest.

Note that the statement of Theorem 2 is in terms of *expected* polynomial time, which is a crucial aspect of the Goldreich-Levin Theorem used in our reduction. Because the runtime is polynomial only in expectation, we cannot directly combine this result with that of Theorem 1.

1.4 Paper Organization

In Section 2, we give the mathematical background needed for our results and formally define the computational problems discussed throughout the paper. Section 3 contains the proof of our first main result, where we use the techniques from [Reg09]. In Section 4, we prove our second main result, where we reduce the hardness of solving search-LWE to the hardness of solving decision-LWE using our new framework [MW18]. We conclude with future directions and open problems in Section 5.

2 Preliminaries

Let $\mathbb{T} := \mathbb{R}/\mathbb{Z}$ denote the additive group of real numbers modulo the integers, i.e. the interval $[0,1)$ with addition modulo 1. $\mathbb{R}_+$ and $\mathbb{Z}_+$ denote the positive real numbers and positive integers, respectively, which do not include zero. Let p be any positive integer (not necessarily prime). $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$ denotes the ring of integers $\{0,1,\ldots,p-1\}$ where addition and multiplication are performed modulo p. We implicitly identify $\mathbb{Z}_p$ with its natural embedding in $\mathbb{Z}$ whenever relevant. For any $n \in \mathbb{Z}_+$, we denote $[n] := \{1,...,n\}$ to be all the integers between 1 and n, inclusive. We use $\langle \cdot, \cdot \rangle$ to denote the standard dot product, so $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^{\mathrm{T}}\mathbf{y}$ for column vectors $\mathbf{x}, \mathbf{y}$ where addition and multiplication are performed according to the domain. We use lowercase boldface letters (such as $\mathbf{v}$), to denote vectors, uppercase boldface letters (such as $\mathbf{B}$) to denote matrices, and calligraphic uppercase boldface letters (such as $\mathcal{A}$) to denote algorithms. We use $\|\cdot\| = \|\cdot\|_2$ to denote the Euclidean norm. Throughout this paper, all norms are assumed to be Euclidean unless specified otherwise. We say that an algorithm is *efficient* if it runs in time polynomial in the size of the input, and use the terms "efficient" and "polynomial-time" interchangeably throughout.

We will need the following standard lemma.

Lemma 1. *Let $Y_1, \ldots, Y_t$ be pairwise independent Bernoulli random variables where $\Pr[Y_i = 1] = p$ for $1 \le i \le t$. Then for any $c > 0$,*

$$\Pr\left[\left|\sum_{i=1}^{t} Y_i - tp\right| \le ctp\right] \ge 1 - \frac{1}{c^2 tp}\,.$$

Proof. Let $Y = Y_1 + \cdots + Y_t$. The expected value of Y is $\mathbb{E}[Y] = tp$ and the variance of Y is $\mathrm{Var}[Y] = tp(1-p)$. Then by the Chebyshev inequality, we have

$$\Pr\left[|Y - tp| \ge ctp\right] \le \frac{tp(1-p)}{c^2 t^2 p^2} = \frac{1-p}{c^2 tp} \le \frac{1}{c^2 tp}\,.$$

□

2.1 Learning with Errors

Definition 1. (LWE Distribution) *Let ϕ be a probability density function on $\mathbb{T}$, and $\mathbf{s} \in \mathbb{Z}_p^n$ denote the unknown secret vector. The* Learning with Errors (LWE) distribution *$A_{\mathbf{s},\phi}$ is the distribution over $\mathbb{Z}_p^n \times \mathbb{T}$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_p^n$ uniformly at random and $e \in \mathbb{T}$ according to ϕ, then outputting $(\mathbf{a}, \frac{1}{p}\langle \mathbf{a}, \mathbf{s} \rangle + e)$.*

The standard Learning with Errors problem has both search and decision variants, defined as follows.

Definition 2. (Search-LWE) *The search variant of the Learning with Errors problem,* search-$LWE_{n,p,\phi}$*, is defined as: given a polynomial number of samples from the distribution $A_{\mathbf{s},\phi}$, recover the secret $\mathbf{s} \in \mathbb{Z}_p^n$.*

Definition 3. (Decision-LWE) *The decision variant of the Learning with Errors problem,* decision-$LWE_{n,p,\phi}$*, is defined as: given a polynomial number of samples either from the distribution $A_{\mathbf{s},\phi}$ or independent and uniformly distributed samples from $\mathbb{Z}_p^n \times \mathbb{T}$, output*

- *YES if the samples are from the LWE distribution $A_{\mathbf{s},\phi}$, or*
- *NO if the samples are uniformly random over $\mathbb{Z}_p^n \times \mathbb{T}$.*

Definition 4. (Binary-LWE) *The Binary Learning with Errors problem,* $\mathsf{binLWE}_{n,p,\phi}$*, is the search-*$\mathsf{LWE}_{n,p,\phi}$ *problem with the restriction that the secret $\mathbf{s}$ is uniform over $\{0,1\}^n$.*

2.2 Lattices

Definition 5. (Lattice) *Let $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subseteq \mathbb{R}^n$ be a set of linearly independent vectors. The* lattice *$\mathcal{L} = \mathcal{L}(\mathcal{B})$ generated by $\mathcal{B}$ is the set of vectors spanned by $\mathcal{B}$ over $\mathbb{Z}$, i.e.*

$$\mathcal{L}(\mathcal{B}) := \left\{ \sum_{i=1}^{m} z_i \mathbf{b}_i \;\middle|\; \mathbf{z} = (z_1, \ldots, z_n)^T \in \mathbb{Z}^n \right\} \subset \mathbb{R}^n .$$

The basis *$\mathcal{B}$ is usually expressed as a matrix $\mathbf{B}$ whose columns are the vectors of $\mathcal{B}$ and we write $\mathcal{L}(\mathbf{B}) := \mathcal{L}(\mathcal{B})$. Here m is the* rank *of the lattice as a free $\mathbb{Z}$-module, and n is the* dimension *of the ambient space.*

Throughout this paper, we assume that the lattices are *full-rank*, meaning that $n = m$.

Definition 6. (Determinant) *The* determinant *of lattice $\mathcal{L}$ generated by $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m) \in \mathbb{R}^{n \times m}$ is*

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

Definition 7. (Dual Lattice) *The* dual lattice *of $\mathcal{L}$ is*

$$\mathcal{L}^* := \{\mathbf{x} \in \mathbb{R}^n \mid \forall\ \mathbf{v} \in \mathcal{L},\ \langle \mathbf{v}, \mathbf{x} \rangle \in \mathbb{Z}\}.$$

Given a basis matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ of a lattice $\mathcal{L}$, we write $\mathbf{B}^*$ to denote the corresponding basis matrix for $\mathcal{L}^*$. This satisfies $(\mathbf{B}^*)^T \mathbf{B} = \mathbf{I}_m$.

Definition 8. (Successive Minima) *For any $k \in \mathbb{Z}_+$, the* k-th successive minimum *of $\mathcal{L}$ (in the Euclidean norm) is*

$$\lambda_k(\mathcal{L}) := \inf\{r \in \mathbb{R} \mid B(\mathbf{0}, r) \text{ contains } k \text{ linearly independent vectors}\},$$

where $B(\mathbf{0}, r)$ denotes the ball of radius r centered at the origin. In particular, $\lambda_1(\mathcal{L})$ is the length of any shortest nonzero vector in $\mathcal{L}$.

Definition 9. (Unique Closest Lattice Vector) *For any vector $\mathbf{v} \in \mathbb{R}^n$ whose distance from the lattice $\mathcal{L}$ is less than $\frac{1}{2}\lambda_1(\mathcal{L})$, there is a unique closest lattice vector to $\mathbf{v}$, which we denote by $\kappa_{\mathcal{L}}(\mathbf{v})$.*

2.3 Probability and Gaussians

Throughout this paper, we frequently use the standard normal distribution over the real numbers. We use the standard notation $N(\mu, \sigma^2)$ to denote the normal distribution with mean μ and variance σ^2.

Definition 10. (Statistical Distance) *Let ϕ_1 and ϕ_2 be probability measures on the space $(X, \mathcal{F})$, where X is the set of outcomes and $\mathcal{F}$ is the collection of events. The* statistical distance *(a.k.a total variation distance) between ϕ_1 and ϕ_2 is*

$$\Delta(\phi_1, \phi_2) := \sup_{A \in \mathcal{F}} \{|\phi_1(A) - \phi_2(A)|\}.$$

In particular, when $X = \mathbb{R}^n$,

$$\Delta(\phi_1, \phi_2) = \frac{1}{2} \int_{\mathbb{R}^n} |\phi_1(\mathbf{x}) - \phi_2(\mathbf{x})| d\mathbf{x}.$$

If X and Y are random variables with distributions ϕ_1 and ϕ_2, respectively, we define $\Delta(X, Y) := \Delta(\phi_1, \phi_2)$. It is immediate that statistical distance satisfies the triangle inequality. Another important property is that it does not increase under the application of any (possibly random) function f [Vad12], i.e.

$$\Delta(f(X), f(Y)) \leq \Delta(X, Y). \tag{$*$}$$

This means that for any algorithm $\mathcal{A}$, the success probability of $\mathcal{A}$ on X differs from the success probability of $\mathcal{A}$ on Y by at most $\Delta(X, Y)$.

Definition 11. (Gaussian Function) *The* Gaussian function *of width $s \in \mathbb{R}_+$ is $\rho_s : \mathbb{R}^n \to \mathbb{R}$, given by*

$$\rho_s(\mathbf{x}) := e^{-\pi \left\| \frac{\mathbf{x}}{s} \right\|^2}.$$

For any countable subset $A \subseteq \mathbb{R}^n$, we write $\rho_s(A) := \sum_{\mathbf{x} \in A} \rho_s(\mathbf{x})$. Furthermore, we denote $\rho := \rho_1$. For any $\mathbf{y} \in \mathbb{R}^n$, we define $\rho_{s,\mathbf{y}}(\mathbf{x}) := \rho_s(\mathbf{x} - \mathbf{y})$.

Note that $\int_{\mathbb{R}^n} \rho_s(\mathbf{x}) d\mathbf{x} = s^n$. Hence

$$\nu_s := \frac{\rho_s}{s^n}$$

is a probability density function on $\mathbb{R}^n$, which we call a continuous Gaussian of width s. Similarly, we write $\nu := \nu_1$. Since a sample from ν_s can be generated by taking n independent samples from the 1-dimensional Gaussian distribution, we assume that we can sample efficiently from ν_s.

Definition 12. (Discrete Gaussian) *For any countable subset A for which $\rho_s(A)$ converges, $D_{A,s} : A \to \mathbb{R}_+$ is the* discrete Gaussian distribution *on A defined by*

$$D_{A,s}(\mathbf{x}) := \frac{\rho_s(\mathbf{x})}{\rho_s(A)}.$$

Definition 13. (Distribution Ψ_γ) *For any $\gamma \in \mathbb{R}_+$, define the distribution $\Psi_\gamma : \mathbb{T} \to \mathbb{R}_+$ by*

$$\Psi_\gamma(r) := \sum_{k \in \mathbb{Z}} \frac{1}{\gamma} e^{-\pi \left(\frac{r-k}{\gamma} \right)^2}.$$

In other words, if X is distributed according to Ψ_γ and $Z \sim N(0, \frac{\gamma^2}{2\pi})$, then X is the image of Z modulo 1.

An important property used in [Reg09] is the fact that Ψ_β does not change much under a small change in the parameter β. In [Reg09], it was shown that the statistical distance between Ψ_β and another distribution Ψ_α, such that β is not too far from α is bounded by a scaling of the ratio $\frac{\beta}{\alpha}$. For our reduction, we need a much tighter bound, so we instead use the ratio between the probability density functions corresponding to Ψ_α and Ψ_β. Formally, we show the following.

Lemma 2. *Let $\alpha \in \mathbb{R}_+$ and $\beta := \alpha(1+\varepsilon)$ for some $\varepsilon > 0$. Denote the probability density functions of distributions Ψ_α and Ψ_β by g_α and g_β, respectively. Then their ratio satisfies*

$$\frac{g_\alpha(x)}{g_\beta(x)} \leq 1+\varepsilon = \frac{\beta}{\alpha}.$$

for any $x \in \mathbb{R}$.

Proof. Suppose X and X' are distributed according to Ψ_α and Ψ_β, respectively. By definition, X is the image of some Z with distribution $N(0, \frac{\alpha^2}{2\pi})$ modulo 1, and similarly, X' is the image of some Z' with distribution $N(0, \frac{\beta^2}{2\pi})$ modulo 1. Then the ratio of the probability density functions f_α and f_β is the same as the ratio of the probability density functions of Z and Z' modulo 1. By this we obtain

$$\frac{g_\alpha(x)}{g_\beta(x)} = \frac{\frac{1}{\alpha}e^{-\pi(\frac{x}{\alpha})^2}}{\frac{1}{\beta}e^{-\pi(\frac{x}{\beta})^2}} = \frac{\beta}{\alpha}e^{-\pi x^2 \frac{1}{\alpha^2}+\pi x^2 \frac{1}{\beta^2}} = \frac{\alpha(1+\varepsilon)}{\alpha}e^{-\pi x^2\left(\frac{1}{\alpha^2}-\frac{1}{\alpha^2(1+\varepsilon)^2}\right)} = (1+\varepsilon)e^{-\pi\frac{x^2}{\alpha^2}\left(1-\frac{1}{(1+\varepsilon)^2}\right)} \leq 1+\varepsilon.$$

The last inequality follows from the fact that $f(x) := e^{-\pi x^2 \frac{1}{\alpha^2}\left(1-\frac{1}{(1+\varepsilon)^2}\right)}$ is a scaling of the Gaussian curve, so $f(x) \leq 1$ for any value of x.

□

Now we prove a multiplicative analog of (∗) for this ratio of probability density functions.

Lemma 3. *Let X and Y be continuous random variables in $\mathbb{R}$ with probability density functions g_X and g_Y, respectively. Suppose that for some fixed $\delta > 0$, their ratio satisfies*

$$\frac{g_X(x)}{g_Y(x)} \leq \delta$$

for all x. Then for any (invertible) function $f : U \to V$ and set $S \subseteq V$,

$$\frac{\Pr[f(X) \in S]}{\Pr[f(Y) \in S]} \leq \delta.$$

Proof. Consider the set $T^* := \left\{u \in U \mid \frac{\Pr[X=u]}{\Pr[Y=u]} > 0\right\}$. This maximises the ratio $\frac{\Pr[X\in T]}{\Pr[Y\in T]}$ over all $T \subseteq U$. This enables us to write

$$\frac{\Pr[f(X) \in S]}{\Pr[f(Y) \in S]} = \frac{\Pr[X \in f^{-1}(S)]}{\Pr[Y \in f^{-1}(S)]} \leq \max_{T\subseteq U}\left\{\frac{\Pr[X \in T]}{\Pr[Y \in T]}\right\} = \frac{\Pr[X \in T^*]}{\Pr[Y \in T^*]}.$$

By definition of the probability density function, we have

$$\frac{\Pr[X \in T^*]}{\Pr[Y \in T^*]} = \frac{\int_{T^*} g_X(x)dx}{\int_{T^*} g_Y(x)dx} \leq \frac{\delta\int_{T^*} g_Y(x)dx}{\int_{T^*} g_Y(x)dx} = \delta.$$

We remark that the statement can easily be extended to any randomised function f. This means that for any algorithm $\mathcal{A}$, the success probability of $\mathcal{A}$ on X differs from the success probability of $\mathcal{A}$ on Y by at most a multiplicative factor of $\frac{1}{\delta}$. We will use Lemmas 2 and 3 in Section 3.2 for the reduction from generised-LWE to standard search-LWE.

2.4 The Smoothing Parameter

For any lattice $\mathcal{L}$, one can show that $\rho_t(\mathcal{L})$ converges for all $t > 0$. In particular, the map $s \mapsto \rho_{1/s}(\mathcal{L}\setminus\{\mathbf{0}\})$ is a strictly decreasing continuous map on $\mathbb{R}_+$, that satisfies $\lim_{s\to\infty}\{\rho_{1/s}(\mathcal{L}\setminus\{\mathbf{0}\})\} = 0$ and $\lim_{s\to 0}\{\rho_{1/s}(\mathcal{L}\setminus\{\mathbf{0}\})\} = \infty$. This enables us to define the following parameter.

Definition 14. (Smoothing Parameter) *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice and $\varepsilon > 0$. The* smoothing parameter *of $\mathcal{L}$ with respect to ε is*

$$\eta_\varepsilon(\mathcal{L}) := \inf\{s \in \mathbb{R}_+ \mid \rho_{\frac{1}{s}}(\mathcal{L}^*\setminus\{\mathbf{0}\}) \le \varepsilon\}.$$

By the above observation on the map $s \mapsto \rho_{1/s}(\mathcal{L}\setminus\{\mathbf{0}\})$, the infimum in the definition above can be achieved with equality. In fact, $s \mapsto \rho_{1/s}(L \setminus \{\mathbf{0}\})$ is a bijection from $\mathbb{R}_+$ to $\mathbb{R}_+$ with inverse $\varepsilon \mapsto \eta_\varepsilon(\mathcal{L})$.

Observe that, using the properties of the Gaussian function and the fact that $(p\,\mathcal{L})^* = p^{-1}\,\mathcal{L}^*$, any scaling of the smoothing parameter can be rewritten as

$$\begin{aligned} p \cdot \eta_\varepsilon(\mathcal{L}) &= \inf\{ps \in \mathbb{R}_+ \mid \rho_{\frac{1}{s}}(\mathcal{L}^*\setminus\{\mathbf{0}\}) \le \varepsilon\} \\ &= \inf\{s' \in \mathbb{R}_+ \mid \rho_{\frac{p}{s'}}(\mathcal{L}^*\setminus\{\mathbf{0}\}) \le \varepsilon\} \\ &= \inf\{s' \in \mathbb{R}_+ \mid \rho_{\frac{1}{s'}}(p^{-1}\,\mathcal{L}^*\setminus\{\mathbf{0}\}) \le \varepsilon\} \\ &= \eta_\varepsilon(p\,\mathcal{L}). \end{aligned}$$

The following upper and lower bounds on the smoothing parameter will be used in our reduction.

Lemma 4. *(Claim 2.13 from [Reg09]) For any n-dimensional lattice $\mathcal{L}$ and $\varepsilon \in \mathbb{R}_+$ we have*

$$\eta_\varepsilon(\mathcal{L}) \ge \sqrt{\frac{1}{\pi}\ln\left(\frac{1}{\varepsilon}\right)} \cdot \frac{1}{\lambda_1(\mathcal{L}^*)} \ge \sqrt{\frac{1}{\pi}\ln\left(\frac{1}{\varepsilon}\right)} \cdot \frac{\lambda_n(\mathcal{L})}{n}.$$

Lemma 5. *(Lemma 3.1 from [GPV08], adapted) For any n-dimensional lattice $\mathcal{L}$ with basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and $\varepsilon \in \mathbb{R}_+$ we have*

$$\eta_\varepsilon(\mathcal{L}) \le \max_{i\in[n]}\{\|\mathbf{b}_i\|\} \cdot \sqrt{\frac{1}{\pi}\ln\left(2n\left(1+\frac{1}{\varepsilon}\right)\right)}.$$

We remark that the original Lemma 3.1 in [GPV08] is tighter, as the maximum is over the Gram-Schmidt orthogonolization of the basis vectors, $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_m$, which satisfy $\|\tilde{\mathbf{b}}_i\| \le \|\mathbf{b}_i\|$ for all $i \in [n]$. The original statement takes the minimum of this maximum norm over all possible bases $\mathbf{B}$ of the lattice.

The following is an elementary bound on the shortest vector length in the dual lattice.

Lemma 6. *(Theorem 3.2 from [Cai98], adapted) For any n-dimensional lattice $\mathcal{L}$ with basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$,*

$$\frac{1}{\lambda_1(\mathcal{L}^*)} \le \max_{i\in[n]}\{\|\mathbf{b}_i\|\}.$$

As in the previous lemma, the statement above is weaker than the original statement in [Cai98], as it uses the weaker bound given by $\|\tilde{\mathbf{b}}_i\| \le \|\mathbf{b}_i\|$ for all $i \in [n]$, instead of the smallest maximum length of the Gram-Schmidt orthogonolization of the basis vectors taken over all possible bases of the lattice. For our purposes, it suffices to take the weaker versions of these bounds stated in the lemmas above.

2.5 Computational Lattice Problems

Definition 15. (SVP) *Let $\gamma \ge 1$. The* γ-approximate Shortest Vector Problem *in the Euclidean norm,* GapSVP_γ*, is the decision problem defined as: given an instance $(\mathbf{B}, d)$ consisting of a basis matrix $\mathbf{B}$ of a rank-n lattice $\mathcal{L}$ and distance parameter $d > 0$, output*

– *YES if* $\lambda_1(\mathcal{L}) \leq d$, *or*
– *NO if* $\lambda_1(\mathcal{L}) \geq \gamma d$.

Definition 16. (BDD) *The* Bounded Distance Decoding problem, BDD_α, *is the search problem defined as: given a basis matrix* $\mathbf{B}$ *of a rank-*n *lattice* $\mathcal{L}$ *and a target vector* $\mathbf{v} \in \mathbb{R}^n$ *with the promise that* $\text{dist}(\mathbf{v}, \mathcal{L}) < \alpha \cdot \lambda_1(\mathcal{L})$, *find a lattice vector closest to* $\mathbf{v}$, *i.e. an* $\mathbf{x} \in \mathcal{L}$ *such that* $\|\mathbf{v} - \mathbf{x}\| < \alpha \cdot \lambda_1(\mathcal{L})$.

Definition 17. (mod-BDD) *Let* $p \in \mathbb{Z}_+$. *The* Modulo-p Bounded Distance Decoding problem, $\mathsf{BDD}_{\alpha,p}$, *is the search problem defined as: given a basis matrix* $\mathbf{B}$ *of a rank-*n *lattice* $\mathcal{L}$ *and a target vector* $\mathbf{v} \in \mathbb{R}^n$ *with the promise that* $\text{dist}(\mathbf{v}, \mathcal{L}) < \alpha \cdot \lambda_1(\mathcal{L})$, *find the coefficient vector of a lattice vector closest to* $\mathbf{v}$ *modulo* p. *i.e. if* $\mathbf{x} \in \mathcal{L}$ *is closest to* $\mathbf{v}$, *then the expected output is* $\mathbf{B}^{-1}\mathbf{x} \pmod p \in \mathbb{Z}_p^n$.

3 BDD to Search-LWE

We now formally state and prove our first main result.

Theorem 3. (BDD$\to$ search-LWE) *Let* $\alpha = \alpha(n) \in (0,1)$ *and* $\gamma \in (0, \frac{1}{2})$. *Suppose there exists a polynomial-time algorithm* $\mathcal{B}$ *that solves* $\mathsf{LWE}_{n,2^n,\Psi_\alpha}$ *with probability* q. *Then there is a PPT algorithm* $\mathcal{A}$ *that, given oracle access to* $\mathcal{B}$ *and a basis* $\mathbf{B}$ *for lattice* $\mathcal{L}$, *solves any* BDD_γ *instance* $(\mathcal{L}, \mathbf{x})$ *where*

$$\text{dist}(\mathcal{L}^*, \mathbf{x}) \leq \frac{\alpha\gamma}{\max_{i\in[n]}\{\|\mathbf{b}_i\|\}} \cdot \left(\frac{1}{\pi}\ln(2n(1+\frac{1}{\varepsilon}))\right)^{-\frac{1}{2}}$$

with probability $q(1+\delta)^{-3} - 6\varepsilon$ *for some* $\varepsilon \in (0, \frac{1}{24})$ *and constant* $\delta > \frac{3}{4}$.

In particular, using Conjecture 1, we assume that $q = 2^{-\kappa n^2}$. Then setting $\delta = 1$ and $\varepsilon = \frac{1}{96}q$, we obtain

$$q(1+\delta)^{-3} - 6\varepsilon = \frac{1}{8}q - \frac{1}{16}q = 2^{-4}q = 2^{-\kappa n^2 - 4} = 2^{-O(n^2)}.$$

So informally, this theorem says that if there is no efficient algorithm that solves BDD_γ with probability $2^{-O(n^2)}$, then there is no efficient algorithm for $\mathsf{LWE}_{n,2^n,\Psi_\alpha}$ that succeeds with probability $2^{-O(n^2)}$.

The proof consists of two parts and uses techniques inspired by Regev's original reduction in [Reg09]. First we give a one-shot reduction from BDD to a generalised LWE problem in Section 3.1. Then we adapt Regev's reduction from this generalised LWE problem to search-LWEwith exponential modulus in Section 3.2, using the multiplicative, rather than additive, difference in distributions. Lastly, we reduce LWE with exponential modulus to binary LWE with polynomial modulus in Section 3.3.

3.1 BDD to Generalised LWE

Consider the following generalised version of LWE, as introduced by Regev in [Reg09].

Definition 18. (Generalised LWE) *The* Generalised Learning with Errors problem, *denoted by* $\mathsf{LWE}_{n,p,\mathcal{D}}$, *is defined as: given a polynomial number of samples from the distribution* $A_{\mathbf{s},\phi}$ *with modulus* p, *where* ϕ *belongs to the family of distributions* $\mathcal{D}$, *recover the secret* $\mathbf{s} \in \mathbb{Z}_p^n$.

Note that in the definition above, any algorithm for the problem may know $\mathcal{D}$, but is not given the specific distribution $\phi \in \mathcal{D}$. Furthermore in any instance of the problem, the input samples all come from the same distribution ϕ. For our proof, we are interested in the family of distributions

$$\Psi_{\leq\alpha} := \{\Psi_\beta \mid 0 < \beta \leq \alpha\}.$$

To obtain the desired bound on the success probability, we would like to minimise the number of calls to the oracle for our target problem. In the chain of reductions $\mathsf{BDD}_\gamma \to \mathsf{BDD}_{\gamma,p} \to \mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ in [Reg09], a

total of n calls is made to the algorithm for $\mathsf{BDD}_{\gamma,p}$, each of which calls the oracle for $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ once. If we insist that the modulus is $p = 2^n$, then we can simplify our analysis by allowing the BDD_γ algorithm to call the $\mathsf{BDD}_{\gamma,p}$ oracle exactly once, so that the total number of calls to the LWE oracle is exactly one.

In our one-shot reduction from BDD_γ to $\mathsf{BDD}_{\gamma,p}$, we call the $\mathsf{BDD}_{\gamma,p}$ oracle on the given BDD_γ instance to obtain a coefficient vector that ideally corresponds to the closest lattice vector to the given target $\mathbf{v}$. We then shift the target vector $\mathbf{v}$ by this closest lattice vector and scale it down by p. Then we run Babai's nearest plane algorithm from [Bab86] on this shifted and scaled vector to find the closest lattice vector to $\mathbf{v}$ within the specified distance. Intuitively, blowing up the modulus to be exponential in the dimension n makes it easy for Babai's nearest plane algorithm to find the exact lattice point we are interested in. Now we formalise this idea.

Lemma 7. ($\mathsf{BDD} \to$ Modulo-BDD) *Let $n \in \mathbb{Z}_+, p = 2^n$, and $\gamma \in (0, \frac{1}{2})$. Suppose there is a polynomial time algorithm $\mathcal{B}$ that solves $\mathsf{BDD}_{\gamma,p}$ with success probability q. Then there exists a polynomial time algorithm $\mathcal{A}$ with oracle access to $\mathcal{B}$ that solves BDD_γ with success probability q.*

Proof. Let $(\mathbf{B}, \mathbf{v})$ be the given instance of BDD_γ. By definition, this defines a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ which satisfies $\mathrm{dist}(\mathbf{v}, \mathcal{L}) < \gamma \cdot \lambda_1(\mathcal{L})$. Consider the following algorithm $\mathcal{A}$:

Algorithm 1: BDD to Modulo-BDD Reduction

Input: BDD_γ instance $(\mathbf{B}, \mathbf{v})$.
Output: Lattice vector $\mathbf{x} \in \mathcal{L}$.

Run $\mathcal{B}$ on $(\mathbf{B}, \mathbf{v})$ to get a vector $\overline{\mathbf{z}} \in \mathbb{Z}_p^n$.
Compute $\mathbf{v}' := \frac{1}{p}(\mathbf{v} - \mathbf{B}\overline{\mathbf{z}})$.
Run Babai's Algorithm on $(\mathbf{B}, \mathbf{v}')$ to get a vector $\mathbf{B}\mathbf{z}' \in \mathcal{L}$.
Output the vector $\mathbf{B}(p\mathbf{z}' + \overline{\mathbf{z}})$.

Since the oracle $\mathcal{B}$ and Babai's Nearest Plane algorithm both run in time polynomial in the dimension of the lattice n, this algorithm clearly runs in polynomial time.

Now we prove correctness and show that if $\mathcal{B}$ answers correctly, then $\mathcal{A}$ outputs a correct answer. If oracle $\mathcal{B}$ answers correctly, it outputs $\overline{\mathbf{z}} = \mathbf{z} \pmod p$ for some coefficient vector $\mathbf{z} = \mathbf{B}^{-1}\mathbf{x} \in \mathcal{L}$, where $\mathbf{x} \in \mathcal{L}$ is a closest lattice vector to $\mathbf{v}$. This means that

$$\|\mathbf{v} - \mathbf{x}\| = \|\mathbf{v} - \mathbf{B}\mathbf{z}\| < \gamma \cdot \lambda_1(\mathcal{L}).$$

The output of $\mathcal{A}$ is correct if and only if $\left\|\mathbf{v} - \mathbf{B}(p\mathbf{z}' + \overline{\mathbf{z}})\right\| < \gamma \cdot \lambda_1(\mathcal{L})$. Since $\|\mathbf{v} - \mathbf{B}\mathbf{z}\| < \gamma \cdot \lambda_1(\mathcal{L})$, it is enough to show that $\mathbf{z} = p\mathbf{z}' + \overline{\mathbf{z}}$. Babai's nearest plane algorithm from [Bab86] guarantees that the output $\mathbf{B}\mathbf{z}'$ satisfies

$$\|\mathbf{v}' - \mathbf{B}\mathbf{z}'\| \leq 2^n \cdot \mathrm{dist}(\mathcal{L}, \mathbf{v}').$$

Observe that since $\mathbf{z} = \mathbf{B}^{-1}\mathbf{x}$ and $\mathbf{x} = \mathbf{B}\mathbf{z} \in \mathcal{L}$ is a lattice vector, $\mathbf{z} \in \mathbb{Z}^n$ must be an integer coefficient vector. By definition, $\overline{\mathbf{z}} = \mathbf{z} \pmod p \in \mathbb{Z}_p^n$ is also a coefficient vector. Combining these two facts and observing that their coordinates can only differ by a multiple of p, we obtain that $\frac{1}{p}(\mathbf{z} - \overline{\mathbf{z}}) \in \mathbb{Z}^n$ is a coefficient vector. Hence $\mathbf{B}\frac{1}{p}(\mathbf{z} - \overline{\mathbf{z}}) \in \mathcal{L}$. By definition, $\mathrm{dist}(\mathcal{L}, \mathbf{v}') \leq \left\|\mathbf{v}' - \mathbf{u}\right\|$ for any lattice vector $\mathbf{u} \in \mathcal{L}$. This yields the bound

$$\|\mathbf{v}' - \mathbf{B}\mathbf{z}'\| \leq 2^n \cdot \mathrm{dist}(\mathcal{L}, \mathbf{v}') \leq 2^n \left\|\mathbf{v}' - \mathbf{B}\frac{1}{p}(\mathbf{z} - \overline{\mathbf{z}})\right\| = 2^n \cdot \frac{1}{p}\|\mathbf{v} - \mathbf{B}\mathbf{z}\| < \gamma \cdot \lambda_1(\mathcal{L}),$$

where the last inequality above follows from the assumption that $p = 2^n$. Thus, the unique closest vector to $\mathbf{v}'$ is in fact $\mathbf{B}\frac{1}{p}(\mathbf{z} - \overline{\mathbf{z}})$. Therefore, $\mathbf{z}' = \frac{1}{p}(\mathbf{z} - \overline{\mathbf{z}})$, so we obtain $\mathbf{z} = p\mathbf{z}' + \overline{\mathbf{z}}$ as desired. Since the oracle $\mathcal{B}$ is correct with probability q and the algorithm $\mathcal{A}$ always answers correctly in this case, the overall success probability of algorithm $\mathcal{A}$ is at least q.

□

Note that in order for the reduction above to work, we need the modulus to be exponential $p = 2^n$. We will reduce this exponential modulus to a polynomial one in Section 3.3. Before we proceed to the second reduction, we introduce the following intermediate results.

Lemma 8. *(Claim 3.8 from [Reg09]) Let $\mathcal{L}$ be a rank-n lattice, $\mathbf{c} \in \mathbb{R}^n$, and $\varepsilon > 0$. For any $r \geq \eta_\varepsilon(\mathcal{L})$,*

$$\rho_r(\mathcal{L}+\mathbf{c}) \in (r^n \det(\mathcal{L}^*)(1-\varepsilon), r^n \det(\mathcal{L}^*)(1+\varepsilon)).$$

This bounds the Gaussian function of any lattice coset by the determinant of the dual lattice. The following lemma bounds the statistical distance between two relevant distributions.

Lemma 9. *(Corollary 3.10 from [Reg09]) Let $\mathcal{L}$ be a rank-n lattice, $\mathbf{w}, \mathbf{v} \in \mathbb{R}^n$ be vectors, and $r, s \in \mathbb{R}_+$. Define $t := \sqrt{(r\|\mathbf{w}\|)^2 + s^2}$. Suppose that for some $\varepsilon \in (0, \frac{1}{2})$*

$$\eta_\varepsilon(L) \leq \frac{1}{\sqrt{\frac{1}{r^2} + \left(\frac{1}{s}\|\mathbf{w}\|\right)^2}}.$$

Define the random variable $X := \langle \mathbf{w}, \mathbf{v} \rangle + e$, where the distribution of $\mathbf{v}$ is $\mathbf{v} \sim D_{L+\mathbf{u},r}$ and $e \sim N\left(0, \frac{s^2}{2\pi}\right)$, and let Φ denote the distribution of X modulo 1. Also let $Z \sim N\left(0, \frac{t^2}{2\pi}\right)$. Then $\Delta(X, Z) \leq 4\varepsilon$, and hence $\Delta\left(\Phi, \Psi_t\right) \leq 4\varepsilon$.

Now we implement the second reduction. Our algorithm requires additional data, namely samples from a discrete Gaussian. We generate these samples using a subroutine from [BLP+13] as a black-box. In [BLP+13], the authors give an efficient algorithm that, for any lattice and sufficiently large width, outputs a sample from a discrete Gaussian distribution. Formally, they prove the following.

Lemma 10. *(Theorem 2.3 from [BLP+13], adapted) There exists a PPT algorithm* DGS *that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, a vector $\mathbf{c} \in \mathbb{R}^n$, and a parameter*

$$r \geq \max_{i \in [n]}\{\|\tilde{\mathbf{b}}_i\|\} \cdot \sqrt{\frac{\ln(2n+4)}{\pi}},$$

outputs a sample with distribution $D_{\mathcal{L}+c,r}$.

Lemma 11. (Modulo-BDD $\rightarrow$ Generalised-LWE) *Let $\varepsilon = \varepsilon(n) \in (0, \frac{1}{24})$, $q = q(n)$, $\alpha = \alpha(n) \in (0,1)$, $p = p(n) \in \mathbb{Z}_+$, $\gamma \in (0, \frac{1}{2})$, and $k \in \mathbb{Z}_+$ be a constant. Suppose there is a polynomial time algorithm $\mathcal{B}$ that, given n^k samples from $A_{s,\Psi_{\leq\alpha}}$, solves $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ with success probability q. Then there is a PPT algorithm $\mathcal{A}$ with oracle access to $\mathcal{B}$ that, given $(\mathbf{B}^*, \mathbf{x})$ corresponding to a lattice $\mathcal{L}^* = \mathcal{L}(\mathbf{B}^*)$ such that $\mathrm{dist}(\mathcal{L}^*, \mathbf{x}) \leq \frac{\alpha p\sqrt{\gamma}}{r}$ for some*

$$r > \frac{p}{\sqrt{\gamma}} \cdot \max_{i \in [n]}\{\|\mathbf{b}_i\|\} \cdot \sqrt{\frac{1}{\pi} \ln\left(2n\left(1+\frac{1}{\varepsilon}\right)\right)},$$

solves $\mathsf{BDD}_{\gamma,p}$ with success probability $q - 6\varepsilon$.

Proof. Let $(\mathbf{B}^*, \mathbf{x})$ be the given $\mathsf{BDD}_{\gamma,p}$ instance. By definition, this defines a lattice $\mathcal{L}^* = \mathcal{L}(\mathbf{B}^*) = (\mathcal{L}(\mathbf{B}))^*$ which is the dual lattice of $\mathcal{L} = \mathcal{L}(\mathbf{B})$. By Lemma 6, the parameter r is bounded by

$$r \geq \frac{p}{\sqrt{\gamma}\lambda_1(\mathcal{L}^*)} \cdot \sqrt{\frac{1}{\pi} \ln\left(2n\left(1+\frac{1}{\varepsilon}\right)\right)} \geq \frac{p}{\sqrt{\gamma}\lambda_1(\mathcal{L}^*)} \cdot \sqrt{\frac{1}{\pi} \ln\left(\frac{1}{\varepsilon}\right)} \geq \frac{\alpha p}{\sqrt{\gamma}\lambda_1(\mathcal{L}^*)}.$$

The last inequality here follows from the upper bound on ϵ and the assumption that $\alpha < 1$. Then for the given parameters, the distance between the given vector and lattice is bounded by

$$\text{dist}(\mathcal{L}^*, \mathbf{x}) \leq \frac{\alpha p\sqrt{\gamma}}{r} \leq \frac{\alpha p\sqrt{\gamma} \cdot \sqrt{\gamma}\lambda_1(\mathcal{L}^*)}{\alpha p} \leq \gamma \cdot \lambda_1(\mathcal{L}^*).$$

Thus, $(\mathbf{B}^*, \mathbf{x})$ is a valid instance of $\mathsf{BDD}_{\gamma,p}$.

First we define a subroutine to efficiently sample from a discrete Gaussian distribution. Note that since $\varepsilon \in (0, \frac{1}{24})$ is a constant, we have $\frac{1}{\varepsilon} \geq \frac{2}{n}$, so the bound on r satisfies

$$r \geq \frac{p}{\sqrt{\gamma}} \cdot \max_{i\in[n]}\{\|\mathbf{b}_i\|\} \cdot \sqrt{\frac{1}{\pi}\ln\left(2n\left(1+\frac{1}{\varepsilon}\right)\right)} \geq \max_{i\in[n]}\{\|\mathbf{b}_i\|\} \cdot \sqrt{\frac{1}{\pi}\ln(2n+4)}.$$

This enables us to run the DGS algorithm from Lemma 10 on this r, the lattice $\mathcal{L}$, and vector $\mathbf{c} = 0$ to generate samples with distribution $D_{\mathcal{L},r}$.

The idea for algorithm $\mathcal{A}$ is to use $\mathbf{x}$ to generate a polynomial number of samples from a distribution Φ that is a good approximation of $A_{\mathbf{s},\Psi_\beta}$, for $\mathbf{s} = (\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}) \bmod p$ and some $\beta \leq \alpha$. Recall that $\kappa_{\mathcal{L}^*}(\mathbf{x})$ denotes the unique closest vector in the lattice $\mathcal{L}^*$ to $\mathbf{x}$. We call the oracle $\mathcal{B}$ on these generated samples to obtain the secret vector $\mathbf{s}$ with probability close to q. We formally define algorithm $\mathcal{A}$ as follows.

Algorithm 2: Modulo-BDD to Generalised-LWE Reduction

Input: $(\mathbf{B}^*, \mathbf{x})$ such that $\text{dist}(\mathcal{L}^*, \mathbf{x}) \leq \frac{\alpha p\sqrt{\gamma}}{r}$.
Output: $\mathbf{s} \in \mathbb{Z}_p^n$.

for $i \in \{1, ..., n^k\}$ **do**
- Run the DGS sampler to obtain a vector $\mathbf{v} \leftarrow D_{\mathcal{L},r}$.
- Compute $\mathbf{a} := \mathbf{B}^{-1}\mathbf{v} \bmod p$.
- Sample some noise $e \leftarrow N(0, \frac{\alpha^2\gamma}{2\pi})$.
- Define $b := \frac{1}{p}\langle \mathbf{x}, \mathbf{v}\rangle + e \bmod 1$.
- Define sample $X_i := (\mathbf{a}, b)$.

end
Run $\mathcal{B}$ on $X_1, ..., X_{n^k} \sim \Phi$ to get a vector $\mathbf{s} \in \mathbb{Z}_p^n$.
Output $\mathbf{s}$.

Since this sampling process is efficient and repeated a polynomial number of times, the overall algorithm is efficient.

Now we prove the correctness of algorithm $\mathcal{A}$. We claim that if $\mathcal{B}$ succeeds, $\mathcal{A}$ generates a good approximation of samples from $A_{\mathbf{s},\Psi_\beta}$. Specifically, we show that the statistical distance between the distributions Φ and $A_{\mathbf{s},\Psi_\beta}$ is ε' for some $\beta \leq \alpha$. Given a polynomial number of samples from $A_{\mathbf{s},\Psi_\beta}$, the oracle $\mathcal{B}$ is guaranteed to find $\mathbf{s}$ with probability q. If the oracle succeeds, its output is $\mathbf{s} = (\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}) \bmod p$, which is precisely the coefficient vector of the closest lattice vector $\mathbf{x} \in \mathcal{L}^*$ modulo p. Hence it is a solution for the given $\mathsf{BDD}_{\gamma,p}$ instance. Since the samples input to $\mathcal{B}$ are from an approximate distribution Φ that is ε' away in statistical distance from the true distribution $A_{\mathbf{s},\Psi_\beta}$, then by $(*)$ the success probability suffers a loss of ε'. Hence, the algorithm $\mathcal{A}$ will succeed with probability $q - \varepsilon'$.

We prove our claim by analysing the distributions of $\mathbf{a}$ and b for any generated sample X_i. First we show that the distribution of $\mathbf{a} \in \mathbb{Z}_p^n$ is close to uniform. Let $\mathcal{Y}$ denote the distribution of $\mathbf{a}$ produced in the algorithm. Fix $\mathbf{a} \in \mathbb{Z}_p^n$. Then the probability that $\mathcal{Y}$ takes the value $\mathbf{a}$ is

$$\Pr[\mathcal{Y} = \mathbf{a}] = \Pr_{\mathbf{v}\leftarrow D_{\mathcal{L},r}}[\mathbf{v} = \mathbf{Ba} \bmod p] = \frac{\rho_r(p\,\mathcal{L} + \mathbf{Ba})}{\rho_r(\mathcal{L})} = \frac{\rho_{\frac{r}{p}}\left(\mathcal{L} + \frac{1}{p}\mathbf{Ba}\right)}{\rho_r(\mathcal{L})},$$

by definition of the discrete Gaussian. By Lemma 5, we have $r > p\sqrt{2}\cdot\eta_\varepsilon(\mathcal{L})$. Then since $\eta_\varepsilon(\mathcal{L}) < r$, Lemma 8 implies

$$\frac{\rho_{\frac{r}{p}}\left(\mathcal{L}+\frac{1}{p}\mathbf{Ba}\right)}{\rho_r(\mathcal{L})} \in \frac{\frac{r^n}{p^n}\det(\mathcal{L}^*)(1\pm\varepsilon)}{r^n\det(\mathcal{L}^*)(1\pm\varepsilon)} = \frac{1}{p^n}\left(1-\frac{2\varepsilon}{1+\varepsilon}, 1+\frac{2\varepsilon}{1-\varepsilon}\right).$$

Then the statistical distance between $\mathcal{Y}$ and the uniform distribution $\mathcal{U}$ over $\mathbb{Z}_p^n$ is bounded by

$$\begin{aligned}
\Delta(\mathcal{Y},\mathcal{U}) &= \frac{1}{2}\sum_{\mathbf{a}\in\mathbb{Z}_p^n}\left|\Pr[\mathcal{Y}=\mathbf{a}] - \Pr[\mathcal{U}=\mathbf{a}]\right| \\
&\le \frac{1}{2}\left(\rho p^n\left(\frac{1}{p^n}\left(1+\frac{2\varepsilon}{1-\varepsilon}\right)-\frac{1}{p^n}\right)\right. \\
&\qquad \left. + (1-\rho)p^n\left(\frac{1}{p^n}\left(1-\frac{2\varepsilon}{1+\varepsilon}\right)-\frac{1}{p^n}\right)\right) \\
&\le \max_{\rho\in[0,1]}\left\{\rho\frac{\varepsilon}{1-\varepsilon}+(1-\rho)\frac{\varepsilon}{1+\varepsilon}\right\} \\
&\le \frac{\varepsilon}{1-\varepsilon}.
\end{aligned}$$

Here $\rho\in[0,1]$ is the fraction of values in $\mathbb{Z}_p^n$ for which $\Pr[\mathcal{Y}] > \Pr[\mathcal{U}]$. Since $\varepsilon\in(0,\frac{1}{24})$, we have $\Delta(\mathcal{Y},\mathcal{U}) \le 2\varepsilon$.

Now we show that the distribution of the second component b of the sample X_i is close to the corresponding LWE distribution. We condition on $\mathbf{a}$ and consider the marginal distribution of b. Define $\mathbf{x}' := \mathbf{x} - \kappa_{\mathcal{L}^*}(\mathbf{x})$. By construction, we have $\|\mathbf{x}'\| \le \operatorname{dist}(\mathcal{L}^*,\mathbf{x}) \le \frac{\alpha p\sqrt{\gamma}}{r}$. Then we can write

$$\frac{1}{p}\left\langle\mathbf{x},\mathbf{v}\right\rangle + e = \frac{1}{p}\left\langle\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{v}\right\rangle + \frac{1}{p}\left\langle\mathbf{x}',\mathbf{v}\right\rangle + e. \tag{$**$}$$

Observe that since $(\mathbf{B}^*)^{\mathrm{T}} = \mathbf{B}^{-1}$, we can write

$$\begin{aligned}
\left\langle\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{v}\right\rangle &= \kappa_{\mathcal{L}^*}(\mathbf{x})^{\mathrm{T}}\mathbf{B}\mathbf{B}^{-1}\mathbf{v} \\
&= \kappa_{\mathcal{L}^*}(\mathbf{x})^{\mathrm{T}}((\mathbf{B}^*)^{-1})^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{v} \\
&= ((\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}))^{\mathrm{T}}(\mathbf{B}^{-1}\mathbf{v}) \\
&= \left\langle(\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{B}^{-1}\mathbf{v}\right\rangle.
\end{aligned}$$

By construction, we have

$$\left\langle\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{v}\right\rangle = \left\langle(\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{B}^{-1}\mathbf{v}\right\rangle \equiv \langle\mathbf{s},\mathbf{a}\rangle \bmod p,$$

so the first term in ($**$) satisfies

$$\frac{1}{p}\left\langle\kappa_{\mathcal{L}^*}(\mathbf{x}),\mathbf{v}\right\rangle \equiv \frac{1}{p}\langle\mathbf{a},\mathbf{s}\rangle \bmod 1.$$

It remains to consider the second term in the expression ($**$). Note that conditioned on $\mathbf{a}$ and since p is fixed, the distribution of $\mathbf{v}$ is the same as the distribution $D_{p\,\mathcal{L}+\mathbf{Ba},r}$. Let $\mathcal{Z}$ denote the distribution of $\frac{1}{p}\left\langle\mathbf{x}',\mathbf{v}\right\rangle + e \bmod 1$ for $\mathbf{v}$ sampled from $D_{p\,\mathcal{L}+\mathbf{Ba},r}$. By construction, e is sampled according to $N\left(0,\frac{\alpha^2\gamma}{2\pi}\right)$. Since $\left\|\frac{1}{p}\mathbf{x}'\right\| \le \frac{\alpha\sqrt{\gamma}}{r}$, we obtain

$$\frac{1}{\sqrt{\frac{1}{r^2}+\left(\frac{1}{\alpha\sqrt{\gamma}}\left\|\frac{1}{p}\mathbf{x}'\right\|\right)^2}} \ge \frac{1}{\sqrt{\frac{1}{r^2}+\frac{1}{r^2}}} = \frac{r}{\sqrt{2}} > r\sqrt{\gamma} \ge p\cdot\eta_\varepsilon(\mathcal{L}) = \eta_\varepsilon(p\,\mathcal{L}).$$

Here the last equality follows from rewriting the scaled smoothing parameter (see subsection 2.4 for the proof). Then by Lemma 9, $\Delta(\mathcal{Z}, \Psi_\beta) \leq 4\varepsilon$, for

$$\beta := \sqrt{\left(r\left\|\frac{1}{p}\mathbf{x}'\right\|\right)^2 + \left(\alpha\sqrt{\gamma}\right)^2} \leq \sqrt{\alpha^2\gamma + \alpha^2\gamma} = \alpha\sqrt{2\gamma} < \alpha.$$

Therefore, by the triangle inequality, the statistical distance between Φ and $A_{\mathbf{s},\Psi_\beta}$ is $\varepsilon' = 2\varepsilon + 4\varepsilon = 6\varepsilon$ for some $\beta \leq \alpha$, as claimed.

□

An immediate corollary of this one-shot reduction is the following bound on the success probability of any polynomial-time algorithm for BDD.

Corollary 1. (BDD $\to$ Generalised-LWE) *Suppose there exists an polynomial-time algorithm $\mathcal{A}$ that solves $\mathsf{LWE}_{n,p,\phi}$, where ϕ is an unknown distribution from the family $\Psi_{\leq\alpha}$, with success probability $q = q(n)$. Then there is a polynomial-time algorithm $\mathcal{B}$ that, given oracle access to $\mathcal{A}$, solves BDD_γ for gap $\gamma \in (0, \frac{1}{2})$ with probability $q - 6\varepsilon$ for some sufficiently small $\varepsilon = \varepsilon(n) \in (0, \frac{1}{24})$.*

Note that the additive loss in success probability can be written as a multiplicative factor:

$$q - \varepsilon' = q\left(1 - \frac{\varepsilon'}{q}\right) = q\left(1 - \frac{6\varepsilon}{q}\right).$$

This probability only makes sense for $0 < q - 6\varepsilon < 1$, which holds if $\epsilon \in \left(\frac{q-1}{6}, \frac{q}{6}\right)$. To obtain a small loss, say $q - \varepsilon' = \frac{q}{2}$, we would need $\epsilon = \frac{q}{12}$. For our application, we are interested in the regime where $q = 2^{-O(n^2)}$, so taking an appropriate ϵ such as this, we can obtain a constant multiplicative loss in success probability.

3.2 Generalised LWE to Standard LWE

In this section, we give a reduction from generalised LWE to LWE by adapting Lemma 3.7 from [Reg09] to work with multiplicative, rather than additive, loss in success probability. In Regev's reduction, we iteratively choose some Gaussian noise from a discrete interval to obtain some optimal noise that guarantees an overwhelming success probability. Since we are concerned with polynomial-time adversaries and the success probability itself, unlike the original reduction, we only sample noise from the interval exactly once. Because we are limited to a single Gaussian noise sample, choosing the interval and parameters requires considerable care.

Lemma 12. (Generalised-LWE $\to$ Search-LWE) *Let $\alpha \in \mathbb{R}_+$, $p = p(n) \in \mathbb{Z}_+$, and $\varepsilon > \frac{3}{4}$ be a constant parameter. Suppose there is an efficient algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,p,\Psi_\alpha}$ with success probability q. Then there is a PPT algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ with success probability at least $\frac{q}{(1+\varepsilon)^3}$.*

Proof. Suppose that $\mathcal{A}$ is given n^k samples for some constant $k \in \mathbb{Z}_+$, distributed according to $A_{\mathbf{s},\Psi_\beta}$, for some $\beta \leq \alpha$. For notational convenience, let $\delta := (1+\varepsilon)^2 - 1 = \varepsilon^2 + 2\varepsilon$ and define

$$Z := \left\{0, \delta\alpha^2, 2\cdot\delta\alpha^2, \ldots, \lfloor\delta\rfloor\delta\alpha^2\right\}$$

to be the set of integer multiples of $\delta\alpha^2$ between 0 and α^2. Consider the following algorithm $\mathcal{A}$:

Algorithm 3: Generalised-LWE to Search-LWE Reduction

Input: Samples $X_1, ..., X_{n^k} \sim A_{\mathbf{s},\Psi_\beta}$.
Output: Secret vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Sample $\gamma \leftarrow Z$ uniformly at random.
for $i \in \{1, ..., n^k\}$ **do**
  Denote $(\mathbf{a}, b) := X_i$.
  Sample some noise $e \leftarrow \Psi_{\sqrt{\gamma}}$.
  Define $Y_i := (\mathbf{a}, b + e)$.
end
Run $\mathcal{B}$ on the generated samples $Y_1, ..., Y_{n^k}$ to get a vector $\mathbf{s}' \in \mathbb{Z}_p^n$.
Output $\mathbf{s}'$.

Since sampling and transforming n^k samples is efficient, and the oracle $\mathcal{B}$ is called once, $\mathcal{A}$ is efficient.

Now we prove correctness of $\mathcal{A}$. The algorithm is given samples of the form $X_i = (\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where e has distribution Ψ_β for some unknown $\beta \leq \alpha$. The algorithm knows the value of α, so it attempts to add noise from $\Psi_{\sqrt{\gamma}}$ in such a way as to obtain samples with noise distribution Ψ_α. In this way, it generates samples of the form $Y_i = (\mathbf{a}, b + e) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e' + e)$ where the noise $e' + e$ has distribution Ψ_σ for $\sigma := \sqrt{\beta^2 + \gamma}$. The error between the noise distribution Ψ_σ generated by $\mathcal{A}$ and the target distribution Ψ_α is determined by the given error parameter ε. Let γ' be the smallest element of Z satisfying $\gamma' \geq \alpha^2 - \beta^2$. Then by construction of Z, we have $\gamma' \leq \alpha^2 - \beta^2 + \delta\alpha^2 = (\delta + 1)\alpha^2 - \beta^2$. By definition, since $\varepsilon > \frac{3}{4}$, we have $\delta > 1$. Together with the fact that $0 < \beta \leq \alpha$, this implies that $0 < \alpha^2 - \beta^2 < (\delta+1)\alpha^2 - \beta^2 < \lfloor \delta \rfloor \delta\alpha^2$. Hence, there exists such an element γ' in Z. There are $|Z| = \frac{\lfloor \delta \rfloor \delta\alpha^2}{\delta\alpha^2} \leq \delta$ elements in Z, so the element γ sampled by the algorithm is $\gamma = \gamma'$ with probability at least $\frac{1}{\delta}$. Let $\sigma' := \sqrt{\beta^2 + \gamma'}$ denote the noise distribution parameter for this special element γ'. Consider the ratio of the probability generating functions corresponding to Ψ_α and $\Psi_{\sigma'}$. Using Lemma 2 and the bounds on γ', this is given by

$$\frac{g_\alpha(x)}{g_{\sigma'}(x)} \leq \frac{\sigma'}{\alpha} = \frac{\sqrt{\beta^2 + \gamma'}}{\alpha} \leq \frac{\sqrt{\beta^2 + (\delta+1)\alpha^2 - \beta^2}}{\alpha} = \frac{\alpha\sqrt{1+\delta}}{\alpha} = \sqrt{(1+\varepsilon)^2} = 1 + \varepsilon.$$

Then by Lemma 3, applying any function to this ratio of probability distribution functions cannot increase the ratio. This implies that the success probability of $\mathcal{A}$ for noise distribution Ψ_α and the success probability of $\mathcal{A}$ for noise distribution $\Psi_{\sigma'}$ have the ratio

$$\frac{\Pr[\mathcal{A} \text{ succeeds for } \Psi_\alpha]}{\Pr[\mathcal{A} \text{ succeeds for } \Psi_{\sigma'}]} = \frac{q}{\Pr[\mathcal{A} \text{ succeeds for } \gamma = \gamma']} \leq 1 + \varepsilon.$$

Hence, for this choice $\gamma = \gamma'$, we know that $\mathcal{A}$ successfully outputs $\mathbf{s}' = \mathbf{s}$ with probability at least $\frac{q}{1+\varepsilon}$. Therefore the overall success probability of $\mathcal{A}$ is at least

$$\Pr[\mathcal{A} \text{ succeeds for } \gamma = \gamma'] \cdot \Pr[\gamma = \gamma'] \geq q \cdot \frac{1}{1+\varepsilon} \cdot \frac{1}{\delta} \geq \frac{q}{(1+\varepsilon)^3}.$$

□

Assuming the success probability of solving $\mathsf{LWE}_{n,p,\Psi_\alpha}$ is $q = p^{-cn}$ for modulus $p = 2^n$, and setting the error parameter to be $\varepsilon = 1$, we obtain the following corollary.

Corollary 2. (Generalised-LWE → Search-LWE) *Suppose there is no efficient algorithm $\mathcal{A}$ for $\mathsf{LWE}_{n,2^n,\Psi_{\leq\alpha}}$ with success probability 2^{-cn^2-3} for some constant $c > 0$. Then there is no efficient algorithm $\mathcal{B}$ for $\mathsf{LWE}_{n,2^n,\Psi_\alpha}$ with success probability 2^{-cn^2}.*

3.3 Reducing the Modulus for Search-LWE

In [BLP+13], Brakerski et al. study the trade-off between the modulus and dimension of decision-LWE instances. In particular, they give a reduction from decision-LWE to decision-LWE that reduces the modulus arbitrarily while preserving the dimension and incurring only a small loss in advantage. Their reduction can also be viewed as a search to search reduction that says the following.

Theorem 4. *(Theorem 4.1. from [BLP+13], rephrased) Let $n \in \mathbb{Z}_+$ and $\alpha = \alpha(n) \in (0,1)$ such that $\frac{1}{\alpha}$ is bounded by a polynomial in n. Then for some prime $p = p(n)$ such that both p and $\frac{p}{\alpha}$ are $n^{\Theta(1)}$, there is a polynomial-time, one-shot reduction from $\mathsf{LWE}_{n,2^n,\Psi_\alpha}$ to $\mathsf{binLWE}_{n^2,p,\Psi_\alpha}$ that preserves the success probability.*

Using the trivial reduction from binLWE to LWE for the same dimension, modulus, and noise distribution, this result allows us to reduce the modulus from exponential to polynomial in n for LWE. For completeness, we include this simple reduction below.

Lemma 13. ($\mathsf{binLWE}_{n,p,\phi} \to \mathsf{LWE}_{n,p,\phi}$) *Suppose there exists an efficient algorithm $\mathcal{B}$ that solves $\mathit{LWE}_{n,p,\phi}$ with success probability q. Then there exists a PPT algorithm $\mathcal{A}$ that solves $\mathit{binLWE}_{n,p,\phi}$ with success probability q.*

Proof. Let $A_{\mathbf{s},\phi}$ be the input distribution for the given $\mathsf{binLWE}_{n,p,\phi}$ samples, where $\mathbf{s} \in \{0,1\}^n$ is a binary secret vector. Consider the following algorithm $\mathcal{A}$:

Algorithm 4: binLWEto LWE Reduction

Input: Samples $X_1, ..., X_{n^k} \sim A_{\mathbf{s},\phi}$.
Output: Secret vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Sample a vector $\mathbf{r} \leftarrow \mathbb{Z}_p^n$ uniformly at random.
for $i \in \{1, ..., n^k\}$ **do**
  Denote $(\mathbf{a}, b) := X_i$.
  Define $Y_i := (\mathbf{a}, b + \langle \mathbf{a}, \mathbf{r} \rangle)$.
end
Run $\mathcal{B}$ on the generated samples $Y_1, ..., Y_{n^k}$ to get a vector $\mathbf{s}' \in \mathbb{Z}_p^n$.
Output $\mathbf{s}' - \mathbf{r}$.

This algorithm transforms a polynomial number of samples and the efficient oracle $\mathcal{B}$ is called once, so $\mathcal{A}$ is efficient. Now we prove correctness. Observe that each sample X_i has $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ for some noise e with distribution ϕ, so the transformed samples have the form

$$Y_i = (\mathbf{a}, b + \langle \mathbf{a}, \mathbf{r} \rangle) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \langle \mathbf{a}, \mathbf{r} \rangle + e) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} + \mathbf{r} \rangle + e).$$

The oracle $\mathcal{B}$ succeeds in recovering the secret vector $\mathbf{s}' = \mathbf{s} + \mathbf{r}$ with probability q, so with the same probability $\mathcal{A}$ outputs the secret binary vector $\mathbf{s} = \mathbf{s}' - \mathbf{r}$.

□

4 Search-LWE to Decision-LWE

In this section, we show how to solve search-LWE given an oracle for decision-LWE, under the condition that the oracle correctly responds YES far more often than it incorrectly responds YES. Formally, show the following result.

Theorem 5. (Search-LWE→ Decision-LWE) *Let $n, p, k \in \mathbb{Z}_+$ be such that $p > 10$ is polynomial in n and k is a constant. Suppose that there exists an efficient algorithm $\mathcal{B}$ for decision-$\mathit{LWE}_{n,p,\phi}$ that,*

- *given n^k LWE samples from $A_{\mathbf{s},\phi}$, outputs YES with probability γ,*
- *given n^k random samples from $\mathbb{Z}_p^n \times \mathbb{T}$, outputs YES with probability δ,*

where $\gamma > 5p^2\delta$. Then there is an algorithm $\mathcal{A}$ for search-$\mathsf{LWE}_{n,p,\phi}$ with oracle access to $\mathcal{B}$ that runs in expected polynomial time and

- *outputs a correct answer with probability $\frac{1}{5p^3}\gamma$ and*
- *outputs $\perp$ with probability $1 - \frac{1}{5p^3}\gamma$.*

We prove this by making the following key observation: If solving search-LWE is hard, then it is hard to determine the secret vector $\mathbf{s}$ from a given polynomial number of LWE samples of the form $(\mathbf{a}, \mathbf{b} = \frac{1}{p}\langle \mathbf{a}, \mathbf{s}\rangle + e)$. Intuitively, this means that the function $f_\phi = (\mathbf{A}, \frac{1}{p}\mathbf{A}\mathbf{s} + \mathbf{e})$ defined by these samples is hard to invert, so it can be viewed as a one-way function. In their seminal work, Goldreich and Levin show how to construct a hard-core predicate from any one-way function [GL89]. This tells us that if we can find the inner product of $\mathbf{s}$ and a given vector $\mathbf{r}$, then we can recover the secret vector $\mathbf{s}$.

Inspired by this connection, we define an intermediate problem we call the *Goldreich-Levin Learning with Errors* (GL-LWE) problem. We reduce search-LWE with polynomial modulus to GL-LWE, then reduce this problem to standard decision-LWE under a reasonable condition.

4.1 Search-LWE to GL-LWE

In [GL89], Goldreich and Levin showed that for any one-way function f, the function $b(\mathbf{x}, \mathbf{z}) := \langle \mathbf{x}, \mathbf{z}\rangle \bmod 2$ is a hard-core predicate for the function $g(\mathbf{x}, \mathbf{z}) := (f(\mathbf{x}), \mathbf{z})$. Levin later improved this result in [Lev12] and showed that the success probability of finding the hard-core predicate is determined by the success probability of inverting the one-way function f. For the formal statement and full proof of Levin's result, we refer the reader to Appendix A.

We generalise Levin's result from modulus 2 to modulus $p > 10$ using the natural generalisation of a hard-core predicate for $\mathbb{Z}_p$, under a certain condition.

Lemma 14. *Let $n, p \in \mathbb{Z}_+$ such that $p > 10$ is polynomial in n and let $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ be an injective one-way function. Suppose there is an efficient algorithm $\mathcal{B}$ that, given $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in \mathbb{Z}_p^n$ and random $\mathbf{r} \in \mathbb{Z}_p^n$, guesses $\langle \mathbf{x}, \mathbf{r}\rangle \bmod p$*

- *correctly with probability $\alpha\beta$,*
- *incorrectly with probability $\alpha(1 - \beta)$, and*
- *outputs $\perp$ with probability $1 - \alpha$,*

where $\beta > 1 - \frac{1}{5p}$. The probability is taken over the randomness of $\mathbf{r}$, and the randomness of the algorithm. Then there is an algorithm $\mathcal{A}$ that runs in expected polynomial time, that given oracle access to $\mathcal{B}$ and $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in \mathbb{Z}_p^n$, finds $\mathbf{x}$ correctly with probability $\frac{1}{5p^2}\alpha\beta$ and outputs $\perp$ with probability $1 - \frac{1}{5p^2}\alpha\beta$.

Proof. Let $\zeta := e^{\frac{2\pi i}{p}}$ denote a primitive p-th root of unity. Note that the group $\mathbb{Z}_p$ is isomorphic to the multiplicative group $\{\zeta, \zeta^2, ..., \zeta^{p-1}\}$. Then without loss of generality, the behaviour of the oracle $\mathcal{B}$ is equivalent to that of an oracle that outputs ζ^a instead of $a \in \mathbb{Z}_p$ and outputs 0 instead of $\perp$. For a fixed input $\mathbf{y} = f(\mathbf{x})$ that is clear from the context, we use the shorthand notation $\mathcal{B}(\mathbf{r})$ for $\mathcal{B}(\mathbf{y}, \mathbf{r})$.

Note that the condition on β requires the oracle $\mathcal{B}$ to be correct with high probability when it does not output $\perp$. Moreover, since $p = \text{poly}(n)$, this β approaches 1 as n increases. Intuitively, this means that $\mathcal{B}$ admits defeat and outputs $\perp$ far more often than it guesses the answer incorrectly.

Consider the following algorithm $\mathcal{A}$:

Algorithm 5: OWF to HCP Reduction

Input: $\mathbf{y} = f(\mathbf{x}) \in \mathbb{Z}_p^n$ for some $\mathbf{x} \in \mathbb{Z}_p^n$.
Output: $\mathbf{x}' \in \mathbb{Z}_p^n$ or $\perp$.

Flip ℓ coins until a 0 is obtained. If $\ell > 2n$, abort and output $\perp$.
Set $k := \ell + \lceil \log_p(4n) \rceil$.
Sample a random matrix $\mathbf{R} \leftarrow \mathbb{Z}_p^{n \times k}$.
for $\mathbf{z} \in \mathbb{Z}_p^k$ **do**
  for $i \in \{1, ..., n\}$ **do**
    Define $g_i(\mathbf{u}) := \mathcal{B}(\mathbf{Ru} + \mathbf{e}_i)$.
    Run FFT on g_i to compute $h_i(\mathbf{z}) := \sum_{\mathbf{u} \in \mathbb{Z}_p^k \setminus \{0\}} \zeta^{-\langle \mathbf{z}, \mathbf{u} \rangle} g_i(\mathbf{u})$.
    if $\left|h_i(\mathbf{z})\right| = 0$ **then**
      Output $\perp$.
    end
    Normalise $h_i(\mathbf{z})$ to obtain a unit vector $h_i^* \in \mathbb{C}$.
    Find the closest p-th root of unity ζ^a to h_i^*.
    Set $x_i' := a$.
  end
  Set $\mathbf{x}' := (x_1', ..., x_n')$.
  if $f(\mathbf{x}') = \mathbf{y}$ **then**
    Output $\mathbf{x}'$.
  end
end
Output $\perp$.

Here FFT denotes the Fast Fourier Transform and $\mathbf{e}_i$ denotes the standard basis vector containing all zeros except a 1 in the i-th coordinate.

The algorithm flips ℓ coins until it obtains a 0. The probability that $\ell = \ell'$ for some fixed $\ell' \in \mathbb{Z}_+$ is $\frac{1}{2^{\ell'}}$. Then $\ell > \ell'$ with probability

$$\Pr_{\ell}[\ell > \ell'] = \left(1 - \Pr_{\ell}[\ell \leq \ell']\right) = \left(1 - \sum_{i=1}^{\ell'} \frac{1}{2^{\ell'}}\right) = \left(1 - \left(1 - \frac{1}{2^{\ell'}}\right)\right) = \frac{1}{2^{\ell'}}.$$

The parameter k is set to be $k := \ell + \lceil \log_p(4n) \rceil$. This is large enough if

$$k > \log_p\left(\frac{4n}{\alpha}\left(2 + \frac{p^2}{\beta}\right) + 1\right) \Longleftarrow k > \left\lceil \log_p\left(\frac{4n}{\alpha\beta}(2\beta + p^2)\right)\right\rceil.$$

This occurs if

$$\ell > \left\lceil \log_p\left(\frac{4n}{\alpha\beta}(2\beta + p^2)\right)\right\rceil - \lceil \log_p(4n) \rceil \Longleftarrow \ell > \log_p\left(\frac{2\beta + p^2}{\alpha\beta}\right) + 1.$$

The probability that ℓ is large enough is at least

$$2^{-\log_p\left(\frac{2\beta+p^2}{\alpha\beta}\right)-1} = 2^{\log_p\left(\frac{\alpha\beta}{2\beta+p^2}\right)} \cdot 2^{-1} = 2^{\log_2\left(\frac{\alpha\beta}{2\beta+p^2}\right)} \cdot 2^{\frac{1}{\log_2(p)}} \cdot 2^{-1} \geq \frac{\alpha\beta}{4\beta + 2p^2}.$$

Note that since p is polynomial in n, the factor $2^{\frac{1}{\log_2(p)}}$ approaches 1 from above as n increases.

The algorithm iterates through all possible guesses $\mathbf{z} \in \mathbb{Z}_p^k$ for $\mathbf{x}^{\mathrm{T}}\mathbf{R}$, so for some guess we will have $\mathbf{z} = \mathbf{x}^{\mathrm{T}}\mathbf{R}$. For this $\mathbf{z}$ and for any coordinate i, we have

$$h_i(\mathbf{z}) = \sum_{\mathbf{u} \in \mathbb{Z}_p^k \setminus \{0\}} \zeta^{-\langle \mathbf{x}, \mathbf{Ru} \rangle} \mathcal{B}(\mathbf{Ru} + \mathbf{e}_i) = \zeta^{x_i} \sum_{\mathbf{u} \in \mathbb{Z}_p^k \setminus \{0\}} \zeta^{-\langle \mathbf{x}, \mathbf{Ru} + \mathbf{e}_i \rangle} \mathcal{B}(\mathbf{Ru} + \mathbf{e}_i).$$

Normalising this $h_i(\mathbf{z})$ gives a point h_i^* on the complex unit circle. We show how, under certain conditions, the closest root of unity to h_i^* is ζ^{x_i} and how in this case we can recover x_i with high probability.

Note that for a uniformly random matrix $\mathbf{R}$, the vectors $\mathbf{Ru}$ for non-zero $\mathbf{u} \in \mathbb{Z}_p^k$ are pairwise independent. Then the vectors $\mathbf{Ru}+\mathbf{e}_i$ for non-zero $\mathbf{u}$ are also pairwise independent. To simplify notation, let $m := p^k - 1$ and enumerate these pairwise independent vectors by $\mathbf{r}_1, ..., \mathbf{r}_m \in \mathbb{Z}_p^n$. The sum above can then be rewritten as

$$h_i(\mathbf{z}) = \zeta^{x_i} \sum_{j=1}^{m} \zeta^{-\langle \mathbf{x}, \mathbf{r}_j \rangle} \mathcal{B}(\mathbf{r}_j).$$

There are three possible cases for each term of this sum. If $\mathcal{B}$ outputs 0 for a given $\mathbf{r}_j$, then the entire summand becomes 0 and is eliminated from the sum. If $\mathcal{B}$ outputs a correct answer for $\mathbf{r}_j$, then we have $\mathcal{B}(\mathbf{r}_j) = \zeta^{\langle \mathbf{x}, \mathbf{r}_j \rangle}$ and the summand becomes 1. Otherwise, if $\mathcal{B}$ outputs a wrong answer, we have $\mathcal{B}(\mathbf{r}_j) = \zeta^a \zeta^{\langle \mathbf{x}, \mathbf{r}_j \rangle}$ for some non-zero $a \in \mathbb{Z}_p$. Using these three cases, we can write

$$\begin{aligned} h_i(\mathbf{z}) &= \zeta^{x_i} \left(0 + \sum_{\substack{\mathbf{r}_j \text{ s.t.} \\ \mathcal{B} \text{ correct}}} 1 + \sum_{\substack{\mathbf{r}_j \text{ s.t.} \\ \mathcal{B} \text{ wrong}}} \zeta^a \right) \\ &= \zeta^{x_i}(c_0 + c_1\zeta^1 + ... + c_{p-1}\zeta^{p-1}). \end{aligned}$$

for some coefficients $c_0, c_1, ..., c_{p-1} \in \mathbb{R}_+$.

Consider the outputs $\mathcal{B}(\mathbf{r}_1), ..., \mathcal{B}(\mathbf{r}_m)$. Let Y_j be the indicator random variable for $\mathcal{B}(r_j) \neq 0$. This is 1 with probability α, so the random variables $Y_1, ..., Y_m$ have Bernoulli distribution with parameter α. Then $Y := \sum_{j=1}^m Y_j$ is the number of non-zero outputs. By Lemma 1, we have

$$\Pr\left[Y < \frac{m\alpha}{2}\right] \leq \Pr\left[|Y - m\alpha| > \frac{m\alpha}{2}\right] \leq \frac{4}{m\alpha}.$$

Hence, with probability at least $1 - \frac{4}{m\alpha}$, at least $\frac{m\alpha}{2}$ of the outputs are non-zero. Let E_1 denote the event that the number of non-zero outputs is at least $\frac{m\alpha}{2}$.

Let t be the number of non-zero values out of these m outputs. Without loss of generality, assume that these t non-zero outputs coincide with $\mathbf{r}_1, ..., \mathbf{r}_t$. Now we bound the number of these non-zero outputs that are correct. Define random variables $Z_1, ..., Z_t$ where Z_j is the indicator random variable for $\mathcal{B}(\mathbf{r}_j) = \zeta^{\langle x_i, \mathbf{r}_j \rangle}$. Since the correctness of these outputs is conditioned on them being non-zero, the probability that any Z_j is 1 is $\frac{\alpha\beta}{\alpha}$. Thus the random variables $Z_1, ..., Z_t$ have Bernoulli distribution with parameter β. Let $c > 0$ be a constant to be determined later. Then by Lemma 1, we have

$$\Pr\left[Z < (1-c)t\beta\right] \leq \Pr\left[|Z - t\beta| < ct\beta\right] \leq \frac{1}{c^2 t\beta}.$$

Hence, with probability at least $1 - \frac{1}{c^2 t\beta}$, the number of correct outputs from among the t non-zero outputs is at least $(1-c)t\beta$. Let E_2 denote the event that the number of correct outputs out of t non-zero outputs is at least $(1-c)t\beta$.

The probability that both E_1 and E_2 occur is bounded by

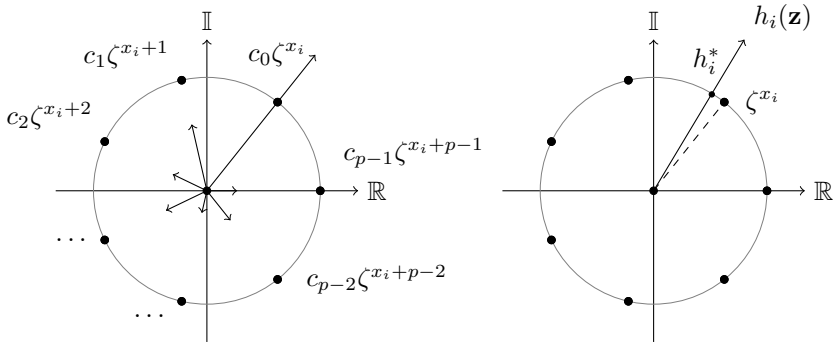
$$\Pr[E_1 \wedge E_2] = \Pr[E_1] \cdot \Pr[E2 \mid E_1] \geq \left(1 - \frac{4}{m\alpha}\right)\left(1 - \frac{2}{c^2 m\alpha\beta}\right) > 1 - \frac{4}{m\alpha} - \frac{2}{c^2 m\alpha\beta}.$$

For $m > \frac{4n}{\alpha}\left(2 + \frac{1}{c^2\beta}\right)$, this bound is at least $1 - \frac{1}{2n}$. This happens if k is large enough. Suppose that both E_1 and E_2 occur. Then the sum

$$h_i(\mathbf{z}) = \zeta^{x_i}(c_0 + c_1\zeta^1 + ... + c_{p-1}\zeta^{p-1}),$$

has coefficients such that $c_0 \geq (1-c)t\beta$ and $c_1 + ... + c_{p-1} \leq t - (1-c)t\beta$.

Now we show that if this value $h_i(\mathbf{z})$ is normalised to h_i^*, the closest root of unity is ζ^{x_i}. Intuitively, the condition on β and choice of parameter c above make the coefficient c_0 much larger than the sum of the other coefficients $c_1 + ... + c_{p-1}$. This bias ensures that $h_i(\mathbf{z})$ is close to a multiple of ζ^{x_i} and hence h_i^* is close to ζ^{x_i}. We illustrate this idea in the diagram below.



Now we formalise this argument. Denote $\omega := c_0 + c_1\zeta + ... + c_{p-1}\zeta^{p-1}$. Since ζ^{x_i} is a factor of length 1 in $h_i(\mathbf{z})$, it is enough to show that the normalisation $\omega^* := \frac{\omega}{|\omega|}$ is close to 1. Since the p-th roots of unity are evenly spaced on the complex unit circle, the angle between any consecutive pair of them is $\frac{2\pi}{p}$. Hence, we need to show that the angle θ between ω^* and the root $1 = \zeta^0$ is strictly less than $\frac{\pi}{p}$. To do this, we use the formula for the length of the chord between ω^* and 1 in terms of θ,

$$|\omega^* - 1| = 2\sin\left(\frac{\theta}{2}\right). \qquad (***)$$

By the assumption on p, we have $\frac{\pi}{p} < \frac{\pi}{2}$. Since $\sin(\phi)$ is an injective increasing function for the range $0 \leq \phi < \frac{\pi}{p}$, showing that $2\sin\left(\frac{\theta}{2}\right) \leq 2\sin\left(\frac{\pi}{2p}\right)$ immediately gives us $\theta < \frac{\pi}{p}$. We prove that $|\omega^* - 1| < 2\sin\left(\frac{\pi}{2p}\right)$ below.

First we use the triangle inequality to bound the normalisation factor by

$$|\omega| = \left|c_0 + c_1\zeta + ... + c_{p-1}\zeta^{p-1}\right| \leq c_0 + c_1 + ...c_{p-1}.$$

The assumption on β implies that $\beta > \frac{1}{2(1-c)}$. This ensures that the coefficients satisfy $c_0 \geq (1-c)t\beta \geq t - (1-c)t\beta \geq c_1 + ... + c_{p-1}$. Then by applying the triangle inequality again, we obtain the lower bound

$$\begin{aligned}
|\omega| &\geq \left|c_0 - \left|c_1\zeta + ... + c_{p-1}\zeta^{p-1}\right|\right| \\
&\geq \left|c_0 - (c_1 + ... + c_{p-1})\right| \\
&\geq c_0 - (c_1 + ... + c_{p-1}) \\
&\geq t(2\beta(1-c) - 1).
\end{aligned}$$

Using these bounds, we can bound the distance between ω^* and 1 by

$$\begin{aligned}
|\omega^* - 1| &= \frac{1}{|\omega|}\big|\omega - |\omega|\big| \\
&\leq \frac{1}{|\omega|}\Big(|\omega - c_0| + \big|\,|\omega| - c_0\big|\Big) \\
&\leq \frac{1}{|\omega|}\left(\left|c_1\zeta + ... + c_{p-1}\zeta^{p-1}\right| + \left|c_0 + c_1 + ...c_{p-1} - c_0\right|\right) \\
&\leq \frac{2}{|\omega|}(c_1 + ... + c_{p-1}) \\
&\leq \frac{2(c_1 + ... + c_{p-1})}{c_0 - (c_1 + ... + c_{p-1})} \\
&\leq \frac{2t(1-(1-c)\beta)}{t(2(1-c)\beta - 1)} \\
&= \frac{2 - 2(1-c)\beta}{2(1-c)\beta - 1}.
\end{aligned}$$

This bound can be rewritten as

$$\frac{2 - 2(1-c)\beta}{2(1-c)\beta - 1} = -\frac{2(1-c)\beta - 1 - 1}{2(1-c)\beta - 1} = \frac{1}{2(1-c)\beta - 1} - 1.$$

This satisfies

$$\frac{1}{2(1-c)\beta - 1} - 1 < 2\sin\left(\frac{\pi}{2p}\right) \iff \beta > \frac{1}{2(1-c)}\left(1 + \frac{1}{1 + 2\sin(\frac{\pi}{2p})}\right).$$

By the assumption on β and taking $c = \frac{1}{p}$, we have

$$1 > \beta > 1 - \frac{1}{5p} > \frac{1}{2(1-c)}\left(1 + \frac{1}{1 + 2\sin(\frac{\pi}{2p})}\right)$$

for any $p > 10$. This lower bound is less than 1 for this choice of parameters and has limit 1 as p tends towards infinity. Since $\beta \in (0, 1)$ by definition, this means that β must be almost 1 for larger p. We set $c = \frac{1}{p}$.

Therefore, if both events E_1 and E_2 occur for every coordinate $i \in \{1, ..., n\}$, we obtain ζ^{x_i} as the closest p-th root of unity to h_i^* for every i. So under these conditions, $\mathcal{A}$ will output $\mathbf{x}' = \mathbf{x}$.

Now we analyse the runtime of $\mathcal{A}$. The expected number of coin flips needed until a 0 is obtained is 2, so in expectation, $k = 2 + \lceil \log_p(4n) \rceil = O(\log_p(n))$. Thus, the algorithm can iterate through all guesses for $\mathbf{x}^{\mathrm{T}}\mathbf{R}$ in expected time polynomial in n. Running FFT on the coefficients g_i to compute $h_i(\mathbf{z})$ takes $O(p^k \log p^k)$ time. Finding the closest p-th root of unity to h_i^* can be done by checking if the bound $(***)$ holds for each root, which takes $O(p)$ time. Since the one-way function f is efficiently computable, the algorithm can efficiently check if $\mathbf{x}' = \mathbf{x}$. Iterating over all n coordinates of $\mathbf{x}$ and by trying all possible values of $\mathbf{z}$, the total expected runtime of $\mathcal{A}$ is

$$n \cdot p^k \cdot (O(p^k \log p^k) + O(p)) = n \cdot p^{O(\log_p(n))} \cdot O(p^{\log_p(n)} \log(p^{\log_p(n)})) = O(n^3 \log n).$$

The success of $\mathcal{A}$ depends on k being sufficiently large and $h_i(\mathbf{z})$ being non-zero for the sampled $\mathbf{R}$. Then the probability that $\mathcal{A}$ succeeds, over the randomness of the input $\mathbf{x}$ and the algorithm's internal randomness,

is given by

$$\begin{aligned}\Pr_{\mathbf{x}}[\mathcal{A}(f(\mathbf{x})) = \mathbf{x}] &= \Pr_{\mathbf{x}}[k \text{ large enough}] \cdot \Pr_{\mathbf{R}}[\forall i \in [n],\ E_1 \wedge E_2 \mid k \text{ large enough}] \\ &> \frac{\alpha\beta}{4\beta + 2p^2} \cdot \left(n\left(1 - \frac{1}{2n}\right) - (n-1) \right) \\ &> \frac{\alpha\beta}{4\beta + 2p^2} \cdot \frac{1}{2} \\ &\geq \frac{1}{5p^2}\alpha\beta.\end{aligned}$$

Since $\mathcal{A}$ verifies its guess for $\mathbf{x}$ and never outputs a wrong answer, it outputs $\perp$ with the remaining probability. □

Now we apply this result to our study of the hardness of LWE. First we define an intermediate worst-case problem inspired by the Goldreich-Levin theorem.

Definition 19. (Goldreich-Levin LWE) *The Goldreich-Levin Learning with Errors (GL-LWE) problem, denoted by GL-LWE$_{n,p,\phi}$ is defined as: given a polynomial number of samples from the distribution $A_{\mathbf{s},\phi}$, where $\mathbf{s} \in \mathbb{Z}_p^n$ is some fixed secret, and a uniformly random vector $\mathbf{r} \in \mathbb{Z}_p^n$, find $\langle \mathbf{s}, \mathbf{r} \rangle \bmod p$.*

We use this to reduce (average-case) search-LWE to (average-case) decision-LWE. First note that search-LWE trivially reduces to worst-case search-LWE: given LWE samples for a uniformly random secret $\mathbf{s}$, the reduction algorithm simply runs its oracle for worst-case search-LWE on the given samples and succeeds in recovering $\mathbf{s}$ with the same probability. Now we interpret Lemma 14 as a reduction from worst-case search-LWE to GL-LWE.

Corollary 3. (Search-LWE → GL-LWE) *Let $n, p \in \mathbb{Z}_+$ such that $p > 10$ is polynomial in n. Suppose that there is an efficient algorithm $\mathcal{B}$ for GL-LWE$_{n,p,\phi}$ that, given a polynomial number of samples from $A_{\mathbf{s},\phi}$ for some fixed secret $\mathbf{s} \in \mathbb{Z}_p^n$, and a uniformly random vector $\mathbf{r} \in \mathbb{Z}_p^n$, outputs a guess for $\langle \mathbf{s}, \mathbf{r} \rangle \bmod p$*

- *correctly with probability $\alpha^*\beta^*$,*
- *incorrectly with probability $\alpha^*(1 - \beta^*)$, and*
- *outputs $\perp$ with probability $1 - \alpha^*$,*

where $\beta^* > 1 - \frac{1}{5p}$. The probability is taken over the randomness of $\mathbf{s}, \mathbf{r}$, and the randomness of the algorithm. Then there is an algorithm $\mathcal{A}$ for search-LWE$_{n,p,\phi}$ that runs in expected polynomial time and, given a polynomial number of samples from $A_{\mathbf{s},\phi}$ for some fixed secret $\mathbf{s} \in \mathbb{Z}_p^n$,

- *correctly outputs $\mathbf{s}$ with probability $\frac{1}{5p^2}\alpha\beta$ and*
- *outputs $\perp$ with probability $1 - \frac{1}{5p^2}\alpha\beta$.*

Proof. Consider the function $f_\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ given by $f_\phi(\mathbf{s}) := (\mathbf{A}, \frac{1}{p}\mathbf{A}\mathbf{s} + \mathbf{e})$, where the rows of $\mathbf{A}$ are uniformly random vectors sampled from $\mathbb{Z}_p^n$ and $\mathbf{e}$ is sampled according to distribution ϕ. This can be used as an injective function, because with high probability there is a unique $\mathbf{s}$ that satisfies the system of equations determined by any given output $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \frac{1}{p}\mathbf{A}\mathbf{s} + \mathbf{e})$. Applying Lemma 14 gives us the desired result. □

4.2 GL-LWE to Decision-LWE

Finally, we finish our chain of reductions by reducing our worst-case intermediate problem to decision-LWE. In the following we show that if there is a (γ, δ)-solver (with $\gamma \gg \delta$) for decision-LWE, then there is an algorithm for GL-LWE that Corollary 3 can be instantiated with to complete the reduction from search-LWE to decision-LWE.

Lemma 15. (GL-LWE $\rightarrow$ Decision-LWE) *Let $k \in \mathbb{Z}_+$ be a constant. Suppose that there exists an efficient algorithm $\mathcal{B}$ for decision-LWE$_{n,p,\phi}$ that,*

- *given n^k LWE samples from $A_{\mathbf{s},\phi}$ for a uniformly random secret $\mathbf{s}$, outputs YES with probability γ,*
- *given n^k random samples from $\mathbb{Z}_p^n \times \mathbb{T}$, outputs YES with probability δ.*

Then there is an algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$ and an instance of GL-LWE$_{n,p,\phi}$ consisting of n^k samples from $A_{\mathbf{s},\phi}$ for a fixed secret $\mathbf{s}$, outputs

- *a correct answer with probability $\frac{1}{p}\gamma$,*
- *a wrong answer with probability $\frac{p-1}{p}\delta$, and*
- *$\perp$ with probability $1 - \frac{1}{p}\gamma - \frac{p-1}{p}\delta$.*

Proof. Let $\mathbf{s} \in \mathbb{Z}_p^n$ be the fixed secret corresponding to the samples given to $\mathcal{A}$ as input. Consider the following algorithm $\mathcal{A}$:

Algorithm 6: GL-LWE to Decision-LWE Reduction

Input: Samples $X_1, ..., X_{n^k} \sim A_{\mathbf{s},\phi}$ and $\mathbf{r} \in \mathbb{Z}_p^n$.
Output: $c \in \mathbb{Z}_p$ or $\perp$.

Sample $c \leftarrow \mathbb{Z}_p$ uniformly at random as a guess for $\langle \mathbf{s}, \mathbf{r} \rangle \bmod p$.
Sample vector $\mathbf{t} \leftarrow \mathbb{Z}_p^n$ uniformly at random.
for $i \in \{1, ..., n^k\}$ **do**
  Denote $(\mathbf{a}, b) := X_i$.
  Define $\mathbf{a}' := \mathbf{a} + \mathbf{r}$ and $b' := b + \frac{1}{p}c + \frac{1}{p}\langle \mathbf{a}', \mathbf{t} \rangle$.
  Define $Y_i := (\mathbf{a}', b')$.
end
Run $\mathcal{B}$ on the transformed samples $Y_1, ..., Y_{n^k}$ to obtain an answer.
if $\mathcal{B}$ *responds YES* **then**
  Output c.
end
Output $\perp$.

Since we efficiently transform a polynomial number of samples and call $\mathcal{B}$ once, algorithm $\mathcal{A}$ is efficient. Now we prove correctness of algorithm $\mathcal{A}$. It is given LWE samples of the form $(\mathbf{a}, b)$ as input, where $b = \frac{1}{p}\langle \mathbf{a}, \mathbf{s} \rangle + e$ for some fixed secret vector $s \in \mathbb{Z}_p^n$ and error $e \in \mathbb{T}$, according to the distribution $A_{\mathbf{s},\phi}$.

Suppose the algorithm guesses correctly and $c = \langle \mathbf{s}, \mathbf{r} \rangle \bmod p$. In this case, the transformed samples $(\mathbf{a}', b')$ are LWE samples. This is because $\mathbf{a}' = \mathbf{a} + \mathbf{r}$ is uniformly random, since $\mathbf{r}$ is a uniformly random vector. We also have

$$\begin{aligned} b' &= b + \frac{1}{p}c + \frac{1}{p}\langle \mathbf{a}', \mathbf{t} \rangle \\ &= \frac{1}{p}\langle \mathbf{a}, \mathbf{s} \rangle + e + \frac{1}{p}\langle \mathbf{r}, \mathbf{s} \rangle + \frac{1}{p}\langle \mathbf{a} + \mathbf{r}, \mathbf{t} \rangle \\ &= \frac{1}{p}\langle \mathbf{a} + \mathbf{r}, \mathbf{s} + \mathbf{t} \rangle + e \\ &= \frac{1}{p}\langle \mathbf{a}', \mathbf{s}' \rangle + e, \end{aligned}$$

for a uniformly random secret $\mathbf{s}' := \mathbf{s} + \mathbf{t}$. Since c and $\mathbf{t}$ were chosen independently and uniformly at random, b' is uniformly random. Hence, the oracle $\mathcal{B}$ is given LWE samples distributed according to $A_{\mathbf{s}',\phi}$ for a uniformly random secret $\mathbf{s}' \in \mathbb{Z}_p^n$. Then it correctly responds YES with probability γ. The guess is $c = \langle \mathbf{s}, \mathbf{r} \rangle \bmod p$ with probability $\frac{1}{p}$, so the probability that $\mathcal{A}$ outputs a correct answer is $\frac{1}{p}\gamma$.

The guess is incorrect with probability $1 - \frac{1}{p}$. In this case, the transformed samples $(\mathbf{a}', b')$ are uniformly distributed over $\mathbb{Z}_p^n \times \mathbb{T}$. This is because $\mathbf{a}'$ is uniform over $\mathbb{Z}_p^n$, since $\mathbf{r}$ is a uniformly random vectors, and $b' = b + \frac{1}{p}c + \frac{1}{p}\langle \mathbf{a}', \mathbf{t} \rangle$ is uniformly random, since c and $\mathbf{t}$ were chosen independently and uniformly at random. Hence, the oracle $\mathcal{B}$ receives uniformly random samples and incorrectly responds YES with probability δ. Thus, $\mathcal{A}$ outputs a wrong answer with probability $\frac{p-1}{p}\delta$.

Then the algorithm outputs $\perp$ with the remaining probability $1 - \frac{1}{p}\gamma - \frac{p-1}{p}\delta$.

□

Now we combine Corollary 3 and Lemma 15 to obtain our second main result.

Corollary 4. (Search-LWE $\to$ Decision-LWE) *Let $n, p, k \in \mathbb{Z}_+$ be such that $p > 10$ is polynomial in n and k is a constant. Suppose that there exists an efficient algorithm $\mathcal{B}$ for decision-LWE$_{n,p,\phi}$ that,*

- *given n^k LWE samples from $A_{\mathbf{s},\phi}$, outputs YES with probability γ,*
- *given n^k random samples from $\mathbb{Z}_p^n \times \mathbb{T}$, outputs YES with probability δ,*

where $\gamma > 5p^2\delta$. Then there is an algorithm $\mathcal{A}$ for search-LWE$_{n,p,\phi}$ with oracle access to $\mathcal{B}$ that runs in expected polynomial time and

- *outputs a correct answer with probability $\frac{1}{5p^3}\gamma$ and*
- *outputs $\perp$ with probability $1 - \frac{1}{5p^3}\gamma$.*

Proof. Set $\alpha := \frac{\gamma + (p-1)\delta}{p}$ and $\beta := \frac{\gamma}{\gamma + (p-1)\delta}$. Then we have $\alpha\beta = \frac{1}{p}\gamma$ and $\alpha(1-\beta) = \frac{p-1}{p}\delta$. By the assumption that $\gamma > 5p^2\delta$, and since $p > 10$, we obtain

$$\beta = \frac{\gamma}{\gamma + (p-1)\delta} > 1 - \frac{p-1}{5p^2 + p - 1} > 1 - \frac{1}{5p}.$$

Consider the following algorithm $\mathcal{A}$: Given an instance of search-LWE, first run the algorithm from Lemma 15 to solve the corresponding instance of GL-LWE. Then run the algorithm from Corollary 3 to solve the corresponding GL-LWE instance. Finally, run the trivial algorithm to solve the given average-case search-LWE instance. By Corollary 3, for these values of α and β, this algorithm $\mathcal{A}$ outputs a correct answer for the given instance of search-LWE with probability

$$\frac{1}{5p^2}\alpha\beta = \frac{1}{5p^3}\gamma,$$

and outputs $\perp$ with the remaining probability.

5 Conclusions and Future Directions

In this paper, we offer a new perspective on the computational complexity of lattice problems by revisiting the notion of characterizing the hardness of a computational problem in terms of the maximum success probability achievable by any probabilistic polynomial-time algorithm.

We show how characterising the hardness in such a way enables us to obtain a much tighter reduction from the worst-case BDD problem for lattices to the average-case search-LWE problem, as well as a tight reduction from search-LWE to decision-LWE. (See Section 1.3 for precise statements.)

We believe that our work should motivate quantifying the hardness of computational problems − especially those relevant to cryptography − using a similar metric. We emphasize that such reductions will be very sensitive to the number of calls made to the oracle, since the success probability will decrease exponentially with the number of oracle calls. In the reductions in this work, our main challenge was to ensure that our reductions make a single call to the oracle, even if that meant the reduction succeeds with a relatively small probability.

6 Acknowledgments

The authors would like to thank Chris Peikert and Vinod Vaikuntanathan for useful conversations during the course of this work.

References

ABGSD21. Divesh Aggarwal, Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. Fine-grained hardness of CVP(P) – Everything that we can prove (and nothing else), 2021. 2

ADRS14. Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n Time via Discrete Gaussian Sampling. *CoRR*, abs/1412.7994, 2014. 2

ADRS15. Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the Shortest Vector Problem in 2^n Time via Discrete Gaussian Sampling, 2015. 2

Ajt96. Miklós Ajtai. Generating Hard Instances of Lattice Problems. *Electron. Colloquium Comput. Complex.*, TR96, 1996. 2

ALS20. Divesh Aggarwal, Zeyong Li, and Noah Stephens-Davidowitz. A $2^{n/2}$-Time Algorithm for $\sqrt{n}$-SVP and $\sqrt{n}$-Hermite SVP, and an Improved Time-Approximation Tradeoff for (H)SVP. *arXiv e-prints*, Jul 2020. 2

AM11. Divesh Aggarwal and Ueli Maurer. The leakage-resilience limit of a computational problem is equal to its unpredictability entropy. In *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 17*, pages 686–701. Springer, 2011. 3

ASD18. Divesh Aggarwal and Noah Stephens-Davidowitz. (Gap/S)ETH Hardness of SVP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 228–238, New York, NY, USA, 2018. Association for Computing Machinery. 2

Bab86. László Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986. 3, 11

BGS17. Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. On the Quantitative Hardness of CVP. *CoRR*, abs/1704.03928, 2017. 2

BKW00. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model. *CoRR*, cs.LG/0010022, 2000. 2, 3

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical Hardness of Learning with Errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA, 2013. Association for Computing Machinery. 2, 12, 17

Bri83. Ernest F. Brickell. Solving Low Density Knapsacks. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 25–37. Plenum, 1983. 2

BV14. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pages 1–12, New York, NY, USA, 2014. Association for Computing Machinery. 2

Cai98. Jin-Yi Cai. A relation of primal-dual lattices and the complexity of shortest lattice vector problem. *Theoretical Computer Science*, 207(1):105–116, 1998. 9

Din02. Irit Dinur. Approximating $\text{SVP}_{\text{infinity}}$ to within almost-polynomial factors is NP-hard. *Theor. Comput. Sci.*, 285(1):55–71, 2002. 2

DKRS03. I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to Within Almost-Polynomial Factors is NP-Hard. *Combinatorica*, 23(2):205–243, Apr 2003. 2

FT87. A. Frank and Éva Tardos. An Application of Simultaneous Diophantine Approximation in Combinatorial Optimization. *Combinatorica*, 7(1):49–65, Jan 1987. 2

Gen09. Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. Association for Computing Machinery. 2

GL89. O. Goldreich and L. A. Levin. A Hard-Core Predicate for All One-Way Functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 25–32, New York, NY, USA, 1989. Association for Computing Machinery. 5, 18

GMSS99. O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. 2

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–â€"206, New York, NY, USA, 2008. Association for Computing Machinery. 9

HR18. Ishay Haviv and Oded Regev. Tensor-based Hardness of the Shortest Vector Problem to within Almost Polynomial Factors. *CoRR*, abs/1806.04087, 2018. 2

Kan87. Ravi Kannan. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research*, 12(3):415–440, 1987. 2

Kho05. Subhash Khot. Hardness of Approximating the Shortest Vector Problem in Lattices. *J. ACM*, 52(5):789–808, Sep 2005. 2

Lev12. Leonid A. Levin. Randomness and Non-determinism. *CoRR*, abs/1211.0071, 2012. 5, 18, 28

LLL82. Arjen Lenstra, Hendrik Lenstra, and Lovász László. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261, 12 1982. 2, 3

LM09. Vadim Lyubashevsky and Daniele Micciancio. On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 577–594, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 3

LO85. Jeffrey Lagarias and Andrew Odlyzko. Solving Low-Density Subset Sum Problems. *J. ACM*, 32:229–246, Jan 1985. 2

MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004. 2

MR09. Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 2

MV13. Daniele Micciancio and Panagiotis Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems Based on Voronoi Cell Computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013. 2

MW18. Daniele Micciancio and Michael Walter. On the Bit Security of Cryptographic Primitives. Cryptology ePrint Archive, Paper 2018/077, 2018. https://eprint.iacr.org/2018/077. 5, 28

Pei09. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 333–342, 2009. 2

PP10. Ramamohan Paturi and Pavel Pudlak. On the Complexity of Circuit Satisfiability. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pages 241–250, New York, NY, USA, 2010. Association for Computing Machinery. 4

Reg06. Oded Regev. Lattice-Based Cryptography. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 131–141, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 1, 2

Reg09. Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM*, 56(6), Sep 2009. 1, 2, 5, 7, 9, 10, 12, 15

Sha85. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 47–53, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg. 2

Vad12. Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. 7

vEB81. Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. *Tecnical Report, Department of Mathmatics, University of Amsterdam*, 1981. 2

Appendix A Proof of Levin's Result in [Lev12]

The original statement of Levin's generalization of the Goldreich-Levin theorem in [Lev12] relates the security of a one-way function to that of a hard-core predicate. The *security* of a computational problem is essentially the minimum of the inverse success probability of any PPT algorithm that tries to solve it. The notion of security can be naturally extended to the primitive that corresponds to the computational problem. For example, we say a one-way function f has security $\sigma \geq 1$ if any polynomial-time algorithm that tries to invert a given element in the image of f will succeed with probability at most $\frac{1}{\sigma}$. This problem of inverting a one-way function is a search problem. On the other hand, the problem of guessing a hard-core predicate can be seen as a decision problem (since a hard-core predicate is binary). As discussed in the introduction, the notion of OPP algorithms needs to be adapted to work for both search and decision problems when studying reductions from one to another.

Micciancio and Walter [MW18] studied the (bit) security of a computational problem and defined this to be compatible with both search and decision problems. If the algorithm is allowed to output a special symbol $\perp$ as an alternative to outputting a correct or wrong answer, then it enables the algorithm to express uncertainty rather than randomly guess an answer that will most likely be wrong. This can often be more informative than giving a (possibly wrong) binary answer.

Consider an algorithm $\mathcal{A}$ for a computational problem that, given any instance, outputs $\perp$ with probability $1-\alpha$, answers correctly with probability $\alpha\beta$, and answers incorrectly with probability $\alpha(1-\beta)$. More precisely, define the *output probability* of $\mathcal{A}$ to be $\alpha := \Pr[A \neq \perp]$ and the *conditional success probability* to be $\beta := \Pr[R(X,A) \mid A \neq \perp]$, where the probabilities are over the randomness of the entire problem and the internal randomness of $\mathcal{A}$. (Recall that for our purposes, we considered $\alpha \gg \beta$.) Using this notation, the success probability of $\mathcal{A}$ is given by $\alpha\beta$. For decision problems, we define the *conditional distinguishing advantage* of $\mathcal{A}$ to be $\delta := 2\beta_{\mathcal{A}} - 1$. Micciancio and Walter define (and prove) the *advantage* of any $\mathcal{A}$ to be $\alpha\beta$ for a search problem and $\alpha(2\beta-1)^2$ for a decision problem.

Here we state the original result in [Lev12] and give a formal proof of Levin's result using the precise definitions above.

Lemma 16. *(adapted from [Lev12]) Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function family that is length-preserving, i.e. $|f(\mathbf{x})| = |\mathbf{x}|$ for any input $\mathbf{x} \in \{0,1\}^n$, and has security s. Then $b : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$, $b(\mathbf{x},\mathbf{r}) := (-1)^{\langle \mathbf{x},\mathbf{r}\rangle \bmod 2}$ is a hard-core predicate for f with security s.*

Proof. Suppose that $\mathcal{G}$ is a PPT algorithm that, given $w \in \mathbb{Z}_+$ fair random coins, $f(\mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^n$, and a vector $\mathbf{r} \in \{0,1\}^n$ as input, guesses $b(\mathbf{x},\mathbf{r})$ with success probability

$$s_{\mathcal{G},f,b} = \frac{\mathbb{E}_{\mathbf{r},\mathbf{x},w}[\mathcal{G}(f(\mathbf{x}),\mathbf{r},w)\cdot b(\mathbf{x},\mathbf{r})]^2}{\mathbb{E}_{\mathbf{x},\mathbf{r},w}[\mathcal{G}(f(\mathbf{x}),\mathbf{r},w)^2]} > \frac{1}{s}.$$

Since finding the hard-core bit is a decision problem, this success probability is the advantage of $\mathcal{G}$ for large n. We show this explicitly later in the proof.

Let the output of $\mathcal{G}$ be in $\{-1,0,1\}$, where outputting "0" indicates that $\mathcal{G}$ does not know and admits failure (this is used instead of $\perp$). We use this $\mathcal{G}$ to construct an algorithm $\mathcal{A}$ that, given $f(\mathbf{x})$ for some $\mathbf{x}$ and w random coins, inverts $f(\mathbf{x})$ with success probability

$$s_{\mathcal{A},f} = \Pr_{\mathbf{x},w}[\mathcal{A}(f(\mathbf{x}),w) = \mathbf{x}' \in f^{-1}(f(\mathbf{x}))] > \frac{1}{s}$$

and runs in expected polynomial time.

Fix the input $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^n$ and randomness w. For this input, we use the shorthand notation $\mathcal{G}(\mathbf{r}) := \mathcal{G}(\mathbf{y},\mathbf{r},w)$. Define

$$c(\mathbf{x}) := \frac{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})\cdot b(\mathbf{x},\mathbf{r})]}{\sqrt{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})^2]}}.$$

We claim that this c is the Walsh-Hadamard transform of $\mathcal{G}$ up to a constant factor. By definition, we can write

$$c(\mathbf{x}) = \frac{\sum_{\mathbf{r}\in\{0,1\}^n} \mathcal{G}(\mathbf{r}) \cdot (-1)^{\langle \mathbf{x},\mathbf{r}\rangle} \cdot \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \cdot b(\mathbf{x},\mathbf{r})]}{\sqrt{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})^2]}} = \frac{1}{\sqrt{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})^2]}} \hat{\mathcal{G}}(\mathbf{r}).$$

Since $\mathbf{r}$ is fixed in the sum, the probability is eliminated, so we obtain a constant (dependent on $\mathbf{r}$) multiple of $\hat{\mathcal{G}}$, the Walsh-Hadamard transform of $\mathcal{G}$.

By definition, we can use the security notation in the preamble to rewrite the following.

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})^2] &= \sum_{\mathbf{r}\in\{0,1\}^n} \mathcal{G}(\mathbf{r})^2 \cdot \Pr[\mathcal{G}(\mathbf{r})^2] \\
&= \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = 1] + (-1)^2 \Pr_{\mathbf{r}}[\mathcal{G}(r) = -1] + 0 \\
&= \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \neq 0] = \alpha_{\mathcal{G}}. \\
\mathop{\mathbb{E}}_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \cdot b(\mathbf{x},\mathbf{r})] &= 0 + \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \cdot b(\mathbf{x},\mathbf{r}) = 1] + (-1) \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \cdot b(\mathbf{x},\mathbf{r}) = -1] \\
&= \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = b(\mathbf{x},\mathbf{r})] - \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = -b(\mathbf{x},\mathbf{r})].
\end{aligned}$$

Dividing these and noticing that $\mathcal{G}$ guesses the bit (correctly or not) only if it does not output 0, we obtain

$$\begin{aligned}
\frac{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \cdot b(\mathbf{x},\mathbf{r})]}{\mathbb{E}_{\mathbf{r}}[\mathcal{G}(\mathbf{r})^2]} &= \frac{\Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = b(\mathbf{x},\mathbf{r})}{\Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \neq 0]} - \frac{\Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = -b(\mathbf{x},\mathbf{r})}{\Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) \neq 0]} \\
&= \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = b(\mathbf{x},\mathbf{r}) \mid \mathcal{G}(\mathbf{r}) \neq 0] - \Pr_{\mathbf{r}}[\mathcal{G}(\mathbf{r}) = -b(\mathbf{x},\mathbf{r}) \mid \mathcal{G}(\mathbf{r}) \neq 0].
\end{aligned}$$

But by definition of conditional distinguishing advantage, this is exactly $\beta_{\mathcal{G}} - (1 - \beta_{\mathcal{G}}) = 2\beta_{\mathcal{G}} - 1 = \delta_{\mathcal{G}}$. Using this notation and the claim above, we can express $c(\mathbf{x})$ as $\sqrt{\alpha_{\mathcal{G}}}\delta_{\mathcal{G}}$. This then implies that the success probability of $\mathcal{G}$ is the same as its advantage for these fixed inputs.

Now we define our inverter. Consider the following algorithm $\mathcal{A}$:

Algorithm 7: OWF to HCP Reduction

Input: $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^n$, w random coins.
Output: $\mathbf{x}' \in f^{-1}(\mathbf{y})$ or $\perp$.

Flip $\ell = \ell(w)$ coins until a 0 is obtained. If $\ell > 2n$, abort and output $\perp$.
Set $k := \ell + \lceil \log_2(4n) \rceil$.
Sample a random matrix $\mathbf{R} \leftarrow \{0,1\}^{n\times k}$.
for $\mathbf{z} \in \{0,1\}^k$ **do**
  for $1 \le i \le n$ **do**
    Define $g_i(\mathbf{u}) := \mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)$.
    Run FFT on g_i to compute $h_i(\mathbf{z}) := \sum_{\mathbf{u}\in\{0,1\}^k\setminus\{\mathbf{0}\}} (-1)^{\langle \mathbf{z},\mathbf{u}\rangle} g_i(\mathbf{u})$.
    Set $x'_i := \mathrm{sign}(h_i(\mathbf{z}))$.
  end
  Define $\mathbf{x}' := (x'_1, ..., x'_n)$.
  if $f(\mathbf{x}') = \mathbf{y}$ **then**
    Output $\mathbf{x}'$.
  end
end
Output $\perp$.

Here FFT denotes the Fast Fourier Transform. We explain the reasoning behind this algorithm, then analyse its performance.

Consider the case where $c(\mathbf{x}) > 0$. (Otherwise, $\delta_{\mathcal{G}}\sqrt{\alpha_{\mathcal{G}}} \le 0$, which implies $\delta \le 0$. This means that $\mathcal{G}$ has no distinguishing advantage. We are only interested in the case where $\mathcal{G}$ has some advantage, so we ignore

this case.) The algorithm flips ℓ coins until it obtains a 0. The probability that $\ell = \ell'$ for some fixed $\ell' \in \mathbb{Z}_+$ is $\frac{1}{2^{\ell'}}$. Then $\ell > \ell'$ with probability

$$\Pr_{\ell}[\ell > \ell'] = \left(1 - \Pr_{\ell}[\ell \leq \ell']\right) = \left(1 - \sum_{i=1}^{\ell'} \frac{1}{2^{\ell'}}\right) = \left(1 - \left(1 - \frac{1}{2^{\ell'}}\right)\right) = \frac{1}{2^{\ell'}}.$$

The parameter k is set to be $k := \ell + \lceil \log_2(4n) \rceil$. This is large enough if

$$k > \log_2\left(\frac{2n}{c(\mathbf{x})^2}\right) \iff \ell > \left\lceil \log_2\left(\frac{2n}{4n \cdot c(\mathbf{x})^2}\right)\right\rceil \geq \log_2\left(\frac{1}{2c(\mathbf{x})^2}\right)$$

Hence, k is large enough with probability $2c(\mathbf{x})^2$.

Let $m := \frac{2^k}{\alpha_{\mathcal{G}}}$. We claim the following: If $\mathbf{r}_1, ..., \mathbf{r}_m \in \{0,1\}^n$ are pairwise independent random vectors, then

$$\Pr_{\mathbf{r}_1,...,\mathbf{r}_m}\left[\sum_{j=1}^{m}(-1)^{\langle \mathbf{x}, \mathbf{r}_j\rangle}\mathcal{G}(\mathbf{r}_j) > 0\right] > 1 - \frac{1}{2n}.$$

Let $Z_j := (-1)^{\langle \mathbf{x}, \mathbf{r}_j\rangle}\mathcal{G}(\mathbf{r}_j)$ be the corresponding random variables and let $Z := \sum_{j=1}^{m} Z_j$ denote their sum. Since $\mathbf{r}_1, ..., \mathbf{r}_m$ are pairwise independent, $Z_1, ..., Z_m$ are also pairwise independent random variables. Since these are identically distributed, their expectation is the same $c := \mathbb{E}[Z_j] = \delta_{\mathcal{G}}\alpha_{\mathcal{G}}$ for all j. This allows us to rewrite the expectation and variance of Z as

$$\mathbb{E}[Z] = \sum_{j=1}^{m}\mathbb{E}[Z_j] = m \cdot c = m \cdot \sqrt{\alpha_{\mathcal{G}}}c(\mathbf{x}).$$
$$\mathrm{Var}[Z] = \sum_{j=1}^{m}\mathrm{Var}[Z_j] \leq m.$$

Note that the expectation of the random variable Z depends on the success rate of $\mathcal{G}$. The bound on the variance follows from the fact that Z_j is distributed over $\{-1, 0, 1\}$. Observe that

$$\Pr_{\mathbf{r}_1,...,\mathbf{r}_m}[Z \leq 0] = \Pr_{\mathbf{r}_1,...,\mathbf{r}_m}[Z - mc \leq -mc] \leq \Pr_{\mathbf{r}_1,...,\mathbf{r}_m}[|Z - mc| \geq mc].$$

By the Chebyshev inequality, we obtain

$$\Pr_{\mathbf{r}_1,...,\mathbf{r}_m}[|Z - mc| \geq mc] \leq \frac{\mathrm{Var}[Z]}{(mc)^2} \leq \frac{1}{mc^2}.$$

For our choice of $m > \frac{2n}{\alpha_{\mathcal{G}}c(\mathbf{x})^2}$, the bound above is less than $\frac{1}{2n}$. Therefore we obtain the desired claim.

Note that for pairwise independent $\mathbf{r}_1, ..., \mathbf{r}_m \in \{0,1\}^n$, the vectors $\mathbf{r}_1 + \mathbf{e}_i$, $..., \mathbf{r}_m + \mathbf{e}_i$ are also pairwise independent for any $1 \leq i \leq n$. For any random matrix $\mathbf{R} \in \{0,1\}^{n\times k}$, the vectors $\mathbf{Ru}$ for non-zero $\mathbf{u} \in \{0,1\}^k$ are pairwise independent vectors. Thus the claim holds for the vectors $\mathbf{Ru} + \mathbf{e}_i$, i.e.

$$\Pr_{\mathbf{R}}\left[\sum_{\mathbf{u}\in\{0,1\}^k\setminus\{\mathbf{0}\}}(-1)^{\langle \mathbf{x}, \mathbf{Ru}+\mathbf{e}_i\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i) > 0\right] > 1 - \frac{1}{2n}.$$

Observe that $(-1)^{\langle \mathbf{x}, \mathbf{Ru}+\mathbf{e}_i\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i) = (-1)^{x_i}(-1)^{\langle \mathbf{x}, \mathbf{Ru}\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)$, where x_i is the i-th bit of $\mathbf{x}$. The sum above is larger than zero if and only if

$$(-1)^{x_i} = \mathrm{sign}\left(\sum_{\mathbf{u}\in\{0,1\}^k\setminus\{\mathbf{0}\}}(-1)^{\langle \mathbf{x}, \mathbf{Ru}\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)\right).$$

By the claim, this happens for at least a $1 - \frac{1}{2n}$ fraction of random matrices $\mathbf{R}$. Note that if $\mathbf{z} := \mathbf{x}^{\mathrm{T}}\mathbf{R}$ were known, then the bit x_i can easily be computed using the expression above, conditioned on the success of $\mathcal{G}$ in guessing for the input $\mathbf{Ru} + \mathbf{e}_i$. $\mathcal{A}$ iterates through all possible $z \in \{0,1\}^k$, so we get $\mathbf{z} = \mathbf{x}^{\mathrm{T}}\mathbf{R}$ for some guess. Note that the algorithm can do this efficiently because 2^k is only logarithmic in n. Then for this desired $\mathbf{z}$ and for any coordinate i, FFT can be used to compute $h_i(\mathbf{z})$, the Walsh-Hadamard transform of $\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)$. This requires $O(2^k \log_2(2^k)) = O(k2^k)$ operations. Iterating over all n coordinates of $\mathbf{x}$ and by trying all possible $\mathbf{z}$, the total runtime of $\mathcal{A}$ becomes

$$n \cdot 2^k \cdot O(k2^k) = O\left(n \log_2\left(\frac{2n}{c(\mathbf{x})^2}\right)\frac{4n^2}{c(\mathbf{x})^4}\right) = O\left(\frac{n^3}{c(\mathbf{x})^4}\log_2\left(\frac{n}{c(\mathbf{x})^2}\right)\right).$$

Thus, for large enough $c(\mathbf{x})^2$, $\mathcal{A}$ runs in time polynomial in n. Note that the *expected* runtime of $\mathcal{A}$ is polynomial, but $\mathcal{A}$ does not necessarily terminate in polynomial time.

The success of $\mathcal{A}$ depends on k being sufficiently large and the sign of $h_i(\mathbf{z})$ being $(-1)^{x_i}$ for the randomly sampled $\mathbf{R}$. Taking all this into account, the probability that $\mathcal{A}$ succeeds is given by

$$\begin{aligned}
s_{\mathcal{A},f} &= \Pr_{\mathbf{x},w}[\mathcal{A}(f(\mathbf{x}), w) = \mathbf{x}' \in f^{-1}(f(\mathbf{x}))] \\
&= \Pr_{\mathbf{x},w}\left[k > \log_2\left(\frac{2n}{c(\mathbf{x})^2}\right)\right] \cdot \Pr_{\mathbf{R}}\left[\forall i,\ (-1)^{x_i} = \operatorname{sign}\left((-1)^{\langle \mathbf{x}, \mathbf{Ru}\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)\right)\right] \\
&\geq 2c(\mathbf{x})^2\left(\sum_{i=1}^{n}\Pr_{\mathbf{R}}\left[(-1)^{x_i} = \operatorname{sign}\left((-1)^{\langle \mathbf{x}, \mathbf{Ru}\rangle}\mathcal{G}(\mathbf{Ru} + \mathbf{e}_i)\right)\right] - (n-1)\right) \\
&> 2c(\mathbf{x})^2\left(n\left(1 - \frac{1}{2n}\right) - n + 1\right) \\
&= c(\mathbf{x})^2.
\end{aligned}$$

This is exactly the success probability of $\mathcal{G}$ when given input $\mathbf{y}, w$. Therefore, $\mathcal{A}$ and $\mathcal{G}$ have the same success probability bound of $\frac{1}{s}$ via this expected polynomial-time reduction.

□