Codes & Lattices:
Computational Complexity and Constructions

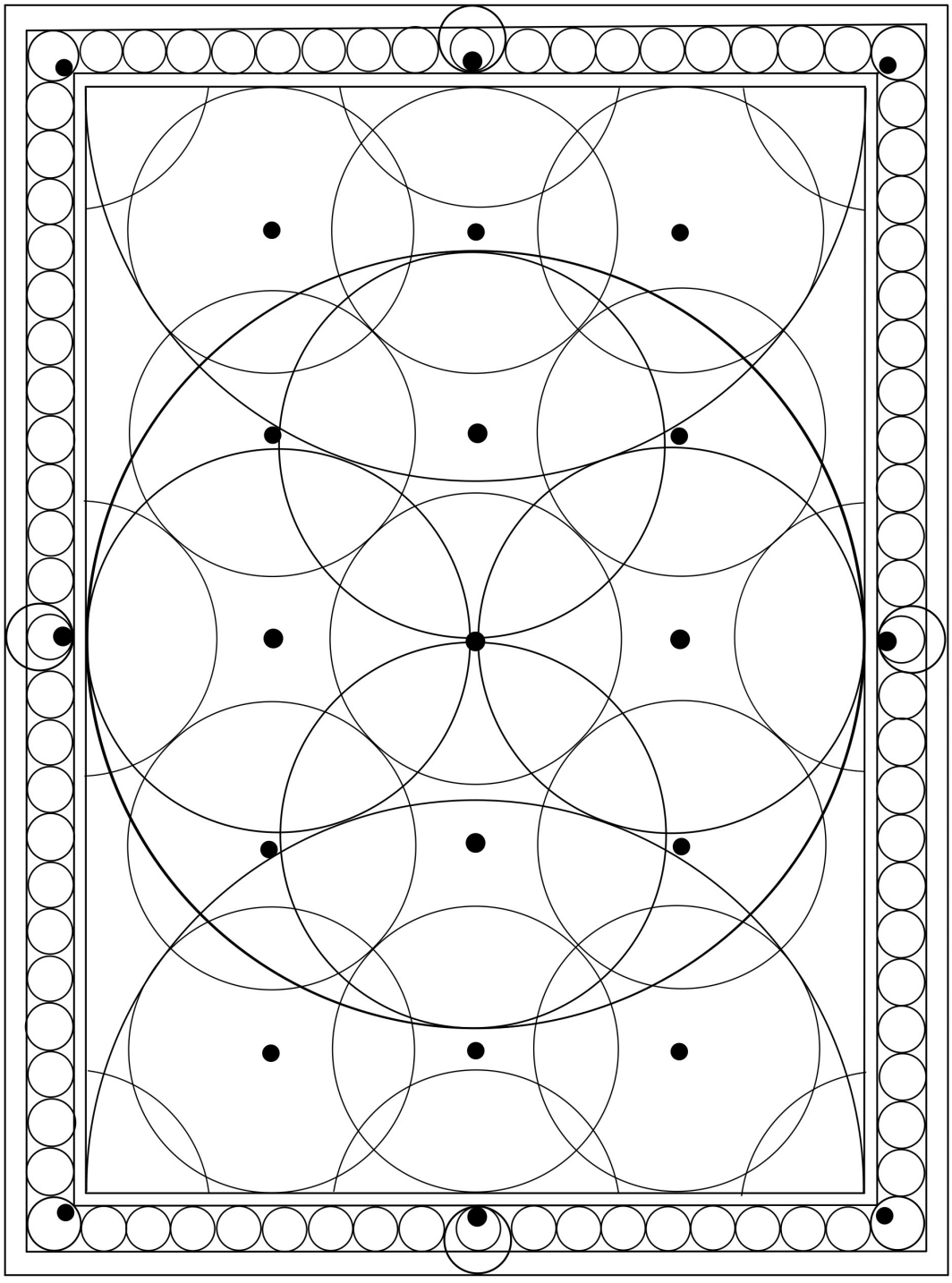
by

Alexandra Veliche Hostetler

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2025

Doctoral Committee:

Associate Professor Mahdi Cheraghchi, Chair
Professor Nikhil Bansal
Associate Professor Viswanath Nagarajan
Professor Chris Peikert
Assistant Professor João Ribeiro, University of Lisbon



Alexandra Veliche Hostetler
aveliche@umich.edu
ORCID iD: 0000-0003-2788-9447

Pentru Bunu

ACKNOWLEDGEMENTS

There are countless people who were instrumental in my education and growth throughout my time in the PhD program at the University of Michigan. In writing this brief acknowledgment, I cannot hope to name them all or express adequate thanks to those that come foremost to mind. Any omission here is due to my own oversight in the present moment (I ask your pardon in advance if you have erroneously not been mentioned). The following is my attempt to convey my gratitude to these people.

First of all, I would like to thank my advisor, Mahdi Cheraghchi, for his guidance. I am very grateful that he accepted me as his student and introduced me to the beautiful realm of coding theory. Throughout my time here, he encouraged me to explore a variety of useful areas, set high expectations, and gave me the freedom to pursue my own intellectual interests. It is largely due to his mentorship that I have gained independence as a researcher.

A special thanks is due to the members of my thesis committee: Mahdi, Nikhil Bansal, Viswanath Nagarajan, Chris Peikert, and João Ribeiro. I appreciate the time they took to listen attentively to my presentations and provide helpful feedback and questions. I am also extremely grateful to those who read through my thesis carefully and provided detailed feedback and suggestions.

I am greatly indebted to my wonderful collaborators, whose work with me has lead to several results discussed in this thesis. João Ribeiro was a patient and experienced guide during my first project as a grad student. Divesh Aggarwal graciously hosted me twice in Singapore – an experience that has been one of the highlights of my graduate years. His persistence when hitting barriers was motivating, and he taught me much about computational complexity. Mahdi and Nikhil Shagrithaya were both fun to work with and our short project turned into a neat result that has left many open questions in its wake. Chris Peikert has been an incredible collaborator and mentor. I am deeply grateful for his patience with my nebulous questions and teaching me by example how to write and think methodically. It is largely due to his work and instruction that I have developed a fascination with lattice theory and cryptography.

I would like to thank the following people for useful discussions on topics related to this thesis and beyond: Alexander Barg, Huck Bennett, Venkata Gandikota, Fernando Granha,

and Daniele Micciancio. I am also grateful to Elena Grigorescu for her kind support, and to Eldon Chung and Dimitrios Myrisiotis for their warm welcome in Singapore.

As a graduate student, I had the opportunity to teach several courses and develop an enthusiasm for teaching. The instructors I worked with include Greg Bodwin, Paul Grubs, Mahdi, Kim Diaz, and the rest of the EECS 203 team. I thank each one for teaching me different approaches to teaching and its challenges.

I would also like to thank my labmates who have shared not only my office space, but also the joys and hardships of graduate student life in the past few years. Among others, these include Nikhil Shagrithaya, Lily Wang, Yi Tang, Dingyu Wang, Gary Hoppenworth, Chaitanya Nalam, Amatya Sharma, Milind Prabhu, Luba Samborska, and Jiwon Kim. I have fond memories of our interesting conversations, research discussions, hard lessons learned, and fun adventures together. Similarly, I thank my flatmate Javiera Jilberto Vallejos.

During my time in Michigan, I have come to know some extraordinary people and make some of the closest relationships I have. It would not be an exaggeration to say that these people have shaped the course of my life over the past few years. I am thankful to all those I have met through the Orthodox Christian Fellowship (OCF) student organization, including Fr. Nick. There are too many others to list here. I am deeply grateful to the nuns of Holy Dormition Monastery, who gave me a second home when I moved away from home, prayed for me constantly, and provided me with a rich practical education in parallel to my graduate studies. I also thank Fr. John for his guidance.

The final acknowledgment goes to my family for their unwavering love and support. My parents and grandparents also provided me with a foundation in mathematics and a thirst to learn and explore the unknown, which has formed the basis of my academic pursuits. My siblings have been the closest friends and conspirators one could wish for. Lastly, I thank my beloved husband Alex. Glory to God for all things!

Alexandra Veliche Hostetler
May 2025

The research presented in this thesis was supported in part by the National Science Foundation under grants CCF-2236931, CCF-2107345, and CCF-2006455 and the (Singapore) National Research Foundation under grant NRF-NRFI09-0005. A number of grants from the Rackham Graduate School and FOCS, ISIT, and TCC conference committees contributed to supporting the travel associated with the research presented herein.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

ABSTRACT

Linear codes and point lattices are two mathematical objects that play a fundamental role in many areas of computer science. In cryptography the computational hardness of problems for codes and lattices is used as a security assumption for most cryptographic schemes proposed to be post-quantum. Understanding the complexity of these problems, and how techniques for codes and lattices are related, is crucial for understanding the security of the corresponding cryptosystems. With this motivation, we explore both the computational complexity and algorithmic sides of coding problems and lattice problems.

We study the computational complexity of two key problems relevant to cryptography: Learning With Errors (LWE) and Code Equivalence (CE). For problems like LWE, we introduce an alternative measure of computational hardness: the maximum success probability achievable by any probabilistic polynomial-time algorithm. This more accurately models the security goals of cryptosystems based on these problems. Under this new perspective, we study the worst-case to average-case hardness of LWE and prove a tight Turing reduction from the Bounded Distance Decoding (BDD) problem to both search and decision variants of LWE. Our reduction improves previous reductions by using only a few oracle calls and explicitly quantifying the loss in success probability. The CE problem has several variants, including Permutation (PCE), Signed Permutation (SPCE), and Linear (LCE) Code Equivalence. We prove polynomial-time Karp reductions from PCE to both LCE and SPCE. Along with a known Karp reduction from SPCE to the Lattice Isomorphism Problem (LIP), our second result implies a reduction from PCE to LIP.

On the algorithmic side, we use lattices and Fourier analytic techniques to construct an algorithm that list-decodes Generalized Reed-Solomon (GRS) codes from worst-case or average-case errors over any ℓ_p (quasi)norm where $0 < p \leq 2$. This is based on the Guruswami-Sudan soft-decision decoding algorithm. Our algorithm generalizes previous algorithms for the ℓ_2 and ℓ_1 norms and achieves a better rate-(decoding) distance trade-off than these algorithms. We also discuss lattice list-decoding capacity bounds for general norms and construct lattices with high density - which achieve the Minkowski bound - using Construction D applied to random linear codes.

CHAPTER 1

Introduction

1.1 A Brief History of Codes & Lattices

Linear codes and point lattices are two mathematical objects that appear throughout cryptography and the theory of error-correcting codes. *Linear codes* are vector subspaces over a finite field, given by the span of a generator matrix. *Lattices* are discrete additive subgroups over the field of real numbers, given by the integer span of a basis matrix. Their close similarity in structure has led to increasing interest in understanding the connections between techniques and computational problems in both areas.

Over the last half-century, computational coding problems and lattice problems have been used as the basis for much of post-quantum cryptography. More precisely, the computational hardness of problems for codes and lattices has been used as a security assumption for most cryptographic schemes proposed to be secure under attacks by quantum computers. Because of this important application and their prevalent use, understanding the complexity of these problems is crucial for understanding the security of cryptosystems based on them. In this thesis, we study the computational hardness of *Learning With Errors* (LWE), a coding problem closely related to several important lattice problems, as well as variants of the *Code Equivalence* (CE) problem. We uncover some new connections between coding and lattice problems.

The theory of error-correcting codes was initiated by Shannon in [Sha48a, Sha48b]. One of its two main branches is that of *channel coding*, which studies how one can communicate reliably over a noisy or unreliable *channel*. In the channel coding model, a *codeword* is transmitted through a channel, which adds some noise to it to produce a perturbed version of the message, called the *received word*. Using a code, the message can be *encoded* before transmission to make it more resilient to noise and allow for reliable *decoding* of the received word. Recovering the exact transmitted message from a given received word is the goal of *unique decoding*. This notion can be relaxed to allow the decoder to output a short *list* of

possible codewords that includes the transmitted message; this is known as *list decoding*. In this thesis, we construct a list-decoding algorithm for a widely-used family of codes, called Generalized Reed-Solomon (GRS) codes, over ℓ_p (quasi)norms.

Just in the last decade, the similarity between codes and lattices has inspired use of lattices for error-correction, particularly for lattice list-decoding. The quality of a lattice for error-correction can be measured in terms of its density in the ambient space under the constraint that the distance between any pair of lattice points is at least one. Several works have constructed lattices with reasonably high density from codes, and provided efficient algorithms for decoding them. In this thesis, we study the conditions sufficient for such a lattice to achieve the highest possible density up to a constant factor and show that these lattices exist using a randomized construction. We also study capacity bounds up to which arbitrary lattices can be list decoded efficiently.

1.2 Our Contributions

In this thesis, we explore the connections between codes and lattice from both a computational complexity and algorithmic perspective.

In Part I, we focus on the computational complexity of coding problems and lattice problems. We specifically study the hardness of the LWE problem – which can be viewed as a coding problem – relative to some well-known lattice problems. We also study the hardness of the CE problem, showing reductions among several useful variants and thereby uncovering a connection to an analogous lattice problem.

In Part II, we focus on algorithms and constructions for codes and lattices. Using the Guruswami-Sudan soft-decision algorithm and techniques from Fourier analysis over lattices, we construct a list-decoding algorithm for Generalized Reed-Solomon codes that works over ℓ_p (quasi)norms. Using bounds from coding theory, we present a sufficient condition needed to construct dense lattices and give a randomized construction that satisfies this with high probability. We also present some probabilistic capacity bounds for list decoding arbitrary lattices over general norms.

Complexity of LWE for Practical Cryptography. Chapter 3 describes our study of the complexity of LWE. The LWE problem is one of the most important computational problems in post-quantum cryptography. Informally, LWE is the problem of decoding a random linear code under some small real-valued error. When Regev introduced this problem in [Reg09], he presented a quantum reduction from a variant of the Shortest Vector Problem for lattices to the Bounded Distance Decoding (BDD) problem for lattices to LWE. Because

LWE is used as a hard problem for cryptographic schemes proposed to be used in practice, we are interested in understanding its hardness under computationally bounded adversaries. Motivated by this practical application, we study the computational complexity of LWE by quantifying the success probability of any (probabilistic) polynomial-time algorithm that attempts to solve it. Under our alternative framework, we adapt reductions from the literature to give a worst-case to average-case Turing reduction from BDD to LWE that makes a single oracle call to the LWE solver and explicitly quantifies the loss in success probability. This reduction is tight in the sense that it gives a lower bound on the success probability of any polynomial-time algorithm for LWE which asymptotically matches the success probability of the best efficient algorithm we know of for LWE.

Complexity of the Code Equivalence Problem. Chapter 4 describes our work on the complexity of CE. The CE problem has found applications in cryptography, and is used as a security assumption for several proposed post-quantum schemes. Informally, CE is the problem of determining whether two codes are equivalent up to some metric-preserving transformation. The variants of the problem specify the type of transformation; for example, Permutation CE (PCE), Signed Permutation CE (SPCE), and Linear CE (LCE). We prove efficient Karp reductions from PCE to LCE and from PCE to SPCE. There is an analogous problem to CE for lattices called the Lattice Isomorphism Problem (LIP). Together with a recent reduction from SPCE to LIP, our results imply a reduction from PCE to LIP.

List-Decoding Algorithm for Generalized Reed-Solomon Codes Chapter 5 describes our work on list-decoding GRS codes over ℓ_p (quasi)norms. Reed-Solomon codes and efficient algorithms for list-decoding them is an active area of interest, as these have many applications in computer science. There has been much work on efficiently decoding GRS codes over the Hamming metric, but only recently has decoding over other metrics and norms been studied. To our knowledge, there are two such algorithms: The first is an algorithm by Roth and Siegel in [RS94] that unique-decodes a special subclass of GRS codes from discrete errors over the Lee metric, which can be viewed as the ℓ_1 norm. The second is a list-decoding algorithm by Mook and Peikert in [MP22] for decoding continuous errors over the ℓ_2 norm. We construct an algorithm that list-decodes GRS codes from continuous errors in the ℓ_p (quasi)norm, where $0 < p \leq 2$. Our algorithm achieves a better rate-distance trade-off than the previous algorithms for the ℓ_1 and ℓ_2 norms. We also consider average-case decoding from errors produced probabilistically.

Conditions and Capacity Bounds for Lattice List-Decoding Chapter 6 describes our study of lattice list-decoding bounds. A popular way of obtaining lattices for list-decoding is to construct them from linear codes. In Section 6.1, we discuss lattice density as a measure of quality for list-decoding and present a sufficient condition for the underlying codes that ensures the corresponding lattice has a high density. We show the existence of such lattices by constructing random linear codes for which the lattice achieves high density with high probability. Motivated by our work on list-decoding over ℓ_p (quasi)norms in Chapter 5, we study analogous lattice list-decoding capacity bounds in Section 6.2.

1.3 A Note on Navigating this Thesis

Given the author's background, this thesis was written primarily with a theoretical computer science audience in mind. For this reason, some familiarity with linear and abstract algebra, probability and combinatorics, algorithmic analysis, and asymptotic notation is assumed of the reader. In an effort to make this thesis accessible and as self-contained as possible, we attempt to provide the mathematical formalism needed to understand the results presented. For reference, a list of definitions and common notation used throughout this thesis is provided in Chapter 2. The individual chapters will often contain concepts and definitions that are only relevant to the work presented in that particular chapter. These are not repeated in Chapter 2. Future directions and open problems are discussed at the end of each chapter. The author hopes that this thesis will be of use to the reader. If the reader finds any errors, please let the author know. Happy reading!

CHAPTER 2

Preliminaries

In this chapter, we introduce the notation and terminology that will be used throughout this thesis. We also recall definitions and some useful facts from coding theory and lattice theory that will be used in later chapters.

2.1 Notation

We use $\mathbb{N}$ to denote the set of natural numbers including zero. For any positive integer n, we write $[n] := \{1, \ldots, n\}$. For any pair of (positive) integers n and m where $n < m$, we use the shorthand $[n, m] := \{n, n+1, \ldots, m-1, m\}$.

Algebraic Notation. For a positive integer p, define the quotient ring $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$ and the additive group $\mathbb{R}_p := \mathbb{R}/p\mathbb{Z}$. For a prime power q, let $\mathbb{F}_q$ denote the finite field of size q. When q is prime, we identify $\mathbb{Z}_q$ with the finite field $\mathbb{F}_q$ in the natural way. We often identify $\mathbb{Z}_q$ with $\mathbb{F}_q$, and with the integers $\{0, 1, \ldots, q-1\}$ using the natural embedding into $\mathbb{Z}$. We will also use $\mathbb{T}$ to denote the additive quotient group $\mathbb{R}/\mathbb{Z}$ of real numbers modulo the integers. For any commutative ring $\mathcal{R}$, we use $\mathcal{R}^*$ to denote its multiplicative subgroup of invertible elements. For any $x \in \mathbb{R}_p$ (which is a coset of $p\mathbb{Z}$), define $\overline{x} \in [-p/2, p/2)$ to be the unique real number such that $\overline{x} = x \pmod{p}$, i.e., the "zero-centered" distinguished representative of x. We also apply this notation entry-wise to vectors over $\mathbb{R}_p$.

For two groups X, Y, their *direct sum* group $X \oplus Y$ is their Cartesian product with the group operation defined component-wise. This notation extends to the direct sum of group *cosets*, which is a coset of the direct sum of the groups.

Linear Algebraic Notation. Vectors are denoted by boldface lowercase letters, such as $\mathbf{v}$. The i-th coordinate of $\mathbf{v}$ is denoted by v_i (not boldface). For any pair of (column) vectors $\mathbf{x}$ and $\mathbf{y}$, we write $\langle \mathbf{x}, \mathbf{y} \rangle$ for the standard dot product given by $\mathbf{x}^{\mathrm{T}}\mathbf{y}$. Matrices are denoted

by boldface uppercase letters, such as $\mathbf{B}$. For any matrix $\mathbf{B}$ and positive integer i, we use $\mathbf{B}[i]$ to denote the i-th column of $\mathbf{B}$. For a pair of positive integers i and j, we use $\mathbf{B}[i, j]$ to denote the entry in the i-th row and j-th column. For positive integers r, r', c, and c' such that $r \le r'$ and $c \le c'$, we will use $\mathbf{B}[r : r', c : c']$ to specify the block submatrix of $\mathbf{B}$ delimited by rows r and r' (inclusive) and columns c and c' (inclusive). For any vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and $0 < p < \infty$, we denote its length in the ℓ_p (quasi)norm by $\|\mathbf{x}\|_p := (x_1^p + \ldots + x_n^p)^{1/p}$. For $p \ge 1$, this is a norm. For $p < 1$, this is a *quasinorm*, which means the triangle inequality axiom is relaxed to require only that $\|\mathbf{x}+\mathbf{y}\| \le \kappa \cdot (\|\mathbf{x}\| + \|\mathbf{y}\|)$ for some fixed κ.

Function Notation. For any function $f\colon D \to \mathbb{C}$ and countable subset $X \subseteq D$, we define $f(X) := \sum_{\mathbf{x}\in X} f(\mathbf{x})$. For any positive $k \in \mathbb{Z}$, we extend the domain to D^k multiplicatively and define

$$f^k(\mathbf{x}) := \prod_{i=1}^{k} f(x_i) \ , \tag{2.1}$$

often omitting the superscript k when it is clear from context. When $D = \mathbb{R}^n$, for any real $s \ne 0$ we define $f_s(\mathbf{x}) := f(\mathbf{x}/s)$. For any two vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ of the same dimension, their coordinate-wise (or *Hadamard*) product is denoted by $\mathbf{x} \odot \mathbf{y} := (x_1 \cdot y_1, \ldots, x_n \cdot y_n)$.

Lemma 2.1. *Let $f\colon D \to \mathbb{R}$ and $X, Y \subseteq D$ be countable subsets of its domain (e.g., lattice cosets). Then*

$$f(X \oplus Y) = f(X) \cdot f(Y) \ .$$

Proof. This follows directly from the definition of direct sum and multiplicativity (Equation (2.1)):

$$f(X \oplus Y) = \sum_{\substack{\mathbf{x}\in X \\ \mathbf{y}\in Y}} f(\mathbf{x} \oplus \mathbf{y}) = \sum_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) \cdot f(\mathbf{y}) = \Big(\sum_{\mathbf{x}} f(\mathbf{x})\Big)\Big(\sum_{\mathbf{y}} f(\mathbf{y})\Big) = f(X) \cdot f(Y) \ .$$

□

Algorithms are often named using calligraphic uppercase letters, such as $\mathcal{A}$, particularly in Chapter 3.

The following function appears in the Gilbert-Varshamov bound (Lemma 6.9).

Definition 2.2 (q-ary Entropy Function)**.** For any integer $q \geq 2$ and $x \in (0,1)$, the *q-ary entropy function* is defined by

$$H_q(x) := x \cdot \log_q(q-1) - x \cdot \log_q(x) - (1-x) \cdot \log_q(1-x) \ .$$

2.2 Coding Definitions

A *linear (error-correcting) code* of *block length* n over the *alphabet* $\mathbb{F}_q$ is a linear subspace $\mathcal{C}$ of $\mathbb{F}_q^n$. As a subspace, $\mathcal{C}$ has a *dimension* k for some integer $0 \leq k \leq n$. We sometimes refer to the *codimension* of a code, defined as $n-k$. The *rate* of a code is given by $R := k/n$. Any linear code can be expressed as the row span of a *generator* matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$. Note that this is not unique, because elementary row operations do not change the span of $\mathbf{G}$. The elements of a code are called *codewords*. The *Hamming distance* between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is the number of coordinates in which they differ and is denoted by $d_H(\mathbf{x}, \mathbf{y}) := |\{i \in [n] : x_i \neq y_i\}|$. The *minimum distance* of a code $\mathcal{C}$ is the minimum Hamming distance between any pair of distinct codewords in $\mathcal{C}$. The trivial code, which only contains $\mathbf{0}$, has minimum distance n by convention. Using the standard notation for linear codes, we say that $\mathcal{C}$ is an $[n, k, d]_q$*-code*. All codes considered in this thesis are linear.

A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is (r, L)-list-decodable if, for any vector $\mathbf{y} \in \mathbb{F}_q^n$, there are at most L codewords within distance r of $\mathbf{y}$, i.e., $|\{\mathbf{c} \in \mathcal{C} : d_H(\mathbf{c}, \mathbf{y}) \leq r\}| \leq L$. We will use a similar notion for lattices (see Section 2.6).

Definition 2.3 ((Generalized) Reed–Solomon code)**.** Let n be a positive integer and q a prime power such that $n \leq q$. For a non-negative integer k, a vector $\boldsymbol{\alpha} \in \mathbb{F}_q^n$ with distinct entries, and a vector $\mathbf{t} \in (\mathbb{F}_q \setminus \{0\})^n$ with (not necessarily distinct) non-zero entries, the *Generalized Reed–Solomon* (GRS) code of dimension k with evaluation points $\boldsymbol{\alpha}$ and twist factors $\mathbf{t}$ is defined as

$$\mathsf{GRS}_{q,k}(\boldsymbol{\alpha}, \mathbf{t}) := \{\mathbf{t} \odot f(\boldsymbol{\alpha}) = (t_1 \cdot f(\alpha_1), \ldots, t_n \cdot f(\alpha_n)) \ : \ f \in \mathbb{F}_p[x], \ \deg(f) < k\} \ .^1$$

A special case of this is a *Reed–Solomon* (RS) code, which is obtained by using trivial twist factors $\mathbf{t} = (1, \ldots, 1)$.

[1] By convention, the zero polynomial has degree $-\infty$.

2.3 Matrix Groups

For any field $\mathbb{F}$ and $n \in \mathbb{N}$, we will consider the following types of $n \times n$ matrices over $\mathbb{F}$. The identity matrix $\mathbf{I}_n$ has ones for the diagonal entries and zeros in all other entries. A *permutation* matrix $\mathbf{P} \in \mathcal{P}_n(\mathbb{F})$ contains exactly one 1 in each row and column and 0s everywhere else. A *signed permutation* matrix $\mathbf{P} \in \mathcal{SP}_n(\mathbb{F})$ contains exactly one non-zero entry in each row and column, and each of these can be either 1 or -1. A *monomial* matrix $\mathbf{M} \in \mathcal{M}_n(\mathbb{F})$ contains exactly one non-zero entry in each row and column, but these can take values in $\mathbb{F}^*$. Any monomial matrix $\mathbf{M}$ can be written as the product $\mathbf{M} = \mathbf{DP}$ of a diagonal matrix $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$, where $d_i \in \mathbb{F}^*$, and a permutation matrix $\mathbf{P} \in \mathcal{P}_n(\mathbb{F})$.

Any permutation matrix $\mathbf{P} \in \mathcal{SP}_n$ defines a bijection over the set of column indices $[n]$; we denote this by $\sigma_{\mathbf{P}} \colon [n] \to [n]$.

Each of these sets of matrices forms a group under matrix multiplication. We use $\mathrm{GL}_n(\mathbb{F})$ to denote the group of invertible matrices, $\mathcal{P}_n(\mathbb{F})$ to denote the set of permutation matrices, $\mathcal{SP}_n(\mathbb{F})$ to denote the set of signed permutation matrices, and $\mathcal{M}_n(\mathbb{F})$ to denote the set of monomial matrices. These groups satisfy $\mathcal{P}_n(\mathbb{F}) \subseteq \mathcal{SP}_n(\mathbb{F}) \subseteq \mathcal{M}_n(\mathbb{F}) \subseteq \mathrm{GL}_n(\mathbb{F})$. If it is clear from context, we do not specify the field for these matrix groups.

2.4 Code Equivalence Problems

Two codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$ are said to be *permutation equivalent* if there exists a permutation of the coordinates of $\mathcal{C}$ that gives $\mathcal{C}'$. We define three variants of this problem in increasing order of generality.

Definition 2.4 (PCE)**.** For positive integers n and k and a field $\mathbb{F}_q$ of size q, the *Permutation Code Equivalence* problem, denoted by PCE, is the following decision problem: Given a pair of generator matrices $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, decide whether there exist an invertible matrix $\mathbf{S} \in \mathrm{GL}_k(\mathbb{F}_q)$ and a permutation matrix $\mathbf{P} \in \mathcal{P}_n(\mathbb{F}_q)$ for which $\mathbf{SGP} = \mathbf{H}$ holds.

Definition 2.5 (SPCE)**.** For positive integers n and k and a field $\mathbb{F}_q$ of size q, the *Signed Permutation Code Equivalence* problem, denoted by SPCE, is the following decision problem: Given a pair of generator matrices $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, decide whether there exist an invertible matrix $\mathbf{S} \in \mathrm{GL}_k(\mathbb{F}_q)$ and a signed permutation matrix $\mathbf{P'} \in \mathcal{SP}_n(\mathbb{F}_q)$ for which $\mathbf{SGP'} = \mathbf{H}$ holds.

Definition 2.6 (LCE)**.** For positive integers n and k and a field $\mathbb{F}_q$ of size q, the *Linear Code Equivalence* problem, denoted by LCE, is the following decision problem: Given a

pair of generator matrices $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, decide whether there exist an invertible matrix $\mathbf{S} \in \mathrm{GL}_k(\mathbb{F}_q)$ and a monomial matrix $\mathbf{M} \in \mathcal{M}_n(\mathbb{F}_q)$ for which $\mathbf{SGM} = \mathbf{H}$ holds.

For brevity, we say that a pair of matrices $(\mathbf{G}, \mathbf{H})$ is in PCE (or SPCE, LCE) if $\mathbf{G}$ and $\mathbf{H}$ satisfy the conditions required for the pair to be a YES instance of the problem. (Note that this treats the decision problem as a language.)

The following is an analogous problem to CE for lattices. Here $\mathcal{O}_n(\mathbb{R})$ denotes the set of orthogonal $n \times n$ matrices over the reals.

Definition 2.7 (LIP)**.** For any positive integer n, the *Lattice Isomorphism Problem*, denoted LIP, is the following decision problem: Given a pair of basis matrices $\mathbf{B}, \mathbf{B}' \in \mathbb{R}^{k \times n}$, decide whether there exists an invertible matrix $\mathbf{S} \in \mathbb{R}^{k \times k}$ and an orthogonal matrix $\mathbf{O} \in \mathcal{O}_n(\mathbb{R})$ for which $\mathbf{SBO} = \mathbf{B}'$.

2.5 Learning With Errors

In this section, we define a matrix problem that is closely related to the computational lattice problems in Section 2.7. Let n be a positive integer, p be a prime modulus, and ϕ be a distribution over $\mathbb{T}$.

Definition 2.8 (LWE Distribution)**.** For any distribution ϕ over $\mathbb{T}$ and secret vector $\mathbf{s} \in \mathbb{Z}_p^n$, the *Learning with Errors (LWE) distribution*, denoted $A_{\mathbf{s},\phi}$, 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 sampling e from ϕ, then outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle / p + e)$.

The standard Learning with Errors problem has both search and decision variants.

Definition 2.9 (Search-LWE)**.** The *search*-$\mathsf{LWE}_{n,p,\phi}$ problem is defined as: Given a polynomial number (in n) of samples from the LWE distribution $A_{\mathbf{s},\phi}$, output the secret vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Definition 2.10 (Decision-LWE)**.** The *decision*-$\mathsf{LWE}_{n,p,\phi}$ problem is defined as: Given a polynomial number (in n) of samples either drawn from the distribution $A_{\mathbf{s},\phi}$ or independently drawn from the uniform distribution over $\mathbb{Z}_p^n \times \mathbb{T}$, 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 2.11 (Binary-LWE)**.** The $\mathsf{binLWE}_{n,p,\phi}$ problem is the search-$\mathsf{LWE}_{n,p,\phi}$ problem restricted to binary secret vectors, i.e., where $\mathbf{s}$ is uniform over the set $\{0,1\}^n$.

2.6 Lattice Definitions

A *lattice* is an infinite set of periodically-arranged points in Euclidean space that contains the origin. More specifically, it is a discrete additive subgroup of Euclidean space. It can be described as the set of all integer linear combinations of a finite set of linearly independent vectors.

Definition 2.12 (Lattice, Basis)**.** An *(n-dimensional, m-rank) lattice* $\mathcal{L} \subset \mathbb{R}^n$ is the set of all integer linear combinations of some m linearly independent *basis* vectors $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{R}^n$:

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

A lattice basis $\mathcal{B}$ can equivalently be expressed as a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ whose columns are the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$, so we also 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.

Note that a given lattice has multiple different bases, which are all related by right-multiplication by *unimodular* matrices in $\mathbb{Z}^{n \times n}$. Any lattice that is a subset of a lattice $\mathcal{L}$ is called a *sublattice* of $\mathcal{L}$. A sublattice of $\mathbb{Z}^n$ is called an *integer lattice*.

From here onward, we assume that all lattices are full rank, i.e., $n = m$. In this case, a lattice is a discrete additive subgroup of $\mathbb{R}^n$ whose $\mathbb{R}$-span is $\mathbb{R}^n$; as such, it defines the quotient group $\mathbb{R}^n/\mathcal{L}$ of *lattice cosets* $\mathbf{x} + \mathcal{L}$ for $\mathbf{x} \in \mathbb{R}^n$.

Definition 2.13 (Determinant)**.** The *determinant* of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ generated by $\mathbf{B} \in \mathbb{R}^{n \times n}$ is given by

$$\det(\mathcal{L}) := |\det(\mathbf{B})| .$$

Note that the determinant of a lattice is invariant under the choice of basis, by the above-mentioned relationship between the bases of a lattice.

Definition 2.14 (Dual Lattice)**.** The *dual lattice* of $\mathcal{L} \subset \mathbb{R}^n$ is

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

Any basis $\mathbf{B} \in \mathbb{R}^{n \times n}$ for $\mathcal{L} = \mathcal{L}(\mathbf{B})$ has a unique *dual basis* $\mathbf{B}^* \in \mathbb{R}^{n \times n}$ that satisfies $(\mathbf{B}^*)^{\mathrm{T}}\mathbf{B} = \mathbf{I}_m$. This $\mathbf{B}^*$ is a basis for $\mathcal{L}^* = \mathcal{L}(\mathbf{B}^*)$.

Definition 2.15 (Successive Minima)**.** For any positive integer k, the *k-th successive minimum* of a lattice $\mathcal{L} \subset \mathbb{R}^n$ with respect to a specified (quasi)norm is

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

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

Definition 2.16 (Unique Closest Lattice Vector)**.** For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and vector $\mathbf{v} \in \mathbb{R}^n$ whose distance from $\mathcal{L}$ is less than $\lambda_1(\mathcal{L})/2$, we denote the unique closest lattice vector to $\mathbf{v}$ by $\kappa_{\mathcal{L}}(\mathbf{v})$.

A natural method to construct a lattice from a linear code was introduced in [BS83]. Given an length-n code over a finite field of size prime p, this construction first embeds the codewords into Euclidean space by mapping them to points in the hypercube $[0, p)^n$, then tiles $\mathbb{R}^n$ with copies of this set of points.

Definition 2.17 (Construction A Lattice)**.** For any positive integer n, prime p, and linear code $\mathcal{C} \subseteq \mathbb{F}_p^n$, let $\widetilde{\mathcal{C}} \subset \mathbb{Z}^n$ denote the set of vectors obtained by lifting the codewords in $\mathcal{C}$ to the integers using the natural identification of the elements of $\mathbb{F}_p$ with the integers $\{0, 1, \ldots, p-1\} \subset \mathbb{Z}$. The *Construction A lattice* associated with $\mathcal{C}$ is

$$\mathcal{L}_A(\mathcal{C}) := \widetilde{\mathcal{C}} + p\mathbb{Z}^n \subseteq \mathbb{Z}^n \ .$$

When it is clear from context, we often drop the tilde and implicitly associate the code $\mathcal{C}$ with its representative set $\widetilde{\mathcal{C}}$.

This construction can be generalized to construct a lattice from a *tower* of nested linear codes instead of a single code. The original definition was given in [CS88], but here we give a simpler formulation that was more recently defined in [MKO18].

Definition 2.18 (Construction D Lattice)**.** For any positive integer n, prime p, and tower of nested linear codes $\mathcal{C}_\ell \subseteq \cdots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_p^n$, where $\mathcal{C}_i$ is a $[n, k_i, d_i]_p$-code for every $i \in [\ell]$, define the upper-triangular matrix

$$\mathbf{G} := \left[\begin{array}{c} \mathbf{G}_\ell \\ \hline \mathbf{G}_{\ell-1} \\ \hline \vdots \\ \hline \mathbf{G}_0 \end{array}\right] \in \mathbb{F}_p^{n \times n},$$

such that the diagonal entries are all 1, $\mathbf{G}_0$ consists of a full-rank identity matrix in the right-most block and zeros in all other entries, and the rows of $\mathbf{G}_\ell, \ldots, \mathbf{G}_i$ generate $\mathcal{C}_i$ for every $i \in [\ell]$. Scale the block matrices and use the natural identification of $\mathbb{F}_p$ with the

integers to define the matrix

$$\mathbf{B} := \begin{bmatrix} \mathbf{G}_\ell \\ \hline p\mathbf{G}_{\ell-1} \\ \hline \vdots \\ \hline p^\ell \mathbf{G}_0 \end{bmatrix} \in \mathbb{Z}^{n \times n}.$$

The *Construction D lattice* is the lattice generated by this basis matrix $\mathbf{B}$:

$$\mathcal{L}_D(\{\mathcal{C}_i\}_{i\in[\ell]}) := \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n \ .$$

For a real $r > 0$, positive integer L, and distance function $d(\cdot,\cdot)$, we say that a lattice $\mathcal{L} \subset \mathbb{R}^n$ is (r, L)-list-decodable if, for any vector $\mathbf{y} \in \mathbb{R}^n$, there are at most L lattice vectors within distance r of $\mathbf{y}$, i.e., $|\{\mathbf{x} \in \mathcal{L} : d(\mathbf{x}, \mathbf{y}) \le r\}| \le L$.

2.7 Computational Lattice Problems

The following two problems are considered to be the most important computational lattice problems.

Definition 2.19 (SVP)**.** The *Shortest Vector Problem*, denoted SVP, is the search problem defined as: Given a basis matrix $\mathbf{B}$ for a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$, output a shortest non-zero lattice vector, i.e., an $\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}$ such that $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$.

Definition 2.20 (CVP)**.** The *Closest Vector Problem*, denoted CVP, is the search problem defined as: Given a basis matrix $\mathbf{B}$ for a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and target vector $\mathbf{t} \in \mathbb{R}^n$, output a lattice vector closest to the target, i.e., an $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{x} - \mathbf{t}\| = \mathrm{dist}(\mathcal{L}, \mathbf{t})$.

We will focus on the following variants of these problems, specifically in Chapter 3.

Definition 2.21 (GapSVP)**.** For any approximation factor $\gamma \ge 1$, the γ-*approximate Shortest Vector Problem*, GapSVP_γ, is the decision problem defined as: Given an instance $(\mathbf{B}, d)$ consisting of a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and distance parameter $d > 0$, with the promise that either $\lambda_1(\mathcal{L}) \le d$ or $\lambda_1(\mathcal{L}) > \gamma \cdot d$, output

- YES if $\lambda_1(\mathcal{L}) \le d$, or
- NO if $\lambda_1(\mathcal{L}) > \gamma \cdot d$.

Definition 2.22 (BDD)**.** For any promise factor $\gamma \in (0, \frac{1}{2})$, the *Bounded Distance Decoding problem*, BDD_γ, is the promise search problem defined as: Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and a target vector $\mathbf{v} \in \mathbb{R}^n$ with the promise that $\mathrm{dist}(\mathbf{v}, \mathcal{L}) < \gamma \cdot \lambda_1(\mathcal{L})$, output the unique lattice vector closest to $\mathbf{v}$, i.e., the $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{x}\| < \gamma \cdot \lambda_1(\mathcal{L})$.

Note that the upper bound on the factor $\alpha < \frac{1}{2}$ guarantees the existence of a *unique* solution.

2.8 Fourier Analysis

In this section we recall some basics of Fourier analysis, restricted to well-behaved functions. Let $f\colon \mathbb{R}^n \to \mathbb{C}$ be a (Borel) measurable function that satisfies $\int_{\mathbb{R}^n} |f(\mathbf{x})|\,\mathrm{d}\mathbf{x} < \infty$. Its *Fourier transform* $\widehat{f}\colon \mathbb{R}^n \to \mathbb{C}$ is defined as

$$\widehat{f}(\mathbf{w}) := \int_{\mathbb{R}^n} f(\mathbf{x}) \cdot \exp(-2\pi i \langle \mathbf{x}, \mathbf{w} \rangle)\,\mathrm{d}\mathbf{x} \ .$$

It satisfies the following standard properties, which follow by routine calculations.

Lemma 2.23 (Multiplicativity)**.** *For any function f as above, $\widehat{f^k} = \widehat{f}^{\,k}$ (where the exponent notation is as defined in Equation* (2.1)*).*

Lemma 2.24 (Time-scaling property)**.** *For any function f as above and real $s \neq 0$,*

$$\widehat{f_s}(\mathbf{w}) = s^n \cdot \widehat{f}_{1/s}(\mathbf{w}) \ .$$

Lemma 2.25 (Time-shift property)**.** *For any function f as above and $\mathbf{c} \in \mathbb{R}^n$, let $g(\mathbf{x}) = f(\mathbf{x} - \mathbf{c})$. Then*

$$\widehat{g}(\mathbf{w}) = \widehat{f}(\mathbf{w}) \cdot \exp(-2\pi i \langle \mathbf{w}, \mathbf{c} \rangle) \ .$$

We say that f is *nice* if it satisfies conditions that are sufficient for the following formula to hold, e.g., those given in [Ser73, pages 106–107]. All of the specific functions f we use in this work are easily seen to be nice.

Lemma 2.26 (Poisson Summation Formula (PSF))**.** *For any lattice $\mathcal{L}$ and nice function f,*

$$f(\mathcal{L}) = \det(\mathcal{L}^*) \cdot \widehat{f}(\mathcal{L}^*) \ .$$

We will use a more general version of the PSF for lattice cosets.

Lemma 2.27 (Generalized PSF)**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, nice function f, and $\mathbf{y} \in \mathbb{R}^n$,*

$$f(\mathbf{y} + \mathcal{L}) = \det(\mathcal{L}^*) \cdot \sum_{\mathbf{w} \in \mathcal{L}^*} \widehat{f}(\mathbf{w}) \cdot \exp(2\pi i \langle \mathbf{w}, \mathbf{y} \rangle) \ .$$

Proof. Define the function $g(\mathbf{x}) := f(\mathbf{x} + \mathbf{y})$. By Lemmas 2.25 and 2.26,

$$\begin{aligned} f(\mathbf{y} + \mathcal{L}) &= g(\mathcal{L}) \\ &= \det(\mathcal{L}^*) \cdot \widehat{g}(\mathcal{L}^*) \\ &= \det(\mathcal{L}^*) \cdot \sum_{\mathbf{w} \in \mathcal{L}^*} \widehat{g}(\mathbf{w}) \\ &= \det(\mathcal{L}^*) \cdot \sum_{\mathbf{w} \in \mathcal{L}^*} \widehat{f}(\mathbf{w}) \cdot \exp(2\pi i \langle \mathbf{w}, \mathbf{y} \rangle) \ . \end{aligned}$$

□

2.9 Probability and Distributions

If a random variable X has distribution ϕ over its support space, we often use the shorthand notation $X \sim \phi$. We use $N(\mu, \sigma^2)$ to denote the standard normal distribution over the real numbers with mean μ and variance σ^2.

Lemma 2.28 (Chebyshev's Inequality)**.** *For any random variable X with expected value $\mu = \mathbb{E}[X]$ and finite variance $\mathrm{Var}[X]$, and $\gamma > 0$,*

$$\Pr\big[|X - \mu| \geq \gamma\big] \leq \frac{\mathrm{Var}[X]}{\gamma^2} \ .$$

The following corollary will be useful for our reductions in Chapter 3.

Lemma 2.29. *For any $p \in [0, 1]$, let $Y_1, \ldots, Y_t$ be pairwise independent Bernoulli random variables over $\{0, 1\}$ such that $\Pr[Y_i = 1] = p$ for all $i \in [t]$. Then for any $c > 0$,*

$$\Pr\left[\left|\sum_{i=1}^{t} Y_i - t \cdot p\right| \leq ctp\right] \geq 1 - \frac{1}{c^2 tp} \ .$$

Proof. Define $Y = Y_1 + \cdots + Y_t$. Its expected value is $\mathbb{E}[Y] = tp$ and its variance is $\mathrm{Var}[Y] = tp(1-p)$. By the Chebyshev inequality (Lemma 2.28),

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

□

For a finite sequence of values $X_1, \ldots, X_n \in \mathbb{R}$, we denote their average by $\mathrm{Avg}_i[X_i] := \frac{1}{n}\sum_{i=1}^{n} X_i$. We use the following special case of the well-known Hoeffding (lower-)tail bound in Chapter 5.

Lemma 2.30 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent identically distributed random variables in $[0,1]$ with common expectation $\mu = \mathbb{E}[X_i]$. Then for any $\gamma \geq 0$,*

$$\Pr\Big[\operatorname*{Avg}_i[X_i] \leq \mu - \gamma\Big] < \exp(-2\gamma^2 n) \ .$$

Definition 2.31 (Statistical Distance)**.** For any set of outcomes X, collection of events $\mathcal{F}$, and two continuous probability distributions with probability density functions ϕ_1 and ϕ_2, 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 $X = \mathbb{R}^n$ this is $\Delta(\phi_1, \phi_2) = \frac{1}{2}\int_{\mathbb{R}^n} |\phi_1(\mathbf{x}) - \phi_2(\mathbf{x})| \, \mathrm{d}\mathbf{x}$.

If X_1 and X_2 are random variables with distributions ϕ_1 and ϕ_2, respectively, we define $\Delta(X_1, X_2) := \Delta(\phi_1, \phi_2)$. Note that, by definition, the statistical distance satisfies the triangle inequality. Another important property is that statistical distance does not increase under the application of any function, as formalized below.

Fact 2.32 (adapted from [Vad12, Lemma 6.3])**.** *For any random variables X_1 and X_2 with probability distributions ϕ_1 and ϕ_2, respectively, and any function f (whose domain is compatible),*

$$\Delta(f(X_1), f(X_2)) \leq \Delta(X_1, X_2) \ .$$

A useful consequence of this fact is that for any algorithm $\mathcal{A}$, the success probability of $\mathcal{A}$ on X_1 differs from that on X_2 by at most $\Delta(X_1, X_2)$.

Definition 2.33 (Gaussian Function)**.** The *Gaussian function* is $\rho : \mathbb{R}^n \to \mathbb{R}$, given by

$$\rho(\mathbf{x}) := \exp(-\pi \|\mathbf{x}\|^2) \ .$$

Using function notation from Section 2.1, for any $\mathbf{y} \in \mathbb{R}^n$ and width parameter $s > 0$, we define $\rho_{s,\mathbf{y}}(\mathbf{x}) := \rho_s(\mathbf{x} - \mathbf{y})$. For simplicity, we denote $\rho := \rho_1$.

Note that while we use this Gaussian function in both Chapters 3 and 5, we make use of this specific notation in Chapter 3 for convenience, since we do not use more general functions there as in Chapter 5.

Definition 2.34 (Discrete Gaussian)**.** For any countable subset A and width $s > 0$ 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 2.35 (Distribution Ψ_γ)**.** For any positive $\gamma \in \mathbb{R}$, the distribution $\Psi_\gamma : \mathbb{T} \to \mathbb{R}$ is defined by

$$\Psi_\gamma(r) := \frac{1}{\gamma} \sum_{k \in \mathbb{Z}} \rho_\gamma(r - k) .$$

In particular, if X is distributed according to Ψ_γ and Z has distribution $N(0, \gamma^2/(2\pi))$, then X is the image of Z modulo 1.

An important property that will be used in Chapter 3 is that Ψ_α does not change much under a small change in the width parameter α. In particular, a small scaling of α to β gives an upper bound on the ratio between the probability density functions corresponding to Ψ_α and Ψ_β.

Lemma 2.36. *For any $\alpha > 0$ and $\varepsilon > 0$ defining $\beta := \alpha(1 + \varepsilon)$, let g_α and g_β denote the probability density functions for distributions Ψ_α and Ψ_β, respectively. Then for any $x \in \mathbb{R}$,*

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

Proof. Suppose X and X' are distributed according to Ψ_α and Ψ_β, respectively. By definition, X and X' are the images of some Z and Z' with distribution $N(0, \alpha^2/(2\pi))$ and $N(0, \beta^2/(2\pi))$, respectively, under 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 mod 1 and Z' mod 1. By definition of β and positivity of ε,

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

The inequality follows from the observation that the exponential factor is a non-zero width-scaling of the Gaussian function ρ_α, which is bounded above by 1. □

We prove a multiplicative analog of Fact 2.32 for this ratio of probability density functions.

Lemma 2.37. *Let X and Y be continuous random variables over some universe U with probability density functions g_X and g_Y, respectively, such that for some fixed $\delta > 0$, their ratio satisfies $g_X(x)/g_Y(x) \leq \delta$ for all x. Then for any invertible function $f : U \to V$ and*

$S \subseteq V$,

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

Proof. Consider the ratio $\Pr[X \in T]/\Pr[Y \in T]$ over all sets $T \subseteq U$. This is maximized by the set of all elements for which the ratio of probabilities is positive, i.e

$$T^* := \left\{ u \in U : \frac{\Pr[X = u]}{\Pr[Y = u]} > 0 \right\} .$$

Then by definition of T^* and the hypothesis, we have

$$\frac{\Pr[f(X) \in S]}{\Pr[f(Y) \in S]} = \frac{\Pr[X \in f^{-1}(S)]}{\Pr[Y \in f^{-1}(S)]} \leq \frac{\Pr[X \in T^*]}{\Pr[Y \in T^*]} = \frac{\int_{T^*} g_X(x)\,\mathrm{d}x}{\int_{T^*} g_Y(x)\,\mathrm{d}x} \leq \delta \ .$$

□

This statement can easily be extended to a randomized function f. Consequently, for any algorithm $\mathcal{A}$, the success probability of $\mathcal{A}$ on X differs from that on Y by at most a multiplicative factor of δ. We will use Lemmas 2.36 and 2.37 in Chapter 3.

2.10 Lattice Smoothing & Roughness

In this section, we describe two useful lattice quantities that will appear throughout this thesis. 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 the positive reals which approaches 0 as s grows, and approaches ∞ as s approaches 0. This enables us to define the following parameter, which was originally introduced in [MR04].

Definition 2.38 (Smoothing Parameter)**.** For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon > 0$, the *smoothing parameter* of $\mathcal{L}$ with respect to ε is

$$\eta_\varepsilon(\mathcal{L}) := \inf\{s > 0 : \rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \varepsilon\} \ .$$

This is the smallest parameter s that makes the function $f_s(\mathbf{y} + \mathcal{L})$ sufficiently "smooth" as a function of $\mathbf{y}$.

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, this map is a bijection over the positive real numbers whose inverse is the map $\varepsilon \mapsto \eta_\varepsilon(\mathcal{L})$.

Lemma 2.39. *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\varepsilon > 0$, and scalar $c > 0$,*

$$c \cdot \eta_\varepsilon(\mathcal{L}) = \eta_\varepsilon(c\mathcal{L}) \ .$$

Proof. By the properties of the Gaussian function and the fact that the dual of a scaled lattice satisfies $(c\mathcal{L})^* = c^{-1}\mathcal{L}^*$,

$$\begin{aligned} c \cdot \eta_\varepsilon(\mathcal{L}) &= \inf\{c \cdot s > 0 : \rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \varepsilon\} \\ &= \inf\{t > 0 : \rho_{c/t}(\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \varepsilon\} \\ &= \inf\{t > 0 : \rho_{1/t}(t^{-1}\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \varepsilon\} \\ &= \eta_\varepsilon(c\mathcal{L}) \ . \end{aligned}$$

□

Lemma 2.40 ([Reg09, Claim 2.13])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon > 0$,*

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

Lemma 2.41 (adapted from [GPV08, Lemma 3.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ and $\varepsilon > 0$,*

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

We remark that the original upper bound in [GPV08] is tighter than the one above. This is because the maximum is taken over the Gram-Schmidt orthogonalization of the basis vectors, $\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_n$, which satisfy $\|\widetilde{\mathbf{b}}_i\| \leq \|\mathbf{b}_i\|$ for all $i \in [n]$. Furthermore, this quantity is minimized over all possible bases $\mathbf{B}$ of the lattice.

Lemma 2.42 (adapted from [Cai98, Theorem 3.2])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ with basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$,*

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

The above lower bound is looser than the original result in [Cai98]. Similar to the previous lemma, we use the upper bound $\|\widetilde{\mathbf{b}}_i\| \leq \|\mathbf{b}_i\|$ for all $i \in [n]$, which yields a weaker bound than the one given by the Gram-Schmidt orthogonolization of the basis vectors minimized over all possible bases of the lattice. For our purposes, the weaker versions of these bounds are sufficient.

For widths greater than the smoothing parameter, the output of the Gaussian function for any lattice coset can be bounded by the determinant of the dual lattice and a power of the width parameter.

Lemma 2.43 ([Reg09, Claim 3.8])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$, $\varepsilon > 0$, and $r \geq \eta_\varepsilon(\mathcal{L})$,*

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

For the functions in Chapter 5, we require the following properties of f.

Assumption 2.44. The function f has range $[0, 1]$ and is nice, and $\widehat{f}$ is *non-negative real* with $\widehat{f}(0) > 0$.

Because f is real, its Fourier transform is conjugate symmetric, i.e., $\widehat{f}(-w) = \widehat{f}(w)^*$ for all w, where the star denotes complex conjugation. Since $\widehat{f}$ is also real, this implies that it is symmetric, i.e., $\widehat{f}(-w) = \widehat{f}(w)$. Finally, note that if f satisfies this assumption, then so does its multiplicative extension f^k.

For these kinds of functions, we introduce an important Fourier-analytic quantity that plays an important role in our analysis in Chapter 5. We adopt the name "roughness" because it is the functional inverse of the smoothing parameter defined above.

Definition 2.45 (Roughness)**.** For a function f, lattice $\mathcal{L} \subset \mathbb{R}^n$, and real $s > 0$, the *roughness* is defined as

$$\varepsilon_{\mathcal{L},s} := \frac{\widehat{f}_s(\mathcal{L}^* \setminus \{\mathbf{0}\})}{\widehat{f}_s(\mathbf{0})} = \frac{\widehat{f}_s(\mathcal{L}^*)}{\widehat{f}_s(\mathbf{0})} - 1 \geq 0. \tag{2.2}$$

More generally, for a (linear) subspace H of $\mathbb{R}^n$, the *H-roughness* is defined as

$$\varepsilon_{\mathcal{L},s}(H) := \frac{\widehat{f}_s(\mathcal{L}^* \setminus H^\perp)}{\widehat{f}_s(\mathcal{L}^* \cap H^\perp)} = \frac{\widehat{f}_s(\mathcal{L}^*)}{\widehat{f}_s(\mathcal{L}^* \cap H^\perp)} - 1 \leq \varepsilon_{\mathcal{L},s}(\mathbb{R}^n) = \varepsilon_{\mathcal{L},s}. \tag{2.3}$$

Here both inequalities follow from the non-negativity of $\widehat{f}_s$.

The following bounds the output of any function f for a lattice coset in terms of the dual lattice and roughness. Note the close similarity to Lemma 2.43 above.

Lemma 2.46 (adapted from [MR07, Lemmas 2.9 and 4.1])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, real $s > 0$, and subspace H of $\mathbb{R}^n$ defining roughness $\varepsilon := \varepsilon_{\mathcal{L},s}(H)$, and any $\mathbf{y} \in H$,*

$$f_s(\mathbf{y} + \mathcal{L}) \in \det(\mathcal{L}^*) \cdot \widehat{f}_s(\mathcal{L}^* \cap H^\perp) \cdot [1 - \varepsilon, 1 + \varepsilon] \ ,$$

with equality against the upper bound when $\mathbf{y} = \mathbf{0}$. *In particular,* $f_s(\mathbf{y}+\mathcal{L}) \in f_s(\mathcal{L}) \cdot [\frac{1-\varepsilon}{1+\varepsilon}, 1]$.

Proof. By the generalized PSF (Lemma 2.27),

$$\begin{aligned} f_s(\mathbf{y}+\mathcal{L}) &= \det(\mathcal{L}^*) \cdot \sum_{\mathbf{w}\in\mathcal{L}^*} \widehat{f}_s(\mathbf{w}) \cdot \exp(2\pi i \langle \mathbf{w}, \mathbf{y}\rangle) \\ &= \det(\mathcal{L}^*) \cdot \left(\widehat{f}_s(\mathcal{L}^* \cap H^\perp) + \sum_{\mathbf{w}\in\mathcal{L}^*\setminus H^\perp} \widehat{f}_s(\mathbf{w}) \cdot \exp(2\pi i \langle \mathbf{w}, \mathbf{y}\rangle \right) \\ &= \det(\mathcal{L}^*) \cdot \left(\widehat{f}_s(\mathcal{L}^* \cap H^\perp) + \sum_{\mathbf{w}\in\mathcal{L}^*\setminus H^\perp} \widehat{f}_s(\mathbf{w}) \cdot \cos(2\pi \langle \mathbf{w}, \mathbf{y}\rangle \right) . \end{aligned}$$

The last equation follows from the symmetry of $\widehat{f}_s$ and by pairing each (non-zero) element of $\mathcal{L}^* \setminus H^\perp$ with its negation, which cancels out the imaginary part of $\exp(2\pi i \langle \mathbf{w}, \mathbf{y}\rangle)$.

Now observe that $\widehat{f}_s(\mathbf{w}) \cdot \cos(2\pi \langle \mathbf{w}, \mathbf{y}\rangle) \in [-\widehat{f}_s(\mathbf{w}), \widehat{f}_s(\mathbf{w})]$, with equality against the upper bound for $\mathbf{y} = \mathbf{0}$, because $\widehat{f}_s$ is non-negative. The claim then follows by the definition of roughness $\varepsilon_{\mathcal{L},s}(H)$. □

CHAPTER 3

Learning With Errors

Based on joint work with Divesh Aggarwal and Leong Jin-Ming in [ALV25].

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 [KS09]. The LWE problem is the task of recovering a secret vector from a set of noisy linear equations (see Definition 2.9 for the formal definition). It can also be seen as the problem of decoding a random linear code. In particular, given an input consisting of a uniformly random matrix $m \times n$ matrix $\mathbf{A}$ with entries in $\mathbb{Z}_p$ and the vector $\mathbf{b} := \mathbf{As} + \mathbf{e}$ determined by a secret uniformly random vector $\mathbf{s}$ and short error vector $\mathbf{e}$ sampled from a Gaussian distribution, the goal is to output the secret vector $\mathbf{s}$. Here the positive integer p is called the modulus, n is the dimension of this LWE problem, and m is the number of samples (polynomial in the dimension n). In Regev's seminal work, he relates LWE to worst-case lattice problems that form the foundation of lattice-based cryptography.

Lattice Problems. 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. In the approximation variant of SVP, denoted by γ-SVP for some approximation factor $\gamma \geq 1$, the goal is to output a short non-zero lattice vector whose length is at most γ times the shortest lattice vector length. In CVP, one is given a target vector and lattice basis 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 shortest distance between the target vector and the lattice. It is known how to reduce SVP

to CVP in polynomial time, while preserving the dimension, rank, and approximation factor [GMSS99]. A closely related problem to CVP is the *Bounded Distance Decoding* (BDD) problem, denoted by BDD_γ for some promise factor $\gamma < \frac{1}{2}$. This is a promise problem in which the goal is to solve CVP under the guarantee that the distance of the target from the lattice is at most γ times the minimum distance between lattice points.

These computational lattice problems, among others, are crucial because of their association with lattice-based cryptography. Specifically, the security of numerous cryptographic systems such as those in [Ajt96, MR04, Reg06, MR09, Reg09, Gen09, BV14] relies on the complexity of solving lattice problems to within a polynomial approximation factor. Aside from their importance in designing cryptosystems, algorithms for solving lattice problems have found applications since the 1980s 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 $\mathrm{poly}(n)$ in the lattice dimension 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 solving these problems up to approximation factor $\gamma = n^c$ for constant $c \geq 0$ is approximately $2^{n/(c+1)}$. For the current state-of-the-art algorithms, we refer the reader to [ALS20].

Hardness of Lattice Problems. There are many results for the hardness of γ-CVP and γ-SVP for various approximation factors γ and distances measured in various ℓ_p norms [vEB81, DKRS03, Din02, Kho05, HR07, Mic12], though we only focus on the Euclidean (ℓ_2) norm for this work. γ-CVP is known to be NP-hard for nearly-polynomial approximation $\gamma = n^{c/\log\log n}$ for some constant $c > 0$ under deterministic reductions [DKRS03]. Solving γ-SVP for such approximation factors cannot be done in polynomial time under the assumption that $\mathsf{NP} \not\subseteq \mathsf{RSUBEXP}$ [HR07]. Exact γ-SVP (where $\gamma = 1$) was shown to be NP-hard under randomized reductions [Ajt98] and unsolvable in polynomial time for constant approximation factors under the assumption that $\mathsf{NP} \not\subseteq \mathsf{RP}$ [Kho05]. 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. It remains an open problem

to prove NP-hardness of γ-SVPunder deterministic reductions, even for the exact version.

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 (like the vector $\mathbf{b}$ described above) and a uniformly random sample from $\mathbb{Z}_p^{m\times(n+1)}$. Regev gave a polynomial-time reduction from BDD to decision-LWE [Reg09]. He also 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. In a subsequent work, Peikert [Pei09] gave a fully classical reduction from GapSVP_γ to decision-LWE, albeit with the modulus p becoming exponential in the dimension n. Later, Brakerski and others [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.

This sequence of classical reductions provides a polynomial-time reduction from lattice problems, namely GapSVP and BDD, of dimension n to decision-LWE of dimension n^2 with modulus p polynomial in n. As a consequence, this means that even if the current best known algorithms for BDD or GapSVP are assumed to be the fastest possible, this reduction only says that decision-LWE in dimension n cannot be solved in faster than $2^{\Omega(\sqrt{n})}$ time. This is a much worse lower bound than one would expect based on the state-of-the-art algorithms for solving LWE [BKW00]. This leads us to the following natural question.

Question 3.1. 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?

3.1 Alternative Measure of Computational Hardness

Cryptographic security models often assume that all possible adversaries are computationally bounded, relative to the state-of-the-art capabilities of modern computers. Often when a cryptographic scheme is declared to be 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 way of studying the complexity of a computational search problem.

They introduced the concept of *unpredictability entropy* for a computational problem, which is 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, witness compression and oracle complexity, are also studied in [AM11]. 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 (or witness) of length w. The *oracle complexity* of problem $\mathcal{P}$ is defined as the number of arbitrary YES/NO questions needed to obtain a solution to $\mathcal{P}$ in probabilistic polynomial time. It was shown in [AM11] that unpredictability entropy, witness compression, and oracle complexity are all equivalent up to lower order additive terms. The common characteristic that unifies these quantities is that they all quantify the hardness of a computational problem in bits.

The authors of [AM11] also gave a straightforward polynomial-time algorithm for both SVP and CVP that has a success probability of $2^{-n^2/4-o(n^2)}$, showing that both problems have unpredictability entropy/witness compression/oracle complexity at most $n^2/4 + o(n^2)$. This algorithm is a straightforward adaptation of the LLL algorithm [LLL82] and Babai's nearest plane algorithm [Bab86]. If the use of LLL in this algorithm is replaced with the slide reduction algorithm from [GN08, ALNSD20] with block length $O(\log n)$, this algorithm still runs in polynomial time and the search space is reduced, thereby increasing the success probability to $2^{-\Theta(n^2/\log n)}$. Despite the various algorithmic techniques available for solving lattice problems, none of these methods seem to improve this algorithm further if we restrict ourselves to probabilistic polynomial time, even if we consider approximation variants of these lattice problems with approximation factor γ polynomial in the lattice dimension n. Moreover, the close relationship between $\mathsf{BDD}_{\gamma'}$ and γ-SVP where the approximation factors $\frac{1}{\gamma'}$ and γ are both polynomial in n [LM09] suggests that it is unlikely for a polynomial-time algorithm for BDD to achieve a significantly better success probability than the current best algorithms. With this in mind, it is reasonable to conjecture the following.

Conjecture 3.2. For any constants $c, c' > 0$, there exists a constant $\kappa = \kappa(c, c') > 0$ such that no algorithm can solve $\mathsf{BDD}_{1/\gamma}$, γ-SVP, and γ-CVP on an arbitrary n-dimensional lattice for approximation factor $\gamma = n^c$ in time $n^{c'}$ with success probability better than $2^{-n^2/\kappa \log n}$.

For solving search-LWE for modulus p and dimension n in polynomial time, there is a trivial lower bound on the success probability of at least p^{-n}. This can be obtained by guessing the secret vector $\mathbf{s} \in \mathbb{Z}_p^n$ uniformly at random. This observation leads us to the following natural question, which offers a new perspective on the worst-case to average-case reductions for LWE.

Question 3.3. Assuming Conjecture 3.2, is there a lower bound close to $p^{-\Omega(n)}$ on the success probability of solving search-LWE via a polynomial-time algorithm?

We answer this question in the affirmative.

Since the security of cryptosystems is based on the hardness of solving decision problems, we formulate the measure of hardness of a decision problem in terms of the success probability of the best PPT algorithm that "solves" it. Defining what it means to solve a probabilistic decision problem is the critical part of this framework.

One-Sided Error. Question 3.3 has been well-studied in the context of average-case NP-hard problems, which any PPT algorithm is expected to be able to solve only with small probability. 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. These work under the assumption that when the algorithm is presented with a NO instance, it always outputs NO (with probability 1), while for a YES instance, the algorithm outputs YES with a small success probability α.

For problems like decision-LWE, whose inputs are chosen according to a distribution, we cannot hope to unfailingly output NO given a NO instance with probability 1. For decision-LWE, this is because a NO instance is a random element which looks like a YES instance (for some secret vector) with non-zero probability. Despite this issue, the notion of OPP algorithms can be reasonably adapted to these problems in the following way: Let $\alpha, \beta \in [0, 1]$ be two parameters such that $\alpha \gg \beta$. We call an algorithm an (α, β)-*solver* for decision-LWE if the algorithm outputs YES with probability at least α when given a YES instance, and probability at most β when given a NO instance. This notation enables us to formulate the following natural question.

Question 3.4. Assuming that no PPT algorithm can succeed in solving search-LWE with probability α, can we prove that an (α', β')-solver for the corresponding decision-LWE problem with $\alpha' \approx \alpha$ and $\beta' \ll \alpha'$ does not exist?

We answer this question in the affirmative.

Our Contributions. We show that if no PPT algorithm can solve BDD for an n-dimensional lattice with success probability greater than $O(q)$, then no PPT algorithm can solve search-LWE for dimension n with success probability greater than $O(q)$. In particular, under Conjecture 3.2, we informally state our first main result as follows.

Theorem 3.5. *(informal) If no PPT algorithm can solve* BDD_γ *for gap* $\gamma \in (0, \frac{1}{2})$ *and rank* n *with success probability greater than* $2^{-n^2/\kappa \log n}$ *for some* $\kappa > 0$*, then no PPT algorithm*

can solve search-LWE for dimension n and polynomial modulus p even for a binary secret vector with success probability $2^{-n^2/\kappa \log n}$.

Note that the above statement can easily be extended to p-ary secret vectors that are uniformly distributed over $\mathbb{Z}_p^n$ using a standard randomization of the secret. We show this explicitly in Section 3.2.3. In the reduction for Theorem 3.5, there is some constant-bounded loss in success probability which we will quantify explicitly. For a worst-case problem like BDD, we can iterate the reduction algorithm a constant number of times to recover the same success probability and still have a polynomial runtime. The complete and formal statement of Theorem 3.5 can be found in Theorem 3.8.

We emphasize here that while our reductions are adaptations of similar reductions in the literature, this adaptation to our alternative framework required great care, because limiting the number of oracle calls made in the reductions is crucial. More precisely, for a reduction from problem $\mathcal{P}$ to problem $\mathcal{Q}$ which 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 results do not have any novel implication in the quantum setting.

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, 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 3.6. *(informal) If no algorithm can solve search-LWE for modulus p polynomial in the dimension n with success probability α 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 3.6 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 3.5.

We remark here that while the techniques used to prove our second main result are similar to those used in [MM11], we do not require that the probability α of the decision-LWE oracle answering correctly is non-negligible. Instead, we only require that α is sufficiently larger than the probability δ of answering incorrectly. In particular, this means that both α and δ can be exponentially small. In our work, we also explicitly quantify the loss in success probability for each of our reductions.

A Note About Concurrent Works. There are two related and independent papers that appeared at the same time as this work. Watanabe and Yasunaga in [WY25] study a related notion of *bit security* in the context of hardness amplification. They present a variant of the XOR lemma using a similar algorithm to ours that can output $\perp$ in addition to a binary output. Micciancio and Schultz-Wu in [MSW25] present a similar notion of bit security for decision problems relevant to cryptography and propose a general way of unifying all such measures. They introduce a measure of security that encompasses both computational and statistical models and adversaries that are not necessarily computationally bounded. Our measure of computational hardness restricted to polynomial-time algorithms can be seen as a special case of the framework presented in [MSW25], but, to our knowledge, our close study of LWE under this framework is entirely new.

Preliminaries. For this entire chapter, all distances and norms are Euclidean (ℓ_2), all lattices are full-rank, and all logarithms are in base 2 unless specified otherwise. We say that a prime modulus p is *polynomial in* n if $p = \mathrm{poly}(n)$, meaning that $p = O(n^k)$ for some positive constant k. We say that p is *exponential in* n if $p = O(2^{\mathrm{poly}(n)})$.

Definition 3.7 (mod-BDD)**.** For any positive modulus $p \in \mathbb{Z}$ and approximation factor $\gamma \in (0, \frac{1}{2})$, the *Modulo-p Bounded Distance Decoding problem*, $\mathsf{BDD}_{\gamma,p}$, is the search problem

defined as: Given a basis matrix $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and a target vector $\mathbf{v} \in \mathbb{R}^n$ with the promise that $\mathrm{dist}(\mathbf{v}, \mathcal{L}) < \gamma \cdot \lambda_1(\mathcal{L})$, output the unique coefficient vector modulo p of the lattice vector closest to $\mathbf{v}$, i.e., if $\mathbf{x} \in \mathcal{L}$ is closest to $\mathbf{v}$, the output should be $\mathbf{B}^{-1}\mathbf{x} \pmod p \in \mathbb{Z}_p^n$.

We formally state our results and prove them in the following sections. These reductions are all (probabilistic) Turing reductions, so we do not specify this in our statements hereafter.

3.2 From BDD to Search-LWE

Theorem 3.8 ($\mathsf{BDD} \to$ Search-LWE)**.** *Let $n \geq 1$ be an integer, $\varepsilon \in (0, \frac{1}{24})$ and $\delta > \frac{3}{4}$ be error parameters, $\alpha \in (0, \sqrt{\ln(2n(1+1/\varepsilon))/\pi})$ a width parameter, p a prime modulus polynomial in n, and $\gamma \in (0, \frac{1}{2})$ an approximation factor. Suppose there is a polynomial-time algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,p,\Psi_\alpha}$ with success probability q. Then there is a (probabilistic) polynomial-time algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves any BDD_γ instance $(\mathbf{B}, \mathbf{v})$ specifying a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and a target vector $\mathbf{v} \in \mathbb{R}^n$, as long as these satisfy*

$$\mathrm{dist}(\mathcal{L}^*, \mathbf{v}) \leq \gamma \cdot \frac{\alpha}{\max_{i \in [n]}\{\|\mathbf{b}_i\|\}} \cdot \sqrt{\frac{\pi}{\ln(2n(1+1/\varepsilon))}}\,,$$

with probability $q(1+\delta)^{-3} - 6\varepsilon$.

Remark 3.9. In particular, we consider Conjecture 3.2 and assume that $q = 2^{-n/(\kappa \log n)}$ for some $\kappa > 0$. Then setting $\varepsilon = q/96$ and $\delta = 1$, we obtain

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

So, Theorem 3.8 says that if no polynomial-time algorithm can solve BDD_γ with probability $2^{-O(n/\log n)}$, then there is no polynomial-time algorithm for $\mathsf{LWE}_{n,p,\Psi_\alpha}$ with exponential modulus $p \approx 2^n$ that succeeds with probability $2^{-O(n/\log n)}$.

Our 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 generalized LWE problem in Section 3.2.1. Then we adapt Regev's reduction from this generalized LWE problem to search-LWE with exponential modulus in Section 3.2.2, using a multiplicative, rather than additive, ratio of distributions. Finally, we reduce LWE with exponential modulus to binary LWE with polynomial modulus in Section 3.2.3.

3.2.1 BDD to Generalised LWE

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

Definition 3.10 (Generalized LWE)**.** For any integer $n \geq 1$, prime modulus p, and family of distributions $\mathcal{D}$, the *Generalized Learning with Errors problem*, denoted $\mathsf{LWE}_{n,p,\mathcal{D}}$, is defined as: Given a polynomial number of samples from the distribution $A_{\mathbf{s},\phi}$, specified by a secret vector $\mathbf{s} \in \mathbb{Z}_p^n$ and distribution ϕ belonging to $\mathcal{D}$, output $\mathbf{s}$.

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

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

To obtain the desired bound on the success probability, we minimize the number of oracle calls in our reductions. In the original 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 $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ one time. If we insist that the modulus is exponential $p = 2^n$, then we can simplify our analysis by allowing the BDD_γ algorithm to call the $\mathsf{BDD}_{\gamma,p}$ oracle exactly once, thereby making a total of one call to the LWE oracle.

In our reduction from BDD_γ to $\mathsf{BDD}_{\gamma,p}$, we call the $\mathsf{BDD}_{\gamma,p}$ oracle once on the given BDD_γ instance of dimension n 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 this down by p. We then run Babai's Nearest Plane Algorithm [Bab86] on this shifted scaled vector as the target; this outputs a lattice vector within a factor of 2^n of the actual distance between the target and the lattice. The key insight here is that setting the modulus to be exponential in the dimension n makes it easy for Babai's algorithm to find the exact desired lattice point, because the margin of error is larger than the search space. We formalize this idea below.

Lemma 3.11 ($\mathsf{BDD} \to$ Modulo-BDD)**.** *For any integer $n \geq 1$, prime modulus p exponential in n, and approximation factor $\gamma \in (0, \frac{1}{2})$, if 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 $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves BDD_γ with success probability q.*

Proof. The given BDD_γ instance $(\mathbf{B}, \mathbf{v})$ specifies a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ and target vector $\mathbf{v} \in \mathbb{R}^n$ that satisfy $\mathrm{dist}(\mathcal{L}, \mathbf{v}) < \gamma \cdot \lambda_1(\mathcal{L})$. 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 $\mathbf{v}' := (\mathbf{v} - \mathbf{B}\overline{\mathbf{z}})/p$.

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 algorithm both run in time polynomial in lattice dimension n, this algorithm runs in polynomial time.

Now we show that if $\mathcal{B}$ answers correctly, $\mathcal{A}$ will output a correct answer. If oracle $\mathcal{B}$ succeeds, 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 $\mathbf{v}$. Then $\|\mathbf{v} - \mathbf{x}\| = \|\mathbf{v} - \mathbf{B}\mathbf{z}\| < \gamma \cdot \lambda_1(\mathcal{L})$. The output of $\mathcal{A}$ is correct if and only if $\|\mathbf{v} - \mathbf{B}(p\mathbf{z}' + \overline{\mathbf{z}})\| < \gamma \cdot \lambda_1(\mathcal{L})$, so it is enough to show that $\mathbf{z} = p\mathbf{z}' + \overline{\mathbf{z}}$. Babai's algorithm [Bab86] guarantees that the output $\mathbf{B}\mathbf{z}'$ satisfies

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

where the inequality follows from hypothesis on the modulus $p > 2^n$. By definition, the coordinates of $\mathbf{z}$ and $\overline{\mathbf{z}}$ can only differ by a multiple of p, so $(\mathbf{z} - \overline{\mathbf{z}})/p$ is a coefficient vector in $\mathbb{Z}^n$. Then $\mathbf{B}((\mathbf{z} - \overline{\mathbf{z}})/p) \in \mathcal{L}$ and $\operatorname{dist}(\mathcal{L}, \mathbf{v}') \leq \|\mathbf{v}' - \mathbf{B}((\mathbf{z} - \overline{\mathbf{z}})/p)\|$. All together,

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

Therefore, the closest vector to $\mathbf{v}'$ is $\mathbf{B}((\mathbf{z} - \overline{\mathbf{z}})/p)$, which gives $\mathbf{z} = p\mathbf{z}' + \overline{\mathbf{z}}$ as desired.

Finally, since the oracle $\mathcal{B}$ is correct with probability q and the algorithm $\mathcal{A}$ always answers correctly when $\mathcal{B}$ succeeds, the success probability of algorithm $\mathcal{A}$ is at least q. □

This exponential modulus $p \approx 2^n$ will be reduced to a polynomial one in Section 3.2.3. Before presenting our second reduction, we introduce some intermediate results. The following lemma bounds the statistical distance between two relevant distributions.

Lemma 3.12 (adapted from [Reg09, Corollary 3.10])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, vectors $\mathbf{w}, \mathbf{u} \in \mathbb{R}^n$, and reals $r, s > 0$ defining $t := \|(r\|\mathbf{w}\|, s)\|$, vector $\mathbf{v} \sim D_{\mathbf{u}+\mathcal{L},r}$, and error $e \sim N(0, s^2/(2\pi))$ defining random variable $X := \langle \mathbf{w}, \mathbf{v} \rangle + e$, and random variable $Z \sim N(0, t^2/(2\pi))$, if there exists an $\varepsilon \in (0, \frac{1}{2})$ such that $\eta_\varepsilon(\mathcal{L}) \leq rs/t$, then*

$$\Delta(X, Z) \leq 4\varepsilon \ .$$

In particular, the distribution Φ of X modulo 1 satisfies $\Delta(\Phi, \Psi_t) \leq 4\varepsilon$.

The algorithm in our second reduction requires additional data, namely samples from a discrete Gaussian. We generate these samples using an explicit subroutine from [BLP+13] as a black-box. This subroutine is an efficient algorithm that, given any lattice and sufficiently large width parameter, outputs a sample from the specified discrete Gaussian distribution. We restate their result below.

Lemma 3.13 (adapted from [BLP+13, Theorem 2.3])**.** *There exists a (probabilistic) polynomial-time algorithm* DGS*, that, given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$, a vector $\mathbf{c} \in \mathbb{R}^n$, and a real r satisfying*

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

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

Lemma 3.14 (Modulo-$\mathsf{BDD} \to$ Generalized-LWE)**.** *Let $n \geq 1$ be an integer, $\varepsilon \in (0, \frac{1}{24})$, $\alpha \in (0, \sqrt{\ln(2n(1+1/\varepsilon))/\pi})$ a width parameter, p a prime modulus, and k a positive integer 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 (probabilistic) polynomial-time algorithm $\mathcal{A}$ with oracle access to $\mathcal{B}$ that, for any approximation factor $\gamma \in (0, \frac{1}{2})$ and*

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

solves any $\mathsf{BDD}_{\gamma,p}$ instance $(\mathbf{B}^*, \mathbf{x})$, where the lattice $\mathcal{L}^* = \mathcal{L}(\mathbf{B}^*) \subset \mathbb{R}^n$ and target vector $\mathbf{x}$ satisfy $\operatorname{dist}(\mathcal{L}^*, \mathbf{x}) \leq \alpha p\sqrt{\gamma}/r$, with success probability $q - 6\varepsilon$.

Proof. The lattice $\mathcal{L}^* = \mathcal{L}(\mathbf{B}^*)$ specified by the given BDD instance has dual $\mathcal{L} = \mathcal{L}(\mathbf{B})$ for some corresponding basis $\mathbf{B}$. By Lemma 2.42 and the upper bound on α,

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

By hypothesis and this lower bound on r, the instance $(\mathbf{B}^*, \mathbf{x})$ satisfies $\operatorname{dist}(\mathcal{L}^*, \mathbf{x}) \leq \gamma \cdot \lambda_1(\mathcal{L}^*)$, and so is a valid input to $\mathsf{BDD}_{\gamma,p}$.

We define a subroutine that efficiently samples from a discrete Gaussian distribution. By the upper bound $\varepsilon < 1/24$ and lower bound $n \geq 1$, which imply $1/\varepsilon \geq 2/n$, and the upper

bound $\gamma < \frac{1}{2}$,

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

By Lemma 3.13, we can apply the DGS algorithm for this r, lattice $\mathcal{L}$, and vector $\mathbf{c} = 0$ to generate samples with distribution $D_{\mathcal{L},r}$.

Now we construct the reduction algorithm $\mathcal{A}$. The idea behind this algorithm is to use the target vector $\mathbf{x}$ to generate a polynomial number of samples from a distribution Φ that is a good approximation of $A_{\mathbf{s},\Psi_\beta}$, for some width parameter $\beta \leq \alpha$ and secret vector $\mathbf{s} = (\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}) \pmod p$. (Recall that $\kappa_{\mathcal{L}^*}(\mathbf{x})$ is the unique closest vector in the lattice $\mathcal{L}^*$ to $\mathbf{x}$ defined in Definition 2.16). We then call the oracle $\mathcal{B}$ on these generated samples to obtain the secret vector $\mathbf{s}$ with probability close to q.

Algorithm 2: Modulo-BDD to Generalized-LWE Reduction

Input: $(\mathbf{B}^*, \mathbf{x})$ such that $\text{dist}(\mathcal{L}^*, \mathbf{x}) \leq \alpha p\sqrt{\gamma}/r$.

Output: $\mathbf{s} \in \mathbb{Z}_p^n$.

for $i \in \{1, \ldots, n^k\}$ **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 $e \leftarrow N(0, \alpha^2\gamma/(2\pi))$.
- Define $b := \langle \mathbf{x}, \mathbf{v} \rangle / p + e \bmod 1$.
- Define sample $X_i := (\mathbf{a}, b)$.

end

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

Output $\mathbf{s}$.

Since the DGS sampling algorithm runs in polynomial time and is called a polynomial number of times, algorithm $\mathcal{A}$ runs in polynomial time.

We claim that $\mathcal{A}$ generates a good approximation of samples from $A_{\mathbf{s},\Psi_\beta}$. Specifically, we show that the statistical distance between the true distribution Φ of the generated samples and the distribution $A_{\mathbf{s},\Psi_\beta}$ for some $\beta \leq \alpha$ is some small ε'. If the oracle $\mathcal{B}$ is given a polynomial number of samples from $A_{\mathbf{s},\Psi_\beta}$, then it is guaranteed to find $\mathbf{s}$ with probability q. If $\mathcal{B}$ succeeds, its output will be $\mathbf{s} = (\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}) \bmod p$, which is precisely the coefficient vector of the closest lattice vector $\mathbf{x} \in \mathcal{L}^*$ modulo p. Hence, $\mathbf{s}$ is a solution to the given $\mathsf{BDD}_{\gamma,p}$ instance. But since the input samples given to $\mathcal{B}$ are from an *approximate* distribution Φ that is ε' away (in statistical distance) from the target distribution $A_{\mathbf{s},\Psi_\beta}$, by Fact 2.32, the success probability suffers a loss of ε'. Therefore, $\mathcal{A}$ will succeed with probability $q - \varepsilon'$.

We prove our claim and show that $\varepsilon' = 6\varepsilon$ by analyzing the distributions of $\mathbf{a}$ and b for any generated sample X_i. First we show that the distribution of $\mathbf{a}$ is close to uniform over

$\mathbb{Z}_p^n$. Let $\mathcal{Y}$ denote the distribution of $\mathbf{a}$. Fix $\mathbf{a} \in \mathbb{Z}_p^n$. Then

$$\begin{aligned}
\Pr[\mathcal{Y} = \mathbf{a}] &= \Pr_{\mathbf{v} \leftarrow D_{\mathcal{L},r}}[\mathbf{v} = \mathbf{B}\mathbf{a} \bmod p] && \text{(definition of } \mathbf{a} \text{ and } \mathcal{Y}) \\
&= \frac{\rho_r(p\mathcal{L} + \mathbf{B}\mathbf{a})}{\rho_r(\mathcal{L})} && \text{(Definition 2.34)} \\
&= \frac{\rho_{r/p}(\mathcal{L} + \mathbf{B}\mathbf{a}/p)}{\rho_r(\mathcal{L})} && \text{(rescaling by } p) \\
&\in \frac{(r/p)^n \cdot \det(\mathcal{L}^*) \cdot (1 \pm \varepsilon)}{r^n \cdot \det(\mathcal{L}^*) \cdot (1 \pm \varepsilon)} && \text{(Lemma 2.43)} \\
&= \frac{1}{p^n} \cdot \left(1 - \frac{2\varepsilon}{1+\varepsilon}, 1 + \frac{2\varepsilon}{1-\varepsilon}\right) && \text{(rearranging).}
\end{aligned}$$

Let $\mathcal{U}$ denote the uniform distribution over $\mathbb{Z}_p^n$ and $\rho \in [0,1]$ be the fraction of values in $\mathbb{Z}_p^n$ for which $\Pr[\mathcal{Y}] > \Pr[\mathcal{U}]$. By definition of statistical distance, the above bounds on $\Pr[\mathcal{Y} = \mathbf{a}]$, and the bounds on ε,

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

Now we show that the distribution of b for any sample X_i is close to the corresponding LWE distribution. We consider the marginal distribution of b conditioned on $\mathbf{a}$. Define $\mathbf{x}' := \mathbf{x} - \kappa_{\mathcal{L}^*}(\mathbf{x})$. Then

$$\langle \mathbf{x}, \mathbf{v} \rangle / p + e = \langle \kappa_{\mathcal{L}^*}(\mathbf{x}), \mathbf{v} \rangle / p + \langle \mathbf{x}', \mathbf{v} \rangle / p + e \ . \tag{3.1}$$

By some simple algebraic manipulations and the fact that $(\mathbf{B}^*)^{\mathrm{T}} = \mathbf{B}^{-1}$, the first summand

can be rewritten as

$$\begin{aligned}\langle \kappa_{\mathcal{L}^*}(\mathbf{x}), \mathbf{v}\rangle &= \kappa_{\mathcal{L}^*}(\mathbf{x})^{\mathrm{T}}\mathbf{B}\mathbf{B}^{-1}\mathbf{v} \\ &= \kappa_{\mathcal{L}^*}(\mathbf{x})^{\mathrm{T}}\big((\mathbf{B}^*)^{-1}\big)^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{v} \\ &= \big\langle (\mathbf{B}^*)^{-1}\kappa_{\mathcal{L}^*}(\mathbf{x}), \mathbf{B}^{-1}\mathbf{v}\big\rangle \\ &\equiv \langle \mathbf{s}, \mathbf{a}\rangle \bmod p\end{aligned}$$

so the first term in Equation (3.1) satisfies $\langle \kappa_{\mathcal{L}^*}(\mathbf{x}), \mathbf{v}\rangle / p \equiv \langle \mathbf{a}, \mathbf{s}\rangle / p \bmod 1$.

It remains to consider the second term in Equation (3.1). Since p is fixed and we condition on $\mathbf{a}$, the distribution $D_{\mathcal{L},r}$ of $\mathbf{v}$ is the same as the distribution $D_{\mathbf{B}\mathbf{a}+p\mathcal{L},r}$. For any such $\mathbf{v}$ and noise e sampled from $N(0, \alpha^2\gamma/(2\pi))$, let $\mathcal{Z}$ denote the distribution of $\langle \mathbf{x}', \mathbf{v}\rangle / p + e \bmod 1$. Then

$$\begin{aligned}\eta_\varepsilon(p\mathcal{L}) &= p \cdot \eta_\varepsilon(\mathcal{L}) && \text{(Lemma 2.39)} \\ &< r \cdot \sqrt{\gamma} && \text{(Lemma 2.41, hypothesis on } r\text{)} \\ &< \frac{1}{\|(1/r, 1/r)\|} && \text{(upper bound on } \gamma\text{, rearranging)} \\ &\le \frac{1}{\|\big((\|\mathbf{x}'\|/p)/(\alpha\sqrt{\gamma}), 1/r\big)\|} && \text{(since } \|\mathbf{x}'\| \le \operatorname{dist}(\mathcal{L}^*, \mathbf{x}) \le \alpha p\sqrt{\gamma}/r\text{)} \\ &= \frac{r \cdot \alpha\sqrt{\gamma}}{\|(r\|\mathbf{x}'\|/p, \alpha\sqrt{\gamma})\|} && \text{(rearranging).}\end{aligned}$$

By $\|\mathbf{x}'\| \le \alpha p\sqrt{\gamma}/r$ and the upper bound $\gamma < \frac{1}{2}$, the denominator is bounded above by

$$\beta := \|(r\|\mathbf{x}'\|/p, \alpha\sqrt{\gamma})\| \le \sqrt{\alpha^2\gamma + \alpha^2\gamma} = \alpha\sqrt{2\gamma} < \alpha\ .$$

By Lemma 3.12, $\Delta(\mathcal{Z}, \Psi_\beta) \le 4\varepsilon$ for this β.

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

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

Corollary 3.15 (BDD $\to$ Generalized-LWE)**.** *Let $n \ge 1$ be an integer, $\alpha \in (0, \sqrt{\ln(2n+4)/\pi})$ a width parameter, ϕ a distribution in the family $\Psi_{\le\alpha}$, p a prime modulus, and $\varepsilon \in (0, \frac{1}{24})$. Suppose there exists a polynomial-time algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,p,\phi}$ with success probability q. Then there is a polynomial-time algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves BDD_γ for any approximation factor $\gamma \in (0, \frac{1}{2})$ with success probability*

$q - 6\varepsilon$.

We remark that the additive loss in success probability can be written as a multiplicative factor in the following way:

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

This quantity is only meaningful if $q - \varepsilon' \in (0, 1)$, which holds if $\varepsilon \in (q-1, q)/6$. To incur only a small loss, say $q - \varepsilon' = q/2$, this requires $\varepsilon = q/12$. For our application (see Remark 3.9), we are interested in the regime where $q = 2^{-O(n/\log n)}$, so taking such an ε gives a valid explicit way to quantify the multiplicative loss in success probability.

3.2.2 Generalised LWE to Standard LWE

In this section, we give a reduction from a more general version of LWE to standard search-LWE by adapting Lemma 3.7 from [Reg09] to work with multiplicative, rather than additive, loss in success probability. In the reduction in [Reg09], Gaussian noise is iteratively chosen from a discrete interval to optimize the noise in a way that guarantees a very high success probability. Since we are concerned with polynomial-time adversaries and explicitly quantify the success probability, we will only sample Gaussian noise from the interval once in our reduction. Because we are limited to a single sample, we need to choose the interval and parameters carefully.

Lemma 3.16 (Generalized-LWE $\to$ Search-LWE)**.** *For any integer $n \geq 1$, width parameter $\alpha > 0$, prime modulus p, and constant $\varepsilon > \sqrt{3} - 1 \approx 0.73205$, if there is a polynomial-time algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,p,\Psi_\alpha}$ with success probability q, then there is a (probabilistic) polynomial-time algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ with success probability at least $q(1+\varepsilon)^{-3}$.*

Proof. Suppose $\mathcal{A}$ is given n^k samples, for some constant positive $k \in \mathbb{Z}$, distributed according to $A_{\mathbf{s},\Psi_\beta}$, for some $0 < \beta \leq \alpha$. For notational convenience, define $\delta := (1+\varepsilon)^2 - 1$ and the set Z of integer multiples of $\delta\alpha^2$ between 0 and $\delta^2\alpha^2$, i.e.

$$Z := \big\{0, \delta\alpha^2, 2 \cdot \delta\alpha^2, \ldots, (\lfloor\delta\rfloor - 1) \cdot \delta\alpha^2\big\} \ .$$

By the lower bound $\varepsilon > \sqrt{3} - 1$, we have $\delta > 2$, so this set Z is well-defined and contains $|Z| = \lfloor\delta\rfloor$ elements, where $2 \leq |Z| < \delta$.

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

Algorithm 3: Generalized-LWE to Search-LWE Reduction

Input: Samples $X_1, \ldots, X_{n^k} \sim A_{\mathbf{s},\Psi_\beta}$.

Output: Secret vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Sample $\gamma \leftarrow Z$ uniformly at random.

for $i \in \{1, \ldots, n^k\}$ **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 $Y_1, \ldots, Y_{n^k}$ to get a vector $\mathbf{s}' \in \mathbb{Z}_p^n$.

Output $\mathbf{s}'$.

Sampling and adding noise to n^k samples is efficient, and the oracle $\mathcal{B}$ is called once, so $\mathcal{A}$ runs in polynomial time.

The algorithm $\mathcal{A}$ 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 $\beta \leq \alpha$. Since the algorithm knows the value of α, it adds noise from $\Psi_{\sqrt{\gamma}}$ with the aim of obtaining samples whose noise distribution is close to Ψ_α. These generated samples then have the form $Y_i = (\mathbf{a}, b + e) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + (e' + e))$ and the noise $e' + e$ has distribution Ψ_σ for width $\sigma := \sqrt{\beta^2 + \gamma}$. We bound the approximation error between the true noise distribution Ψ_σ and the target distribution Ψ_α to obtain a lower bound on the total success probability of $\mathcal{A}$.

Let γ' be the smallest element of Z satisfying $\gamma' \geq \alpha^2 - \beta^2$. By the assumption $0 < \beta \leq \alpha$ and the lower bound $\delta > 2$, we have $0 \leq \alpha^2 - \beta^2 < \alpha^2 < (\lfloor \delta \rfloor - 1) \cdot \delta\alpha^2$, so there exists such an element γ' in Z that satisfies

$$\gamma' \leq \alpha^2(1 + \varepsilon)^2 - \beta^2 \leq (\lfloor \delta \rfloor - 1)\delta\alpha^2 \, . \tag{3.2}$$

Let $\sigma' := \sqrt{\beta^2 + \gamma'}$ denote the noise distribution parameter for this particular element γ', and consider the ratio of the probability generating functions for Ψ_α and $\Psi_{\sigma'}$. By Lemma 2.36 and Equation (3.2),

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

By Lemma 2.37, applying any function to this ratio cannot increase it. In particular, applying the (randomized) function given by the probability that $\mathcal{B}$ succeeds when run on the samples

Y_i generated with a specific noise distribution, by hypothesis we obtain

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

So for this value γ', the oracle $\mathcal{B}$ successfully outputs $\mathbf{s}' = \mathbf{s}$ with probability at least $q/(1+\varepsilon)$. By construction of the algorithm $\mathcal{A}$, the total success probability of $\mathcal{A}$ is bounded below by the product of the probability that $\gamma = \gamma'$ and the probability that $\mathcal{B}$ succeeds for this γ'. Since $|Z| < \delta$, the probability that γ' is selected is greater than $1/\delta$. Therefore, the total success probability of $\mathcal{A}$ is at least

$$\frac{q}{1+\varepsilon} \cdot \frac{1}{\delta} \geq \frac{q}{(1+\varepsilon)^3} \ .$$

□

For our application, in which we assume that $\mathsf{LWE}_{n,p,\Psi_\alpha}$ can be solved with success probability $q = p^{-n/(\kappa \log n)}$ for exponential modulus $p \approx 2^n$, we set $\varepsilon = 1$ in Lemma 3.16 to obtain the following result.

Corollary 3.17 (Generalized-LWE → Search-LWE)**.** *For any integer $n \geq 1$, prime modulus p exponential in n, and width parameter $\alpha > 0$, if no polynomial-time algorithm can solve $\mathsf{LWE}_{n,p,\Psi_{\leq\alpha}}$ with success probability $2^{-n/(\kappa \log n)-3}$ for some constant $\kappa > 0$, then there is no polynomial-time algorithm that can solve $\mathsf{LWE}_{n,p,\Psi_\alpha}$ with success probability $2^{-n/(\kappa \log n)}$.*

3.2.3 Reducing the Modulus for Search-LWE

In this subsection, we show how to reduce our exponential modulus to a polynomial one. To do this, we use a reduction from [BLP+13]. In their work, the authors study the trade-off between the modulus and the dimension of decision-LWE instances. In particular, they give a reduction from decision-LWE to decision-LWE that reduces the modulus arbitrarily, while incurring a quadratic loss in the dimension and a small loss in success probability.

We observe that their result also gives a reduction from search-LWE to search-LWE, because theirs is a *transformation reduction*, meaning that it transforms the input and applies the oracle to the result [BLP+13] to map one LWE instance to another. We restate their results in the context of search-LWE.

Theorem 3.18 (adapted from [BLP+13, Theorem 4.1.])**.** *For any positive integer n, width $\alpha > 0$ 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.*

Now we give a trivial reduction from binLWE to LWE for the same dimension, modulus, and noise distribution. Together with the reduction above, this allows us to reduce the modulus from exponential in to polynomial in the dimension.

Lemma 3.19 (Binary-LWE $\to$ Search-LWE)**.** *For any positive integer n, prime modulus p, and distribution ϕ over $\mathbb{T}$, if there exists a polynomial-time algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,p,\phi}$ with success probability q, then there is a (probabilistic) polynomial-time algorithm $\mathcal{A}$ that, given oracle access to $\mathcal{B}$, solves $\mathsf{binLWE}_{n,p,\phi}$ with success probability q.*

Proof. For the given $\mathsf{binLWE}_{n,p,\phi}$ samples, let $\mathbf{s} \in \{0,1\}^n$ denote the binary secret vector and $A_{\mathbf{s},\phi}$ be the corresponding distribution. Consider the following algorithm $\mathcal{A}$:

Algorithm 4: binLWE to LWE Reduction

Input: Samples $X_1, \ldots, X_{n^k} \sim A_{\mathbf{s},\phi}$.

Output: Secret vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Sample a vector $\mathbf{r} \leftarrow \mathbb{Z}_p^n$ uniformly at random.

for $i \in \{1, \ldots, n^k\}$ **do**

  Denote $(\mathbf{a}, b) := X_i$.

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

end

Run $\mathcal{B}$ on $Y_1, \ldots, Y_{n^k}$ 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 calls $\mathcal{B}$ once, so $\mathcal{A}$ runs in polynomial time. 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 $\mathcal{A}$ outputs the secret binary vector $\mathbf{s} = \mathbf{s}' - \mathbf{r}$ with the same probability. □

3.3 From 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 gives a correct response far more often than it gives a wrong one. The formal statement of our result is below.

Theorem 3.20. (Search-LWE$\to$ Decision-LWE) *Let $n \geq 1$ be an integer, $p > 10$ a prime modulus polynomial in n, and k a positive integer. Suppose there exists an efficient algorithm*

$\mathcal{B}$ for decision-$\mathsf{LWE}_{n,p,\phi}$ *that, given* n^k

- *LWE samples from* $A_{\mathbf{s},\phi}$*, outputs YES with probability* γ*,*

- *random samples from* $\mathbb{Z}_p^n \times \mathbb{T}$*, outputs YES with probability* δ*,*

where $\gamma, \delta \in (0,1)$ *satisfy* $\gamma > 5p^2\delta$*. Then there is an algorithm* $\mathcal{A}$ *for search-*$\mathsf{LWE}_{n,p,\phi}$ *that, given oracle access to* $\mathcal{B}$ *and* n^k *samples, runs in expected polynomial time and outputs*

- *a correct answer with probability* $\gamma/(5p^3)$ *and*

- $\perp$ *with probability* $1 - \gamma/(5p^3)$*.*

Note that we do not make any assumptions here on how large γ and δ must be; in particular, they need not be negligibly small. 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 distributed according to $A_{\mathbf{s},\phi}$. Intuitively, this means that the function 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 [GL89] show how to construct a hard-core predicate from any one-way function. 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 the *Goldreich-Levin Learning with Errors* (GL-LWE) problem, and use this as an intermediate problem in our reduction from search-LWE to decision-LWE. We reduce search-LWE with polynomial modulus to GL-LWE, then reduce this problem to standard decision-LWE under a reasonable condition.

3.3.1 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 generalize Levin's result from modulus 2 to modulus $p > 10$ using the natural generalization of a hard-core predicate for $\mathbb{Z}_p$. We remark that this lower bound on p is required for our analysis. Intuitively, it ensures that the approximate roots of unity found in our reduction algorithm are close enough for it to find the correct solution with high probability.

Lemma 3.21. *Let $n \geq 1$ be an integer, $p > 10$ a prime modulus polynomial in n and $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ 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 a 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 the probability $\alpha, \beta \in (0,1)$ satisfies $\beta > 1 - 1/(5p)$ and is determined by the randomness of $\mathbf{r}$ and that of the algorithm. Then there is an algorithm $\mathcal{A}$ running in expected polynomial time, that given oracle access to $\mathcal{B}$ and $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in \mathbb{Z}_p^n$, outputs $\mathbf{x}$ correctly with probability $\alpha\beta/(5p^2)$ and outputs $\perp$ with probability $1 - \alpha\beta/(5p^2)$.

Proof. Let $\zeta := e^{\frac{2\pi i}{p}}$ denote a primitive p-th root of unity. This ζ generates a multiplicative group $\{\zeta, \zeta^2, \ldots, \zeta^{p-1}\}$ isomorphic to $\mathbb{Z}_p$. 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 is polynomial in 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.

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

Sample ℓ bits 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 $i \in \{1, \ldots, n\}$ **do**

    Define $g_i(\mathbf{u}) := \mathcal{B}(\mathbf{R}\mathbf{u} + \mathbf{e}_i)$.

    Run FFT on g_i to compute

    $h_i(\mathbf{z}) := \sum_{\mathbf{u} \in \mathbb{Z}_p^k \setminus \{0\}} \zeta^{-\langle \mathbf{z}, \mathbf{u} \rangle} \cdot g_i(\mathbf{u})$.

    if $|h_i(\mathbf{z})| = 0$ **then**

      Output $\perp$.

    end

    Normalize $h_i(\mathbf{z})$ 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 $\mathbf{x}' := (x_1', \ldots, x_n')$.

  if $f(\mathbf{x}') = \mathbf{y}$ **then**

    Output $\mathbf{x}'$.

  end

end

Output $\perp$.

The algorithm samples ℓ bits until it obtains a 0. This random variable ℓ has Bernoulli distribution with probability $\frac{1}{2}$, so the probability that $\ell = \ell'$ for some fixed positive integer ℓ' is $1/2^{\ell'}$. Then $\ell > \ell'$ with probability

$$\Pr_{\ell}[\ell > \ell'] = 1 - \Pr_{\ell}[\ell \leq \ell'] = \Big(1 - \sum_{j=1}^{\ell'} 2^{-j}\Big) = \big(1 - \big(1 - 2^{-\ell'}\big)\big) = 2^{-\ell'} \,. \tag{3.3}$$

The algorithm sets $k := \ell + \lceil \log_p(4n) \rceil$. This is large enough if

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

which is satisfied if

$$\ell > \log_p\Big(\frac{2\beta + p^2}{\alpha\beta}\Big) + 1 := \ell^* \ .$$

By Equation (3.3) and some algebraic manipulations, ℓ satisfies this lower bound with probability

$$2^{-\ell^*} = 2^{\log_2(\alpha\beta/(2\beta+p^2))} \cdot 2^{1/\log_2(p)-1} \geq \frac{\alpha\beta}{4\beta + 2p^2} \ .$$

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

The algorithm $\mathcal{A}$ iterates through all possible guesses $\mathbf{z}$ in $\mathbb{Z}_p^k$ for $\mathbf{x}^{\mathrm{T}}\mathbf{R}$. For a uniformly random matrix $\mathbf{R}$, any two vectors of the form $\mathbf{Ru}$ for non-zero $\mathbf{u} \in \mathbb{Z}_p^k$ are pairwise independent. Similarly, vectors of the form $\mathbf{Ru} + \mathbf{e}_i$ are also pairwise independent. To simplify notation, let $m := p^k - 1$ and enumerate these pairwise independent vectors by $\mathbf{r}_1, \ldots, \mathbf{r}_m \in \mathbb{Z}_p^n$. For the correct value of $\mathbf{z} = \mathbf{x}^{\mathrm{T}}\mathbf{R}$ and any coordinate i,

$$h_i(\mathbf{z}) = \sum_{\mathbf{u} \in \mathbb{Z}_p^k \setminus \{0\}} \zeta^{-\langle \mathbf{x}, \mathbf{Ru} \rangle} \mathcal{B}(\mathbf{Ru} + \mathbf{e}_i) = \zeta^{x_i} \sum_{j=1}^{m} \zeta^{-\langle \mathbf{x}, \mathbf{r}_j \rangle} \mathcal{B}(\mathbf{r}_j) \ . \tag{3.4}$$

Normalizing 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}, in which case we can recover x_i with high probability.

There are three possible cases for each term in this sum. If $\mathcal{B}$ outputs 0 for a given $\mathbf{r}_j$, then the summand becomes 0. If $\mathcal{B}$ outputs a correct answer for $\mathbf{r}_j$, then $\mathcal{B}(\mathbf{r}_j) = \zeta^{\langle \mathbf{x}, \mathbf{r}_j \rangle}$ and the summand becomes 1. Otherwise, if $\mathcal{B}$ outputs a wrong answer, $\mathcal{B}(\mathbf{r}_j) = \zeta^a \zeta^{\langle \mathbf{x}, \mathbf{r}_j \rangle}$ for some non-zero $a \in \mathbb{Z}_p$. Using these three cases, Equation (3.4) becomes

$$h_i(\mathbf{z}) = \zeta^{x_i}\Big(0 + \sum_{\mathbf{r}_j : \mathcal{B} \text{ correct}} 1 + \sum_{\mathbf{r}_j : \mathcal{B} \text{ wrong}} \zeta^a\Big) = \zeta^{x_i}(c_0 + c_1\zeta^1 + \ldots + c_{p-1}\zeta^{p-1}) \ .$$

for some positive real coefficients $c_0, c_1, \ldots, c_{p-1}$.

Consider the outputs $\mathcal{B}(\mathbf{r}_1), \ldots, \mathcal{B}(\mathbf{r}_m)$ and let Y_j be the indicator random variable for $\mathcal{B}(\mathbf{r}_j) \neq 0$ for any j. These random variables $Y_1, \ldots, Y_m$ have Bernoulli distribution with probability α of being 1. Then $Y := \sum_{j=1}^{m} Y_j$ counts the number of non-zero outputs. By Lemma 2.29, we have

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

So, at least $m\alpha/2$ of the outputs are non-zero with probability at least $1 - 4/(m\alpha)$. Let E_1 denote this event.

Let t be the number of non-zero values among these m outputs. Without loss of generality, assume that these t non-zero outputs correspond to $\mathbf{r}_1, \ldots, \mathbf{r}_t$. We bound the number of correct values among these t non-zero outputs. Define random variables $Z_1, \ldots, Z_t$ 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, Z_j has value 1 with probability $(\alpha\beta)/\alpha$. Hence $Z_1, \ldots, Z_t$ have Bernoulli distribution with parameter β. Let $c > 0$ be a constant to be determined later. Then by Lemma 2.29,

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

So, the number of correct outputs among the t non-zero outputs is at least $(1-c)t\beta$ with probability at least $1 - 1/(c^2 t\beta)$. Let E_2 denote this event.

By the above bounds, the probability that both events E_1 and E_2 occur is

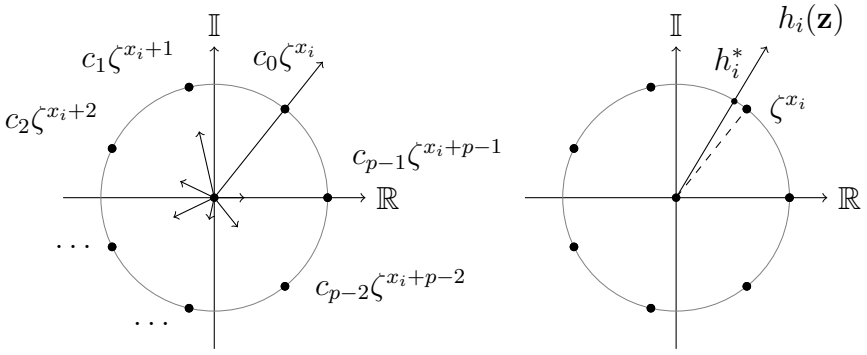
$$\Pr[E_1 \wedge E_2] \geq \left(1 - \frac{4}{m\alpha}\right)\left(1 - \frac{2}{c^2 m\alpha\beta}\right) > 1 - \frac{4}{m\alpha} - \frac{2}{c^2 m\alpha\beta} .$$

For large enough k, we have $m > 4n/(\alpha(2 + 1/(c^2\beta)))$, which makes the right hand side of the inequality above least $1 - 1/(2n)$. If both E_1 and E_2 occur,

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

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

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 $c_1 + \ldots + c_{p-1}$. 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 formalize this argument. Denote $\omega := c_0 + c_1\zeta + \ldots + c_{p-1}\zeta^{p-1}$. The factor ζ^{x_i} in

$h_i(\mathbf{z})$ has length 1, so it is enough to show that the normalization $\omega^* := \omega/|\omega|$ 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 these is $2\pi/p$. Hence, we need to show that the angle θ between ω^* and the root $1 = \zeta^0$ is strictly less than π/p. To do this, we use the formula for the length of the chord between ω^* and 1 in terms of θ,

$$|\omega^* - 1| = 2\sin(\theta/2). \tag{3.5}$$

By the lower bound on p, we have $\pi/p < \pi/2$. Since $\sin(\phi)$ is an injective increasing function for the range $0 \le \phi < \frac{\pi}{p}$, showing that $2\sin(\theta/2) \le 2\sin(\pi/(2p))$ will immediately imply $\theta < \pi/p$. Now we show that $|\omega^* - 1| < 2\sin(\pi/(2p))$. By the triangle inequality and hypothesis on β, which implies $\beta > 1/(2(1-c))$, we obtain an upper bound

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

Similarly, by the triangle inequality and lower bound on β, we obtain the lower bound

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

Together, this upper and lower bound on $|\omega|$ implies

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

By hypothesis on β, lower bound $p > 10$, and setting $c = 1/p$,

$$\beta > 1 - \frac{1}{5p} > \frac{1}{2(1-c)}\Big(1 + \frac{1}{1 + 2\sin(\pi/(2p))}\Big) .$$

This lower bound is less than 1 for this choice of parameters and approaches 1 as p grows. Since $\beta \in (0, 1)$, this means that β is nearly 1 for large values of p. For this β, we have

$$|\omega^* - 1| \leq \frac{1}{2(1-c)\beta - 1} - 1 < 2\sin(\pi/(2p)) .$$

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

Now we analyze the runtime of $\mathcal{A}$. The expected number of bit samples 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 $h_i(\mathbf{z})$ requires $O(p^k \log p^k)$ time. Finding the closest p-th root of unity to h_i^* can be done by checking if the bound in Equation (3.5) 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 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,

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

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 generalized result to our study of the hardness of LWE. First we define

our intermediate worst-case problem.

Definition 3.22 (GL-LWE)**.** The *Goldreich-Levin Learning with Errors problem*, denoted $\mathsf{GL\text{-}LWE}_{n,p,\phi}$ is defined as: Given a polynomial number of samples from the distribution $A_{\mathbf{s},\phi}$, determined by a (fixed) secret vector $\mathbf{s} \in \mathbb{Z}_p^n$ and distribution ϕ, and a uniformly random vector $\mathbf{r} \in \mathbb{Z}_p^n$, output $\langle \mathbf{s}, \mathbf{r} \rangle \bmod p$.

We use this as an intermediate problem to reduce (average-case) search-LWE to (average-case) decision-LWE. We first remark that the standard average-case search-LWE problem reduces to the worst-case version of search-LWE via the following straightforward reduction: 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 3.21 as a reduction from worst-case search-LWE to GL-LWE.

Corollary 3.23 ((Search-LWE $\to$ GL-LWE))**.** *Let $n \geq 1$ be an integer, $p > 10$ a prime modulus polynomial in n, and k a positive integer constant. Suppose that there is an efficient algorithm $\mathcal{B}$ for $\mathsf{GL\text{-}LWE}_{n,p,\phi}$ that, given n^k 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 $\alpha^*(1-\beta^*)$, or*
- *outputs $\perp$ with probability $1-\alpha^*$,*

where the probabilities $\alpha^*, \beta^* \in (0,1)$ satisfy $\beta^* > 1 - 1/(5p)$ and are determined by the randomness of $\mathbf{s}, \mathbf{r}$, and that of the algorithm. Then there is an algorithm $\mathcal{A}$ for search-$\mathsf{LWE}_{n,p,\phi}$ running in expected polynomial time that, given n^k samples from $A_{\mathbf{s}',\phi}$ for some fixed secret $\mathbf{s}' \in \mathbb{Z}_p^n$,

- *correctly outputs $\mathbf{s}'$ with probability $\alpha\beta/(5p^2)$ or*
- *outputs $\perp$ with probability $1 - \alpha\beta/(5p^2)$.*

Proof. Consider the function $f_\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^n$ given by $f_\phi(\mathbf{s}) := (\mathbf{A}, \mathbf{A}\mathbf{s}/p + \mathbf{e})$, 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 $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A}\mathbf{s}/p + \mathbf{e})$. The result then follows immediately from Lemma 3.21. $\square$

3.3.2 GL-LWE to Decision-LWE

Finally, we complete our chain of reductions by reducing GL-LWE to decision-LWE. We show that if there is a (γ, δ)-solver with $\gamma \gg \delta$ for decision-LWE, then this can be used as an oracle for GL-LWE and combined with Corollary 3.23 to complete the reduction from search-LWE to decision-LWE.

Lemma 3.24 (GL-LWE $\rightarrow$ Decision-LWE)**.** *Let $n \geq 1$ be an integer, p a prime modulus, and k a positive integer constant. Suppose that there exists an efficient algorithm $\mathcal{B}$ for decision-$\mathsf{LWE}_{n,p,\phi}$ that, given n^k*

- *LWE samples from $A_{\mathbf{s},\phi}$ for some uniformly random $\mathbf{s}$, outputs YES with probability γ,*
- *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 $\mathsf{GL\text{-}LWE}_{n,p,\phi}$ consisting of n^k samples from $A_{\mathbf{s},\phi}$ for a fixed secret $\mathbf{s}$, outputs

- *a correct answer with probability γ/p,*
- *a wrong answer with probability $(1-1/p)\delta$, and*
- *$\perp$ with probability $1 - \gamma/p - (1-1/p)\delta$.*

Proof. Let $\mathbf{s} \in \mathbb{Z}_p^n$ be the fixed secret and $e \in \mathbb{T}$ be the error with distribution ϕ that correspond to the given samples from $A_{\mathbf{s},\phi}$. Consider the following algorithm $\mathcal{A}$:

Algorithm 6: GL-LWE to Decision-LWE Reduction

Input: Samples $X_1, \ldots, X_{n^k} \sim A_{\mathbf{s},\phi}$ 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 $i \in \{1, \ldots, n^k\}$ **do**
  Denote $(\mathbf{a}, b) := X_i$.
  Define $\mathbf{a}' := \mathbf{a} + \mathbf{r}$ and $b' := b + c/p + \langle \mathbf{a}', \mathbf{t} \rangle / p$.
  Define $Y_i := (\mathbf{a}', b')$.
end
Run $\mathcal{B}$ on $Y_1, \ldots, Y_{n^k}$ to obtain an answer.
if $\mathcal{B}$ responds YES **then**
  Output c.
end
Output $\perp$.

Since a polynomial number of samples is transformed and the oracle $\mathcal{B}$ is called once, algorithm $\mathcal{A}$ runs in polynomial time.

$\mathcal{A}$ is given LWE samples of the form $(\mathbf{a}, b)$ as input, where $b = \langle \mathbf{a}, \mathbf{s} \rangle / p + e$. Suppose $\mathcal{A}$ guesses c correctly so that $c = \langle \mathbf{s}, \mathbf{r} \rangle \bmod p$. In this case, the transformed samples $(\mathbf{a}', b')$ are LWE samples, because both $\mathbf{a}' = \mathbf{a} + \mathbf{r}$ and $\mathbf{r}$ are uniformly random. By construction, for uniformly random secret $\mathbf{s}' := \mathbf{s} + \mathbf{t}$,

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

Since c and $\mathbf{t}$ were chosen independently and uniformly at random, b' is uniformly random. Hence, the oracle $\mathcal{B}$ is given valid LWE samples distributed according to $A_{\mathbf{s}',\phi}$. It correctly responds YES with probability γ. Then the guess c has value $\langle \mathbf{s}, \mathbf{r} \rangle \bmod p$ with probability $1/p$, so the probability that $\mathcal{A}$ outputs a correct answer is γ/p.

The algorithm guesses c incorrectly with probability $1 - 1/p$. 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 $\mathbb{Z}_p^n$, $\mathbf{r}$ is uniformly random, and, since c and $\mathbf{t}$ were chosen independently and uniformly at random, $b' = b + c/p + \langle \mathbf{a}', \mathbf{t} \rangle / p$ is uniformly random. Hence, the oracle $\mathcal{B}$ receives uniformly random samples and incorrectly responds YES with probability δ. Then $\mathcal{A}$ outputs a wrong answer with probability $(1 - 1/p)\delta$.

Lastly, the algorithm outputs $\perp$ with the remaining probability $1 - \gamma/p - (1 - 1/p)\delta$. $\square$

Now we combine Corollary 3.23 and Lemma 3.24 to obtain our second main result.

Corollary 3.25 (Search-LWE $\rightarrow$ Decision-LWE)**.** *Let $n \geq 1$ be an integer, $p > 10$ a prime modulus polynomial in n, and k a positive integer constant. Suppose that there exists an efficient algorithm $\mathcal{B}$ for decision-$\mathsf{LWE}_{n,p,\phi}$ that, given n^k*

- *LWE samples from $A_{\mathbf{s},\phi}$, outputs YES with probability γ,*

- *random samples from $\mathbb{Z}_p^n \times \mathbb{T}$, outputs YES with probability δ,*

where $\gamma, \delta \in (0, 1)$ satisfy $\gamma > 5p^2\delta$. Then there is an algorithm $\mathcal{A}$ for search-$\mathsf{LWE}_{n,p,\phi}$ running in expected polynomial time that, given oracle access to $\mathcal{B}$ and n^k samples, outputs

- *a correct answer with probability $\gamma/(5p^3)$ and*

- *$\perp$ with probability $1 - \gamma/(5p^3)$.*

Proof. Set $\alpha := (\gamma + (p-1)\delta)/p$ and $\beta := \gamma/(\gamma + (p-1)\delta)$. These satisfy $\alpha\beta = \gamma/p$ and $\alpha(1-\beta) = (1-1/p)\delta$. By the assumption $\gamma > 5p^2\delta$ and lower bound $p > 10$,

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

Consider the following algorithm $\mathcal{A}$: Given an instance of search-LWE, first run the algorithm from Lemma 3.24 to solve the corresponding instance of GL-LWE. Then run the algorithm from Corollary 3.23 to solve the corresponding GL-LWE instance. Finally, run the trivial algorithm to solve the given average-case search-LWE instance. By Corollary 3.23, for these values of α and β, this algorithm $\mathcal{A}$ outputs a correct answer for the given instance of search-LWE with probability

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

and outputs $\perp$ with the remaining probability. □

3.4 Future Directions

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

We show how characterizing 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.

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.

CHAPTER 4

Code Equivalence

Based on joint work with Mahdi Cheraghchi and Nikhil Shagrithaya in [CSV25].

The *Code Equivalence* (CE) problem asks if two given codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are "equivalent" in some metric-preserving way; variants of the CE problem specify the type of equivalence. *Permutation Code Equivalence* (PCE) asks if the codes are the same up to permutation of the coordinates of codewords, while *Signed Permutation Code Equivalence* (SPCE) allows equivalence up to signed permutations. More generally, *Linear Code Equivalence* (LCE) allows an equivalence up to permutation and multiplication by a (non-zero) constant. (For formal definitions of these problems, see Definitions 2.4 to 2.6 in Section 2.4.) Variants of CE belong to a larger class of isomorphism problems that ask the following question: Given two objects of the same kind, is there an isomorphism that transforms one object into the other? Another such problem is the Graph Isomorphism problem.

Besides being interesting problems in their own right, CE problems have many important applications. Perhaps most notably, the conditional hardness of CE variants has been used as a security assumption for several cryptographic schemes proposed to be post-quantum. These include the seminal McEliece public-key encryption scheme [McE78], a recent NIST post-quantum standardization submission called "Classic McEliece" [ABC+22], and the more recent LESS identification scheme [BBPS21, BBPS22].

Prior Work. Given the relevance of these problems to cryptography, there has been a considerable amount of work on designing efficient algorithms that solve these problems. Leon [Leo03] introduced an algorithm for the search version of PCE that works well for a large number of codes, but still requires exponential time in the worst case. The Support Splitting Algorithm developed by Sendrier in [Sen00] and extended by Sendrier and Simos in [SS13a] gives an algorithm for linear codes that is efficient for codes with small hull, where the hull of a code is defined by the intersection of the code and its dual. This algorithm, however, does not work for the case where the dimension of the hull is zero, but this case

was later handled by Bardet, Otmani, and Saeed-Taha in [BOS19]. In the latter paper, the authors reduce the problem of deciding PCE to a weighted version of Graph Isomorphism, and then use a variant of Babai's algorithm for solving the latter problem [Bab16] to give a quasi-polynomial time algorithm that computes PCE for the zero hull case. Recently, Bennett and others [BBB+25] show how to use Babai's algorithm for PCE [BCGQ11] to construct a provable deterministic algorithm that solves LCE for any arbitrary code of block length n over any finite field of size q in time $2^{n+o(n+q)}$. The authors also give provable algorithms for solving PCE and LCE for arbitrary codes of block length n and any field size q in randomized $2^{n/2+o(n+q)}$-time and quantum $2^{n/3+o(n+q)}$-time.

On the other side of the cryptographic coin, considerable attention has been devoted towards understanding the computational hardness of these problems. Petrank and Roth in [PR97] showed that Graph Isomorphism reduces to PCE, and that PCE is not NP-complete unless the polynomial hierarchy collapses. This was used as evidence for the computational hardness of PCE until Babai introduced his quasi-polynomial time algorithm for deciding Graph Isomorphism [Bab16]. Even under the assumption that the polynomial hierarchy does not collapse, however, it is still possible that both Graph Isomorphism and PCE cannot be decided in polynomial time.

In [SS13a], Sendrier and Simos give a reduction from LCE to PCE that runs in time polynomial in the block length n and alphabet size q of the codes. They use the *closure* of a code; for any code of block length n over a field of size q, this is the set obtained by taking every codeword and multiplying each of its n coordinates with all the non-zero field elements to produce a new vector of length $n(q-1)$. Then using the fact that multiplication by any non-zero field element induces a permutation on the elements in $\mathbb{F}_q^*$, they reframe scalar multiplication as a permutation over $\mathbb{F}_q^*$. In [DG23], Ducas and Gibbons use a specific form of this closure to prove a reduction from SPCE to PCE. Biasse and Micheli in [BM23] give a search-to-decision reduction for PCE, which implies that the decision version of PCE is at least as hard as the search version. Bennett and Win in [BW24] give several reductions between CE variants and other isomorphism problems. They extend the closure technique from [SS13a] to give a reduction from LCE to SPCE, and also give a reduction from SPCE to the *Lattice Isomorphism Problem* (LIP).

LIP is an analogous problem to CE for lattices. Two lattices are said to be *isomorphic* if there exists an orthogonal transformation that transforms one lattice (basis) into the other. LIP asks if such an isomorphism exists (see Definition 2.7 for formal definition). In their reduction from SPCE to LIP, Bennet and Win use Construction A (see Definition 2.17) to lift a linear code over a prime finite field $\mathbb{F}_p$ to a lattice in $\mathbb{R}^n$. Because this construction is only well-defined for prime fields, the reduction only works for prime fields. It remains an

open problem to find a reduction from any CE variant to LIP for non-prime fields.

Our Contributions. Now we state our main results and place them into context of the prior work described above. In [BW24], the authors prove Karp reductions from LCE to PCE and from LCE to SPCE. In our work, we show a reverse Karp reduction from PCE to LCE.

Theorem 4.1 (PCE reduces to LCE, informal)**.** *For linear codes with block length n over a field of size q, there is a Karp reduction from* PCE *to* LCE *that runs in* $\mathrm{poly}(n, \log q)$ *time.*

The formal statement is given in Theorem 4.4. This result, along with the Karp reduction in [SS13b, BW24] from LCE to PCE running in $\mathrm{poly}(n, q)$ time, implies that the problems LCE and PCE are computationally equivalent up to factors in the runtime that are polynomial in n and q.

Additionally, the Karp reduction from LCE to SPCE detailed in [BW24, Theorem 4.4, Corollary 4.5] combines with our result to give a reduction from PCE to SPCE running in $\mathrm{poly}(n, q)$ time. In situations where q is much larger than n (as is the case for Reed-Solomon codes, for example), it is desirable to have the runtime depend only logarithmically on q. Note that at least $\log q$ time is required to express a single element of the field, so the dependence on q cannot be smaller than $\log q$. Our second result is a Karp reduction from PCE to SPCE whose runtime depends only logarithmically on q.

Theorem 4.2 (PCE reduces to SPCE, informal)**.** *For linear codes with block length n over a field of size q, there is a Karp reduction from* PCE *to* SPCE *that runs in* $\mathrm{poly}(n, \log q)$ *time.*

The formal statement is given in Theorem 4.5. This result, together with a Karp reduction from SPCE to PCE from [DG23, Lemma 10] running in $\mathrm{poly}(n, \log q)$ time, implies that PCE and SPCE are computationally equivalent problems up to factors in the runtime that are polynomial in n and $\log q$.

Finally, by combining the above result with the Karp reduction from SPCE over prime fields to LIP from [BW24, Theorem 5.1], we obtain the following corollary.

Corollary 4.3 (PCE reduces to LIP)**.** *For any prime q, there is a Karp reduction from* PCE *over a field of order q to* LIP *that runs in* $\mathrm{poly}(n, \log q)$ *time.*

4.1 Reductions from PCE to LCE and SPCE

We formally state our results below and prove them in this section. These reductions are all deterministic Karp reductions, so we do not specify this in our statements hereafter.

Theorem 4.4 (PCE reduces to LCE)**.** *There is a reduction from* PCE *to* LCE *that runs in* $\mathrm{poly}(n, \log q)$ *time, where* n *is the block length and* q *is the field size of the input code pair.*

Theorem 4.5 (PCE reduces to SPCE)**.** *There is a reduction from* PCE *to* SPCE *that runs in* $\mathrm{poly}(n, \log q)$ *time, where* n *is the block length and* q *is the field size of the input code pair.*

The following observation will be used throughout our reduction.

Observation 4.6. *Any invertible matrix* $\mathbf{S} \in \mathrm{GL}_k(\mathbb{F})$ *induces a bijective map on* $\mathbb{F}^k$*. In particular, for any* $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$*, we have* $\mathbf{Sx} = \mathbf{Sy}$ *if and only if* $\mathbf{x} = \mathbf{y}$*. Then when* $\mathbf{S}$ *is multiplied by some matrix* $\mathbf{A}$ *of the appropriate dimension, it maps identical (distinct) columns in* $\mathbf{A}$ *to identical (distinct) columns in* $\mathbf{SA}$*.*

We will use the following three lemmas to justify why certain assumptions about the input can be made without loss of generality.

Lemma 4.7 (adapted from [Mac33, Corollary 27.2])**.** *For any prime power* q *and matrices* $\mathbf{A}, \mathbf{B}$ *over* $\mathbb{F}_q$*, the equivalence* $\mathbf{A} = \mathbf{PBQ}$ *holds for some* $\mathbf{P}, \mathbf{Q}$ *over* $\mathbb{F}_q$ *if and only if* $\mathbf{A}$ *and* $\mathbf{B}$ *have equal rank.*

Lemma 4.8. *Let* $k, n \in \mathbb{N}$ *and* q *be a prime power. For any matrices* $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$*, if* $(\mathbf{G}, \mathbf{H})$ *is in* PCE*, then no column of* $\mathbf{G}$ *appears in* $\mathbf{G}$ *more times than a column of* $\mathbf{H}$ *appears in* $\mathbf{H}$*.*

Proof. By definition, there must be some $\mathbf{S} \in \mathrm{GL}_k$ and $\mathbf{P} \in \mathcal{P}_n$ such that $\mathbf{SGP} = \mathbf{H}$. By Observation 4.6, $\mathbf{S}$ takes identical columns in $\mathbf{G}$ to identical columns in $\mathbf{SG}$. When multiplied with $\mathbf{G}$ on the right, $\mathbf{P}$ only permutes the columns of $\mathbf{G}$ and so does not change the frequency with which any column appears in the matrix $\mathbf{SG}$. Therefore for every column $\mathbf{c}$ in $S\mathbf{G}$, its corresponding column in $\mathbf{H}$ must appear the same number of times as $\mathbf{c}$ appears in $\mathbf{SG}$. □

Lemma 4.9. *For any* $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$*, let* $\overline{\mathbf{G}}$ *and* $\overline{\mathbf{H}}$ *be the submatrices of* $\mathbf{G}$ *and* $\mathbf{H}$*, respectively, obtained by removing all columns equal to* $\mathbf{0} \in \mathbb{F}_q^k$*. Then* $(\mathbf{G}, \mathbf{H})$ *is in* PCE *if and only if* $(\overline{\mathbf{G}}, \overline{\mathbf{H}})$ *is in* PCE*.*

This lemma follows from the following trivial reduction: If $\mathbf{G}$ and $\mathbf{H}$ satisfy the PCE condition, there exist matrices $\mathbf{S} \in \mathrm{GL}_k$ and $\mathbf{P} \in \mathcal{P}_n$ such that $\mathbf{SGP} = \mathbf{H}$. Because $\mathbf{S}$ is a linear map, it will always map $\mathbf{0} \in \mathbb{F}_q^k$ to itself, so there is an invertible submatrix of $\mathbf{S}$ and a permutation submatrix of $\mathbf{P}$ for which $\overline{\mathbf{G}}$ and $\overline{\mathbf{H}}$ satisfy the PCE condition.

As a result of Lemma 4.9 and Lemma 4.8, we can assume without loss of generality that any given input pair of matrices do not contain a zero column and have the same column frequency.

Now we define the construction that will transform the input matrices in our reduction.

Construction 4.10. Given a $k \times n$ matrix $\mathbf{A}$ over $\mathbb{F}_q$ with column vectors $\mathbf{A}[1], \ldots, \mathbf{A}[n] \in \mathbb{F}_q^k$, let $m_{\mathbf{A}}$ denote the maximum number of times a column appears in $\mathbf{A}$. Denote $m := m_{\mathbf{A}} + 1$. We construct the $k \times nm$ matrix $\widehat{\mathbf{A}}$ by appending m copies of each column vector in order. More explicitly, for every $i \in [n]$ and $j \in [m]$, we set $\widehat{\mathbf{A}}[(i-1)m + j] := \mathbf{A}[i]$. Define the $(k+1) \times n$ matrix $\mathbf{A}'_1$ by appending a row of all ones at the bottom of matrix $\mathbf{A}$, define the $(k+1) \times nm$ matrix $\mathbf{A}'_2$ by appending a row of all zeros at the bottom of matrix $\widehat{\mathbf{A}}$, and define the $(k+1) \times (nm+1)$ matrix $\mathbf{A}'_3$ by placing ones in every entry of the last row and zeros everywhere else. Denoting $n' := n + 2nm + 1$, we obtain the final $(k+1) \times n'$ matrix $\mathbf{A}'$ by concatenating the block matrices $\mathbf{A}'_1$, $\mathbf{A}'_2$, and $\mathbf{A}'_3$:

$$\mathbf{A}' := \left[\begin{array}{c|c|c} \mathbf{A}'_1 & \mathbf{A}'_2 & \mathbf{A}'_3 \end{array} \right].$$

$\mathbf{A}$	$\widehat{\mathbf{A}}$	$\mathbf{0}$
1 1 ... 1	0 0 0	1 1 1

Figure 4.1: The matrix A' obtained by Construction 4.10.

Note that if the given matrix $\mathbf{A}$ has full row rank, then $\mathbf{A}'$ must have full row rank.

With these assumptions and definitions in place, we now prove the main result.

Proof of Theorem 4.4: Given a pair of matrices $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$ as input, we construct matrices $\mathbf{G}', \mathbf{H}'$ according to Construction 4.10. By Lemma 4.7 and Lemma 4.9, we can assume without loss of generality that $\mathbf{G}$ and $\mathbf{H}$ have full row rank and do not contain any zero columns. Let $m_{\mathbf{G}}$ denote the maximum number of times a column appears in $\mathbf{G}$, and define $m_{\mathbf{H}}$ similarly. By Lemma 4.8, we must have $m_{\mathbf{G}} = m_{\mathbf{H}}$. Denoting $m := m_{\mathbf{G}} + 1 = m_{\mathbf{H}} + 1$ and $n' := n + 2nm + 1$, we obtain $(k+1) \times n'$ matrices $\mathbf{G}'$ and $\mathbf{H}'$. Note that because $m \leq n + 1$, these matrices can be constructed deterministically in $\mathrm{poly}(n, \log q)$ time. The claim in Theorem 4.4 then follows from Lemma 4.11 and Corollary 4.16, which are stated and proven below. □

The proof of Theorem 4.5 is nearly identical to the above proof, but with a restriction to the set of signs $\{-1, +1\}$ instead of all non-zero field elements in the proof of Corollary 4.16.

4.1.1 From PCE to LCE

First we show the forward direction. We prove that our construction preserves the permutation equivalence of the input pair, which gives the following stronger result.

Lemma 4.11. *For any $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, let $\mathbf{G}', \mathbf{H}'$ be the corresponding matrices given by Construction 4.10. If $(\mathbf{G}, \mathbf{H})$ is in* PCE*, then $(\mathbf{G}', \mathbf{H}')$ is in* PCE *(and therefore in* LCE*).*

Proof. By definition, $(\mathbf{G}, \mathbf{H})$ is in PCE if and only if there exist an invertible matrix $\mathbf{S} \in \mathrm{GL}_k$ and a permutation matrix $\mathbf{P} \in \mathcal{P}_n$ such that $\mathbf{SGP} = \mathbf{H}$. From these matrices we will construct $\mathbf{S}' \in \mathrm{GL}_{k+1}$ and $\mathbf{P}' \in \mathcal{P}_{n'}$ such that $\mathbf{S}'\mathbf{G}'\mathbf{P}' = \mathbf{H}'$. Define $\mathbf{S}' \in \mathrm{GL}_{k+1}$ such that the top-left block submatrix is $\mathbf{S}'[1:k, 1:k] = \mathbf{S}$, the bottom-right entry is $\mathbf{S}'[k+1, k+1] = 1$, and all other entries are zero.

Let $\sigma_{\mathbf{P}} \colon [n] \to [n]$ be the permutation map described by $\mathbf{P}$. Define a new permutation $\sigma_{\mathbf{P}'} \colon [n'] \to [n']$ that permutes the first n columns of $\mathbf{G}'$ among each other according to P, i.e. $\sigma_{\mathbf{P}'}(x) := \sigma_{\mathbf{P}}(x)$ for all $x \in [n]$. We also define $\sigma_{\mathbf{P}'}$ such that it permutes each of the nm columns in the second block according to how its corresponding column was permuted in the first block. Formally, for every $x \in [n+1, n+nm]$, we write $x = n + m(i-1) + j$ uniquely for $i \in [n]$ and $j \in [m]$ and set $\sigma_{\mathbf{P}'}(x) := n + m(\sigma_{\mathbf{P}}(i) - 1) + j$. Lastly, we define $\sigma_{\mathbf{P}'}$ such that it sends each of the last $nm+1$ columns identically to itself, i.e. $\sigma_{\mathbf{P}'}(x) := x$ for every $x \in [n+nm+1, n+2nm+1]$. Finally we let $\mathbf{P}' \in \mathcal{P}_{n'}$ be the permutation matrix defined by $\sigma_{\mathbf{P}'}$.

We claim that $\mathbf{S}'\mathbf{G}'\mathbf{P}' = \mathbf{H}'$. This holds if and only if $\mathbf{S}'\mathbf{G}'\mathbf{P}'[x] = \mathbf{H}'[x]$ for every column $x \in [n']$. We show that this is the case for every column in each of the three blocks. For every $x \in [n]$, by the definition of $\sigma_{\mathbf{P}'}$, the fact that each column in the first block has 1 as the last entry, and the PCE assumption,

$$\mathbf{S}'\mathbf{G}'\mathbf{P}'[x] = \mathbf{S}'\mathbf{G}'[\sigma_{\mathbf{P}'}(x)] = \mathbf{SG}[\sigma_{\mathbf{P}}(x)] \parallel 1 = \mathbf{SG}[x]\mathbf{P} \parallel 1 = \mathbf{SGP}[x] \parallel 1 = \mathbf{H}[x] \parallel 1 = \mathbf{H}'[x].$$

Here "$\parallel$" denotes concatenation of an additional entry at the end of the column vector. For $x \in [n+1, n+nm]$, we can write $x = n + m(i-1) + j$ for some $i \in [n]$ and $j \in [m]$. Then by the definition of $\sigma_{\mathbf{P}'}$, the fact that each column in the second block has 0 as the last entry, and the PCE assumption,

$$\begin{aligned}\mathbf{S}'\mathbf{G}'\mathbf{P}'[x] &= \mathbf{SG}[\sigma_{\mathbf{P}'}(x)] \parallel 0 = \mathbf{SG}[n + m(\sigma_{\mathbf{P}}(i) - 1) + j] \parallel 0 = \mathbf{SG}[\sigma_{\mathbf{P}}(i)] \parallel 0 \\ &= \mathbf{SGP}[i] \parallel 0 = \mathbf{H}[i] \parallel 0 = \mathbf{H}[n + m(i-1) + j] \parallel 0 = \mathbf{H}'[x].\end{aligned}$$

Lastly, for all $x \in [n+nm+1, n+2nm+1]$, by definition of $\sigma_{\mathbf{P}'}$, the fact that each column

in the last block is identical to $\mathbf{e}_{k+1}$, and the PCE assumption,

$$\mathbf{S}'\mathbf{G}'\mathbf{P}'[x] = \mathbf{S}\mathbf{G}[\sigma_{\mathbf{P}'}(x)] \parallel 1 = \mathbf{S}\mathbf{G}[x] \parallel 1 = \mathbf{S}\mathbf{G}\mathbf{P}[x] \parallel 1 = \mathbf{H}[x] \parallel 1 = \mathbf{H}'[x].$$

Therefore, $(\mathbf{G}', \mathbf{H}')$ is in PCE. □

Because the matrix groups are nested $\mathcal{P}_n \subseteq \mathcal{SP}_n \subseteq \mathcal{M}_n$, Lemma 4.11 immediately implies the first direction of both statements Theorem 4.4 and Theorem 4.5.

4.1.2 From LCE to PCE

Now we prove the other direction, and show that if the constructed matrix pair $(\mathbf{G}', \mathbf{H}')$ is in LCE, then the original matrix pair $(\mathbf{G}, \mathbf{H})$ must be in PCE. This requires some careful analysis of how the change-of-basis matrix $\mathbf{S}$ and monomial matrix $\mathbf{M}$ affect each block of the matrices $\mathbf{G}'$ and $\mathbf{H}'$.

First we show that for any monomial matrix $\mathbf{M} = \mathbf{D}\mathbf{P}$ for which $(\mathbf{G}', \mathbf{H}')$ is in LCE, the permutation matrix $\mathbf{P}$ must respect the "boundaries" between the three block matrices of $\mathbf{G}'$.

Lemma 4.12. *For any $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, if $\mathbf{S}'\mathbf{G}'\mathbf{M}' = \mathbf{H}'$ for some $\mathbf{S}' \in \mathrm{GL}_{k+1}$ and $\mathbf{M}' = \mathbf{D}'\mathbf{P}' \in \mathcal{M}_{n'}$, where $\mathbf{P}' \in \mathcal{P}_{n'}$, the corresponding permutation map $\sigma_{\mathbf{P}'} \colon [n'] \to [n']$ satisfies the following:*

(i) For every $i \in [1, n]$, $\sigma_{\mathbf{P}'}(i) \in [1, n]$.

(ii) For every $i \in [n+1, n+mn]$, $\sigma_{\mathbf{P}'}(i) \in [n+1, n+mn]$.

(iii) For every $i \in [n+nm+1, n']$, $\sigma_{\mathbf{P}'}(i) \in [n+mn+1, n']$.

Proof. As a result of Lemma 4.9, we can assume without loss of generality that $\mathbf{G}$ and $\mathbf{H}$ do not contain a zero column. Suppose $\mathbf{S}' \in \mathrm{GL}_{k+1}$ and $\mathbf{M}' = \mathbf{D}'\mathbf{P}' \in \mathcal{M}_{n'}$ satisfy $\mathbf{S}'\mathbf{G}'\mathbf{M}' = \mathbf{H}'$. By definition, $\mathbf{S}'$ has an inverse $\mathbf{S}'^{-1} \in \mathrm{GL}_{k+1}$, so we can rewrite

$$\mathbf{G}'\mathbf{M}' = \mathbf{S}'^{-1}\mathbf{H}'. \tag{4.1}$$

By Observation 4.6, $\mathbf{S}'^{-1}$ maps identical columns in $\mathbf{H}'$ to identical columns in $\mathbf{S}'^{-1}\mathbf{H}'$. To prove each part of the claim, we analyze the effect of $\mathbf{S}'^{-1}$ on each block $\mathbf{H}'_1, \mathbf{H}'_2, \mathbf{H}'_3$ of matrix $\mathbf{H}'$.

First we prove part (iii). By construction, the last $nm+1$ columns of $\mathbf{H}'$, contained in $\mathbf{H}'_3$ are identical to $\mathbf{e}_{k+1}$. By assumption, $\mathbf{H}$ does not contain a zero column, so no other

column in $\mathbf{H}'$ outside of the block $\mathbf{H}'_3$ is equal to $\mathbf{e}_{k+1}$. Then each of the $nm+1$ columns in $\mathbf{S}'^{-1}\mathbf{H}'_3$, and no other column of $\mathbf{S}'^{-1}\mathbf{H}'$, is equal to $\mathbf{S}'^{-1}\mathbf{e}_{k+1}$.

By Equation (4.1), the last $nm+1$ columns of $\mathbf{G}'\mathbf{M}'$ must be identical and equal to $\mathbf{S}'^{-1}\mathbf{e}_{k+1}$. Since $\mathbf{M}' = \mathbf{D}'\mathbf{P}'$ for some diagonal matrix $\mathbf{D}'$ and permutation $\mathbf{P}'$, it maps each column in $\mathbf{G}'$ to a (possibly) scaled permutation of that column in $\mathbf{G}'\mathbf{M}'$. Thus, the matrix $\mathbf{M}'$ must scale and permute the columns of $\mathbf{G}'$ to produce $nm+1$ identical columns in the last block of $\mathbf{G}'\mathbf{M}'$. We claim that multiplying by $\mathbf{M}'$ cannot cause the number of copies of a column in $\mathbf{G}'$ to increase in $\mathbf{G}'\mathbf{M}'$. In particular, no two columns from different blocks of $\mathbf{G}'$ can be scaled to produce identical columns in $\mathbf{G}'\mathbf{M}'$. By construction, the last row of $\mathbf{G}'$ ensures that no column of $\mathbf{G}'_1$ can be scaled to produce a column of $\mathbf{G}'_2$, so $\mathbf{M}'$ cannot map columns from $\mathbf{G}'_1$ and $\mathbf{G}'_2$ to identical columns in $\mathbf{G}'\mathbf{M}'$. Additionally, $\mathbf{M}'$ maps every $\mathbf{e}_{k+1}$ column in $\mathbf{G}'_3$ to a scaling of $\mathbf{e}_{k+1}$ in $\mathbf{G}'\mathbf{M}'$, and since no column in $\mathbf{G}'_1$ or $\mathbf{G}'_2$ can be scaled to produce a multiple of $\mathbf{e}_{k+1}$, all scalings of $\mathbf{e}_{k+1}$ in $\mathbf{G}'\mathbf{M}'$ can only come from $\mathbf{G}'_3$. The only column in $\mathbf{G}'$ that appears $nm+1$ times is $\mathbf{e}_{k+1}$ from $\mathbf{G}'_3$. In this way, $\mathbf{M}'$ can only produce $nm+1$ identical columns in the last block of $\mathbf{G}'\mathbf{M}'$ by mapping from the $nm+1$ columns of $\mathbf{G}'_3$. Therefore, $\sigma_{\mathbf{P}'}(i)$ maps every column with index in $[n+mn+1, n']$ to a column with index in $[n+mn+1, n']$.

Next we show part (ii). By part (iii), any column with index $i \in [n+1, n+mn]$ is permuted to a column with index $\sigma_{\mathbf{P}'}(i) \leq n+mn$, so it is enough to show that $\sigma_{\mathbf{P}'}(i) \geq n+1$. Suppose for the sake of contradiction that there is a column with index $i \in [n+1, n+nm]$ in $\mathbf{G}'$ that is mapped to a column with index $\sigma_{\mathbf{P}'}(i) \in [1, n]$. By design, the last entry of the columns in $\mathbf{G}'_1$ is 1 and differs from the last entry 0 of the columns in $\mathbf{G}'_2$, so each column in $\mathbf{G}'_1$ appears strictly less than m times. Then, if $\mathbf{M}'$ permutes any number of columns from $\mathbf{G}'_1$ with the same number of columns in $\mathbf{G}'_2$, there will be some column in the second block of $\mathbf{G}'\mathbf{M}'$ that appears less than m times. But by Observation 4.6, every column in $\mathbf{S}'^{-1}\mathbf{H}'_2$ appears at least m times, and since the second block of $\mathbf{G}'\mathbf{P}'$ must be equal to $\mathbf{S}'^{-1}\mathbf{H}'_2$, this gives a contradiction.

Finally, part (i) follows immediately from parts (ii) and (iii). □

This result gives the following corollary.

Corollary 4.13. *For any $\mathbf{G}', \mathbf{H}', \mathbf{S}', \mathbf{M}'$ as in Lemma 4.12, the monomial matrix $\mathbf{M}'$ is comprised of three block matrices $\mathbf{M}_1 \in \mathcal{M}_n$, $\mathbf{M}_2 \in \mathcal{M}_{nm}$, $\mathbf{M}_3 \in \mathcal{M}_{nm+1}$, such that $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ only act on the first n, next nm, and last $nm+1$ columns of $\mathbf{G}'$, respectively.*

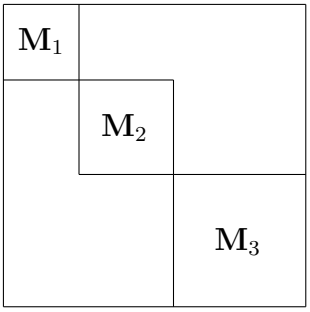


Figure 4.2: The structure of matrix $\mathbf{M}'$.

Now, we use this lemma to show that any invertible matrix for which $(\mathbf{G}', \mathbf{H}')$ is in LCE must contain only zeros in the last row and last column, except for the last entry.

Lemma 4.14. *For any $\mathbf{G}', \mathbf{H}', \mathbf{S}', \mathbf{M}'$ as in Lemma 4.12, the change-of-basis matrix $\mathbf{S}'$ satisfies the following properties:*

(i) The last column of $\mathbf{S}'$ contains zeros in the first k entries.

(ii) The last row of $\mathbf{S}'$ contains zeros in the first k entries.

(iii) The entry $\mathbf{S}'[k+1, k+1]$ is non-zero.

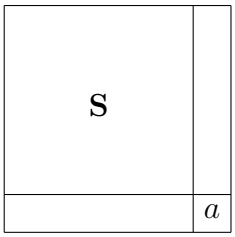


Figure 4.3: The structure of matrix $\mathbf{S}'$.

Proof. By definition, $\mathbf{S}'$ has an inverse $\mathbf{S}'^{-1} \in \mathrm{GL}_{k+1}$, so we can write $\mathbf{G}'\mathbf{M}' = \mathbf{S}'^{-1}\mathbf{H}'$. By Observation 4.6, $\mathbf{S}'^{-1}$ maps identical columns in $\mathbf{H}'$ to identical columns in $\mathbf{S}'^{-1}\mathbf{H}'$. To prove each part of the claim, we analyze the effect of $\mathbf{S}'^{-1}$ on each block $\mathbf{H}'_2$ and $\mathbf{H}'_3$ in the right-hand side of the equation.

By Corollary 4.13, we know that $\mathbf{M}'$ is comprised of three block matrices $\mathbf{M}_1 \in \mathcal{M}_n$, $\mathbf{M}_2 \in \mathcal{M}_{nm}$, and $\mathbf{M}_3 \in \mathcal{M}_{nm+1}$ that affect the first n, next nm, and last $nm+1$ columns of $\mathbf{G}'$, respectively. This allows us to write $\mathbf{G}'_2\mathbf{M}_2 = \mathbf{S}'^{-1}\mathbf{H}'_2$ and $\mathbf{G}'_3\mathbf{M}_3 = \mathbf{S}'^{-1}\mathbf{H}'_3$.

For part (i), consider the equation $\mathbf{G}'_3\mathbf{M}_3 = \mathbf{S}'^{-1}\mathbf{H}'_3$. By construction, all columns of $\mathbf{G}'_3$ and $\mathbf{H}'_3$ are identical and equal to $\mathbf{e}_{k+1}$. Since $\mathbf{M}_3$ permutes all columns of $\mathbf{G}'_3$ and multiplies

them by a non-zero scalar, all columns of $\mathbf{G}_3'\mathbf{M}_3$ must be of the form $a \cdot \mathbf{e}_{k+1}$ for some non-zero a. By Observation 4.6, all columns of $\mathbf{S}'^{-1}\mathbf{H}_3'$ are identical. Then for any column $a \cdot \mathbf{e}_{k+1}$ of $\mathbf{G}_3'\mathbf{M}_3$, we have $\mathbf{S}'(a \cdot \mathbf{e}_{k+1}) = \mathbf{e}_{k+1}$. If any of the first k entries of the last column of $\mathbf{S}'$ is non-zero, then the corresponding entry in $\mathbf{S}'(a \cdot \mathbf{e}_{k+1}) = \mathbf{e}_{k+1}$ must be non-zero, but this is not the case. Therefore, the last column of $\mathbf{S}'$ must contain zeros in the first k entries.

For part (ii), consider the equation $\mathbf{G}_2'\mathbf{M}_2 = \mathbf{S}'^{-1}\mathbf{H}_2'$. By construction, the last rows of $\mathbf{G}_2'$ and $\mathbf{H}_2'$ only contains zeros. By Lemma 4.12, no columns in $\mathbf{G}_2'\mathbf{M}_2$ could have been mapped from outside $\mathbf{G}_2'$, so the last row of $\mathbf{G}_2'\mathbf{M}_2$ must only contain zeros. Since $\mathbf{G}$ is a submatrix of $\mathbf{G}_2'$ with full row rank k, the first k rows of $\mathbf{G}_2'$, denoted by $\widehat{\mathbf{G}}$ in Construction 4.10, form a submatrix of rank k. Then $\widehat{\mathbf{G}}$ also has column rank k, and because the last row of $\mathbf{G}_2'$ only contains zeros, $\mathbf{G}_2'$ must contain k linearly independent columns. By Corollary 4.13, we know that $\mathbf{G}_2'\mathbf{M}_2$ is entirely comprised of columns of $\mathbf{G}_2'$ multiplied by a non-zero scalar, so $\mathbf{G}_2'\mathbf{M}_2$ has column rank k. Then the first k rows of $\mathbf{G}_2'\mathbf{M}_2$ form a submatrix of rank k. By definition, the last row of $\mathbf{S}'$ contains the coefficients that specify the linear combination of rows of $\mathbf{G}_2'\mathbf{M}_2$ which gives the last row of $\mathbf{H}_2$. But since the first k rows of $\mathbf{G}_2'\mathbf{M}_2$ are linearly independent, no linear combination of these can produce the all-zero last row of $\mathbf{H}_2$. Therefore, the first k entries in the last row of $\mathbf{S}'$ must be zero.

Part (iii) follows immediately from parts (i) and (ii) and the fact that $\mathbf{S}'$ is invertible. □

Finally, we show how Lemmas 4.12 and 4.14 combine to ensure the existence of an unsigned permutation for which $(\mathbf{G}, \mathbf{H})$ is in PCE.

Corollary 4.15. *Let $\mathbf{M}' \in \mathcal{M}_n$ be a permutation with block submatrices $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3$ as described in Corollary 4.13. Then, $\mathbf{M}_1 = a \cdot \mathbf{P}$ for some permutation $\mathbf{P} \in \mathcal{P}_n$ and non-zero scalar a.*

Proof. By Corollary 4.13, we can write $\mathbf{G}_1'\mathbf{M}_1 = \mathbf{S}'^{-1}\mathbf{H}_1'$. By Lemma 4.14 and Observation 4.6, the last row of $\mathbf{S}'^{-1}\mathbf{H}_1'$, and hence the last row of $\mathbf{G}_1'\mathbf{M}_1$, must be of the form $(a, a, \ldots, a)$ for some non-zero scalar a. By construction, the last row of $\mathbf{G}_1'$ contains only ones. Since $\mathbf{M}_1$ acts on $\mathbf{G}_1'$ by permuting and scaling the columns of $\mathbf{G}_1'$, and since the last row of $\mathbf{G}_1'\mathbf{M}_1$ contains identical entries a, we infer that $\mathbf{M}_1$ must multiply each column by the same scalar a. Therefore, $\mathbf{M}_1 = a \cdot \mathbf{P}$ for some unsigned permutation $\mathbf{P} \in \mathcal{P}_n$ and non-zero scalar a. □

Finally, we use the structure of the permutation and change-of-basis matrix for any LCE matrices $\mathbf{G}'$ and $\mathbf{H}'$ to show that the original matrices $\mathbf{G}$ and $\mathbf{H}$ must be in PCE.

Corollary 4.16. *For any $\mathbf{G}, \mathbf{H} \in \mathbb{F}_q^{k \times n}$, if $(\mathbf{G}', \mathbf{H}')$ is in LCE, then $(\mathbf{G}, \mathbf{H})$ is in PCE.*

Proof. If $(\mathbf{G}', \mathbf{H}')$ is in LCE, then there exists a monomial matrix $\mathbf{M}' \in \mathcal{M}_{n'}$ and an invertible matrix $\mathbf{S}' \in \mathrm{GL}_{k+1}$ such that $\mathbf{S}'\mathbf{G}'\mathbf{M}' = \mathbf{H}'$. By Corollary 4.13, we know that $\mathbf{M}'$ is comprised of three block matrices $\mathbf{M}_1 \in \mathcal{M}_n$, $\mathbf{M}_2 \in \mathcal{M}_{nm}$, and $\mathbf{M}_3 \in \mathcal{M}_{nm+1}$ which act exclusively on the first n, next nm, and last $nm+1$ columns of $\mathbf{G}'$, respectively. By Corollary 4.15, we know that $\mathbf{M}_1 = a \cdot \mathbf{P}$ for some unsigned permutation $\mathbf{P} \in \mathcal{P}_n$ and non-zero scalar a. By Lemma 4.14, the last row and last column of $\mathbf{S}'$ contain only zeros, except in the last entry. Let $\mathbf{S} \in \mathrm{GL}_k$ denote the top-left block submatrix of $\mathbf{S}'$ consisting of the intersection of the first k rows and k columns (see Figure 4.3). Since $\mathbf{S}'$ must have a non-zero determinant, this implies that $\mathbf{S}$ must be invertible. Then since a is non-zero, $a \cdot \mathbf{S}$ is also an invertible matrix.

By construction of $\mathbf{G}'$ and $\mathbf{H}'$, and the implications of Lemma 4.14 and Corollary 4.13, we have the block matrix product $\mathbf{S}\mathbf{G}\mathbf{M}_1 = \mathbf{H}$. Then for the matrices $(a \cdot \mathbf{S})$ and $\mathbf{P}$,

$$(a \cdot \mathbf{S})\mathbf{G}\mathbf{P} = \mathbf{S}\ \mathbf{G}\ (a \cdot \mathbf{P}) = \mathbf{S}\mathbf{G}\mathbf{M}_1 = \mathbf{H}.$$

Therefore, $(\mathbf{G}, \mathbf{H})$ is in PCE. □

For the reverse direction of Theorem 4.5, the proof is nearly identical to the one for LCE, but with the assumption that all non-zero scalars are restricted to the set of signs $\{-1, +1\}$.

4.2 Future Directions

Our results imply that the CE variants PCE, SPCE, and LCE are computationally equivalent, from the perspective of polynomial-time algorithms. In particular, we have shown that LCE and SPCE are at least as hard as PCE. So, to study the hardness of the former two problems, it suffices to consider the hardness of PCE. We now have more reductions among the CE variants, and in consequence, from these variants to LIP. It remains, however, an open problem to reduce LIP to any CE problem. While there has been some small progress in this direction, namely Ducas and Gibbons' Turing reduction from LIP for Construction A lattices to SPCE for codes with zero hull [DG23], no other progress has yet been made.

Another interesting open problem is to improve the runtime of the reduction from LCE to PCE proven in [SS13a] from $\mathrm{poly}(n, q)$ to $\mathrm{poly}(n, \log q)$. In light of our results, this would give a tighter equivalence between LCE and PCE. The reduction from [SS13a] requires time polynomial in the alphabet size q because the closure (as described in Chapter 4) increases the block length of the codes from n to $(q-1)n$. Any reduction running in $\log q$ time would either need to use a new type of closure that requires only a $\log q$ increase in block length, or a more efficient way to transform scalar multiplication operations into permutations.

CHAPTER 5

List-Decoding Generalized Reed-Solomon Codes

Based on joint work with Chris Peikert.

The concept of list-decoding was introduced independently by Elias [Eli57] and Wozencraft [Woz58] to enable decoding from a large amount of error that causes ambiguity in determining the original codeword. The goal of list-decoding is to find *all* possible codewords within a certain distance of a received word. The size of the list of possible codewords can be limited so that the list can be efficiently searched. While list-decoding began as a problem of improving reliable communication, it has found numerous applications throughout computer science.

Reed-Solomon codes are a family of codes that have found applications in many areas and are widely used in practice for data transmission and wireless communication. In their seminal work [GS99], Guruswami and Sudan presented an algorithm that efficiently list-decodes Reed-Solomon codes from errors in the Hamming metric. The Guruswami-Sudan (GS) algorithm was later extended by Guruswami in [Gur01] and Koetter and Vardy in [KV03] into a *soft-decision* algorithm, which takes as input a vector of "weights" that can indicate the likelihood that a symbol was transmitted in a particular coordinate. In this work, we will focus on a more general family of codes, called (Generalized) Reed-Solomon (GRS) codes (see Definition 2.3 for formal definition).

Prior Work. There has been some work on decoding GRS codes from error not measured in Hamming distance. Roth and Siegel in[RS94] presented a unique-decoding algorithm for decoding from discrete errors in the *Lee metric*, which is a generalization of the Hamming metric equivalent to the ℓ_1 norm with wrap-around. Their algorithm applies to a certain type of GRS codes and a larger class of codes they call (shortened) BCH codes. Recently, Mook and Peikert in [MP22] presented a list-decoding algorithm for the ℓ_2 norm, which uses

the GS algorithm as a black box. Their results are stated for a family of subfield subcodes of Reed-Solomon codes, called BCH codes, but apply more generally to GRS codes.

Our Contributions. In this work, we present an algorithm based on the GS soft-decision algorithm that list-decodes Generalized Reed-Solomon codes from error in the ℓ_p (quasi)norm for any $0 < p \leq 2$ (see Section 2.1 for definition). Our algorithm translates a given received word into a weight vector parameterized by a function that, conceptually, matches the channel's probability density function (even if the channel is adversarial). For each coordinate of the received word, this assigns weights to all possible symbols in the alphabet so that the ones closest to the received coordinate have the largest weight but all symbols have non-zero weight. In contrast, the algorithm in [MP22] only considers the two neighboring symbols.

Our algorithm achieves a better rate-distance trade-off than the prior algorithms for ℓ_1 and ℓ_2 described above. For $p = 2$, our algorithm works for arbitrarily large decoding distance and for larger rates than the one from [MP22] when the relative distance is at least a quantity slightly larger than half (see Section 5.3.2 for details). For $p = 1$, our algorithm can *list*-decode a more general family of GRS codes from *continous* error for arbitrarily large decoding distances, while the previous algorithm from [RS94] could only unique-decode from discrete error. Our algorithm works for larger rates for any relative distance slightly larger than three-quarters (see Section 5.4.2 for details).

In addition to decoding worst-case errors incurred by an adversarial channel, our algorithm also works for average-case errors produced by a probabilistic channel. A simple approach is to bound random error with high probability and then apply a worst-case algorithm. To our knowledge, ours is the first algorithm to list-decode GRS codes from average-case errors in ℓ_p (quasi)norms that outperforms this trivial approach.

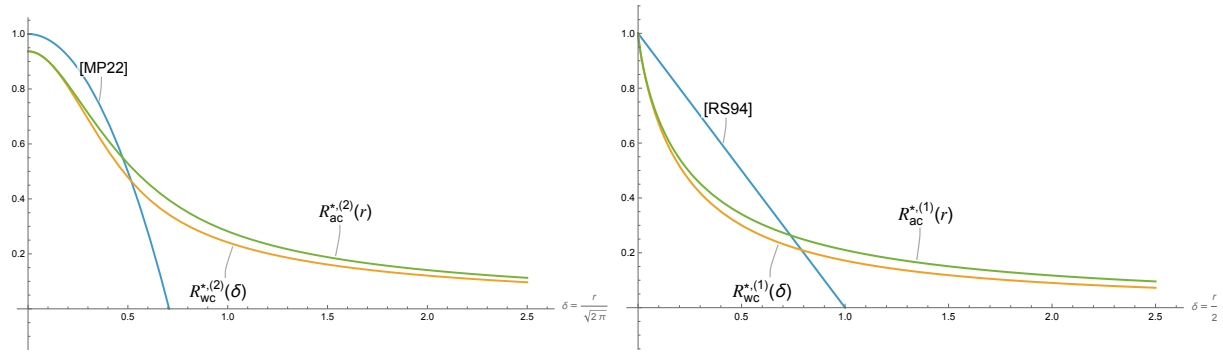


Figure 5.1: Plots of the adjusted rate $R^{*,(p)}$, as a function of the relative ℓ_p decoding distance $\delta = d/n^{1/p}$ or corresponding channel error width $r = p^{1/p} \cdot c_p \cdot \delta$, for which our algorithm can list decode GRS codes in the worst case (wc) or average case (ac), respectively, for $p = 2$ (left) and $p = 1$ (right). (For simplicity, these plots assume a field size $q \gg \delta, r$.) For comparison, also shown are the corresponding functions from the prior work on list decoding GRS codes in the ℓ_2 norm [MP22], and for *unique* decoding a *special subclass* of GRS codes in the ℓ_1 norm [RS94].

5.1 List-Decoding Reed–Solomon Codes

5.1.1 Soft-Decision Decoding

To list-decode Reed–Solomon codes under various norms and probabilistic channel models, we use the "weighted," or *soft-decision*, list decoder of Guruswami and Sudan (hereafter GS) [GS99], as elaborated upon in Guruswami's thesis [Gur01, Section 6.2.10] and the work of Koetter and Vardy [KV03]. A soft-decision decoder takes a "weight vector" as input, and outputs a set of codewords.

Definition 5.1. A *weight vector* for an n-length code over $\mathbb{F}_q$ is some $W := (W_1, \ldots, W_n) \in [0,1]^{qn}$ where each block $W_i \in [0,1]^q$ is indexed by $\mathbb{F}_q$; equivalently, it is a function $W_i \colon \mathbb{F}_q \to [0,1]$.

Conceptually, each block W_i of a weight vector may be thought of as specifying a (posterior) probability distribution Π_i over $\mathbb{F}_q$, where $\Pi_i(x)$ is proportional to the probability that the ith transmitted symbol was $x \in \mathbb{F}_q$, given what was received from the channel (which need not be an element of $\mathbb{F}_q$). At a formal level, this interpretation makes sense only when the channel is *probabilistic* (for average-case decoding), but it still serves as useful intuition when the channel is *adversarial* (for worst-case decoding). We consider both types of channels in our results below.

For $c \in \mathbb{F}_q$, define $[c] \in [0,1]^q$ to be the binary indicator vector indexed by $\mathbb{F}_q$ that has a 1

in coordinate c and 0s elsewhere. Similarly, for any vector $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$, define the weight vector $[\mathbf{c}] := ([c_1], \ldots, [c_n]) \in [0,1]^{qn}$. Observe that its Euclidean norm is $\|[\mathbf{c}]\| = \sqrt{n}$.

Definition 5.2. The *correlation* between a weight vector $W \in [0,1]^{qn}$ and a word $\mathbf{c} \in \mathbb{F}_q^n$ is defined as their length-normalized inner product (or the cosine of the angle between them):

$$\operatorname{corr}(W, \mathbf{c}) := \frac{\langle W, [\mathbf{c}] \rangle}{\|W\| \cdot \sqrt{n}} \,. \tag{5.1}$$

Theorem 5.3 (adapted from [GS99, Theorem 18] and [Gur01, Theorem 6.21])**.** *For a prime power q, let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a Generalized Reed–Solomon code of dimension k and adjusted rate $R^* := (k-1)/n$. There is a deterministic algorithm that, given a weight vector W and a "tolerance" $\tau > 0$, outputs in time* $\operatorname{poly}(n, q, 1/(\tau\|W\|))$ *the set of all codewords $\mathbf{c} \in \mathcal{C}$ that satisfy*

$$\operatorname{corr}(W, \mathbf{c}) \geq \sqrt{R^*} + \tau \,. \tag{5.2}$$

We remark that the above theorem is originally stated for *rational* weights, but the supporting argument (from [Gur01, Lemma 6.20]) easily adapts to handle *real*-valued weights that can be lower bounded to any needed precision in polynomial time, as all of ours can be.

5.1.2 From Received Words to Weight Vectors

Here we describe a general approach for translating a received word to a weight vector. This translation is parameterized by a function that, conceptually, can be viewed as (proportional to) the channel's probability density function, even if the channel is not actually probabilistic.

Let $f\colon \mathbb{R} \to [0,1]$ be a function that satisfies Assumption 2.44, extended multiplicatively to $\mathbb{R}^n$ as in Equation (2.1), and recall that $f_s(x) := f(x/s)$ for any constant $s > 0$. Next let q be a positive integer, and recall that we identify $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ with $\mathbb{F}_q$ in the natural way when q is prime. Let the set of possible received values be $\mathbb{R}_q = \mathbb{R}/q\mathbb{Z}$, and for any such value $y \in \mathbb{R}_q$, define the weight function $W_{s,y}\colon \mathbb{Z}_q \to [0,1]$ by

$$W_{s,y}(x) := f_s(y - x + q\mathbb{Z}) \,. \tag{5.3}$$

Notice that here f_s is applied to a *coset* of $q\mathbb{Z}$, which represents an infinite series; for all our concrete choices, these series converge and so the function $W_{s,y}$ is well defined. This function can also be seen as the vector $W_{s,y} := (W_{s,y}(x))_{x \in \mathbb{Z}_q} \in [0,1]^q$, indexed by $\mathbb{Z}_q$.

In line with the probabilistic conception of weight vectors from Section 5.1.1 above, the function $W_{s,y}$ can be seen as follows. Suppose that a uniformly random symbol in $\mathbb{Z}_q$ is

sent over a channel, which adds (modulo q) noise drawn from a distribution over $\mathbb{R}$ whose probability density function is proportional to f_s. Then the probability that the sent symbol was $x \in \mathbb{Z}_q$, conditioned on receiving y, is proportional to $W_{s,y}(x)$. This is because the coset $y - x \in \mathbb{R}_q$ is the set of all noise values that yield y if x is sent. Note that in the definition of $W_{s,y}$ we do *not* normalize by the total weight $W_{s,y}(\mathbb{Z}_q) = f_s(y + \mathbb{Z})$ (which may vary based on the received value y); this turns out to yield simpler analyses and tighter results in the end.

Definition 5.4. For a function f_s as above and any received vector $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}_q^n$, define the corresponding weight vector as

$$W_{s,\mathbf{y}} := (W_{s,y_1}, \ldots, W_{s,y_n}) \in [0,1]^{nq} \ . \tag{5.4}$$

In order to use the soft-decision algorithm (Theorem 5.3) for decoding under an adversarial channel, it suffices to show that we can choose a suitable s so that for any received word $\mathbf{y}$ and any sufficiently close codeword $\mathbf{c}$ (in the norm of interest), the correlation $\mathrm{corr}(W_{s,\mathbf{y}}, \mathbf{c})$ satisfies (5.2). Similarly, for decoding under a probabilistic channel, it suffices to show that with high probability over the channel noise $\mathbf{e}$, the transmitted codeword $\mathbf{c}$ has large enough correlation with the weight vector $W_{s,\mathbf{y}}$ of the received word $\mathbf{y} = \mathbf{c} + \mathbf{e}$ (again, for some suitably chosen s). To this end, in what follows we give a lower bound on $\langle W_{s,\mathbf{y}}, [\mathbf{c}] \rangle$ and an upper bound on $\|W_{s,\mathbf{y}}\|$, in terms of f_s and the difference $\mathbf{y} - \mathbf{c}$ between the received word and the codeword of interest.

5.1.3 Main Theorem

Here we state and prove the main result of this section. For this we define the two-dimensional integer lattice $\mathcal{L}_q$ that consists of all shifts of the lattice $q\mathbb{Z}^2$ by (z, z) for an integer z, i.e.,

$$\mathcal{L}_q := \bigcup_{x \in \mathbb{Z}_q} (x \oplus x) = \bigcup_{z \in \mathbb{Z}} ((z,z) + q\mathbb{Z}^2) \supset q\mathbb{Z}^2. \tag{5.5}$$

We have that $\det(\mathcal{L}_q) = q$, and so $\det(\mathcal{L}_q^*) = 1/q$. We sometimes omit the q subscript when it is clear from context or its value is unimportant.

Theorem 5.5. *For any $s > 0$ and $\mathbf{y} \in \mathbb{R}_q^n$ defining $W = W_{s,\mathbf{y}}$, and any $\mathbf{c} \in \mathbb{Z}_q^n$,*

$$\mathrm{corr}(W, \mathbf{c}) \geq \frac{\mathrm{Avg}_{i \in [n]}[f_s(y_i - c_i)]}{\sqrt{f_s(\mathcal{L}_q)}} \geq \frac{f_s(\mathbf{y} - \mathbf{c})^{1/n}}{\sqrt{f_s(\mathcal{L}_q)}} \ .$$

Proof. This follows immediately from the following lower and upper bounds on the numerator and denominator of $\mathrm{corr}(W, \mathbf{c}) = \frac{\langle W, [\mathbf{c}] \rangle / n}{\|W\|/\sqrt{n}}$. For the numerator, by the definitions of W and $[\mathbf{c}]$,

$$\langle W, [\mathbf{c}] \rangle / n = \operatorname*{Avg}_{i \in [n]} [W_{s,y_i}(c_i)] = \operatorname*{Avg}_{i \in [n]} [f_s(y_i - c_i)] \geq f_s(\mathbf{y} - \mathbf{c})^{1/n} \ ,$$

where the last step follows by the inequality of arithmetic and geometric means, and the non-negativity and multiplicativity of f_s over direct sums of cosets (Lemma 2.1). For the denominator, the upper bound $\|W\|/\sqrt{n} \leq \sqrt{f_s(\mathcal{L}_q)}$ is proved in Lemma 5.7 below. □

Remark 5.6. If certain coordinates of $\mathbf{y}$ are "very far" from the corresponding entries of $\mathbf{c}$, we may get a better lower bound on $\mathrm{corr}(W, \mathbf{c})$ by *restricting* to "good" coordinates. Specifically, for any nonempty $G \subseteq [n]$ of cardinality $g = |G|$, by the non-negativity of f_s,

$$\operatorname*{Avg}_{i \in [n]} [f_s(y_i - c_i)] \geq \frac{g}{n} \cdot \operatorname*{Avg}_{i \in G} [f_s(y_i - c_i)] \geq \frac{g}{n} \cdot f_s(\mathbf{y}_G - \mathbf{c}_G)^{1/g} \ ,$$

where $\mathbf{x}_G$ is the vector obtained by restricting $\mathbf{x}$ to the coordinates in G.

Lemma 5.7. *Adopting the notation from Theorem 5.5, and letting* $\tilde{\varepsilon} = \varepsilon_{\mathcal{L}_q,s}(H)$ *where* $H = \mathrm{span}(1, 1)$*,*

$$\|W\|^2/n \in f_s(\mathcal{L}_q) \cdot \Big[\frac{1 - \tilde{\varepsilon}}{1 + \tilde{\varepsilon}}, 1\Big] \ .$$

Proof. By definition of W,

$$\|W\|^2/n = \operatorname*{Avg}_{i \in [n]} \Big[\sum_{x \in \mathbb{Z}_q} f_s(y_i - x)^2\Big] \ .$$

To bound this, let $y \in \mathbb{R}_q$ be arbitrary. By Lemma 2.1,

$$\begin{aligned}
\sum_{x \in \mathbb{Z}_q} f_s(y - x)^2 &= \sum_{x \in \mathbb{Z}_q} f_s((y - x) \oplus (y - x)) \\
&= \sum_{x \in \mathbb{Z}_q} f_s((y \oplus y) - (x \oplus x)) \\
&= f_s((\overline{y}, \overline{y}) + \mathcal{L}_q) \\
&\in f_s(\mathcal{L}_q) \cdot \Big[\frac{1 - \tilde{\varepsilon}}{1 + \tilde{\varepsilon}}, 1\Big] \ ,
\end{aligned}$$

where the last step follows by the latter part of Lemma 2.46 on the lattice $\mathcal{L}_q$ with subspace H, and noting that $(\overline{y}, \overline{y}) \in H$. The claim follows by averaging over $i \in [n]$. □

5.1.4 Average-Case Decoding

Here we consider list-decoding in the *average case*, where the channel is probabilistic (not worst case) and the goal is to output a list of codewords that includes the transmitted one. We consider channels that add independent, identically distributed random error (drawn from some specified distribution) to each coordinate of the transmitted codeword; this is often known as a *memoryless additive channel*. Specifically, we assume that the channel's error distribution (for each coordinate) is proportional to f_r for some $r > 0$, i.e., it has probability density function

$$D_r(x) := \frac{f_r(x)}{\widehat{f_r}(0)} \, . \tag{5.6}$$

For example, if f_r is a Gaussian function, this is known as the *additive white Gaussian noise* (AWGN) channel model. In some settings one may also consider a *discrete* channel distribution, e.g., over $\mathbb{Z}$, in which case its probability mass function is $D_r(x) := f_r(x)/f_r(\mathbb{Z})$. For any $s > 0$ (which may differ from r), define

$$\mu_{r,s} := \mathop{\mathbb{E}}_{e \leftarrow D_r} [f_s(e)] \, . \tag{5.7}$$

In Section 5.2 we will use the following bound for a specific family of functions f to show that the transmitted codeword is recovered with high probability over the channel error.

Lemma 5.8. *For any $r, s > 0$ and T defining $\gamma := \mu_{r,s} - T \cdot \sqrt{f_s(\mathcal{L}_q)} \geq 0$, and any $\mathbf{c} \in \mathbb{Z}_q^n$,*

$$\Pr_{\mathbf{e} \leftarrow D_r^n} \big[\mathrm{corr}(W_{s,\mathbf{c}+\mathbf{e}}, \mathbf{c}) \leq T \big] < \exp(-2\gamma^2 n) \, .$$

Proof. The error coordinates e_i are drawn independently from D_r, so by Assumption 2.44 the values $f_s(e_i)$ are independent and identically distributed random variables in $[0, 1]$, with expected value $\mu_{r,s}$. Then by Theorem 5.5 and Hoeffding's inequality (Lemma 2.30),

$$\begin{aligned} \Pr_{\mathbf{e}}[\mathrm{corr}(W_{s,\mathbf{c}+\mathbf{e}}, \mathbf{c}) \leq T] &\leq \Pr_{\mathbf{e}} \big[\mathop{\mathrm{Avg}}_{i \in [n]} [f_s(e_i)] \leq T \cdot \sqrt{f_s(\mathcal{L}_q)} \big] \\ &< \exp(-2\gamma^2 n) \, . \end{aligned}$$

□

5.2 General ℓ_p (Quasi)Norms

For any $p > 0$, define $\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$ for $\mathbf{x} \in \mathbb{R}^n$. It is well known that this is a norm if and only if $p \geq 1$, and is a *quasinorm* for any $p > 0$.[1] In this section we define weight vectors via Definition 5.4 using the function $f : \mathbb{R} \to [0, 1]$ defined as

$$f(x) = f^{(p)}(x) := \exp\bigl(-(c_p\,|x|)^p\bigr) \qquad (5.8)$$
$$\text{where } c_p := 2 \cdot \Gamma(1 + 1/p)\ ,$$

where the gamma function $\Gamma(z) = \int_0^\infty u^{z-1} \exp(-u)\,\mathrm{d}u$ for $z > 0$, and satisfies $\Gamma(1) = 1$ and $\Gamma(1 + z) = z \cdot \Gamma(z)$. As two important examples, $c_1 = 2$ and $c_2 = \sqrt{\pi}$.

Note that by multiplicativity (Equation (2.1)),

$$f(\mathbf{x}) = \prod_{i=1}^n f(x_i) = \exp\Bigl(-\sum_{i=1}^n (c_p\,|x_i|)^p\Bigr) = \exp\bigl(-(c_p\|\mathbf{x}\|_p)^p\bigr) = f(\|\mathbf{x}\|_p)\ .$$

Regarding the Fourier transform of f, the "normalizing" constant c_p has been defined to make $\widehat{f}(0) = 1$:

$$\widehat{f}(0) = \int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 2\int_0^\infty \exp\bigl(-(c_p x)^p\bigr)\,\mathrm{d}x = \frac{2}{p \cdot c_p}\int_0^\infty u^{1/p-1}\exp(-u)\,\mathrm{d}u = \frac{2 \cdot \Gamma(1/p)}{p \cdot c_p} = 1\ ,$$

by the change of variable $u = (c_p x)^p$. It is also known that $\widehat{f}$ is non-negative for $0 < p \leq 2$; this follows immediately from an elegant lemma and proof due to Logan, given in [EOR91, Lemma 5]. For $p > 2$ this is no longer the case; this fact prohibits our framework from supporting ℓ_p norms for larger values of p. So, f satisfies Assumption 2.44 for such p. Another immediate consequence of Logan's lemma is that as s grows, $\widehat{f}_s(w)/s$ strictly decreases and approaches zero for every $w \neq 0$.

We will need the following simple lemma.

Lemma 5.9. *For any $s > 0$ and $\mathbf{y} \in \mathbb{R}_q^n$ defining $W = W_{s,\mathbf{y}}$, we have that $\|W\|_2 \geq \sqrt{n}/\exp(c_p^p/(2s)^p)$.*

Proof. Observe that each of the n blocks of W has an entry indexed by some $x \in \mathbb{Z}_q$ for which $|\overline{y_i - x}| \leq 1/2$. This entry's contribution to $\|W\|^2$ is at least $f_s(1/2)^2 = \exp(-2c_p^p/(2s)^p)$. □

[1] A quasinorm relaxes the triangle inequality axiom to require only that $\|\mathbf{x}+\mathbf{y}\| \leq K(\|\mathbf{x}\|+\|\mathbf{y}\|)$ for some fixed K. We do not use the triangle inequality, or even its relaxation, so we can consider $p < 1$.

5.2.1 Worst-Case Decoding

We now address list-decoding in the ℓ_p (quasi)norm for $0 < p \leq 2$, under worst-case error. Consider decoding distance $d = \delta \cdot n^{1/p}$, where n is the code length, and δ can be seen as the *relative* decoding distance (relative to $n^{1/p}$, which is the most natural normalization factor for ℓ_p). For $s > 0$, relative distance $\delta \geq 0$, and positive integer modulus q, define

$$B^{(p)}_{q,\delta}(s) := \frac{f_s(\delta)}{\sqrt{f_s(\mathcal{L}_q)}} = \frac{\exp\big(-(c_p \cdot \delta/s)^p\big)}{\sqrt{f_s(\mathcal{L}_q)}} \geq 0 \ . \tag{5.9}$$

By Theorems 5.3 and 5.5, to decode a GRS code of adjusted rate R^* over a prime field $\mathbb{F}_q$ to within ℓ_p distance $\delta \cdot n^{1/p}$ using the GS algorithm, it suffices to set $s > 0$ so that $B^{(p)}_{q,\delta}(s) > \sqrt{R^*}$. In other words, we can decode under relative distance δ for any R^* less than

$$R^{*,(p)}_{\mathrm{wc},q}(\delta) := \sup_{s>0} B^{(p)}_{q,\delta}(s)^2 \ . \tag{5.10}$$

The following theorem makes this formal.

Theorem 5.10. *For any $0 < p \leq 2$, $\delta \geq 0$, and prime q, the GS soft-decision algorithm using weight vector given by $f^{(p)}_s$ for any $s > 0$ list-decodes, up to ℓ_p distance $d = \delta \cdot n^{1/p}$, any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate $R^* < B^{(p)}_{q,\delta}(s)^2$, in time polynomial in n, q, and $\exp(1/s^p)/(B^{(p)}_{q,\delta}(s) - \sqrt{R^*})$.*[2]

Proof. We invoke the GS algorithm on the weight vector $W = W_{s,\mathbf{y}}$ given by the choice of s and the received word $\mathbf{y}$, and tolerance $\tau = B^{(p)}_{q,\delta}(s) - \sqrt{R^*} > 0$.[3] The running time is polynomial in n, q, and $1/(\tau\|W\|_2) \leq \exp(c_p^p/(2s)^p)/(\tau\sqrt{n})$, by Lemma 5.9.

Now let $\mathbf{c} \in \mathcal{C}$ be a codeword within distance d of $\mathbf{y}$, i.e., $\|\overline{\mathbf{y} - \mathbf{c}}\|_p \leq d$. By Theorem 5.5, Assumption 2.44, and Equation (5.9),

$$\mathrm{corr}(W, \mathbf{c}) \geq \frac{f_s(\mathbf{y} - \mathbf{c})^{1/n}}{\sqrt{f_s(\mathcal{L}_q)}} \geq \frac{f_s(d)^{1/n}}{\sqrt{f_s(\mathcal{L}_q)}} = B^{(p)}_{q,\delta}(s) = \sqrt{R^*} + \tau \ .$$

So, by Theorem 5.3, the output of the GS algorithm includes $\mathbf{c}$, as needed. □

Remark 5.11. Following Remark 5.6, suppose that the received word $\mathbf{y}$ is within relative distance δ of a codeword $\mathbf{c}$ on some subset $G \subseteq [n]$ of $g = |G|$ coordinates, i.e., $\|\overline{\mathbf{y}_G - \mathbf{c}_G}\| \leq$

[2] We remark that in many cases, the bound on the polynomial running time can be improved using a better lower bound for $\|W\|_2$, such as the one given by Lemma 5.7.

[3] To be more precise, we can invoke GS on any approximation of τ in $[\tau/2, \tau]$, say. This can be computed by approximating $f_s(\mathcal{L}_q)$ from above to the needed precision, by enumerating sufficiently many points of $\mathcal{L}_q$ near the origin, and upper-bounding the contribution of the remaining points in the "tails" using, e.g., Lemma 5.18.

$\delta \cdot g^{1/p}$, and the remaining coordinates of $\mathbf{y}$ may be *arbitrary*. Then $\mathrm{corr}(W_{s,\mathbf{y}}, \mathbf{c}) \geq (g/n) \cdot B^{(p)}_{q,\delta}(s)$. So, we can list-decode all such codewords as long as the adjusted rate $R^* < (g/n)^2 \cdot R^{*,(p)}_{\mathrm{wc},q}(\delta)$.

Remark 5.12. Interestingly, as δ, q/δ, and n grow (and the other parameters remain fixed), the product of the relative distance δ and the adjusted rate R^* for which we can decode approaches the *relative radius of a unit-volume ℓ_p ball*. To see this, first observe that as q/s grows, $f_s(\mathcal{L}_q)$ approaches

$$f_s(\mathcal{L}_q \cap \mathrm{span}(1,1)) = \sum_{z \in \mathbb{Z}} f_s(z,z) = \sum_{z \in \mathbb{Z}} f_{s/2^{1/p}}(z) = f_{s/2^{1/p}}(\mathbb{Z}) = \widehat{f_{s/2^{1/p}}}(\mathbb{Z}) \ ,$$

where the last equality is by the PSF (Lemma 2.26). Then as s grows, the above approaches $s/2^{1/p}$, because $\widehat{f}(0) = 1$ and hence $\widehat{f_{s/2^{1/p}}}(0) = s/2^{1/p}$ by Lemma 2.25, and the other Fourier coefficients approach zero.

So, as s and q/s grow, the bound $B^{(p)}_{q,\delta}(s)^2$ on the adjusted rate approaches $2^{1/p} \cdot \exp\bigl(-(2^{1/p} c_p \cdot \delta/s)^p\bigr)/s$. A straightforward calculation using the change of variable $t = (2^{1/p} c_p \cdot \delta/s)^p$ shows that this is maximized for $t = 1/p$ (and hence $s = (2p)^{1/p} c_p \cdot \delta$), so we can decode to within relative ℓ_p distance δ for an adjusted rate R^* approaching

$$\frac{1}{(ep)^{1/p} \cdot c_p \cdot \delta} \ .$$

By comparison, it is known that the volume of an n-dimensional ℓ_p ball of unit relative radius (i.e., radius $n^{1/p}$) has nth root

$$\frac{2 \cdot \Gamma(1 + 1/p)}{\Gamma(1 + n/p)^{1/n}} \cdot n^{1/p} \longrightarrow (ep)^{1/p} \cdot c_p \ ,$$

using Stirling's approximation for the denominator as n grows. So, the relative radius of a unit-volume ℓ_p ball is the reciprocal of this, which is what $R^* \cdot \delta$ approaches.

5.2.2 Average-Case Decoding

We now consider average-case decoding under a memoryless additive (continuous or discrete) channel whose density function is proportional to a scaling of $f = f^{(p)}$. Specifically, we consider the continuous distribution with probability density function $D_r(x) := f_r(x)/r$, and the discrete distribution over $\mathbb{Z}$ with probability mass function $D_r(x) := f_r(x)/f_r(\mathbb{Z})$.

Following Section 5.1.4, for any $r, s > 0$ define

$$\mu_{r,s}^{(p)} := \mu_{r,s} = \mathop{\mathbb{E}}_{e \leftarrow D_r} [f_s(e)] \, .$$

For these channel distributions we derive suitable bounds on $\mu_{r,s}^{(p)}$, then reach the conclusion via Lemma 5.8 and Theorem 5.3.

Lemma 5.13. *For any $0 < p \leq 2$, any $r > 0$ defining a continuous or discrete distribution D_r, and $s > 0$,*

$$\mu_{r,s}^{(p)} \geq \frac{s}{\|(r,s)\|_p} \, ,$$

with equality in the continuous case and strict inequality in the discrete case.

Proof. First note that for $t := (rs)/(r^p + s^p)^{1/p}$, we have $f_r(x) \cdot f_s(x) = f_t(x)$, since $1/t^p = 1/r^p + 1/s^p$. For the continuous case,

$$\mu_{r,s}^{(p)} = \int_{\mathbb{R}} D_r(e) \cdot f_s(e) \, \mathrm{d}e = \frac{1}{r} \int_{\mathbb{R}} f_t(e) \, \mathrm{d}e = \frac{t}{r} = \frac{s}{\|(r,s)\|_p} \, .$$

For the discrete case, since $t < r$, we have that $\widehat{f}_t(w)/t > \widehat{f}_r(w)/r$ for all $w \in \mathbb{Z}$, so by Lemma 2.26,

$$\mu_{r,s}^{(p)} = \sum_{e \in \mathbb{Z}} D_r(e) \cdot f_s(e) = \frac{f_t(\mathbb{Z})}{f_r(\mathbb{Z})} = \frac{t}{r} \cdot \frac{\widehat{f}_t(\mathbb{Z})/t}{\widehat{f}_r(\mathbb{Z})/r} > \frac{t}{r} \, .$$

□

Now, for any channel parameter $r > 0$ and for $s > 0$, define

$$A_{q,r}^{(p)}(s) := \frac{\mu_{r,s}^{(p)}}{\sqrt{f_s(\mathcal{L}_q)}} \geq \frac{s}{\|(r,s)\|_p \cdot \sqrt{f_s(\mathcal{L}_q)}} \, , \tag{5.11}$$

where the inequality is by Lemma 5.13. By Theorems 5.3 and 5.5, to decode (with high probability) a GRS code of adjusted rate R^* over a prime field $\mathbb{F}_q$ under a channel with parameter r, it suffices to set $s > 0$ so that $A_{q,r}^{(p)}(s) > \sqrt{R^*}$. In other words, we can decode under channel parameter r for any R^* less than

$$R_{\mathrm{ac},q}^{*,(p)}(r) := \sup_{s>0} A_{q,r}^{(p)}(s)^2 \, . \tag{5.12}$$

See Theorem 5.14 below for the formal statement.

We remark that Theorem 5.14 outperforms Theorem 5.10 (for worst-case decoding) by a factor that approaches $(e/2)^{1/p}$ in the adjusted rate R^* it can handle, as r and q/r grow.

Specifically, consider a channel with parameter r. A calculation reveals that its relative error (in the ℓ_p norm, relative to $n^{1/p}$) is tightly concentrated around $\delta = r/(p^{1/p} \cdot c_p)$, so following the analysis in Remark 5.12, Theorem 5.10 applies for R^* that approaches $1/(r \cdot e^{1/p})$. By comparison, Theorem 5.14 applies for R^* that approaches $1/(r \cdot 2^{1/p})$.

Theorem 5.14. *Let $0 < p \leq 2$, $r > 0$, $\alpha \in (0,1)$, and q be prime. Under a memoryless additive (continuous or discrete) channel with distribution D_r, the GS soft-decision algorithm, using weight vector given by $f_s^{(p)}$ for any $s > 0$, list-decodes any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate $R^* < A_{q,r}^{(p)}(s)^2$, in time polynomial in n, q, and $\exp(1/s^p)/(A_{q,r}^{(p)}(s) - \sqrt{R^*})$, except with probability less than*

$$\exp\left(-2n \cdot f_s(\mathcal{L}_q) \cdot \alpha^2 \cdot \left(A_{q,r}^{(p)}(s) - \sqrt{R^*}\right)^2\right) .$$

Proof. Throughout the proof let $A(s) := A_{q,r}^{(p)}(s)$. We invoke the GS algorithm on the weight vector $W = W_{s,\mathbf{y}}$ given by the choice of s and the received word $\mathbf{y}$, and tolerance $\tau = T - \sqrt{R^*} > 0$, where

$$\begin{aligned} T &= A(s) - \alpha(A(s) - \sqrt{R^*}) && (5.13)\\ &= \sqrt{R^*} + (1-\alpha)(A(s) - \sqrt{R^*}) \in (\sqrt{R^*}, A(s)) . && (5.14) \end{aligned}$$

The running time is polynomial in n, q, and $1/(\tau\|W\|_2) \leq \exp(c_p^p/(2s)^p)/(\tau\sqrt{n})$, by Lemma 5.9.

Now suppose that $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} \in \mathcal{C}$ is the transmitted codeword and $\mathbf{e} \leftarrow D_r^n$ is the channel error. To show that the list output by the GS algorithm contains $\mathbf{c}$ with the claimed probability, by Theorem 5.3 it suffices to show that $\mathrm{corr}(W, \mathbf{c}) \geq \sqrt{R^*} + \tau = T$. Following the setup of Lemma 5.8, let

$$\begin{aligned} \gamma &= \mu_{r,s} - T \cdot \sqrt{f_s(\mathcal{L}_q)} \\ &= \sqrt{f_s(\mathcal{L}_q)} \cdot (A(s) - T) && \text{(Equation (5.11))} \\ &= \sqrt{f_s(\mathcal{L}_q)} \cdot \alpha(A(s) - \sqrt{R^*}) > 0 && \text{(Equation (5.13)).} \end{aligned}$$

So, by Lemma 5.8, $\Pr_{\mathbf{e}}[\mathrm{corr}(W, \mathbf{c}) < T] < \exp(-2\gamma^2 n)$, which yields the claim. □

Remark 5.15. Following Remark 5.6, and similarly to Remark 5.11, suppose that there exists a subset $G \subseteq [n]$ of $g = |G|$ "good" coordinates for which the channel generates the received word $\mathbf{y}$ from the transmitted codeword $\mathbf{c}$ by adding independent noise from distribution D_r on those coordinates, and sets the remaining coordinates *arbitrarily*. Then our lower bound

on $\mathrm{corr}(W_{s,\mathbf{y}}, \mathbf{c})$ from Theorem 5.5 involves an extra g/n factor and an average over just the coordinates in G. So, we can correctly list-decode for any adjusted rate $R^* < (g/n)^2 \cdot A^{(p)}_{q,r}(s)^2$. More precisely, the statement and proof of Theorem 5.14 hold with every occurrence of $A^{(p)}_{q,r}(s)$ having an additional g/n factor, and with g in place of n in the failure probability.

5.3 The ℓ_2 Norm and Gaussian Error

In the remainder of the paper we instantiate our general list-decoding results for ℓ_p norms (Theorems 5.10 and 5.14) for specific norms of interest and memoryless additive channels. In this section, we consider the ℓ_2 norm and Gaussian channels.

We specialize Equation (5.8) to $p = 2$, i.e., the Gaussian function

$$f(x) := f^{(2)}(x) = \exp(-\pi x^2) \ .$$

By a straightforward calculation it can be seen that this function is its own Fourier transform: $\widehat{f} = f$. Note that $\widehat{f_s} = s \cdot f_{1/s}$ by the time-scaling property of the Fourier transform (Lemma 2.24). Finally, recalling that $f(\mathbf{x}) = f(\|\mathbf{x}\|_2)$, we get that f is invariant under rotations.

5.3.1 Bounds

In this subsection we derive fairly tight bounds on the factor $f_s(\mathcal{L}_q)$ that appears in the quantities that govern the adjusted rates under which we can decode in the worst and average cases (Equations (5.9) and (5.11), respectively). For this purpose we need to define a suitable "fudge factor." For $r \geq r_0 := \sqrt{\ln(4)/\pi} \approx 0.66428$, define

$$E(r) := 1 - 2\exp(-\pi r^2/2) \in [0, 1) \ . \tag{5.15}$$

Notice that $E(r)$ is positive for $r > r_0$, is strictly increasing, and rapidly approaches 1 as r increases. Next, for real s, q such that $s \in [r_0, q/r_0]$, define

$$E_q(s) := \sqrt{E(q/s) \cdot E(s)} \in [0, 1) \ . \tag{5.16}$$

Similarly, $E_q(s)$ is positive for $s \in (r_0, q/r_0)$, and rapidly approaches 1 as both $s, q/s$ increase.

Lemma 5.16. *For any real s and positive integer q such that $s \in (r_0, q/r_0)$,*

$$\frac{1}{f_s(\mathcal{L}_q)} > \frac{\sqrt{2}}{s} \cdot E_q(s)^2 \ .$$

Proof. This follows directly from Lemmas 5.17 and 5.19 below. Specifically, let $\varepsilon' = \varepsilon_{\mathbb{Z},q/(s\sqrt{2})}$ and $\tilde{\varepsilon} = \varepsilon_{\mathcal{L}_q,s}(H)$. By Lemma 5.19 (applied twice, with $r = q/s$ and $r = s$, which are both greater than r_0),

$$\frac{1}{(1+\varepsilon')(1+\tilde{\varepsilon})} > E_q(s)^2 \ .$$

The result then follows by Lemma 5.17. □

Lemma 5.17. *For any real $s > 0$ and positive integer q, let $\varepsilon' = \varepsilon_{\mathbb{Z},q/(s\sqrt{2})}$ and $\tilde{\varepsilon} = \varepsilon_{\mathcal{L}_q,s}(H)$ where $H = \mathrm{span}(1,1)$. Then*

$$f_s(\mathcal{L}_q) = \frac{s}{\sqrt{2}} \cdot (1+\varepsilon') \cdot (1+\tilde{\varepsilon}) \ .$$

Proof. Recall that $\mathcal{L} = \mathcal{L}_q$ has determinant $\det(\mathcal{L}) = q$, hence its dual has determinant $\det(\mathcal{L}^*) = 1/q$. A basis for $\mathcal{L}$ consists of the vectors $(1,1)$ and $(q,0)$, and its dual basis consists of the vectors $(0,1)$ and $(1,-1)/q$.

Since $H^\perp = \mathrm{span}(1,-1)$, we have that $\mathcal{L}^* \cap H^\perp$ consists merely of all the integer multiples of the dual basis vector $(1,-1)/q$, which has Euclidean norm $\sqrt{2}/q$. Therefore, $\mathcal{L}^* \cap H^\perp$ is a rotation of $(\sqrt{2}/q)\mathbb{Z}$. So,

$$\begin{aligned}
f_s(\mathcal{L}) &= (1/q) \cdot \widehat{f_s^2}(\mathcal{L}^* \cap H^\perp) \cdot (1+\tilde{\varepsilon}) && \text{(Lemma 2.46 and definition of } \tilde{\varepsilon}) \\
&= (s^2/q) \cdot f_{1/s}^2(\mathcal{L}^* \cap H^\perp) \cdot (1+\tilde{\varepsilon}) && \text{(Lemma 2.24)} \\
&= (s^2/q) \cdot f_{q/(s\sqrt{2})}(\mathbb{Z}) \cdot (1+\tilde{\varepsilon}) && \text{(rotational invariance of } f^2\text{, rescaling)} \\
&= (s/\sqrt{2}) \cdot (1+\varepsilon') \cdot (1+\tilde{\varepsilon}) && \text{(Lemma 2.46 and definition of } \varepsilon').
\end{aligned}$$

□

Next we bound the roughness quantities $\varepsilon', \tilde{\varepsilon}$ from Lemmas 5.16 and 5.17, using the following classic tail inequality.

Lemma 5.18 (adapted from [Ban95, Lemma 2.4])**.** *For any lattice $\mathcal{L}$, unit vector $\mathbf{u}$, and $s, t > 0$, let $T_{\mathbf{u},t} = \{\mathbf{x} : |\langle \mathbf{x}, \mathbf{u} \rangle| \geq t\}$. Then*

$$f_s(\mathcal{L} \cap T_{\mathbf{u},t}) < 2\exp(-\pi t^2/s^2) \cdot f_s(\mathcal{L}) \ .$$

Lemma 5.19. *Let $r > r_0$ and $H = \mathrm{span}(1,1)$. Then*

$$\frac{1}{1+\varepsilon_{\mathbb{Z},r/\sqrt{2}}} \ , \ \frac{1}{1+\varepsilon_{\mathcal{L}_q,r}(H)} > E(r) = 1 - 2\exp(-\pi r^2/2) \ .$$

Proof. We first bound $\varepsilon_{\mathbb{Z},r/\sqrt{2}}$. Let $s = r/\sqrt{2} > \sqrt{\ln(2)/\pi}$. By the definition of roughness (Definition 2.45) and the facts that $\mathbb{Z}^* = \mathbb{Z}$, $\widehat{f}_s = s \cdot f_{1/s}$ (Lemma 2.24), and $f_{1/s}(0) = 1$,

$$1 + \varepsilon_{\mathbb{Z},s} = \widehat{f}_s(\mathbb{Z})/\widehat{f}_s(0) = f_{1/s}(\mathbb{Z}) \ .$$

Now, by Lemma 5.18 and rearranging (where the denominator is positive due to the lower bound on s),

$$f_{1/s}(\mathbb{Z}) = f_{1/s}(0) + f_{1/s}(\mathbb{Z} \cap T_{1,1}) < 1 + 2\exp(-\pi s^2) \cdot f_{1/s}(\mathbb{Z}) \leq \frac{1}{1 - 2\exp(-\pi s^2)} \ ,$$

which yields the claim.

For $\varepsilon_{\mathcal{L},r}(H)$ where $\mathcal{L} = \mathcal{L}_q$, again by Definition 2.45 and Lemma 2.24,

$$1 + \varepsilon_{\mathcal{L},r}(H) = \frac{\widehat{f}_r(\mathcal{L}^*)}{\widehat{f}_r(\mathcal{L}^* \cap H^\perp)} = \frac{f_{1/r}(\mathcal{L}^*)}{f_{1/r}(\mathcal{L}^* \cap H^\perp)} \ .$$

Because $H^\perp = \text{span}(1,-1)$ and the vectors $(0,1), (1,-1)/q$ form a basis of $\mathcal{L}^*$, every point in $\mathcal{L}^*$ lies one of the lines (i.e., affine subspaces) $L_k = k \cdot (1,0) + H^\perp$ for some $k \in \mathbb{Z}$. The unit vector $\mathbf{u} = (1,1)/\sqrt{2}$ is orthogonal to $H^\perp$, so for any $\mathbf{x} \in L_k$, we have that $\langle \mathbf{x}, \mathbf{u} \rangle = k/\sqrt{2}$, and hence $|\langle \mathbf{x}, \mathbf{u} \rangle| \geq 1/\sqrt{2}$ if $k \neq 0$. Therefore, $\mathcal{L}^*$ can be partitioned as the disjoint union

$$\mathcal{L}^* = (\mathcal{L}^* \cap H) \cup (\mathcal{L}^* \cap T_{\mathbf{u},1/\sqrt{2}}) \ .$$

So, by Lemma 5.18 and rearranging (where again the denominator is positive due to the bound on r),

$$\begin{aligned} f_{1/r}(\mathcal{L}^*) &= f_{1/r}(\mathcal{L}^* \cap H) + f_{1/r}(\mathcal{L}^* \cap T_{\mathbf{u},1/\sqrt{2}}) \\ &< f_{1/r}(\mathcal{L}^* \cap H) + 2\exp(-\pi r^2/2) \cdot f_{1/r}(\mathcal{L}^*) \\ &\leq \frac{f_{1/r}(\mathcal{L}^* \cap H)}{1 - 2\exp(-\pi r^2/2)}. \end{aligned}$$

The result follows by dividing $f_{1/r}(\mathcal{L}^* \cap H)$ by both sides. □

5.3.2 Worst-Case Decoding

We now address list-decoding in the ℓ_2 norm, under worst-case error of bounded norm, by specializing the material of Section 5.2.1 to $p = 2$ and using our bounds on $f_s(\mathcal{L}_q)$ from Section 5.3.1. So, we consider decoding distance $d = \delta\sqrt{n}$, where n is the code length and δ

is the relative decoding distance. Then by Equations (5.9) and (5.10), we can list-decode for any R^* less than

$$R^{*,(2)}_{\mathrm{wc},q}(\delta) = \sup_{s>0} B^{(2)}_{q,\delta}(s)^2 > \sup_{s\in(r_0,q/r_0)} \frac{\sqrt{2}\cdot\exp(-2\pi\delta^2/s^2)}{s}\cdot E_q(s)^2\ , \tag{5.17}$$

where the inequality follows by Lemma 5.16.

Corollary 5.20 below is obtained by nearly maximizing the right-hand side of (5.17). More specifically, a standard calculation shows that taking $s = \delta\sqrt{4\pi}$ maximizes the "main term" $\sqrt{2}\cdot\exp(-2\pi\delta^2/s^2)/s$, to have value $1/(\delta\sqrt{2\pi e})$. For moderate or larger values of δ (and hence s), this very nearly maximizes the entire expression, because $E_q(s) \geq E(s)$ since $q/s \geq s$, and $E(s)$ rapidly approaches 1 as s grows. For example, $E(s)^2 \geq 1-10^{-8}$ for $\delta \geq 1$. So, as δ grows, the R^* for which we can list-decode rapidly approaches $1/(\delta\sqrt{2\pi e})$.

Corollary 5.20. *For any $\delta > \sqrt{\ln(4)}/(2\pi) \approx 0.1874$ and prime $q \geq 4\pi\delta^2$, the GS algorithm using weight vector given by f_s for $s = \delta\sqrt{4\pi}$ list-decodes, up to ℓ_2 distance $\delta\sqrt{n}$ in time $\mathrm{poly}(n, q, 1/(\sqrt{\widetilde{R}^{*,(2)}_{wc,q}(\delta)} - \sqrt{R^*}))$, any GRS code with adjusted rate*

$$R^* < \widetilde{R}^{*,(2)}_{wc,q}(\delta) := \frac{1}{\delta\sqrt{2\pi e}}\cdot E_q(\delta\sqrt{4\pi})^2\ .$$

Proof. For $s = \delta\sqrt{4\pi}$, the lower bounds on δ and q imply that $s = \delta\sqrt{4\pi} \in (r_0, q/r_0)$. Then by hypothesis and Lemma 5.16 and Equation (5.9),

$$R^* < \frac{1}{\delta\sqrt{2\pi e}}\cdot E_q(\delta\sqrt{4\pi})^2 < \frac{\exp(-2\pi\delta^2/s^2)}{f_s(\mathcal{L}_q)} = B^{(2)}_{q,\delta}(s)^2\ .$$

The claim then follows directly by Theorem 5.10. □

Comparison to [MP22]. The previous best result for list-decoding (Generalized) Reed–Solomon codes in the ℓ_2 norm was given by Mook and Peikert [MP22].[4]

Proposition 5.21 ([MP22, Theorem 3.4])**.** *For any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with any adjusted rate $R^* < 1$ and any $\varepsilon > 0$, there is a $\mathrm{poly}(n, q, 1/\varepsilon)$-time algorithm that list-decodes $\mathcal{C}$ up to ℓ_2 distance $d = \sqrt{n(1-R^*)(1-\varepsilon)/2}$.*

Equivalently, for a relative decoding distance $\delta = d/\sqrt{n} > 0$, the result from [MP22]

[4]By a standard reduction, the result from [MP22] also applies to GRS codes, not just RS codes as was originally stated.

works for adjusted rates R^* approaching $1 - 2\delta^2$, so it applies only for

$$\delta \leq \sqrt{(1 - R^*)/2} \leq 1/\sqrt{2} \, .$$

By contrast, our Theorem 5.10 works for *any* (arbitrarily large) $\delta > 0$ (and Corollary 5.20 gives a simpler and more explicit rate bound for any $\delta > 0.1875$). Moreover, for those δ for which both Theorem 5.10 and Proposition 5.21 apply, our result works for a larger R^* as long as $R^{*,(2)}_{\mathrm{wc},q}(\delta) > 1 - 2\delta^2$ (see (5.10)). For typical (moderate or larger) q, this holds for all $\delta \gtrapprox 0.51797$, which corresponds to $R^* \lessapprox 0.46342$. (For tiny $\delta \approx 0$, Theorem 5.10 works for $R^* \approx 0.93700$, whereas [MP22] works for $R^* \approx 1$, so the latter is better for very small distances.)

We also point out that [MP22] proves that for any $\delta \leq 1/2$, which corresponds to $R^* \geq 1/2$, its (very simple) choice of weight vector gives an *optimal* tradeoff between δ and R^* *for the GS/KV soft-decision algorithm and analysis*. However, the optimality argument breaks down for $\delta > 1/2$ (equivalently, for $R^* < 1/2$). And indeed, as we have just seen, we obtain a better distance-rate tradeoff than [MP22] for *almost all* such δ. This highlights the interesting question of determining an optimal choice of weights for the GS soft-decision algorithm for $\delta > 1/2$ (especially at the low end of this range).

5.3.3 Average-Case Decoding

We now consider average-case decoding under a memoryless additive (continuous or discrete) Gaussian channel, by specializing the material of Section 5.2.2 to $p = 2$ and using our bounds on $f_s(\mathcal{L}_q)$ from Section 5.3.1. Consider a Gaussian channel of parameter $r > 0$. Then by Equations (5.11) and (5.12), we can list-decode for any R^* less than

$$R^{*,(2)}_{\mathrm{ac},q} = \sup_{s>0} A^{(2)}_{q,r}(s)^2 > \sup_{s \in (r_0, q/r_0)} \frac{s\sqrt{2}}{r^2 + s^2} \cdot E_q(s)^2 \, , \tag{5.18}$$

where the inequality is by Lemma 5.16.

Corollary 5.22 below is obtained by nearly maximizing the right-hand side of (5.18). More specifically, setting $s = r$ maximizes the "main term" $s\sqrt{2}/(r^2 + s^2)$, to have value $1/(r\sqrt{2})$. As above, for moderate or larger values of r (and hence s), this very nearly maximizes the entire expression, because $E_q(s)$ rapidly approaches 1 as s grows.[5] So, as r grows, the rate R^* for which we can list-decode rapidly approaches $1/(r\sqrt{2})$.

[5] By contrast, $E_q(s) \ll 1$ for values of s very close to r_0, in which case the bound is maximized by taking s somewhat larger than r.

Corollary 5.22. *For any $r \in (r_0, q/r_0)$, $\alpha \in (0,1)$, and prime q, the GS algorithm using weight vector given by f_r list-decodes, in time* $\mathrm{poly}(n, q, 1/(\sqrt{\widetilde{R}^{*,(2)}_{ac,q}(r)} - \sqrt{R^*}))$*, any GRS code with adjusted rate*

$$R^* < \widetilde{R}^{*,(2)}_{ac,q}(r) := \frac{1}{r\sqrt{2}} \cdot E_q(r)^2 \ ,$$

except with probability less than $\exp\left(-\sqrt{2}n \cdot \alpha^2 \cdot r \cdot \left(\sqrt{\widetilde{R}^{*,(2)}_{ac,q}(r)} - \sqrt{R^*}\right)^2\right)$.

Proof. By hypothesis, Lemmas 5.13 and 5.16 and Equation (5.11),

$$R^* < \frac{1}{r\sqrt{2}} \cdot E_q(r)^2 < \frac{\mu^2_{r,r}}{f_r(\mathcal{L}_q)} = A^{(2)}_{q,r}(r)^2 \ .$$

The claim then follows directly by Theorem 5.14, and the fact that $f_r(\mathcal{L}_q) > r/\sqrt{2}$ by Lemma 5.17. □

5.4 The ℓ_1 Norm and Laplacian Error

In this section, we consider the ℓ_1 norm and Laplacian channels. We specialize Equation (5.8) to $p = 1$, i.e., the Laplacian function

$$f(x) := f^{(1)}(x) = \exp(-2\,|x|) \ .$$

(The Fourier transform of this function is given by $\widehat{f}(w) = 1/(1 + (\pi w)^2)$, but we will not use this; as already noted earlier, $f^{(1)}$ satisfies Assumption 2.44.)

Throughout this section we use the hyperbolic tangent function

$$\tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} < 1$$

and its reciprocal $\coth(x) = 1/\tanh(x) > 1$. Observe that $\tanh(x)$ approaches 1 as x grows; it also satisfies $\tanh(x) < x$ for all $x > 0$, and approaches x as x approaches zero.[6]

5.4.1 Bounds

In this subsection, we analyze the exact value of $f_s(\mathcal{L}_q)$ and derive an asymptotic bound. This appears in the quantities that govern the adjusted rates under which we can decode in the worst and average cases (Equations (5.9) and (5.11), respectively). For this purpose, we

[6]Both facts can be seen from the Taylor series $\tanh(x) = x - x^3/3 + \cdots$, valid for $|x| < \pi/2$.

define a suitable "fudge factor". For any real $x > 0$, define

$$E(x) := \Big(\coth(x) + \frac{4x \cdot e^{2x}}{(e^{2x}-1)^2}\Big)^{-1} \in (0,1) \ , \tag{5.19}$$

where the upper bound comes from the fact that $\coth(x) > 1$. Note that, as x grows, the first term in the sum rapidly approaches one, and the second term rapidly approaches zero. More precisely, a brief calculation reveals that

$$E(x) = 1 - O(x \cdot e^{-2x}) \ . \tag{5.20}$$

Lemma 5.23. *For any $s > 0$ and positive integer q,*

$$\frac{1}{f_s(\mathcal{L}_q)} > \tanh(2/s) \cdot E(q/s) \ .$$

Note that by Equation (5.20), for any fixed $s > 0$, as q (or equivalently, q/s) grows, $1/f_s(\mathcal{L}_q)$ rapidly approaches $\tanh(2/s)$. In turn, this approaches $2/s$ as s grows.

Proof. This follows directly from Lemma 5.24 below and Equation (5.19). By Lemma 5.24, the bound $\coth(2/s) > s/2$, and the definition of $E(x)$,

$$f_s(\mathcal{L}_q) < \coth(2/s)\Big(\coth(q/s) + \frac{2q \cdot e^{2q/s}}{(e^{2q/s}-1)^2} \cdot \frac{2}{s}\Big) = \frac{\coth(2/s)}{E(q/s)} \ .$$

The claim then follows by taking reciprocals. □

Lemma 5.24. *For any $s > 0$ and positive integer q,*

$$f_s(\mathcal{L}_q) = \coth(2/s) \cdot \coth(q/s) + \frac{2q \cdot e^{2q/s}}{(e^{2q/s}-1)^2} \ .$$

Proof. By Lemma 2.1, we can write

$$f_s(\mathcal{L}_q) = \sum_{\mathbf{v} \in \mathcal{L}_q} f_s(\mathbf{v}) = \sum_{x=0}^{q-1} f_s((x,x) + q\mathbb{Z}^2) = \sum_{x=0}^{q-1} f_s(x + q\mathbb{Z})^2.$$

Rewriting the term in the summation as a sum of two geometric series,

$$\begin{aligned}
f_s(x+q\mathbb{Z}) &= \sum_{w\in\mathbb{Z}} f_s(x+qw) \\
&= \sum_{w\geq 0} \exp\bigl(-\tfrac{2}{s}\,|x+qw|\bigr) + \sum_{w<0} \exp\bigl(-\tfrac{2}{s}\,|x+qw|\bigr) \\
&= \sum_{w\geq 0} \exp\bigl(-\tfrac{2}{s}\,|x+qw|\bigr) + \sum_{w\geq 0} \exp\bigl(-\tfrac{2}{s}\,|x-q(w+1)|\bigr) \\
&= \sum_{w\geq 0} \exp\bigl(-\tfrac{2}{s}(x+qw)\bigr) + \sum_{w\geq 0} \exp\bigl(-\tfrac{2}{s}(q-x+qw)\bigr) \\
&= \frac{e^{-2x/s}+e^{-2(q-x)/s}}{1-e^{-2q/s}}\ .
\end{aligned}$$

Substituting this term in the summation, we obtain

$$\begin{aligned}
f_s(\mathcal{L}_q) &= \sum_{x=0}^{q-1} \Bigl(\frac{e^{-2x/s}+e^{-2(q-x)/s}}{1-e^{-2q/s}}\Bigr)^2 \\
&= \frac{1}{(1-e^{-2q/s})^2} \cdot \sum_{x=0}^{q-1} \bigl(e^{-4x/s}+e^{-4(q-x)/s}+2e^{-2q/s}\bigr) \\
&= \frac{1}{(1-e^{-2q/s})^2} \cdot \Bigl(2q\cdot e^{-2q/s} + \frac{e^{4/s}(1-e^{-4q/s})}{e^{4/s}-1} - \frac{1-e^{-4q/s}}{1-e^{4/s}}\Bigr) \\
&= \frac{1}{(1-e^{-2q/s})^2} \cdot \Bigl(2q\cdot e^{-2q/s} + (1-e^{-4q/s})\cdot \frac{e^{4/s}+1}{e^{4/s}-1}\Bigr) \\
&= \frac{1}{(1-e^{-2q/s})^2} \cdot \Bigl(2q\cdot e^{-2q/s} + (1-e^{-4q/s})\cdot \coth(2/s)\Bigr) \\
&= \frac{2q\cdot e^{2q/s}}{(1-e^{2q/s})^2} + \frac{e^{4q/s}-1}{(1-e^{2q/s})^2}\cdot\coth(2/s) \\
&= \frac{2q\cdot e^{2q/s}}{(e^{2q/s}-1)^2} + \coth(q/s)\cdot\coth(2/s)\ .
\end{aligned}$$

□

5.4.2 Worst-Case Decoding

Now we address list-decoding in the ℓ_1 norm, under worst-case error of bounded norm, by specializing the material of Section 5.2.1 to $p = 1$ and using our bound on $f_s(\mathcal{L}_q)$ from Lemma 5.23. We consider decoding distance $d = \delta n$, where n is the code length and δ is the relative decoding distance. Then by Equations (5.9) and (5.10) and Lemma 5.23, we can

list-decode for any R^* less than

$$R^{*,(1)}_{\mathrm{wc},q}(\delta) = \sup_{s>0} B^{(1)}_{q,\delta}(s)^2 > \sup_{s>0} \exp(-4\delta/s) \cdot \tanh(2/s) \cdot E(q/s) \; . \tag{5.21}$$

Corollary 5.25 below is obtained by maximizing the "main term" $\exp(-4\delta/s) \cdot \tanh(2/s)$ of the right-hand side of (5.21). By calculus, this is done by taking $s = 4/\ln(D(\delta)) > 0$, where

$$D(\delta) := \sqrt{1 + \frac{1}{\delta^2}} + \frac{1}{\delta} > 1 \; .$$

Substituting, this means we can list-decode for any R^* less than

$$\widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta) := \frac{\tanh(\ln \sqrt{D(\delta)})}{D(\delta)^\delta} \cdot E(q \ln(D(\delta))/4) = \frac{D(\delta) - 1}{D(\delta) + 1} \cdot \frac{E(q \ln(D(\delta))/4)}{D(\delta)^\delta} \; . \tag{5.22}$$

We consider this quantity's asymptotic behavior for large and small δ:

- As δ grows, $D(\delta) = 1 + 1/\delta + O(1/\delta^2)$ and $D(\delta)^\delta$ approaches e, hence $\widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta)$ approaches $1/(2e\delta)$ as q/δ also grows. This is consistent with Remark 5.12.
- As δ approaches zero, $D(\delta)$ approaches $2/\delta$ and $D(\delta)^\delta$ approaches 1, hence $\widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta)$ approaches 1 as q/δ also grows.

Alternatively, we can get a simpler but cruder bound by replacing $\tanh(2/s)$ in Equation (5.21) with its upper bound of $2/s$. Then the resulting "main term" of $2\exp(-4\delta/s)/s$ is maximized at $s = 4\delta$; substituting, this means we can list decode for any R^* less than

$$e^{-1} \cdot \tanh(1/(2\delta)) \cdot E(q/(4\delta)) \; .$$

This bound approaches $1/(2e\delta)$ as δ and q/δ grow, which matches the behavior of $\widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta)$ as described above. However, as δ approaches zero (and q/δ grows), the above bound merely approaches $1/e$, which is much worse than the limit of 1 for $\widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta)$.

Corollary 5.25. *For any $\delta > 0$ and prime q, the GS algorithm using weight vector f_s for $s = 4/\ln(D(\delta))$ list-decodes, up to ℓ_1 distance δn in time* $\mathrm{poly}(n, q, 1/(\sqrt{\widetilde{R}^{*,(1)}_{wc,q}(\delta)} - \sqrt{R^*}))$, *any GRS code with adjusted rate $R^* < \widetilde{R}^{*,(1)}_{wc,q}(\delta)$ (see Equation (5.22)).*

Proof. By hypothesis and Lemma 5.23 and Equation (5.9),

$$R^* < \widetilde{R}^{*,(1)}_{\mathrm{wc},q}(\delta) = \frac{\tanh(\ln \sqrt{D(\delta)})}{D(\delta)^\delta} \cdot E(q/s) < \frac{\exp(-4\delta/s)}{f_s(\mathcal{L}_q)} = B^{(1)}_{q,\delta}(s)^2 \; .$$

The claim then follows directly by Theorem 5.10. □

Comparison to [RS94]. To our knowledge, the only prior result for decoding Reed–Solomon codes in the ℓ_1 norm is due to Roth and Siegel [RS94], who gave a *unique* decoding algorithm for *discrete* error up to half the minimum distance using Euclid's algorithm for polynomials.[7]

Proposition 5.26 (adapted from [RS94, Section 5])**.** *For any prime-field GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ and any adjusted rate $R^* \in (0,1)$, there is a* $\mathrm{poly}(n,q)$ *time algorithm that (uniquely) decodes $\mathcal{C}$ under* discrete *error of relative ℓ_1 distance $\delta < 1 - R^* - 1/n < 1$.*

By contrast, our Corollary 5.25 works for continuous error for *any* (arbitrarily large) $\delta > 0$, by taking sufficiently small $R^* > 0$. For large n, the above distance bound approaches $1 - R^*$. In this regime, our rate-distance trade-off from Corollary 5.25 surpasses that of Proposition 5.26 for all relative distances $\delta \gtrapprox 0.78988$, which corresponds to rates $R^* \lessapprox 0.21012$.

5.4.3 Average-Case Decoding

We now consider average-case decoding under a memoryless additive (continuous or discrete) Laplacian channel, by specializing the material of Section 5.2.2 to $p = 1$ and using our bound on $f_s(\mathcal{L}_q)$ from Lemma 5.23. Consider a Laplacian channel of parameter $r > 0$. Then by Equations (5.11) and (5.12), we can list-decode for any R^* less than

$$R^{*,(1)}_{\mathrm{ac},q}(r) = \sup_{s>0} A^{(1)}_{q,r}(s)^2 > \sup_{s>0} \frac{s^2 \cdot \tanh(2/s)}{(r+s)^2} \cdot E(q/s)\ , \tag{5.23}$$

where the inequality is by Lemma 5.23.

Corollary 5.27 below is obtained by nearly maximizing the right-hand side of (5.23), at least for moderate or large values of r. Specifically, we use the bound $\tanh(2/s) < 2/s$ to approximate the "main term" of (5.23) by $2s/(r+s)^2$. This is maximized at $s = r$, which makes the original main term equal to $\tanh(2/r)/4$. Note that $R^{*,(1)}_{\mathrm{ac},q}(r)$ does indeed approach this value as r and q/r grow, because $\tanh(2/r)$ approaches $2/r$, and $E(q/r)$ rapidly approaches 1 (see Equation (5.20)).

However, for small values of r, the expression in (5.23) is maximized for s significantly larger than r, to have value much larger than $\tanh(2/r)/4 < 1/4$. This maximization can be computed numerically, and indeed, $R^{*,(1)}_{\mathrm{ac},q}(r)$ approaches 1 as r approaches 0.

[7] The algorithm of [RS94] actually applies to a more general family of codes, called *(shortened) BCH codes*, over any prime-power field. By restricting to prime fields and using a standard reduction, the result from [RS94] also applies to all prime-field GRS codes. We also remark that the results in [RS94] are stated for the *Lee metric*, which is equivalent to the ℓ_1 norm in prime fields.

Corollary 5.27. *For any $r > 0$, $\alpha \in (0,1)$, and prime q, the GS algorithm using weight vector given by f_r list-decodes, in time* $\mathrm{poly}(n, q, 1/(\sqrt{R^{*,(1)}_{ac,q}} - \sqrt{R^*}))$*, any GRS code with adjusted rate*

$$R^* < \widetilde{R}^{*,(1)}_{ac,q}(r) := \frac{\tanh(2/r)}{4} \cdot E(q/r) \ ,$$

except with probability less than $\exp\left(-n \cdot \alpha^2 \cdot r \cdot \left(\sqrt{\widetilde{R}^{*,(1)}_{ac,q}(r)} - \sqrt{R^*}\right)^2\right)$.

Proof. By hypothesis, Lemmas 5.13 and 5.23 and Equation (5.11),

$$R^* < \widetilde{R}^{*,(1)}_{\mathrm{ac},q}(r) = \frac{\tanh(2/r)}{4} \cdot E(q/r) < \frac{\mu^2_{r,r}}{f_r(\mathcal{L}_q)} = A^{(1)}_{q,r}(r)^2 \ .$$

The claim then follows directly by Theorem 5.14, and (for the probability bound) the fact that $f_r(\mathcal{L}_q) > \coth(2/r) > r/2$ by Lemma 5.24. □

5.5 Future Directions

In this work, we present an efficient algorithm for list-decoding GRS codes from either worst-case errors or average-case errors in the ℓ_p (quasi)norm for $0 < p \leq 2$. Remarkably, the product of the relative distance and the adjusted rate for which we can decode worst-case errors approaches the relative radius of a unit-volume ℓ_p ball. This relationship arose naturally from the Guruswami-Sudan framework and our analysis, but we do not know what the deeper explanation is. Explaining this phenomenon in the geometric or coding theoretic context is left open.

As described in Section 5.3, instantiating our general list-decoding algorithm with $p = 2$ gives an efficient list-decoding algorithms for GRS codes in the ℓ_2 norm. Our algorithm performs better than the previously known one in [MP22] in that it works for arbitrarily large decoding distances and larger rates. For decoding distances less than $1/2$, however, the algorithm from [MP22] was shown to be an optimal instantiation of the GS algorithm. It remains an open question to determine an optimal choice of weights for the GS algorithm for decoding distances larger than $1/2$.

In Section 5.4, we instantiated our general list-decoding algorithm with $p = 1$ to obtain an efficient list-decoding algorithms for GRS codes in the ℓ_1 norm. To our knowledge, only a *unique*-decoding algorithm GRS codes in the ℓ_1 norm was known [RS94], so ours is the first *list*-decoding algorithm. Improving our algorithm is an open direction.

CHAPTER 6

Lattice List-Decoding Bounds

Based on joint work with Mahdi Cheraghchi and on joint work with Chris Peikert.

Given the structural similarity between codes and lattices as vector linear subspaces over a field, a lattice can be viewed as an (infinite) code in Euclidean space. For wireless communication, it is natural to represent transmitted messages by real-valued vectors. Lattices provide a convenient structure for encoding and decoding in this context, which makes *lattice list-decoding* a natural object of study. A lattice with larger "density", and whose minimum distance is at least one, contains more codewords within a finite bounded space, so such a lattice has a higher "rate" when used for error-correction; this allows more information to be sent for a fixed blocklength. Thus, a lattice ideal for use in error-correction would have high density and be efficiently list-decodable.

Lattice Density and Minkowski's Inequality. In order to quantify how good a lattice is for error-correction, we need analogous notions for rate and minimum distance. While the minimum distance of a lattice is well-defined as the length of the shortest non-zero vector (see Definition 2.15), defining the rate of a lattice is not so straightforward. The rate of a code measures the density of the codewords relative to the finite ambient space. Lattices, on the other hand, are defined over *infinite* and *dense* Euclidean space, so the closest analog of rate is given by its determinant (see Definition 2.13). Since the determinant of a lattice is dependent on the minimum distance, a natural way to measure the density of a lattice is to compare their ratio (see [CS88] for background).

In particular, we use the *normalized minimum distance* of a lattice $\mathcal{L} \subset \mathbb{R}^n$, which is given by $\gamma(\mathcal{L}) := (\lambda_1(\mathcal{L})/\det(\mathcal{L})^{1/n})^2$. A related quantity is the *Hermite constant*, which quantifies the largest normalized minimum distance for a given dimension; this is defined as $\gamma_n := \sup_{\mathcal{L} \subset \mathbb{R}^n} \gamma(\mathcal{L})$. The following result bounds this quantity for any lattice.

Theorem 6.1 (adapted from [CS88, Chapter 1, Equation 28])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$,*

$$\frac{\lambda_1(\mathcal{L})}{\det(\mathcal{L})^{1/n}} \leq \sqrt{n} \ .$$

Equivalently, the Hermite constant for dimension n is bounded by $\gamma_n \leq n$.

We refer to this inequality hereafter as *Minkowski's bound*.

Prior Work. The study of efficient lattice list-decoding was initiated by Grigorescu and Peikert in [GP17], who gave an efficient algorithm for list-decoding a family of lattices, called Barnes-Wall lattices. Mook and Peikert in [MP22] later showed how to efficiently list-decode Barnes-Sloane lattices, which achieve Minkowski's bound within a factor of $O(\sqrt{\log(n)})$ and have better density than Barnes-Wall lattices. Soon after, Bennett and Peikert [BP22] showed how to efficiently list-decode lattices obtained from Reed-Solomon codes, which achieve Minkowski's bound within the same $O(\sqrt{\log(n)})$ factor. Most recently, Kirshanova and Malygina in [KM23] constructed efficiently list-decodable lattices by applying Construction D to a tower of algebraic-geometry codes, known as the Garcia-Stichtenoth tower. These lattices achieve Minkowski's bound within a smaller factor of $O(\log(n)^{\varepsilon+o(1)})$ for some $\varepsilon > 0$.

All of these works list-decode lattices that can be obtained from codes using Construction D; in fact, the recursive nature of this construction forms the basis of these efficient iterative algorithms. None of these lattices, however, achieve the maximum density given by Minkowski's bound.

Our Contributions. In this work, we probabilistically construct dense Construction D lattices and prove capacity bounds for lattice list-decoding over general norms using a probabilistic argument. In Section 6.1, we determine a sufficient condition for a Construction D lattice to achieve Minkowski's inequality up to a *constant* factor. In particular, it is sufficient that each code with blocklength n and minimum distance d in the input tower of linear codes have codimension $\Theta(d \log(n/d))$. Random linear codes satisfy this codimension bound with high probability. Using this observation, we show that instantiating Construction D with a tower of random linear codes produces a lattice that achieves Minkowski's inequality. In Section 6.2, we determine probabilistic capacity bounds for list-decoding lattices over the Euclidean norm and other norms.

Throughout this work, all distance are measured in the Euclidean (ℓ_2) norm unless specified otherwise. All lattices are assumed to be full-rank.

6.1 Dense Lattices from Random Linear Codes

Results in this section are based on joint work with Mahdi Cheraghchi.

First we describe some properties of lattices obtained via Construction D. If the construction is instantiated with binary linear codes, the minimum distance and determinant can be expressed explicitly as follows.

Lemma 6.2 (adapted from [BS83, Theorem 1] and [CS88, Chapter 8, Theorem 13])**.** *For any positive integers n and ℓ, and tower of codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$ where $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_2^n$ and each $\mathcal{C}_i$ is a $[n, k_i, d_i]_2$-code with distance $d_i \geq 4^i$, the Construction D lattice $\mathcal{L} = \mathcal{L}_D(\{\mathcal{C}_i\}_{i=0}^{\ell})$ has minimum distance $\lambda_1(\mathcal{L}) = 2^\ell$ and determinant*

$$\det(\mathcal{L}) = 2^{\sum_{i=1}^{\ell}(n-k_i)} .$$

We generalize this property to codes over prime fields. Recall that Construction D requires the field characteristic to be prime. We remark that [MP22] mentions this generalization but does not explicitly prove it, so we include a proof here for completion.

Lemma 6.3. *For any positive integers n and ℓ, prime p, and tower of codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$ where $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_p^n$ and each $\mathcal{C}_i$ is a $[n, k_i, d_i]_p$-code with distance $d_i \geq p^{2i}$, the Construction D lattice $\mathcal{L} = \mathcal{L}_D(\{\mathcal{C}_i\}_{i=0}^{\ell})$ has minimum distance $\lambda_1(\mathcal{L}) = p^\ell$ and determinant*

$$\det(\mathcal{L}) = p^{\sum_{i=1}^{\ell}(n-k_i)} = p^{\ell n - \sum_{i=1}^{\ell} k_i} .$$

Proof. By Definition 2.18, the Construction D lattice $\mathcal{L}$ is generated by a basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$ which is obtained by scaling the rows of the generator matrix $\mathbf{G}$ for the given tower of codes. Recall that $\mathbf{G}$ has block matrices $\mathbf{G}_0, \ldots, \mathbf{G}_\ell$ such that the rows of $\mathbf{G}_i, \ldots, \mathbf{G}_\ell$ generate code $\mathcal{C}_i$ for every $i \in [\ell]$ and $\mathbf{G}_0$ is the identity matrix. Then for each i, the block matrix consisting of $\mathbf{G}_i, \ldots, \mathbf{G}_\ell$ has k_i rows. The basis matrix $\mathbf{B}$ is obtained by scaling the block matrices of $\mathbf{G}$ so that the corresponding block matrices of $\mathbf{B}$ are $p^\ell \mathbf{G}_0, p^{\ell-1}\mathbf{G}_1, \ldots, p^0 \mathbf{G}_\ell$. By construction, $\mathbf{B}$ is also an upper-triangular matrix. Since the diagonal entries of $\mathbf{G}$ are all 1, the diagonal entries of $\mathbf{B}$ are all multiples of p. By the fact that the determinant of an upper-triangular matrix is the product of its diagonal entries,

$$\det(\mathcal{L}) = \det(\mathbf{B}) = p^{n-k_\ell} \cdot p^{n-k_{\ell-1}} \cdot \ldots \cdot p^{n-k_1} = p^{\sum_{i=1}^{\ell}(n-k_i)} .$$

Now we prove the minimum distance claim. First we show the upper bound. Since the basis matrix $\mathbf{B}$ is upper-triangular, has non-zero determinant, and has integer entries, $\mathbb{Z}^n$ lies

in the integer span of $\mathbf{B}$. This implies that the scaled integer lattice $p^\ell\mathbb{Z}^n$ is a sublattice of $\mathcal{L}$; this has minimum distance p^ℓ. Since adding points to $p^\ell\mathbb{Z}^n$ cannot increase the minimum distance, and $\mathcal{L}$ is a superset of $p^\ell\mathbb{Z}^n$, we have $\lambda(\mathcal{L}) \leq p^\ell$.

Now let $\mathbf{v} = (v_1, \ldots, v_n) \in \mathcal{L}$ be an arbitrary non-zero lattice vector. Then $\mathbf{v} = \mathbf{uB}$ for some non-zero coefficient vector $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{Z}^n$. Let $j \in [n]$ be the smallest index such that the corresponding coefficient u_j is non-zero. This coefficient scales the j-th row of $\mathbf{B}$ and is contained in the block matrix $\mathbf{G}_i$ for some $i \in [\ell]$. Then, by definition, every coordinate of $\mathbf{u}$ with index smaller than j must be zero, so the first $j-1$ coordinates of $\mathbf{v}$ are $v_1 = \ldots = v_{j-1} = 0$. By definition of $\mathbf{B}$, every entry of the block matrices $\mathbf{B}_i, \ldots, \mathbf{B}_\ell$ is a multiple of $p^{\ell-i}$, so the the non-zero coordinates of $\mathbf{v}$ are all multiples of $p^{\ell-i}$. In particular, we can express $\mathbf{v}$ in the form $\mathbf{v} = (0, \ldots, 0, p^{\ell-i} \cdot w_j, \ldots, p^\ell \cdot w_n)$ for some $w_j, \ldots, w_n \in \mathbb{Z}$, at least one of which is non-zero. Consider the normalization of $\mathbf{v}$ by $p^{\ell-i}$, given by $\mathbf{v}' := p^{i-\ell}\mathbf{v}$, and define $\mathbf{c}' := \mathbf{v}' \bmod p \in \mathbb{F}_p^n$. Suppose that the last row of $\mathbf{B}_i$ is the j'-th row of $\mathbf{B}$. Then using this notation,

$$\begin{aligned}
\mathbf{c}' &\equiv p^{i-\ell} \cdot (0, \ldots, 0, p^{\ell-i} \cdot w_j, \ldots, p^{\ell-i} \cdot w_{j'}, p^{\ell-i+1} \cdot w_{j'+1}, \ldots, p^\ell \cdot w_n) \\
&\equiv (0, \ldots, 0, w_j, \ldots, w_{j'}, p \cdot w_{j'+1}, \ldots, p^i \cdot w_n) \\
&\equiv (0, \ldots, 0, w_j, \ldots, w_{j'}, 0, \ldots, 0) \bmod p \,.
\end{aligned}$$

By Definition 2.18, this $\mathbf{c}'$ must be a non-zero codeword of $\mathcal{C}_i$, which by hypothesis has minimum distance d_i. Then $\mathbf{c}'$, and consequently $\mathbf{v}'$, must have at least d_i non-zero coordinates. Then the lattice vector $\mathbf{v}$ has length

$$\|\mathbf{v}\|^2 = \|p^{\ell-i}\mathbf{v}'\|^2 = p^{2(\ell-i)} \cdot \|\mathbf{v}'\|^2 \geq p^{2(\ell-i)} \cdot d_i \,.$$

By hypothesis, $d_i \geq p^{2i}$ for all $0 \leq i \leq \ell$, and minimizing both sides of this inequality,

$$\lambda_1(\mathcal{L})^2 = \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|^2 \geq \min_i p^{2(\ell-i)} \cdot d_i \geq p^{2\ell} \,.$$

This gives the lower bound $\lambda_1(\mathcal{L}) \geq p^\ell$. Together with the upper bound, we obtain $\lambda_1(\mathcal{L}) = p^\ell$ as claimed. □

6.1.1 A Sufficient Condition for Achieving Minkowski's Bound

In this section, we determine a sufficient conditions for a Construction D lattice from binary linear codes to achieve Minkowski's inequality. We also show that random linear binary codes satisfy this condition with high probability.

Remark 6.4. For Construction D lattices from a tower of binary codes in $\mathbb{F}_2^n$, a simple calculation reveals that the density ratio on the left-hand side of the inequality in Theorem 6.1 is equal to the right-hand side $\sqrt{n}$ if the sum of all the code dimensions is $n\log_2(\sqrt{n})$. This follows immediately from Lemma 6.2. The following statements give an asymptotic condition for *each* code in the tower; it is more clear how to satisfy this condition for individual codes in the tower, rather than satisfy the dimension sum.

Theorem 6.5 (Theorem 6.6, informal)**.** *For any positive integers n and ℓ satisfying $\ell = \log(\Theta(n)) < \log_4(n/2)$, and tower of binary codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$ where $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_2^n$ and each $\mathcal{C}_i$ is a $[n, k_i, d_i]_2$-code with minimum Hamming distance $d_i = 4^i$ and codimension $n - k_i = \Theta(d_i \log(n/d_i))$, the Construction D lattice $\mathcal{L} = \mathcal{L}_D(\{\mathcal{C}_i\}_{i=0}^{\ell})$ satisfies*

$$\frac{\lambda_1(\mathcal{L})}{\det(\mathcal{L})^{1/n}} = \Theta(\sqrt{n}) \ .$$

This result is analogous to [MP22, Theorem 5.5].

Theorem 6.6. *For any positive integer n, $\alpha \in (0, \frac{1}{2})$, tower length $\ell = \lfloor \log_4(\alpha n) \rfloor$, real $\gamma > 0$, and $\kappa \in (0, 1)$ satisfying*

$$\log_2(\kappa) \geq \log_2(\alpha)\Big(\frac{4\alpha\gamma}{3} + \frac{1}{2}\Big) - \frac{8\alpha\gamma}{9} + \frac{4\gamma \cdot \log_2(n)}{3 \cdot n} + \frac{8\gamma}{9 \cdot n} \ ,$$

the tower of binary codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$, where $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_2^n$ and each $\mathcal{C}_i$ is a $[n, k_i, d_i]_2$-code with minimum Hamming distance $d_i = 4^i$ and codimension $n - k_i = \gamma_i \cdot d_i \log(n/d_i)$ for some constant $\gamma_i \leq \gamma$, produces a Construction D lattice $\mathcal{L} = \mathcal{L}_D(\{\mathcal{C}_i\}_{i=0}^{\ell})$ that achieves

$$\frac{\lambda_1(\mathcal{L})}{\det(\mathcal{L})^{1/n}} \leq \kappa\sqrt{n} \ .$$

Note that this approximation factor κ exists for any blocklength n by taking sufficiently small values of α, as long as the tower length ℓ is still a positive integer.

Proof. By Lemma 6.2 and hypothesis on ℓ, we have $\lambda_1(\mathcal{L}) = 2^\ell = \sqrt{\alpha n}$ and $\det(\mathcal{L}) = 2^{\sum_i n - k_i}$. So to show that their ratio satisfies the claimed inequality, it is enough to lower bound the sum of the code codimensions by $n \cdot \log_2(\sqrt{\alpha}/\kappa)$. By hypothesis on the codimension and distance and $\gamma_i \leq \gamma$, and some logarithmic manipulations,

$$\sum_{i=1}^{\ell} n - k_i = \sum_{i=1}^{\ell} \gamma_i \cdot 4^i \log_2(n/4^i) \leq \gamma \log_2(n) \cdot \sum_{i=1}^{\ell} 4^i - 2\gamma \cdot \sum_{i=1}^{\ell} i \cdot 4^i \ .$$

Using identities for the finite geometric series and the second series above [1] and some logarithmic manipulations

$$\sum_{i=1}^{\ell} n - k_i \leq \frac{4\gamma}{3}(\alpha n - 1) \cdot \log_2(n) - \frac{8\gamma}{9}(\alpha n(3\log_4(\alpha n) - 1) + 1)$$
$$= n\Big(\frac{8\alpha\gamma}{9} - \frac{4\alpha\gamma\log_2(\alpha)}{3}\Big) - \frac{4\gamma \cdot \log_2(n)}{3} - \frac{8\gamma}{9} .$$

Dividing both sides by n and applying the hypothesis on $\log_2(\kappa)$, we obtain

$$\frac{1}{n} \cdot \sum_{i=1}^{\ell} n - k_i \geq \log_2(\sqrt{\alpha}/\kappa) ,$$

which proves the claim. □

Using Lemma 6.3 and a proof nearly identical to that of Theorem 6.6, we obtain a more general result for codes over larger prime fields.

Theorem 6.7. *For any positive integer n, $\alpha \in (0, \frac{1}{2})$, tower length $\ell = \lfloor \log_p(\sqrt{\alpha n}) \rfloor$, real $\gamma > 0$, and $\kappa \in (0, 1)$ satisfying*

$$\log_2(\kappa) \geq \log_p(\alpha)\Big(\frac{p^2\alpha\gamma}{p^2-1} + \frac{1}{2}\Big) - \frac{2p^2\alpha\gamma}{(p^2-1)^2} + \frac{p^2\gamma \cdot \log_p(n)}{(p^2-1)\cdot n} + \frac{2p^2\gamma}{(p^2-1)^2 \cdot n} ,$$

the tower of codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$, where $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_p^n$ and each $\mathcal{C}_i$ is a $[n, k_i, d_i]_p$-code with minimum Hamming distance $d_i = p^{2i}$ and codimension $n - k_i = \gamma_i \cdot d_i \log(n/d_i)$ for some constant $\gamma_i \leq \gamma$, produces a Construction D lattice $\mathcal{L} = \mathcal{L}_D(\{\mathcal{C}_i\}_{i=0}^{\ell})$ that achieves

$$\frac{\lambda_1(\mathcal{L})}{\det(\mathcal{L})^{1/n}} \leq \kappa\sqrt{n} .$$

6.1.2 Constructing Codes that Satisfy the Condition

In this section, we give a Varshamov-like probabilistic construction for generating a tower of linear codes that satisfy the conditions in Theorem 6.7. For any blocklength n and prime field size p, we construct a tower of random linear codes $\mathbb{F}_p^n = C_0 \supseteq C_1 \supseteq \cdots \supseteq C_\ell$ by sampling row vectors from $\mathbb{F}_p^n$ uniformly at random and iteratively concatenating them to form parity-check matrices. The following construction describes the formal procedure.

Let n be a positive integer, p a prime, and ℓ a positive integer such that $\ell < n/\log_p(n)$.

[1] In particular, we use the well-known identity $\sum_{i=1}^{\ell} a^i = (a^{\ell+1} - a)/(a-1)$ and the identity $\sum_{i=1}^{\ell} i \cdot a^i = a(a^\ell(\ell(a-1)-1)+1)/(a-1)^2$. Both can be shown using a straightforward induction proof.

Construction 6.8. Set $r_0 := 1$. Choose rank parameters $r_1, \ldots, r_\ell \leftarrow [n]$ uniformly at random such that $r_1 \leq r_2 \leq \ldots \leq r_\ell$. Define $\mathbf{H}_0 := \mathbf{0} \in \mathbb{F}_p^{1 \times n}$. For all $1 \leq i \leq \ell$, construct the matrix $\mathbf{H}_i \in \mathbb{F}_p^{r_i \times n}$ as follows: Set $s_i := r_1 + \ldots + r_{i-1}$.If $r_i > s_i$, sample $r_i - s_i$ new row vectors $\mathbf{h}_1, \ldots, \mathbf{h}_{r_i - s_i}$ independently ad uniformly at random from $\mathbb{F}_p^n$. Concatenate these row vectors to the bottom of $\mathbf{H}_{i-1}$ to form the matrix $\mathbf{H}_i$. These matrices have the form

$$\mathbf{H}_1 = \begin{bmatrix} -\mathbf{h}_1^{(1)} - \\ \vdots \\ -\mathbf{h}_{r_1}^{(1)} - \end{bmatrix}, \ \mathbf{H}_2 = \begin{bmatrix} \mathbf{H}_1 \\ -\mathbf{h}_1^{(2)} - \\ \vdots \\ -\mathbf{h}_{r_2 - r_1}^{(2)} - \end{bmatrix}, \ldots, \ \mathbf{H}_\ell = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_{\ell-1} \\ -\mathbf{h}_1^{(\ell)} - \\ \vdots \\ -\mathbf{h}_{r_\ell - s_\ell}^{(\ell)} - \end{bmatrix}.$$

Figure 6.1: The parity-check matrices generated by Construction 6.8.

For every $0 \leq i \leq \ell$, define $\mathcal{C}_i := \ker(\mathbf{H}_i) \subseteq \mathbb{F}_p^n$ to be the code whose parity-check matrix is $\mathbf{H}_i$. Output the tower of codes $\{\mathcal{C}_i\}_{i=0}^{\ell}$.

This probabilistic algorithm runs in time polynomial in n.

By construction, the subspaces corresponding to the iteratively-constructed matrices $\mathbf{H}_0, \ldots, \mathbf{H}_\ell$ are nested, so that $\ker(H_i) \subseteq \ker(H_{i-1})$ for all $1 \leq i \leq \ell$. Hence, $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_p^n$ produced by Construction 6.8 is a nested tower of linear p-ary codes, where each $\mathcal{C}_i$ is a $[n, k_i, d_i]_p$-code with codimension $r_i = n - k_i$. Since the rows of these matrices are sampled independently and uniformly at random, and additional random rows are concatenated in each iteration, every matrix $\mathbf{H}_i$ is a uniformly random matrix over its domain $\mathbb{F}_p^{r_i \times n}$.

We require the tower length to be at most $\ell < n/\log_p(n)$ to ensure that the generated codes do not trivialize before all ℓ codes are constructed. More precisely, if the iterative construction of the matrices $\mathbf{H}_i$ is repeated for more than $n/\log_p(n)$ iterations, the matrix $\mathbf{H}_i$ will contain n linearly independent rows and form the trivial code $\mathcal{C}_i = \{\mathbf{0}\}$.

For our application in Theorem 6.7, we want each code $\mathcal{C}_i$ to have distance $d_i = p^{2i}$ and codimension $r_i \leq \gamma \cdot d_i \log_p(n/d_i)$ for some fixed constant $\gamma > 0$. We will show that the tower of codes produced by Construction 6.8 achieves the lattice density in Theorem 6.7 for an appropriate $\kappa \in (0, 1)$ with high probability. To do this, we will use the following rate-distance trade-off bound for codes generated by a uniformly random generator matrix.

Lemma 6.9 (Gilbert-Varshamov Bound, adapted from [GRS19])**.** *For any positive integer n, prime p, $\delta \in (0, 1-1/p)$, and $\varepsilon \in (0, 1-H_p(\delta))$ defining $k := n(1-H_p(\delta)-\varepsilon)$, the code $\mathcal{C} \subseteq \mathbb{F}_p^n$ generated by a uniformly random matrix $\mathbf{G} \in \mathbb{F}_p^{k\times n}$ has rate at least $k/n \geq 1 - H_p(\delta) - \varepsilon$ and relative minimum distance $\delta = d/n$ with probability at least $1 - p^{-\varepsilon n}$.*

Recall that $H_p(\cdot)$ here is the p-ary entropy function defined in Definition 2.2. This result can be reformulated in terms of parity-check matrices to obtain a bound on the codimension of the code as follows.

Corollary 6.10. *For any positive integer n, prime p, $\delta \in (0, 1-1/p)$, and $\varepsilon \in (0, 1-H_p(\delta))$ defining $k := n(1 - H_p(\delta) - \varepsilon)$, the code $\mathcal{C} \subseteq \mathbb{F}_p^n$ whose parity-check matrix is a uniformly random matrix $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$ has codimension at most $n - k \leq n(H_p(\delta) - \varepsilon)$ and minimum distance $d = \delta n$ with probability at least $1 - p^{-\varepsilon n}$.*

Now we apply this result to the codes generated by Construction 6.8.

Theorem 6.11. *For any positive integer n, prime p, $\alpha \in (0, \frac{1}{2})$, and positive integer ℓ satisfying $\ell = \lfloor \log_p(\sqrt{\alpha n})\rfloor < n/\log_p(n)$, let $\mathcal{C}_\ell \subseteq \ldots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \mathbb{F}_p^n$ be the tower of codes generated by Construction 6.8, where each $\mathcal{C}_i$ is a $[n, k_i, d_i]_p$-code. Then for any $\gamma > 0$ defining $\widetilde{\varepsilon} := \min_{i\in[\ell]} H_p(d_i/n) + \gamma \cdot (d_i/n) \cdot \log_p(d_i/n)$ and $\varepsilon_1, \ldots, \varepsilon_\ell$ such that $\varepsilon_i \in (\widetilde{\varepsilon}, 1 - H_p(d_i/n))$, every code $\mathcal{C}_i$ in the tower satisfies the codimension bound $n - k_i \leq \gamma \cdot d_i \log_p(n/d_i)$ with probability at least $1 - \ell/p^{\widetilde{\varepsilon}n}$.*

Note that the input d_i/n to the entropy function is in the required domain $(0, 1)$, so $\widetilde{\varepsilon}$ is well-defined. For large values of p, the function H_p is strictly increasing until shortly before 1; this factor dominates the value of $\widetilde{\varepsilon}$ for large values of n and p. The probability in the statement above approaches 1 as n grows, because the height of the tower ℓ is logarithmic in (a fixed fraction of) n, while $\widetilde{\varepsilon}n$ grows with n, so the denominator in the probability expression grows more rapidly than the numerator.

Proof. It is enough to show that at least one code in the tower does not satisfy its codimension bound with probability at most $\ell/p^{\widetilde{\varepsilon}n}$. By the union bound, the probability that some $\mathcal{C}_i$ does not satisfy its codimension bound is bounded above by

$$\Pr\left[\exists i \in [\ell] : n - k_i > \gamma \cdot d_i \log_p(n/d_i)\right] \leq \sum_{i=0}^{\ell} \Pr\left[n - k_i > \gamma \cdot d_i \log_p(n/d_i)\right] .$$

Fix an arbitrary $i \in [\ell]$. By Corollary 6.10 with ε_i and the definition of $\widetilde{\varepsilon}$, the codimension of code $\mathcal{C}_i$ satisfies

$$n - k_i \leq n(H_p(d_i/n) - \varepsilon_i) \leq n(H_p(d_i/n) - \widetilde{\varepsilon}) \leq \gamma \cdot d_i \log_p(n/d_i)$$

with probability at least $1 - p^{-\varepsilon_i n} \geq 1 - p^{-\widetilde{\varepsilon} n}$. Taking the complement and summing over all codes in the tower, we obtain

$$\Pr\left[\exists i \in [\ell] : n - k_i > \gamma \cdot d_i \log_p(n/d_i)\right] \leq \ell \cdot p^{-\widetilde{\varepsilon} n} .$$

□

As a consequence of Theorems 6.7 and 6.11, we obtain that dense Construction D lattices exist and can be constructed from a tower of p-ary linear codes using Construction 6.8.

6.2 Lattice List-decoding Capacity

Results in this section are based on joint work with Chris Peikert.

The list-decoding capacity of a family of codes gives the optimal trade-off between the rate of a code and the relative distance up to which it can be list-decoded. This trade-off is known for the Hamming metric (see [GRS19, Theorem 7.4.1] for example). We derive a similar optimal rate-distance trade-off for the Lee metric for sufficiently small list-decoding radii. We also present a lattice list-decoding capacity bounds for decoding over general norms.

6.2.1 List-decoding Capacity for Lee Metric

The *Lee metric* can be seen as a generalization of the Hamming metric, which instead of indicating whether two symbols are the same, it quantifies *how many* symbols apart they are. Formally, for any positive integer p, a real element $x \in [0, p)$ has *Lee weight*

$$\overline{x} := \begin{cases} x, & \text{if } 0 \leq x < \frac{p}{2} \\ x - p, & \text{if } \frac{p}{2} \leq x < p \end{cases} .$$

The *Lee distance* between any two elements $x, y \in \mathbb{R}_p$ is given by $\overline{x - y}$. This extends naturally to vectors over $\mathbb{R}_p$: For any $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}_p^n$, its Lee weight is defined as the sum of the Lee weights of its coordinates, i.e., $\overline{\mathbf{x}} := \sum_{i=1}^{n} \overline{x_i}$. Similarly, the Lee distance between any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}_p^n$ of the same length is given by $\overline{\mathbf{x} - \mathbf{y}}$.

Theorem 6.12 (List-decoding Capacity for Lee Metric)**.** *Let n and L be positive integers, p a prime, and $r \in (0, p/2)$. If a code $\mathcal{C} \subseteq \mathbb{F}_p^n$ is (r, L)-list-decodable over the Lee metric and has rate at least $1 - \log_p(V_r^{(n)})/n$, then the list size L must be exponential in n.*

For radius $r < p/2$, the volume of the n-dimensional r-radius Lee ball is known to be

$$V_r^{(n)} := \sum_{i=0}^{n} 2^i \binom{n}{i} \binom{r}{i} .$$

Proof. We use the probabilistic method to show that if $\mathcal{C}$ has large enough rate, there exists a vector $\mathbf{y} \in \mathbb{F}_p^n$ such that the number of codewords within Lee distance r of $\mathbf{y}$ is exponentially large. Let $\mathcal{B}(\mathbf{0}, r)$ denote the Lee ball of radius r centered at the origin. Define the indicator function $g(\mathbf{x}) = \mathbb{1}[\mathbf{x} \in \mathcal{B}(\mathbf{0}, r)] \in \{0, 1\}$, which is 1 if and only if $\mathbf{x}$ lies in $\mathcal{B}(\mathbf{0}, r)$, and also define the function $f : \mathbb{F}_p^n \to \mathbb{Z}$ by $f(\mathbf{z}) := \sum_{\mathbf{c} \in \mathcal{C}} g(\mathbf{z} - \mathbf{c})$. This function f counts the number of codewords within Lee distance of r from $\mathbf{y}$, i.e.

$$f(\mathbf{y}) = |\{\mathbf{c} \in \mathcal{C} : \|\mathbf{y} - \mathbf{c}\|_L \leq r\}| .$$

Let $\mathbf{y} \in \mathbb{F}_p^n$ be a uniformly random vector. We compute the expected number of close codewords to $\mathbf{y}$. By definition of f and linearity of expectation, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{y}}[f(\mathbf{y})] &= \sum_{\mathbf{c} \in \mathcal{C}} \Pr_{\mathbf{y}}[\mathbf{y} - \mathbf{c} \in \mathcal{B}(\mathbf{0}, r)] \\
&= \sum_{\mathbf{c} \in \mathcal{C}} \Pr_{\mathbf{y}}[\mathbf{y} \in \mathcal{B}(\mathbf{c}, r)] \\
&= \frac{|\mathcal{C}| \cdot \mathrm{vol}(\mathcal{B}(\mathbf{0}, r))}{p^n} \\
&= \frac{p^k \cdot V_r^{(n)}}{p^n} \\
&= \left(\frac{{V_r^{(n)}}^{1/n}}{p^{1-R}} \right)^n .
\end{aligned}$$

By assumption on the rate R, we have ${V_r^{(n)}}^{1/n} > p^{1-R}$, and so this expected value is exponential in n. Therefore, there must exist a vector $\mathbf{y}$ such that the list size $|\mathcal{C} \cap \mathcal{B}(\mathbf{y}, r)|$ is exponentially large. □

6.2.2 Lattice List-decoding Capacity

We generalize the approach used in Theorem 6.12 to lattices for large list-decoding radii. Let $\mathcal{B}_r^n$ denote the n-dimensional ball of radius r centered at the origin.

Theorem 6.13 (Lattice List-decoding Capacity for General Norms)**.** *For any positive integers n and L and real $r > 0$, if a lattice $\mathcal{L} \subset \mathbb{R}^n$ is (r, L)-list-decodable and has determinant*

at most $\det(\mathcal{L}) < \mathrm{vol}(\mathcal{B}_r^n)$*, then the list size* L *must be exponential in* n*.*

Proof. We use the probabilistic method to show that there exists a vector $\mathbf{y} \in \mathbb{R}^n$ such that the number of lattice points within the r-radius ball centered at $\mathbf{y}$ is exponentially large. Let $\mathcal{B}(\mathbf{0}, r)$ be the ball of radius r centered at the origin. Define the indicator function $g(\mathbf{x}) = \mathbb{1}[\mathbf{x} \in \mathcal{B}(\mathbf{0}, r)] \in \{0, 1\}$, which is 1 if and only if $\mathbf{x}$ lies in $\mathcal{B}(\mathbf{0}, r)$, and also define the function $f : \mathbb{F}_p^n \to \mathbb{Z}$ by $f(\mathbf{z}) := \sum_{\mathbf{x} \in \mathcal{L}} g(\mathbf{z} - \mathbf{x})$. This function f is periodic and counts the number of lattice points at distance of at most r from $\mathbf{y}$, i.e.

$$f(\mathbf{y}) = |\{\mathbf{v} \in \mathcal{L} \ : \ \|\mathbf{y} - \mathbf{v}\| \leq r\}| \ .$$

Let $\mathbf{y} \in \mathbb{R}^n$ be a vector drawn uniformly at random from the fundamental parallelipiped $\mathcal{P} = \mathcal{P}(\mathcal{L})$ of the lattice. Since $\mathcal{P}$ tiles $\mathbb{R}^n$ by lattice translates $\mathbf{v} + \mathcal{P}$ where $\mathbf{v} \in \mathcal{L}$, the number of lattice vectors close to $\mathbf{y}$ is independent of which lattice translate it lies in. So without loss of generality we can choose a vector in $\mathcal{P}$.

Now we compute the expected number of close lattice points to $\mathbf{y}$. By definition of f, linearity of expectation, the fact that $\mathcal{P}$ partitions $\mathbb{R}^n$ by lattice translates, and that the volume of $\mathcal{B}(\mathbf{0}, r)$ is equal to the sum of the volumes of the portion of the ball that lies in each lattice translate of $\mathcal{P}$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{y}}[f(\mathbf{y})] &= \sum_{\mathbf{v} \in \mathcal{L}} \Pr_{\mathbf{y}}[\mathbf{y} - \mathbf{v} \in \mathcal{B}(\mathbf{0}, r)] \\
&= \sum_{\mathbf{v} \in \mathcal{L}} \Pr_{\mathbf{y}}[\mathbf{y} \in \mathcal{B}(\mathbf{v}, r)] \\
&= \sum_{\mathbf{v} \in \mathcal{L}} \Pr_{\mathbf{y}}[\mathbf{y} \in \mathcal{B}(\mathbf{v}, r) \cap \mathcal{P}] \\
&= \sum_{\mathbf{v} \in \mathcal{L}} \Pr_{\mathbf{y}}[\mathbf{y} \in \mathcal{B}(\mathbf{0}, r) \cap (\mathcal{P} - \mathbf{v})] \\
&= \sum_{\mathbf{v} \in \mathcal{L}} \frac{\mathrm{vol}(\mathcal{B}(\mathbf{0}, r) \cap (\mathcal{P} - \mathbf{v}))}{\mathrm{vol}(\mathcal{P})} \\
&= \frac{\mathrm{vol}(\mathcal{B}_r^n)}{\det(\mathcal{L})} \\
&= \left(\frac{\mathrm{vol}(\mathcal{B}_r^n)^{1/n}}{\det(\mathcal{L})^{1/n}} \right)^n .
\end{aligned}$$

By hypothesis, $\mathrm{vol}(\mathcal{B}_r^n)^{1/n} > \det(\mathcal{L})^{1/n}$, so this expected value is exponential in n. Hence there must exist a $\mathbf{y}$ such that the list size $L = |\mathcal{L} \cap \mathcal{B}(\mathbf{y}, r)|$ is exponential. □

We are particularly interested in the list-decoding capacity for the ℓ_1 and ℓ_2 norms.

Instantiating Theorem 6.13 with these norms gives the following two bounds.

Corollary 6.14 (Lattice List-decoding Capacity for ℓ_2 norm)**.** *For any positive integers n and L, real $r > 0$, and lattice $\mathcal{L} \subset \mathbb{R}^n$, if $\mathcal{L}$ is (r, L)-list-decodable for radius at least $r > \Theta(\sqrt{n}) \cdot \det(\mathcal{L})^{1/n}$, then the list size L must be exponential in n.*

This can be shown using a nearly identical proof to that of Theorem 6.13 and using the fact that the r-radius ball $\mathcal{B}_r^n$ in the ℓ_2 norm has volume

$$\mathrm{vol}(\mathcal{B}_r^n) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} \cdot r^n = \Theta\Big(\frac{r^n}{n^{n/2}}\Big) .$$

Corollary 6.15 (Lattice List-decoding Capacity for ℓ_1 norm)**.** *For any positive integers n and L, real $r > 0$, and lattice $\mathcal{L} \subset \mathbb{R}^n$, if $\mathcal{L}$ is (r, L)-list-decodable for radius at least $r > 2/(n! \det(\mathcal{L}))^{1/n}$, then the list size L must be exponential in n.*

This can be shown using a nearly identical proof to that of Theorem 6.13 and using the fact that the r-radius ball $\mathcal{B}_r^n$ in the ℓ_1 norm has volume

$$\mathrm{vol}(\mathcal{B}_r^n) = \frac{2^n}{n!} \cdot r^n = \frac{(2r)^n}{\Theta(n)} .$$

In the particular case of Construction A lattices, where $\mathcal{L} = \tilde{\mathcal{C}} + p\mathbb{Z}^n$ for some linear code $\mathcal{C} \subseteq \mathbb{F}_p^n$ over a prime field, if the code has rate $R = k/n$, the determinant of the lattice $\mathcal{L}$ has the form $\det(\mathcal{L}) = p^{n-k}$ by Lemma 6.3. Then by Theorem 6.13, in the ℓ_2 norm the list size becomes exponential if

$$\mathrm{vol}(\mathcal{B}_r^n)^{1/n} = \Theta\Big(\frac{r}{\sqrt{n}}\Big) > \det(\mathcal{L})^{1/n} = p^{1-R} .$$

So if the rate is greater than $R > 1 - \log_p(\Theta(r/\sqrt{n}))$, the list size L is exponential in n.

6.3 Future Directions

The probabilistic construction of dense lattices from Section 6.1 succeeds with high probability and requires a fairly strict condition on the minimum distances and codimensions for the underlying codes. Our result only implies the existence of codes that satisfy the conditions sufficient to construct a dense Construction D lattice. It is still unknown how to explicitly construct and efficiently list-decode lattices that achieve Minkowski's inequality within a constant factor. It is also unknown how to efficiently list-decode Construction D

lattices obtained from an *arbitrary* tower of linear codes. All previous lattice list-decoding algorithms, namely those in [MP22, GP17, KM23], rely on efficient decoders for the underlying family of codes in addition to the recursive structure of Construction D. Thus, constructing an *explicit and efficiently list-decodable lattice* that achieves Minkowski's bound – even without using Construction D – remains an open problem.

We presented some probabilistic lattice list-decoding capacity bounds in Section 6.2. These only give an idea of the asymptotic relationship between the radius, determinant, and list size. A more fundamental problem is that of determining the maximum density of a list-decodable lattice for *any* given dimension, radius, and list size. This is a purely geometric problem: The error-correcting capability of a lattice can be represented by spheres centered at each lattice point with radius equal to the decoding distance. For this lattice to be *efficiently* list-decodable, any point in the space should lie in the decoding radius of at most a polynomial number L (in the dimension) of lattice points. All together, this means that any given point must lie in at most L overlapping decoding-radius spheres. In this way, determining the fundamental limits of lattice list-decoding is precisely this multi-sphere packing problem of determining the densest lattice packing for a given radius that allows a limited number of sphere overlaps. Some limited progress has been made on this problem in [IHKU14, Yan80, Pur73, ZV22, Tót83].

APPENDIX A

Proof of Levin's Result

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 probabilistic polynomial-time 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 $1/\sigma$. 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 reduction in Chapter 3, 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 A.1 (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 polynomial-time algorithm that, given w 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

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

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

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

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

and the expected runtime is polynomial.

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,

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

Since $\mathbf{r}$ is fixed in the sum, the probability is eliminated, so we obtain a constant (dependent on $\mathbf{r}$) multiple of $\widehat{\mathcal{G}}$, 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 $\sqrt{\alpha_{\mathcal{G}}} \cdot \delta_{\mathcal{G}}$. 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 $g_i(\mathbf{u}) := \mathcal{G}(\mathbf{R}\mathbf{u} + \mathbf{e}_i)$.

    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 $\mathbf{x}' := (x'_1, \ldots, x'_n)$.

  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 analyze its performance.

Consider the case where $c(\mathbf{x}) > 0$. (Otherwise, $\delta_{\mathcal{G}} \cdot \sqrt{\alpha_{\mathcal{G}}} \le 0$, 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 positive integer ℓ' is $2^{-\ell'}$. Then $\ell > \ell'$ with probability

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

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

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

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

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

random vectors, then

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

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 $\mathbf{r}_1,\ldots,\mathbf{r}_m$ are pairwise independent, $Z_1,\ldots,Z_m$ 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

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

By the Chebyshev inequality, we obtain

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

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

Note that for pairwise independent $\mathbf{r}_1,\ldots,\mathbf{r}_m \in \{0,1\}^n$, the vectors $\mathbf{r}_1 + \mathbf{e}_i$, $\ldots,\mathbf{r}_m + \mathbf{e}_i$ 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 $(-1)^{\langle \mathbf{x},\mathbf{Ru}+\mathbf{e}_i\rangle} \cdot \mathcal{G}(\mathbf{Ru}+\mathbf{e}_i) = (-1)^{x_i}(-1)^{\langle \mathbf{x},\mathbf{Ru}\rangle} \cdot \mathcal{G}(\mathbf{Ru}+\mathbf{e}_i)$, where x_i is the ith

bit of $\mathbf{x}$. The sum above is larger than zero if and only if

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

By the claim, this happens for at least a $1-1/(2n)$ 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

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

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

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

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 $1/s$ via this expected polynomial-time reduction. $\square$

BIBLIOGRAPHY

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. NIST PostQuantum Cryptography Standardization Project submission, 2022.

[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.

[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.

[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.

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[Ajt98] Miklós Ajtai. The shortest vector problem in l2 is np-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, 1998.

[ALNSD20] Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. Slide Reduction, Revisited—Filling the Gaps in SVP Approximation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 274–295, Cham, 2020. Springer International Publishing.

[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.

[ALV25] Divesh Aggarwal, Jin Ming Leong, and Alexandra Veliche. Worst-Case to Average-Case Hardness of LWE: An Alternative Perspective. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography Conference*, pages 308–336, Cham, 2025. Springer Nature Switzerland.

[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.

[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.

[Bab86] László Babai. On Lovász Lattice Reduction and the Nearest Lattice Point Problem. *Combinatorica*, 6:1–13, 1986.

[Bab16] László Babai. Graph Isomorphism in Quasipolynomial Time. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016.

[Ban95] Wojciech Banaszczyk. Inequalites for Convex Bodies and Polar Reciprocal Lattices in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 13:217–231, 1995.

[BBB+25] Huck Bennett, Drisana Bhatia, Jean-François Biasse, Medha Durisheti, Lucas LaBuff, Vincenzo Pallozzi Lavorante, and Philip Waitkevich. Asymptotic Improvements to Provable Algorithms for the Code Equivalence Problem. Cryptology ePrint Archive, Paper 2025/187, 2025.

[BBPS21] Alessandro Barenghi, Jean-Francois Biasse, Edoardo Persichetti, and Paolo Santini. LESS-FM: Fine-Tuning Signatures from the Code Equivalence Problem. Cryptology ePrint Archive, Paper 2021/396, 2021.

[BBPS22] Alessandro Barenghi, Jean-Francois Biasse, Edoardo Persichetti, and Paolo Santini. On the Computational Hardness of the Code Equivalence Problem in Cryptography. Cryptology ePrint Archive, Paper 2022/967, 2022.

[BCGQ11] László Babai, Paolo Codenotti, Joshua Grochow, and Youming Qiao. Code Equivalence and Group Isomorphism. pages 1395–1408, 01 2011.

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

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

[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.

[BM23] Jean-François Biasse and Giacomo Micheli. A Search-to-Decision Reduction for the Permutation Code Equivalence Problem. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 602–607, 2023.

[BOS19] Magali Bardet, Ayoub Otmani, and Mohamed Saeed-Taha. Permutation Code Equivalence is Not Harder than Graph Isomorphism When Hulls Are Trivial. In *IEEE International Symposium on Information Theory, ISIT 2019, Paris, France, July 7-12, 2019*, pages 2464–2468. IEEE, 2019.

[BP22] Huck Bennett and Chris Peikert. Hardness of the (Approximate) Shortest Vector Problem: A Simple Proof via Reed-Solomon Codes. In *International Workshop and International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2022.

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

[BS83] E. Barnes and N. Sloane. New Lattice Packings of Spheres. *Canadian Journal of Mathematics*, 35, February 1983.

[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.

[BW24] Huck Bennett and Kaung Myat Htay Win. Relating Code Equivalence to Other Isomorphism Problems. Cryptology ePrint Archive, Paper 2024/782, 2024.

[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.

[CS88] John Conway and Neil Sloane. *Sphere Packings, Lattices and Groups*, pages 232–233. Springer New York, NY, January 1988.

[CSV25] Mahdi Cheraghchi, Nikhil Shagrithaya, and Alexandra Veliche. Reductions Between Code Equivalence Problems. To appear at IEEE International Symposium on Information Theory, 2025.

[DG23] Léo Ducas and Shane Gibbons. Hull Attacks on the Lattice Isomorphism Problem. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 177–204. Springer, 2023.

[Din02] Irit Dinur. Approximating SVP_∞ to Within Almost-Polynomial Factors is NP-Hard. *Theor. Comput. Sci.*, 285(1):55–71, 2002.

[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.

[Eli57] Peter Elias. List Decoding for Noisy Channels. Technical Report 335, Research Laboratory of Electronics, 1957.

[EOR91] N. D. Elkies, A. M. Odlyzko, and J. A. Rush. On the Packing Densities of Superballs and Other Bodies. *Inventiones Mathematicae*, 105:613–639, December 1991.

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

[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.

[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.

[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.

[GN08] Nicolas Gama and Phong Q Nguyen. Finding Short Lattice Vectors within Mordell's Inequality. In *Proceedings of the fortieth annual ACM symposium on Theory of Computing*, pages 207–216, 2008.

[GP17] Elena Grigorescu and Chris Peikert. List-decoding Barnes–Wall lattices. *Computational Complexity*, 26:365–392, June 2017.

[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—206, New York, NY, USA, 2008. Association for Computing Machinery.

[GRS19] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential Coding Theory, March 2019.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. Preliminary version in FOCS 1998.

[Gur01] Venkatesan Guruswami. *List Decoding of Error Correcting Codes*. PhD thesis, Massachusetts Institute of Technology, 2001.

[HR07] Ishay Haviv and Oded Regev. Tensor-Based Hardness of the Shortest Vector Problem to within Almost Polynomial Factors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 469–477, New York, NY, USA, 2007. Association for Computing Machinery.

[IHKU14] Mabel Iglesias-Ham, Michael Kerber, and Caroline Uhler. Sphere Packing with Limited Overlap, 2014.

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

[Kho05] Subhash Khot. Hardness of Approximating the Shortest Vector Problem in Lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[KM23] Elena Kirshanova and Ekaterina Malygina. Construction-D Lattice from Garcia-Stichtenoth Tower Code. *Designs, Codes, and Cryptography*, 92:1127–1142, December 2023.

[KS09] Adam R. Klivans and Alexander A. Sherstov. Cryptographic Hardness for Learning Intersections of Halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009. Learning Theory 2006.

[KV03] Ralf Koetter and Alexander Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.

[Leo03] Jeffrey Leon. Computing Automorphism Groups of Error-Correcting Codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 2003.

[Lev12] Leonid A. Levin. Randomness and Non-Determinism, 2012.

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

[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.

[LO85] Jeffrey C Lagarias and Andrew M Odlyzko. Solving Low-Density Subset Sum Problems. *Journal of the ACM (JACM)*, 32(1):229–246, 1985.

[Mac33] C. C. Macduffee. *The Theory of Matrices*. Verlag von Julius Springer, 1933.

[McE78] Robert J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. DSN Progress Report, 1978.

[Mic12] Daniele Micciancio. Inapproximability of the Shortest Vector Problem: Toward a Deterministic Reduction. *Theory of Computing*, 8(1):487–512, 2012.

[MKO18] Toshiki Matsumine, Brian M. Kurkoski, and Hideki Ochiai. Construction D Lattice Decoding and Its Application to BCH Code Lattices. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In *Annual Cryptology Conference*, pages 465–484. Springer, 2011.

[MP22] Ethan Mook and Chris Peikert. Lattice (List) Decoding Near Minkowski's Inequality. *IEEE Transactions on Information Theory*, 68(2):863–870, 2022.

[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.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

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

[MSW25] Daniele Micciancio and Mark Schultz-Wu. Bit Security: Optimal Adversaries, Equivalence Results, and a Toolbox for Computational-Statistical Security Analysis. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography*, pages 224–254, Cham, 2025. Springer Nature Switzerland.

[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.

[MW18] Daniele Micciancio and Michael Walter. On the Bit Security of Cryptographic Primitives. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–28. Springer, 2018.

[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.

[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.

[PR97] Erez Petrank and Ron M. Roth. Is Code Equivalence Easy to Decide? *IEEE Trans. Inf. Theory*, 43(5):1602–1604, 1997.

[Pur73] G. B. Purdy. The Lattice Triple Packing of Spheres in Euclidean Space. *Transactions of the American Mathematical Society*, 181:457–470, 1973.

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

[Reg09] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM*, 56(6), Sep 2009.

[RS94] Ron M. Roth and Paul H. Siegel. Lee-Metric BCH Codes and Their Application to Constrained and Partial-Response Channels. *IEEE Trans. Inf. Theory*, 40(4):1083–1096, 1994.

[Sen00] Nicolas Sendrier. Finding the Permutation Between Equivalent Linear Codes: The Support Splitting Algorithm. *IEEE Trans. Inf. Theory*, 46(4):1193–1203, 2000.

[Ser73] Jean-Pierre Serre. *A Course in Arithmetic*. Springer New York, NY, 1973.

[Sha48a] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Sha48b] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(4):623–656, 1948.

[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.

[SS13a] Nicolas Sendrier and Dimitrios E. Simos. How Easy is Code Equivalence over $\mathbb{F}_q$? In *International Workshop on Coding and Cryptography - WCC 2013*, Bergen, Norway, April 2013.

[SS13b] Nicolas Sendrier and Dimitris E. Simos. The Hardness of Code Equivalence over $\mathbb{F}_q$ and Its Application to Code-Based Cryptography. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 203–216, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Tót83] G Fejes Tóth. New Results in the Theory of Packing and Covering. In *Convexity and its Applications*, pages 318–359. Springer, 1983.

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

[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.

[Woz58] John M. Wozencraft. List Decoding. Quarterly Progress Report, Research Laboratory of Electronics, 1958.

[WY25] Shun Watanabe and Kenji Yasunaga. Bit-Security Preserving Hardness Amplification. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography*, pages 195–223, Cham, 2025. Springer Nature Switzerland.

[Yan80] L. J. Yang. Multiple Lattice Packings and Coverings of Spheres. *Monatshefte für Mathematik*, 89:69–76, 1980.

[ZV22] Yihan Zhang and Shashank Vatedka. List Decoding Random Euclidean Codes and Infinite Constellations. *IEEE Transactions on Information Theory*, 68(12):7753–7786, 2022.