TECHNISCHE
UNIVERSITÄT
DARMSTADT

Technische Universität Darmstadt
Department of Computer Science
Cryptography and Computer Algebra

Master's thesis
June 26, 2017

Solving Learning With Errors Instances Using Quantum Reductions

Sebastian Bugge
Technische Universität Darmstadt
Department of Mathematics

Supervisors: Prof. Dr. Johannes Buchmann
Dr. Rachid El Bansarkhani
Nabil Alkeilani Alkadri

Contents

1 Introduction **1**

2 Notation and Preliminaries **4**

2.1 Notation 4

2.2 A Brief Introduction to Quantum Computing 6

2.2.1 Qubits and Registers 6

2.2.2 Quantum Circuits and Reversible Computation 9

2.2.3 Quantum Measurements 11

2.2.4 The Quantum Fourier Transform 13

2.3 Lattice-based Cryptography 15

2.3.1 Lattices 15

2.3.2 Probability Theory and Gaussian Distributions on Lattices 17

2.3.3 Lattice Problems 19

Learning With Errors 20

2.4 The Dihedral Group 21

3 The Hidden Subgroup Problem (HSP) **24**

3.1 Solving the Hidden Subgroup Problem in the Abelian Case 24

4 Some Reductions of Lattice Problems **30**

4.1 A Reduction from BDD to uSVP 31

4.2 A Reduction from uSVP to DHSP 40

4.3 A Reduction from DCP to the Subset Sum Problem 48

5 Algorithms for DHSP and SSP **59**

5.1 DHSP Algorithms 59

5.1.1 Kuperberg's First Algorithm and Regev's Improvement 59

5.1.2 Kuperberg's Second Algorithm for DHSP 61

5.2 A Quantum Algorithm for SSP 65

6 Solving LWE Instances via Reductions to SSP and DHSP **70**
6.1 Viewing LWE as BDD 70
6.2 Combining the Reductions 74
7 Room for Improvements and Conclusions **80**
7.1 Room for Improvements 80
7.2 Conclusions 81
Bibliography **83**

Acknowledgements

First, I would like to thank my supervisors Prof. Dr. Johannes Buchmann, Dr. Rachid El Bansarkhani and Nabil Alkeilani Alkadri for assigning my master's thesis.

During the preparation of this thesis I had regular meetings and discussions with my supervisors Dr. Rachid El Bansarkhani and Nabil Alkeilani Alkadri. I would like to thank them especially for their time, their helpful remarks and conversations with me.

Finally I would like to thank my friends Thomas Eiter, Konrad Flindt, Thuy Linh Luu and Thomas Schneider, who were so kind to proofread earlier drafts of my thesis.

Affidavit

I hereby confirm that my thesis is the result of my own work. I have used no sources and aids other than those indicated. Where I used other sources I marked them as not my own. All sources and materials applied are listed.

The thesis was not submitted in the same or in a substantially similar version to another examination board and was not published elsewhere.

Sebastian Bugge

1 Introduction

Quantum computers, although mostly still a theoretical concept, have already impacted modern cryptography immensely. Since the celebrated algorithm for integer factorization by Peter Shor ([Sho94], [Sho99]), it has become clear that quantum computers change the landscape of problems that are considered to be hard and are therefore the basis of cryptographic systems. For example, large scale quantum computers would make it possible to break public-key cryptosystems such as the widely used RSA scheme, by using Shor's or related algorithms. Thus cryptographers have been looking for other seemingly hard problems that even quantum computers are not able to solve efficiently and that cryptographic systems can be based on. This field of research is known as *post-quantum cryptography*. Recent research suggests that *lattice-based cryptography*, i.e. cryptographic primitives based on lattices, are somewhat resilient to attacks not only by classical, but also by quantum computers. A variety of lattice problems, such as finding a shortest non-zero vector in a given norm or the task to find the closest lattice point to a given target, are conjectured to be hard problems, i.e. there are no known efficient algorithms to solve these problems until today, despite immense efforts in recent cryptographic research.

Another problem worth mentioning in this context is the *learning with errors problem* (LWE) first introduced by Oded Regev in 2005 ([Reg05]). This is a problem in machine learning which asks to recover some secret vector from a number of linear equations which have been disturbed by a random additive error term. Without the error the problem would be simple and could be solved by standard Gaussian elimination. However, adding the noise to the 'right side' of the system of linear equations makes the problem seemingly very hard. Regev showed in [Reg05] that LWE is at least as hard as certain lattice problems in the worst-case. Since its introduction, the LWE problem has been subject to a lot of research. There are quite a variety of cryptographic constructions based on the conjectured hardness of LWE, such as public-key cryptosystems (e.g. [Reg05], [Pei09]), digital signature schemes (e.g. [BG14], [DDLL13]) and key exchange protocols (e.g. [DXL12]). However, there are algorithms – both classical and quantum – for solving LWE, but they all require superpolynomial resources, e.g. [BKW03].

A good deal of this thesis will be dedicated to explore two ways of solving instances of LWE by using quantum reductions from LWE to other computational problems and then to solve these problems with quantum algorithms. More explicitely we will show how it is possible to view an instance of LWE as an instance of the *bounded distance decoding*

problem (BDD), which can be reduced to the *dihedral hidden subgroup problem* (DHSP) as well as the *subset sum problem* (SSP) by combining a variety of reductions from the literature due to Bai et al. ([SBW16]) and Regev ([Reg04a]). Then, we use solvers due to Kuperberg ([Kup05], [Kup11]), Regev ([Reg04b]) and Bernstein et al. [BJLM13] to solve DHSP and SSP and thus BDD/LWE. We analyse the complexity of the combined reductions together with the solver algorithms to present a complete analysis of our approach to solve LWE instances with quantum algorithms for DHSP and SSP. We also specify what kind of LWE instances, i.e. what choices for the LWE parameters, can be solved with the help of quantum reductions to DHSP or SSP, respectively. In this context the *failure parameter* of the *dihedral coset problem* (DCP), which is a variant of DHSP, will be important. Finally, we discuss some approaches for further research in order to make the approach of solving LWE instances with quantum reductions to DHSP and SSP more efficient.

Structure of this thesis

First, we collect the required preliminaries in the topics of quantum computing as well as lattice-based cryptography in Chapter 2, where we also cover the notation used in this thesis (Section 2.1). In Section 2.2 we will cover the minimal requirements to provide the necessary vocabulary and techniques required to understand the various quantum algorithms presented later on. A very good introduction to the topic of quantum computation and quantum information is the book by Nielsen and Chuang [NC00]. In Section 2.3 we will provide some background on the topic of lattices, as well as some results on probability theory which we will require later. We also give a short survey on some of the standard lattice problems considered in this thesis. Afterwards, we will review the *dihedral group* since the hidden subgroup problem on this non-abelian group will be of great importance to us.

In Chapter 3 we will introduce the *hidden subgroup problem* (HSP), which plays a somewhat prominent role in the theory of quantum computing, since it captures problems like integer factorisation, discrete logarithms as well as graph isomorphism. We will present an algorithm from the literature (e.g. [Lom04] or [NC00]) that solves HSP in the abelian case.

In Chapter 4 we consider various reductions of computational problems from the literature. The reductions are due to Bai et al. ([SBW16]) and Regev ([Reg04a]) and will enable us to solve instances of the *bounded distance decoding problem* (BDD) with quantum algorithms for the hidden subgroup problem over the dihedral group (DHSP) as well as the *subset sum problem* (SSP).

In Chapter 5, we will take a look at two algorithms from the literature for solving DHSP and SSP. The first subexponential algorithm for DHSP was due to Greg Kuperberg ([Kup05]) and is from 2003. After an improvement of the algorithm by Regev ([Reg04b]) in 2004 , which only needed a polynomial amount of quantum space, Kuperberg generalised Regev's improved algorithm even further in 2011 ([Kup11]). Kuperberg's second

and most recent algorithm for DHSP is the one we will use as a solver. The solver algorithm for SSP is due to Bernstein et al. ([BJLM13]) and is based on an SSP algorithm by Howgrave-Graham and Joux ([HGJ10]) as well as on techniques from a quantum walk algorithm by Ambainis ([Amb07]).

Afterwards, in Chapter 6 we will explain how to view an instance of LWE as an instance of the lattice problem BDD and apply the reductions established earlier in order to solve LWE with the quantum algorithms for DHSP and SSP. We will then analyse the time and space requirements for solving LWE instance by making use of the given reductions to DHSP and SSP.

Finally, in Chapter 7 we will point out certain parts of the reductions from Chapter 4 whose improvement would lead to significantly better results when we solve LWE instances with algorithms for DHSP and SSP in the way presented in this thesis and we will also compare our results with algorithms that are used to solve other hard lattice problems.

2 Notation and Preliminaries

In this chapter, we will cover the necessary preliminaries from the topics of quantum computing as well as lattice-based cryptography we will require later on. We will also establish some general notation that is used throughout this thesis.

2.1 Notation

In this section we will collect some frequently used and fairly standard notation. We denote the natural, integer, rational, real and complex numbers by the symbols $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$, respectively. In this thesis, the natural numbers do not include 0. Therefore, we write $\mathbb{N}_0$ for the set $\mathbb{N} \cup \{0\}$. For any positive integer p we denote the set $\{0, \dots, p-1\}$ by $\mathbb{Z}_p$ and we do not always rigorously distinguish between $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}_p$. Unless stated otherwise logarithms are always to base 2. We write id or E for the identity map or matrix and, if we want to emphasise the dimension d, we write id_d or E_d. The symbol $\cong$ denotes the isomorphy of appropriate objects, e.g. groups, rings or vector spaces. We write $\mathrm{dist}(x, y)$ for the distance between two vectors $x, y \in \mathbb{R}^n$ in some given metric. Unless stated otherwise we always refer to the euclidean distance. For a set $S \subset \mathbb{R}^n$ and a vector $t \in \mathbb{R}^n$ we set $\mathrm{dist}(t, S) \coloneqq \inf_{s \in S} \mathrm{dist}(s, x)$. With $\oplus$ we denote the bitwise addition XOR. The term $\mathrm{poly}(n)$ refers to an unspecified, positive-valued polynomial in $n \in \mathbb{N}$. We use the notation $\mathcal{N}(\mu, \sigma)$ for a normal distribution with mean μ and standard deviation σ (see Def. 2.35) as well as $\mathcal{U}(S)$ for the discrete or continuous uniform distribution on some set $S \subseteq \mathbb{R}$.

Since we will be concerned with, for example, the running time of various algorithms and reductions and since we are particularly interested in the asymptotical examination of these, we will make heavy use of the Landau symbols.

Definition 2.1 (Landau symbols)**.** Let $f, g \colon \mathbb{R} \to \mathbb{R}$ be some real valued functions and $a \in \mathbb{R} \cup \{-\infty, \infty\}$. The Landau symbols are defined as follows:

- $f \in \mathcal{O}(g) :\iff \limsup_{x \to a} \left| \frac{f(x)}{g(x)} \right| < \infty$,
- $f \in \Omega(g) :\iff \liminf_{x \to a} \left| \frac{f(x)}{g(x)} \right| > 0$,
- $f \in \Theta(g) :\iff f \in \mathcal{O}(g) \cap \Omega(g)$,

- $f \in o(g) :\Longleftrightarrow \lim_{x\to a} \left|\frac{f(x)}{g(x)}\right| = 0$,
- $f \in \omega(g) :\Longleftrightarrow \lim_{x\to a} \left|\frac{f(x)}{g(x)}\right| = \infty$.

Unless stated otherwise, we will always consider the case $a = \infty$. We use the common (abusive) notation $f = \mathcal{O}(g)$ instead of the more accurate $f \in \mathcal{O}(g)$, and we will also make use of the expression $\widetilde{\mathcal{O}}$ when we choose to ignore logarithmic factors, that is $f = \widetilde{\mathcal{O}}(g) :\Longleftrightarrow f \in \mathcal{O}(g \log^c(g))$ for some $c \in \mathbb{R}_{\geq 0}$.

An important term in cryptography, when it comes to analysing algorithms, is 'efficiency'. Although the usage of the word 'efficient' may vary in some cases, it means in general that the algorithm has time and space complexity *polynomial* in the input size. Whenever we are talking about a reduction from a problem P to another problem P', we mean an algorithm that solves P that has access to a (maybe hypothetical) subroutine that solves P'. When we talk about a *polynomial-time reduction,* the term polynomial-time refers to the time the reduction takes, excluding the time of the subroutine.

Since we do not restrict ourselves to deterministic algorithms in cryptography, another important quantity in the analysis of (probabilistic) algorithms is the success probability. In this context the terms *negligible* or *negligible functions* are of great importance.

Definition 2.2 (Negligibility)**.** Let n denote the input size of a problem. A function $\varepsilon\colon \mathbb{N} \to \mathbb{R}$ is called *negligible,* if for every polynomial $p\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ there exists an $N_0 \in \mathbb{N}$ such that

$$\varepsilon(n) \leq \frac{1}{p(n)}$$

holds for all $n \geq N_0$. If two functions α, β differ only by a negligible amount, we write $\alpha \approx \beta$.

A standard example for a negligible function is 2^{-n}. Some useful and easily proved properties of negligible functions are the following.

Lemma 2.3. *Let $\varepsilon, \varepsilon_1, \varepsilon_2$ be negligible functions and δ a non-negligible function. Let $p\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a polynomial. Then*

- *$p \cdot \varepsilon$ is negligible,*
- *$\varepsilon_1 + \varepsilon_2$ is negligible,*
- *$\delta - \varepsilon$ is not negligible.*

Coming back to the success probability of an algorithm, we say that an algorithm solves a certain problem with 'good' or 'high' probability, if its success probability is bounded from below by a non-negligible function, e.g. $1/\mathrm{poly}(n)$. This success probability might

be amplified to a desired degree by running the algorithm several (polynomially many) times. Finally, when we say that an algorithm has a success probability 'exponentially close to 1' we mean that its success probability differs from 1 by an amount that is exponentially small in the input size and therefore negligible.

2.2 A Brief Introduction to Quantum Computing

In this section we will give a brief overview on some basic concepts in the topic of *quantum computing*. At first, we will consider the fundamental component of quantum computing, the *qubit*. Afterwards, we will take a look at the computational model underlying quantum computers and introduce the concept of *quantum measurements*. In the last section we will be concerned with a component of many prominent quantum algorithms: the *quantum Fourier transform*.

2.2.1 Qubits and Registers

A quantum bit, or *qubit*, is the quantum analogue to a classical bit. Just like usual bits, qubits also have states. The main difference is that a classical bit can only be in one of the two states 0 and 1 but a qubit, as a two-state quantum mechanical system, can be in a *superposition* of states. A superposition can be thought of as some kind of undisturbed overlapping of physical quantities, e.g. waves. In the context of quantum mechanics it usually refers to an 'overlapping', i.e. a linear combination, of waves functions, or state vectors of some quantum mechanical system. We will begin by defining single qubits in a mathematical context, and by making use of the tensor product we will be able to describe systems with more than one qubit. According to the postulates of quantum mechanics, any isolated physical system is associated to a complete complex vector space with an inner product, which is called the *state space*. The system is then completely described by a unit vector, called the *state vector*, in the state space of the system.

Remark 2.4. (Dirac notation) Throughout this thesis we will use the Dirac notation, which is fairly standard in quantum mechanics. Expressions like $|\varphi\rangle$ are so-called 'kets' and represent vectors of a complex Hilbert spaces $\mathcal{H}$. Expressions like $\langle\psi|$ are so-called 'bras' and represent elements of the dual space $\mathcal{H}^\star$. For finite-dimensional Hilbert spaces with a fixed orthonormal basis – this will always be the case in this thesis – we can think of the kets as column and of the bras as row vectors. For a given ket $|\psi\rangle$, the corresponding bra $\langle\psi|$ is given by

$$\langle\psi| = |\psi\rangle^\dagger := \overline{|\psi\rangle}^T.$$

Bras and kets can be paired in two different ways. A 'braket' $\langle\psi|\varphi\rangle$ denotes the inner product of the two kets $|\psi\rangle$ and $|\varphi\rangle$, whereas the 'ketbra' $|\psi\rangle\langle\varphi|$ denotes the linear

operator acting on $x \in \mathcal{H}$ according to $(|\psi\rangle\langle\varphi|)(x) = \langle\varphi|x\rangle|\psi\rangle$. In the setting of this thesis we can therefore think of ketbras as matrices.

Definition 2.5 (The qubit)**.** A *qubit* is a two-state quantum-mechanical system. The state space $\mathcal{H}$ is a *Hilbert space*, i.e. a complete inner product space, of dimension two. Let further $\{|0\rangle, |1\rangle\}$ denote an orthonormal basis, i.e. a basis of pairwise orthogonal unit vectors, of $\mathcal{H}$. Then any state vector is of the form

$$|\psi\rangle = a|0\rangle + b|1\rangle,$$

for $a, b \in \mathbb{C}$ satisfying $|a|^2 + |b|^2 = 1$.

Example 2.6. Consider the complex vector space $\mathbb{C}^2$ with the usual (complex) inner product $\langle\cdot,\cdot\rangle$. Then $(\mathbb{C}^2, \langle\cdot,\cdot\rangle)$ is a Hilbert space. We use the orthonormal basis

$$|0\rangle \coloneqq \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle \coloneqq \begin{pmatrix}0\\1\end{pmatrix}$$

and call these the *computational basis states*. For complex numbers $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$ we say that

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

is a *qubit state* or a *superposition* of the basis states. We will interpret the amplitudes $|\alpha|^2 = |\langle 0|\psi\rangle|^2$ and $|\beta|^2 = |\langle 1|\psi\rangle|^2$ as the probabilities of the qubit being in the state $|0\rangle$ or $|1\rangle$, respectively, when measured (see Section 2.2.3).

As we have already mentioned, we would like to consider systems with more than one qubit. We can achieve this by using the tensor product, which we denote by $\otimes$. The formal definition reads as follows.

Definition 2.7 (Tensor product)**.** Let $\mathbb{K}$ be a field and V, W two $\mathbb{K}$-vector spaces. Let $\mathcal{X}$ denote the $\mathbb{K}$-vector space whose basis is formally given by all objects (v, w) with $v \in V$ and $w \in W$, and let $\mathcal{U}$ be the subspace of $\mathcal{X}$ that is generated by all elements of the form

(i) $(v, w_1 + w_2) - (v, w_1) - (v, w_2)$

(ii) $(v_1 + v_2, w) - (v_1, w) - (v_2, w)$

(iii) $(av, w) - a(v, w)$

(iv) $(v, aw) - a(v, w)$,

where $a \in \mathbb{K}, v, v_1, v_2 \in V, w, w_1, w_2 \in W$. The *tensor product* of V and W is defined as

$$V \otimes W \coloneqq \mathcal{X}/\mathcal{U}.$$

It is itself a $\mathbb{K}$-vector space of dimension $\dim(V \otimes W) = \dim V \cdot \dim W$. Its elements, denoted by $v \otimes w \coloneqq [(v, w)]$, are called tensors. If $\{e_i\}$ and $\{f_j\}$ are bases of V and W, respectively, then $\{e_i \otimes f_j\}$ forms a basis of $V \otimes W$.

The rather abstract definition of the tensor product can be somewhat irritating at first. A more concrete understanding of the tensor product may be obtained through the *Kronecker product.*

Definition 2.8 (Kronecker product)**.** Let $(a_{ij}) = A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{r\times s}$ represent linear maps $\varphi\colon V_1 \to W_1$ and $\psi\colon V_2 \to W_2$, respectively, between the $\mathbb{C}$-vector spaces V_1, V_2, W_1, W_2. Then the Kronecker product

$$A \otimes B \coloneqq \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix} \in \mathbb{C}^{mr\times ns}$$

represents the tensor product $\varphi \otimes \psi\colon V_1 \otimes W_1 \to V_2 \otimes W_2,\ v_1 \otimes v_2 \mapsto \varphi(v_1) \otimes \psi(v_2)$.

Example 2.9. We can describe a system with two qubits as follows. Consider the tensor product $\mathbb{C}^2 \otimes \mathbb{C}^2$ with the following basis

$$|00\rangle \coloneqq |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix}, \quad |01\rangle \coloneqq |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix},$$

$$|10\rangle \coloneqq |1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix}, \quad |11\rangle \coloneqq |1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}.$$

Using the Kronecker product, we have

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \quad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \quad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

Similarly, we define

$$\underbrace{|0\dots0\rangle}_{n \text{ times}} \coloneqq |0\rangle^{\otimes n} = \begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}, \quad |0\dots01\rangle \coloneqq \begin{pmatrix}0\\1\\\vdots\\0\end{pmatrix}, \quad \dots \quad \underbrace{|1\dots1\rangle}_{n \text{ times}} \coloneqq |1\rangle^{\otimes n} = \begin{pmatrix}0\\\vdots\\0\\1\end{pmatrix}.$$

as a basis for $(\mathbb{C}^2)^{\otimes n}$. For convenience, we usually omit the tensor sign $\otimes$ in our notation and write for example $|0,0\rangle$ instead of $|0\rangle \otimes |0\rangle$. Sometimes we write $|\psi\rangle\,|\varphi\rangle$ rather than $|\psi,\varphi\rangle$ to emphasise that we want to consider the entries in the kets separately. We call $|\psi\rangle\,|\varphi\rangle$ a two qubit *quantum register.*

2.2.2 Quantum Circuits and Reversible Computation

We will use *circuits* as our computational model. Just as classical circuits, which consist of wires and logical gates, quantum circuits are built from wires and *quantum gates*. The gates, classical or quantum, can be represented as matrices. If a gate is supposed to have n (qu)bits as input and m (qu)bits as output, it can be represented by a $2n \times 2m$ matrix. For example, the (classical) AND-gate can be represented as

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

when we use the notation of Example 2.6. The main difference between classical and quantum circuits is that in a quantum circuit all computations must be *reversible,* i.e. they have to be represented by an invertible matrix. Because a qubit state of n qubits is a unit vector in $\mathbb{C}^{2n}$, we also require the matrix representing a quantum gate to be an isometry with respect to the Euclidean norm. Therefore, the appropriate condition that a matrix U representing a quantum gate has to fulfil is that it is *unitary,* i.e.

$$U^{-1} = U^\dagger := \overline{U}^T.$$

Vice versa, every unitary $2n \times 2n$ matrix defines a quantum gate acting on n qubits. Next we give an example of two important quantum gates: the CNOT-gate and the *Hadamard gate*.

Example 2.10. In the setting of Example 2.9, consider the two qubit gate given by the matrix

$$U_{CN} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

which acts on the basis states by

$$|00\rangle \mapsto |00\rangle\,, \quad |01\rangle \mapsto |01\rangle\,, \quad |10\rangle \mapsto |11\rangle\,, \quad |11\rangle \mapsto |10\rangle\,.$$

So U_{CN} flips the second qubit, the so-called target qubit, if and only if the first qubit, the so-called control qubit, is 1. Next we will take a look at the single qubit gate defined by

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

called the Hadamard gate. It acts on the basis states according to

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

So the Hadamard gate takes a basis state, namely $|0\rangle$ or $|1\rangle$, and puts it into a superposition of the two basis states with a probability of $1/2$ to measure either 0 or 1.

Example 2.11. We can use the Kronecker product to construct the Hadamard transform on n qubits by using the matrix representation H from above. We set $H_1 \coloneqq H$ and define recursively for $1 < j \leq n$

$$H_j \coloneqq H_1 \otimes H_{j-1} = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{j-1} & H_{j-1} \\ H_{j-1} & -H_{j-1} \end{pmatrix}.$$

For example, the Hadamard transform for a two qubit system is given by the unitary matrix

$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Just as in the case $n = 1$, the Hadamard transform acts on every basis state by creating a superposition of all basis states such that the probability to measure one of the states $|00\rangle, |01\rangle, |10\rangle$ or $|11\rangle$ is $1/4$.

We have observed that the AND-gate, which computes the logical function $(a, b) \mapsto a \wedge b$, is not reversible and is therefore not a valid quantum gate. However, we would obviously like to have an AND-gate in our repertoire. Fortunately, it is possible to simulate any classical irreversible circuit as a reversible circuit and therefore on a quantum computer by allowing the use of additional 'ancilla' input bits as well as the output of some, for the following computations unnecessary, 'garbage' output bits. This can be seen by using either the *Fredkin gate* or the *Toffoli gate,* which both act on three bits and are universal gates of reversible computation. The Fredkin gate swaps its first two bits if and only if its control bit, which remains unchanged, is equal to 1. For the Toffoli gate, the first two bits are control bits, which remain unchanged, and the third bit is negated if and only if both control bits are 1. The universality of the Toffoli gate can be realised by noting that the NAND-gate can be built by using the input bits for the NAND-gate as the two control bits of the Toffoli gate and setting the ancilla target bit to 1. Then the output of the target bit is $\neg(a \wedge b)$. The above reasoning, together with the well-known universality of the NAND-gate, gives us the following proposition.

Proposition 2.12. *Given any (perhaps irreversible) classical circuit that computes a function $f\colon D \to I, x \mapsto f(x)$, there is a reversible circuit which on input x, together with some ancilla bits in a standard state, computes $f(x)$ together with some additional garbage output denoted by $g(x)$.*

The dependence of the garbage bits $g(x)$ on the initial input x may cause problems in the quantum computer setting because it may affect interference properties crucial to quantum computations. However, it is possible to modify the computation in such a way that the garbage output bits do not depend on x anymore.

Proposition 2.13. *Assuming we are in the situation of Proposition 2.12, there is a reversible circuit $\mathcal{C}$ that computes the action*

$$(x, y) \overset{\mathcal{C}}{\mapsto} (x, y \oplus f(x))$$

for every possible input x to f, where y is some arbitrary starting state.

Proof. We assume that we have access to the reversible variants of the NOT-gate and the CNOT-gate. Since we have seen that the Toffoli gate is universal, this poses no problem. We begin by adding two registers: one to store $f(x)$ and one to store the garbage bits $g(x)$. By making use of the NOT-gates, we may assume that all ancilla bits are initially zero, thus we can compute

$$(x, 0, 0, y) \mapsto (x, f(x), g(x), y).$$

By using an appropriate number of CNOT-gates, we can bitwise add the value of the second register to the fourth one, resulting in

$$(x, f(x), g(x), y \oplus f(x)).$$

Since the computation of f is reversible and does not affect the fourth register at all, we can apply the reverse circuit to obtain

$$(x, 0, 0, y \oplus f(x)).$$

The statement follows by omitting the two zero registers. □

Remark 2.14. To simplify the notation, it is a standard procedure to refer to the modified circuit above as *the* reversible circuit that computes f. Most of the time we will not specifically state that the computation is reversible in the setting of quantum computers.

Remark 2.15. The complexity classes P and NP are the same, no matter if we talk about reversible or irreversible computation. This is because the overhead for the extra controlled-NOT operations is linear in the number of bits that are involved in the circuit, and the number of required ancilla bits scales at most linearly with the number of gates. Also, the number of gates in the reversible circuit is the same as in the irreversible one up to a constant factor representing the number of Toffoli gates which are needed to simulate a single element of the irreversible circuit. For a more detailed discussion of the content of this subsection, we refer to [NC00], Sections 1.3. and 3.2.5.

2.2.3 Quantum Measurements

Of course, we would like to observe the quantum mechanical systems in question at some point. Therefore another important concept for us is the *measurement* or the *partial measurement* of a quantum mechanical system. Such an interaction results in the system not being closed anymore, thus the mathematical description of this 'observation' is no longer required to be represented by unitary operators as before. The measurement destroys the quantum state and collapses it into a classical system.

Definition 2.16 (Quantum measurements)**.** Mathematically any quantum measurement is described by a collection of linear *measurement operators* $\{M_i\}_{i\in I}$, where I is an index set whose elements represent the possible outcomes of the quantum measurement. The measurement operators must fulfil the *completeness equation*

$$\sum_{i\in I} M_i^\dagger M_i = \mathrm{id},$$

where id is the identity operator on the Hilbert space in question. If the state of the quantum system is $|\psi\rangle$, then the probability p_i of measuring the outcome i is given by

$$p_i = \langle\psi| M_i^\dagger M_i |\psi\rangle ,$$

and the state $|\psi'\rangle$ of the system after the measurement is obtained by the *re-normalisation*

$$|\psi'\rangle = \frac{M_i |\psi\rangle}{\sqrt{\langle\psi| M_i^\dagger M_i |\psi\rangle}}.$$

Example 2.17. Consider the one qubit system from Example 2.6. Let $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$ be the measurement operators. It is easy to verify that they satisfy the completeness equation. If the system is in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ as above, we have

$$p_0 = |\alpha|^2, \quad p_1 = |\beta|^2$$

as we have stated earlier.

Remark 2.18. When describing quantum algorithms, it is standard to just say that a quantum register is measured rather than specifying the measurement operators. From now on, we will do this as well.

Often, we will be interested in measuring only a subset of the qubits used in our algorithms. Thus the remaining qubits will collapse into a state compatible with the measurement.

Example 2.19. Consider the quantum state

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle , \text{ with } |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1.$$

If we decide to measure only the left qubit we get 0 with probability $|\alpha|^2 + |\beta|^2$. Let us assume that the result of our measurement is indeed 0, then the state $|\psi\rangle$ above would collapse to

$$|\psi'\rangle = \frac{1}{\sqrt{|\alpha|^2 + |\beta|^2}}(\alpha|00\rangle + \beta|01\rangle)$$

because the squared values of the coefficients, or amplitudes, of the basis states must once again sum up to 1.

Sometimes it can be beneficial to perform a measurement in a basis different from the computational one as the following example illustrates.

Example 2.20. Consider the *Hadamard basis* on one qubit defined by

$$|+\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |-\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Let us assume we have somehow managed to prepare the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i d}\,|1\rangle)$$

for some integer d. Our goal is to find the least significant bit of d, i.e. we want to determine the parity of d. A simple calculation shows that the state $|\psi\rangle$ has the following representation in the Hadamard basis:

$$|\psi\rangle = \frac{1 - e^{\pi i d}}{2}\,|-\rangle + \frac{1 + e^{\pi i d}}{2}\,|+\rangle\,.$$

Thus a measurement in the Hadamard basis would produce the basis states $|+\rangle$ and $|-\rangle$ with probabilities

$$p_{|+\rangle} = \left|\frac{1 + (\cos(\pi d) + i\sin(\pi d))}{2}\right|^2 = \frac{1 + \cos(\pi d)}{2}$$
$$p_{|-\rangle} = \left|\frac{1 - (\cos(\pi d) + i\sin(\pi d))}{2}\right|^2 = \frac{1 - \cos(\pi d)}{2}.$$

From this we see immediately that we measure $|+\rangle$ with probability 1 if d is even, and $|-\rangle$ if d is odd.

2.2.4 The Quantum Fourier Transform

We will take a look at a part of many quantum algorithms: the *quantum Fourier transform* (QFT). From now on we will often use the notation $|j\rangle$ for some $j \in \mathbb{N}_0$, by identifying j with its binary representation $j_1 \dots j_n$, such that $j = j_1 \cdot 2^{n-1} + \dots + j_{n-1} \cdot 2 + j_n$. This is compatible with our former notation of the computational basis.

Definition 2.21. Let $N \in \mathbb{N}$ and let $|0\rangle, \dots, |N-1\rangle$ denote an orthonormal basis of a Hilbert space $\mathcal{H}$. The quantum Fourier transform (QFT) $\mathcal{F}\colon \mathcal{H} \to \mathcal{H}$ is the linear operator defined by

$$|j\rangle \overset{\mathcal{F}}{\mapsto} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N}\,|k\rangle\,, \quad \forall j \in \{0, \dots, N-1\}. \tag{2.1}$$

The action on an arbitrary state is given by

$$\sum_{j=0}^{N-1} x_j\,|j\rangle \overset{\mathcal{F}}{\mapsto} \sum_{k=0}^{N-1} y_k\,|k\rangle\,,$$

where

$$y_k \coloneqq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N}, \quad \text{for all } k \in \{0, \ldots, N-1\}$$

is the discrete Fourier transform of the N points $x_0, \ldots, x_{N-1}$.

Lemma 2.22. *For $N = 2^n$ let $\mathcal{F}$ denote the quantum Fourier transform acting on n qubits and $\omega = \exp(2\pi i/N)$. Then $\mathcal{F}$ is unitary and given by the matrix*

$$F \coloneqq \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \ldots & \omega^{(N-1)^2} \end{pmatrix}.$$

Proof. The matrix representation of $\mathcal{F}$ follows directly from the definition of the quantum Fourier transform (2.1). It remains to check that F is a unitary matrix. Let c_k denote the k-th column of the matrix F. On the one hand, we have

$$\langle c_i | c_j \rangle = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{(i-j)k} = \frac{1 - \omega^{(i-j)N}}{N(1 - \omega^{i-j})} = 0$$

for $i, j \in \{0, \ldots, N-1\}$, $i \neq j$, because ω^{i-j} is an N-th root of unity. On the other hand, if $i = j$, then

$$\langle c_i | c_i \rangle = \frac{1}{N} \sum_{k=0}^{N-1} 1 = 1.$$

Therefore, the columns of F form an orthonormal basis. So F is unitary. □

Remark 2.23. From Lemma 2.22 we see immediately that the inverse Fourier transform $\mathcal{F}^{-1}$ is given by $F^\dagger = \overline{F}$. It acts on the basis states according to

$$|k\rangle \xmapsto{\mathcal{F}^{-1}} \frac{1}{2^{n/2}} \sum_{j=0}^{2^n-1} e^{-2\pi i j k/2^n} |j\rangle,$$

which is easily seen because of $\overline{\omega} = \omega^{-1}$. Also note that the Hadamard transform is just the Fourier transform on one qubit, or as we will see later (in Section 3.1), the Fourier transform in the additive group of two elements $\mathbb{Z}/2\mathbb{Z}$.

The QFT has the following nice product representation.

Lemma 2.24. *Let $N = 2^n$ and $|0\rangle, \ldots, |N-1\rangle$ the computational basis for an n bit quantum system. For $m \leq n$ we write $0.j_1 j_2 \ldots j_m$ to represent the binary fraction $j_1/2 + j_2/4 + \ldots + j_m/2^m$. The QFT can be given the following product representation.*

$$|j_1, \ldots j_n\rangle = \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i \cdot 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i \cdot 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i \cdot 0.j_1 \ldots j_n} |1\rangle\right).$$

Proof. We write out j and k as their binary representations and calculate:

$$\begin{aligned}
|j\rangle = |j_1 \dots j_n\rangle &\overset{\mathcal{F}}{\mapsto} \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j (\sum_{l=1}^{n} k_l 2^{-l})} |k_1 \dots k_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right) \\
&= \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i \cdot 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i \cdot 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i \cdot 0.j_1 \dots j_n} |1\rangle\right).
\end{aligned}$$

□

Remark 2.25. The product representation can be used to show how to implement the QFT efficiently using $\mathcal{O}(n^2)$ quantum gates. It is possible to show that the QFT can be implemented efficiently for arbitrary $N \in \mathbb{N}$ by constructing it for any odd integer and then combining this with the construction for a power of 2 via the tensor product. For more details concerning the efficient implementation of the QFT, we refer to [NC00] Chapter 5 or [Lom04] Section 3.4 and Appendix A.

2.3 Lattice-based Cryptography

In this section we present some standard definitions, notations and well-known facts regarding lattices, and probability distributions we will need later on.

2.3.1 Lattices

Definition 2.26. Let $n \in \mathbb{N}$. A subset $\mathcal{L} \subset \mathbb{R}^n$ that is a discrete additive subgroup of $\mathbb{R}^n$ is called a *lattice*.

Remark 2.27. Every lattice $\mathcal{L}$ can be written as

$$b_1\mathbb{Z} + \ldots + b_m\mathbb{Z}$$

for some linearly independent vectors $b_1, \ldots, b_m \in \mathbb{R}^n$. The set $B \coloneqq \{b_1, \ldots, b_m\}$ is called a *basis* of $\mathcal{L}$. The integer m is called the *rank* of $\mathcal{L}$. If $m = n$, we say $\mathcal{L}$ has full

rank. It can be convenient to write a lattice as

$$\mathcal{L} \coloneqq \mathcal{L}(B) \coloneqq B \cdot \mathbb{Z}^m = \left\{ \sum_{i=1}^{m} z_i b_i : z_i \in \mathbb{Z} \right\},$$

where B is the matrix whose columns are the basis vectors $b_1, \ldots, b_m$. In a slight abuse of notation we will speak of B as the basis of $\mathcal{L}$ as well. Of course, a lattice basis is not unique. In fact, take any matrix $U \in \mathbb{Z}^{m \times m}$ with $|\det(U)| = 1$, i.e. U is a unimodular matrix, then $B \cdot U$ is also a basis of $\mathcal{L}$ and every basis of $\mathcal{L}$ can be obtained that way.

From now on we will be dealing with full-rank lattices exclusively. For every full-rank lattice, there is a dual lattice, which is defined as follows.

Definition 2.28 (The dual lattice)**.** Let $\mathcal{L}$ be a lattice and let $\langle \cdot, \cdot \rangle$ denote the standard scalar product on $\mathbb{R}^n$. The *dual lattice* of $\mathcal{L}$ is defined as

$$\mathcal{L}^\star \coloneqq \{x \in \mathbb{R}^n : \langle \mathcal{L}, x \rangle \subseteq \mathbb{Z}\}.$$

Remark 2.29. If B is a basis of $\mathcal{L}$, then $B^\star \coloneqq (B^T)^{-1}$ is a basis of $\mathcal{L}^\star$. We also have $\mathcal{L} = (\mathcal{L}^\star)^\star$ for every lattice $\mathcal{L}$.

Lattices have certain invariants, which do not depend on the chosen basis. Some of the invariants we will be dealing with are introduced in the following.

Definition 2.30. The *determinant of a lattice* $\mathcal{L}$ is

$$\det(\mathcal{L}) \coloneqq |\det(B)|$$

for some lattice basis B of $\mathcal{L}$. Note, that this definition does not depend on the choice of basis, see Remark 2.27. Also note that $\det(\mathcal{L}^\star) = 1/\det(\mathcal{L})$.

The determinant of a lattice also gives us the volume of the lattice, as the following proposition demonstrates. See for example [Neu13, Chapter 1 §4].

Proposition 2.31. *Let $\mathcal{L}$ be a full-rank lattice with basis (matrix) B. The* volume *of $\mathcal{L}$ is defined as the volume (Lebesgue measure) of the* fundamental parallelepiped

$$\mathcal{P}(B) \coloneqq B \cdot [0,1)^n = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in [0,1) \right\}$$

and is given by

$$\mathrm{vol}(\mathcal{P}(B)) = \det(\mathcal{L}).$$

Definition 2.32. Let $i \in \mathbb{N}$. The i-th successive minimum $\lambda_i(\mathcal{L})$ of a lattice $\mathcal{L}$ in a certain norm is the smallest $r \in \mathbb{R}$ such that there exist i linear independent vectors in $\mathcal{L}$ of norm at most r. Especially,

$$\lambda_1(\mathcal{L}) = \min_{v \in \mathcal{L} \setminus \{0\}} \|v\|$$

is the length of a shortest non-zero vector in $\mathcal{L}$.

We will require some useful facts about lattice bases, which have been reduced by the well-known LLL-algorithm, we will present in the following proposition. Roughly speaking, the LLL-algorithm computes a basis of rather short and nearly orthogonal lattice vectors. An LLL-reduced basis can be found for any lattice in polynomial-time. The running time of the LLL-algorithm can be stated as $\mathcal{O}(nd^5 \log^3(\|B\|_\infty))$, where d is the rank of the lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\|B\|_\infty = \max_{1 \leq i \leq j \leq d} |B_{i,j}|$. For more details see [LLL82], [EJ16] as well as [Kal83].

Proposition 2.33. *Let $\{b_1, \ldots, b_n\}$ be an LLL-reduced basis and $\{b_1^*, \ldots, b_n^*\}$ its Gram-Schmidt orthogonalisation. That is, b_i^* is the component of b_i orthogonal to the subspace spanned by $b_1, \ldots, b_{i-1}$. Then it holds*

$$\|b_i^*\| \leq \sqrt{2}\|b_{i+1}^*\|, \quad \text{for all} \quad 1 \leq i \leq n$$

and for $i > j$ we have

$$|\langle b_i, b_j^* \rangle| \leq \frac{1}{2}\|b_j^*\|^2.$$

Proof. See [Reg04a] or [LLL82]. □

Lemma 2.34. *Let $\mathcal{L}$ be an n-dimensional lattice and $u \in \mathcal{L}$ with $\lambda_1(\mathcal{L}) = \|u\|$, i.e. u is a shortest vector in $\mathcal{L}$. Let $\{b_1, \ldots, b_n\}$ be an LLL-reduced basis of $\mathcal{L}$ and*

$$u = \sum_{i=1}^{n} u_i b_i,$$

then $|u_i| \leq 2^{2n}$ for $i \in \{1, \ldots, n\}$.

Proof. See [Reg04a, Lemma 3.3.]. □

2.3.2 Probability Theory and Gaussian Distributions on Lattices

In this section we collect some well-known facts from probability theory, we will need later, and we introduce the standard Gaussian probability distribution on lattices.

Definition 2.35 (The normal distribution)**.** The *normal* or *Gaussian distribution* is the continuous probability distribution whose density function is given as

$$f(x) := \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

for some $\mu, \sigma^2 \in \mathbb{R}$ with $\sigma^2 > 0$. The normal distribution has mean μ and variance σ^2. We denote this distribution by $\mathcal{N}(\mu, \sigma^2)$.

Proposition 2.36 (Hoeffding's inequality [Hoe63])**.** *Let $X_1, X_2, \dots, X_m$ be independent and identically distributed random variables such that $a_i \leq X_i - \mathrm{E}[X_i] \leq b_i$ for real $a_1, \dots, a_m$, $b_1, \dots, b_m$ almost surely, and let $c > 0$ be any real constant, then*

$$\mathrm{Prob}\left[\sum_{i=1}^{m}(X_i - \mathrm{E}[X_i]) \geq c\right] \leq \exp\left(-\frac{2c^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right).$$

Applying Hoeffding's inequality to Bernoulli random variables leads to the following often used bound.

Proposition 2.37 (Chernoff-Hoeffding bound)**.** *Let $X_1, X_2, \dots X_m$ be independent and identically distributed Bernoulli random variables, i.e they take values in $\{0, 1\}$. Then for any $0 \leq \gamma \leq 1$ we have*

$$\mathrm{Prob}\left[\sum_{i=1}^{m} X_i \geq (\mathrm{E}[X_1] + \gamma)m\right] \leq e^{-2m\gamma^2}$$

and

$$\mathrm{Prob}\left[\sum_{i=1}^{m} X_i \leq (\mathrm{E}[X_1] - \gamma)m\right] \leq e^{-2m\gamma^2}.$$

In particular, it holds

$$\mathrm{Prob}\left[\left|\sum_{i=1}^{m} X_i - m\mathrm{E}[X_1]\right| \geq m\gamma\right] \leq 2e^{-2m\gamma^2}.$$

Proposition 2.38 (Chebyshev bound)**.** *Let X be a random variable with $\mu = \mathrm{E}[X]$ and $\sigma^2 = \mathrm{Var}(X) < \infty$. Then for all real $k > 0$ we have*

$$\mathrm{Prob}[|X - \mu| \geq k] \leq \frac{\sigma^2}{k^2}.$$

Proof. See for example [Kle13, Thm. 5.11] □

Definition 2.39. Let $A \subset \mathbb{R}^n$ be a countable subset, such that $\rho_s(A) \coloneqq \sum_{y \in A} \rho_s(y)$ converges, the *discrete Gaussian probability distribution* with parameter $s > 0$ is defined as

$$D_{A,s}(x) \coloneqq \frac{\rho_s(x)}{\rho_s(A)}, \quad x \in A,$$

where $\rho_s(x) \coloneqq \exp(-\pi\|x/s\|)$.

Remark 2.40. Since $\int \rho_s(x)\,\mathrm{d}x = s^n$, the function ρ_s/s^n is a density function of a continuous Gaussian distribution with mean 0 and standard deviation $\sigma = s/\sqrt{2\pi}$.

An important lattice quantity, introduced by Micciancio and Regev in [MR07], is the so-called *smoothing parameter*. Basically, it is the smallest width such that a discrete Gaussian measure on a lattice 'behaves like a continuous one'. For the sake of completeness, we include the formal definition.

Definition 2.41. Let $\mathcal{L}$ be an n-dimensional lattice and $\varepsilon = \varepsilon(n) > 0$ (typically some negligible function of n). The *smoothing parameter* $\eta_\varepsilon(\mathcal{L})$ is defined as the smallest s such that $\rho_{1/s}(\mathcal{L}^\star \setminus \{0\}) \leq \varepsilon$.

Remark 2.42. For more details on the smoothing parameter, especially in the LWE context, we refer to [MR07] as well as [Reg09].

2.3.3 Lattice Problems

In this section we will give a compact overview of some of the most prominent lattice problems. In the following definitions an n-dimensional lattice $\mathcal{L}$ is given to us by some lattice basis B and $\gamma(n)$ denotes some function of the lattice dimension, which we will think of as an approximation factor. We say, an algorithm solves one of the following problems efficiently, if it solves it in time $\mathcal{O}(\text{poly}(n))$.

Definition 2.43 (Approximate shortest vector problem (SVP_γ))**.** Given an n-dimensional lattice $\mathcal{L}$ and some $\gamma = \gamma(n)$, the *approximate shortest vector Problem* (SVP_γ) is to find a vector $v \in \mathcal{L}$ such that $\|v\| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$. The special case where $\gamma(n) \equiv 1$, i.e. $\|v\| = \lambda_1(\mathcal{L})$ is called the *shortest vector problem* (SVP).

Definition 2.44 (Decisional approximate shortest vector problem (GapSVP_γ))**.** Given an n-dimensional lattice $\mathcal{L}$ and some $\gamma = \gamma(n)$, where it is promised that either $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, determine which is the case.

Definition 2.45 (Approximate shortest independent vectors problem (SIVP_γ))**.** Given an n-dimensional lattice $\mathcal{L}$ and some $\gamma = \gamma(n)$, the *approximate shortest independent vectors problem* (SIVP_γ) is to find a set $\{s_1, \ldots, s_n\}$ of linearly independent vectors of $\mathcal{L}$ which satisfy $\|s_i\| \leq \gamma(n) \cdot \lambda_i(\mathcal{L})$ for all $i \in \{1, \ldots, n\}$.

Definition 2.46 (Unique shortest vector problem(uSVP_γ))**.** Given an n-dimensional lattice $\mathcal{L}$ and some $\gamma = \gamma(n)$ such that $\lambda_2(\mathcal{L}) > \gamma(n) \cdot \lambda_1(\mathcal{L})$, the *unique shortest vector problem* (uSVP_γ) is to find a vector $v \in \mathcal{L}$ with $\|v\| = \lambda_1(\mathcal{L})$. The quantity $\lambda_2(\mathcal{L})/\lambda_1(\mathcal{L})$ is sometimes called the *uSVP-gap*.

Definition 2.47 (Bounded distance decoding problem (BDD_γ))**.** Given an n-dimensional lattice $\mathcal{L}$, some $\gamma = \gamma(n)$ as well as a *target vector* $t \in \mathbb{R}^n$ with the guarantee that $\text{dist}(t, \mathcal{L}) \leq \gamma(n)\lambda_1(\mathcal{L})$, the *bounded distance decoding problem* (BDD_γ) is to find a lattice vector $c \in \mathcal{L}$ such that $\|c - t\| = \min_{v \in \mathcal{L}} \|v - t\|$.

Remark 2.48. Note that the closest vector is unique if and only if $\gamma < \frac{1}{2}$.

Remark 2.49. We remark that uSVP is harder for smaller values of the parameter γ whereas the BDD problem gets easier for smaller values of γ.

Learning With Errors

In the following we give the definition of the main problem which we would like to solve using quantum reductions. It is the *learning with errors problem* (LWE) introduced by Regev in [Reg05].

Definition 2.50 (Learning with errors (LWE))**.**

- Parameters and components: The description of an LWE instance consists of a positive integer n, an integer modulus $q \geq 2$ and an error distribution χ on $\mathbb{Z}_q$.
- The LWE distribution: For any vector $s \in \mathbb{Z}_q^n$ (the secret), we sample from the LWE distribution $A_{s,\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ by choosing $a \in \mathbb{Z}_q^n$ uniformly at random and choosing e according to χ. The output is $(a, b = \langle s, a\rangle + e \mod q)$, the so-called *LWE sample*.
- Search version of LWE: We are given a number of m independent LWE samples, drawn from $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$, the goal is to find s.
- Decision version of LWE: We are given a number of m independent LWE samples $(a_i, b_i)_{1\leq i\leq m} \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, distinguish whether these samples were drawn from $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ or from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

We say, that an algorithm solves the search or the decision version of LWE if it outputs the secret s with probability exponentially close to 1 or if it can distinguish with non-negligible advantage between samples, drawn from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ and the LWE-distribution $A_{s,\chi}$, respectively.

Remark 2.51. In most applications the error distribution over $\mathbb{Z}_q$ is as follows: Recall that the density function of the normal distribution with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma > 0$ is

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}\,.$$

Let the rounded normal distribution on $\mathbb{Z}_q$ be defined via

$$h^q_{\mu,\sigma}\colon \mathbb{Z}_q \to [0,1] \quad h^q_{\mu,\sigma}(n) \coloneqq \sum_{k\in\mathbb{Z}} \int_{(n+kq)-1/2}^{(n+kq)+1/2} f_{\mu,\sigma}(x)\,\mathrm{d}x.$$

Then the error distribution for LWE is taken to be $\chi = \Psi_\beta \coloneqq h^q_{0,\beta}$, where $\beta = \alpha q/\sqrt{2\pi}$ for some $0 < \alpha = \alpha(n) < 1$. For the other LWE parameters we usually have $q \leq \mathrm{poly}(n)$ and $n \leq m \leq \mathrm{poly}(n)$.

Lemma 2.52. *If the modulus q is prime, then the search and the decision versions of LWE are equivalent.*

Proof. It is obvious that an algorithm that can solve the search version, can also solve the decision version of LWE. For the other direction, consider for any $k \in \mathbb{Z}_q$ the transformation on a given sample according to $\varphi_k(a, b) \coloneqq (a + (l, 0, \ldots, 0), b + l \cdot k \mod q)$ for some $l \in \mathbb{Z}_q$ chosen uniformly at random. We assume that we have access to a polynomial-time algorithm $\mathcal{S}$ that solves the decision version of LWE with probability exponentially close to 1. If the original samples are uniformly distributed on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, then so are the transformed ones. However, if the samples are drawn from the LWE-distribution $A_{s,\chi}$, then the transformed samples are distributed according to $A_{s,\chi}$ if $k = s_1$ and uniformly otherwise. For the last statement we require that q is a prime number. Thus, we can use a solver for the decision version of LWE, i.e. a distinguisher between the uniform and the LWE-distribution, to determine the first coordinate of the secret s by trying all possible values for $k \in \mathbb{Z}_q$. Since $q \leq \text{poly}(n)$, this can be done in polynomial-time. We can find the other coordinates of s in a similar way and thus we can solve the search version with only a polynomial number of calls to $\mathcal{S}$. □

Remark 2.53. There is also a worst-case to average-case reduction of the decision version of LWE. The proof can be found in [Reg09] along with more variants of the LWE problem.

Remark 2.54. We remark here that the search-version of LWE can be seen as an instance of BDD on the lattice

$$\mathcal{L}(A) = \{A^T s : s \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m, \tag{2.2}$$

where A is the $n \times m$-matrix whose columns are the m LWE samples $a_i \in \mathbb{Z}_q^n$. This gives us $b = A^T s + e$, where b and e are the vectors consisting of the b_i or e_i from the samples, respectively. This connection of LWE and BDD is very important to us and will be further discussed in Section 6.1. Another important property of the LWE problem is that it is at least as hard as the lattice problems GapSVP and SIVP as was shown by Regev in [Reg09, Thm. 3.1], who gave a (quantum) reduction from GapSVP and SIVP to LWE.

Remark 2.55. The connection between LWE and GapSVP or SIVP, respectively, was established using the discrete Gaussian sampling problem (DGS), which is defined as follows: Given a basis B of an n-dimensional lattice $\mathcal{L}$, some real-valued function on lattices φ and a number $r > \varphi(\mathcal{L})$, the *discrete Gaussian sampling problem* (DGS_φ) is to output a sample from $D_{\mathcal{L},r}$.

2.4 The Dihedral Group

In Section 3 we will discuss the hidden subgroup problem (HSP) for the finite abelian case. Because of the great importance the HSP seems to have when it comes to quantum algorithms, which provide a superpolynomial speed-up over classical algorithms, it is not surprising that the HSP on non-abelian groups is also of great interest. Unfortunately,

until today there are no known efficient quantum algorithms for solving the HSP on non-abelian groups. However, it is known that a solution for the *symmetric group*, i.e. the group consisting of the all permutations of a fixed, finite number of (labelled) elements, leads to a solution of the graph isomorphism problem. Another non-abelian group, for which the HSP has a certain application is the *dihedral group*, which is the symmetry group of the regular N-sided polygon for some number $N \in \mathbb{N}$, which we will denote by D_N. Regev showed ([Reg04a, Thm. 1.1]) that a solution to the *dihedral coset problem* (DCP), which is essentially the hidden subgroup problem over the dihedral group, would lead to a solution of $\Theta(n^{1/2+2f})$-uSVP where f is some failure parameter in the description of DCP. We now give a brief overview of the dihedral group, DCP as well as the connection of the latter to DHSP.

Definition 2.56 (The dihedral group, first definition)**.** For $N \in \mathbb{N}$ we define the *dihedral group* to be the abstract group generated by an element r of order N and an element s of order 2 subject to the relation $rs = sr^{-1}$. That is,

$$D_N \coloneqq \{s^i r^j : r^N = s^2 = srsr = \mathrm{id},\ 1 \leq j \leq N,\ i \in \{1, 2\}\}.$$

If we interpret r^j as a rotation and sr^j as a reflection, we see that D_N can indeed be viewed as the symmetry group of the regular N-sided polygon.

Another way of defining the dihedral group is via the *semidirect product* of groups.

Definition 2.57 (The dihedral group, second definition)**.** For $N \in \mathbb{N}$ consider the group homomorphism $\varphi\colon \mathbb{Z}/2\mathbb{Z} \to \mathrm{Aut}(\mathbb{Z}/N\mathbb{Z}), 1 \mapsto -\mathrm{id}$. Where $\mathrm{Aut}(\mathbb{Z}/N\mathbb{Z})$ denotes the group of all automorphisms of $\mathbb{Z}/N\mathbb{Z}$. Then the *dihedral group* is defined as the (outer) semidirect product

$$D_N \coloneqq \mathbb{Z}/2\mathbb{Z} \ltimes_\varphi \mathbb{Z}/N\mathbb{Z}.$$

Following the second of these equivalent definitions of D_N, we will denote elements as $(s, r) \in \mathbb{Z}/2\mathbb{Z} \ltimes \mathbb{Z}/N\mathbb{Z}$, where the group operation is given by

$$(s_1, r_1)(s_2, r_2) = (s_1 + s_2, r_1 + \varphi(s_1)r_2) = (s_1 + s_2, r_1 + (-1)^{s_1} r_2).$$

Proposition 2.58 (Subgroups of the dihedral group)**.** *Every subgroup of D_N is either generated by one element r^d for some $d \mid N$ or by two elements $r^d, r^j s$ for some $d \mid N$ and $0 \leq j \leq d - 1$.*

Proof. See [Con09, Thm. 3.1]. □

Definition 2.59 (Dihedral coset problem (DCP_f))**.** Let $N \in \mathbb{N}$, $d \in \{0, \ldots, N-1\}$ some fixed value and f a *failure parameter*. Given a polynomial (in $\log N$) number of $1 + \lceil \log N \rceil$ qubit registers as input, each in the state

$$\frac{1}{\sqrt{2}}(|0, x\rangle + |1, x + d \mod N\rangle),$$

with probability of at least $1 - 1/(\log N)^f$ and otherwise in an arbitrary state $|b, x\rangle$ with $b \in \{0, 1\}$ and $x \in \{0, \ldots, N-1\}$. The *dihedral coset problem* (DCP_f) with failure parameter f is to find the value d.

Remark 2.60. The 'good' input states in the definition above can be interpreted as the uniform superposition of a coset of the subgroup $\{(0,0),(1,d)\} \subset D_N$. It is well-known that the DHSP can be reduced to the case where the hidden subgroup is of the above form. Let H denote the hidden subgroup of G. We note that if $H \neq \langle sr^t \rangle$ for some $1 \leq t \leq N$, then $H' \coloneqq H \cap \{0\} \times \mathbb{Z}/N\mathbb{Z} \neq \{e\}$. Here, e denotes the neutral element of D_N. We can find the subgroup $H' \subset \{0\} \times \mathbb{Z}/N\mathbb{Z}$ efficiently since this is just an instance of the abelian hidden subgroup problem. Hence we can compute the quotient group H/H', which is either trivial or a reflection in G/H'.

Lemma 2.61. *If there is an efficient solution to the dihedral hidden subgroup problem that samples cosets, then there exists a quantum algorithm that solves the dihedral coset problem with some failure parameter f.*

Proof. By assumption, there is a solution to the dihedral hidden subgroup problem. Let $\mathfrak{I}_{\mathrm{DHSP}}$ denote the number of coset samples that the DHSP algorithm requires as input. Then the probability, that all input registers of the DCP instance are 'good', is at least

$$\left(\frac{(\log N)^f - 1}{(\log N)^f}\right)^{\mathfrak{I}_{\mathrm{DHSP}}} \xrightarrow{f\to\infty} 1.$$

So for f large enough, we have that with high probability every input register is 'good'. Hence the algorithm that solves DHSP also solves DCP with failure parameter f. □

Remark 2.62. Unfortunately, this reduction does not provide any information on the required size of the failure parameter for DCP in order to make use of an algorithm that solves DHSP. It would depend on the actual application to choose a minimal probability for all DCP input registers to be 'good', which would then lead to a failure parameter f.

3 The Hidden Subgroup Problem (HSP)

In this chapter we will present one important problem in current research, the *hidden subgroup problem* (HSP). The reason why this problem is of such importance is that many of the currently known quantum algorithms that provide an exponential speedup over classical algorithms can be seen as a solver for a special case of the hidden subgroup problem. While it is possible to solve the HSP (efficiently) for any abelian group, there is no known solution of polynomial complexity in the general case. We will begin by formulating the HSP and giving some background information on representation theory, which we will need to present an efficient quantum algorithm that solves the HSP in the (finite) abelian case.

Definition 3.1 (The hidden subgroup problem (HSP))**.** Given a group G, a subgroup $H \subseteq G$, a set X and a function $f\colon G \to X$ such that

$$\forall g_1, g_2 \in G:\ f(g_1) = f(g_2) \iff g_1 H = g_2 H,$$

i.e. the function f is constant on every (left) coset of H in G and takes different values on different cosets, the hidden subgroup problem is to determine a generating set for the subgroup H using evaluations of f. Such a function is said to *separate cosets* of H.

3.1 Solving the Hidden Subgroup Problem in the Abelian Case

In the following we will need some results from representation theory (of finite groups) as well as some character theory. All results are basically standard, for the omitted proofs we refer to [FH13].

Definition 3.2. Let G be a finite group, V a vector space over $\mathbb{C}$ of dimension $d < \infty$ and $\mathrm{GL}(V)$ its group of automorphisms. A representation of G is a group homomorphism $\rho\colon G \to \mathrm{GL}(V)$. The dimension of ρ is $\dim(V) = d$. A representation ρ of G is called *irreducible* if V has no non-trivial linear subspace which is G-invariant, i.e. for any linear subspace $U \subseteq V$ with $\rho_g(U) \subseteq U$, for all $g \in G$ we have $U = \{0\}$ or $U = V$, where $g \overset{\rho}{\mapsto} \rho(g) =: \rho_g$.

One can show that there are only finitely many pairwise non-isomorphic irreducible representations for every finite group, say n, and that $\sum_{i=1}^{n} d_i^2 = |G|$, where d_i denotes the

dimension of the i-th irreducible representation. One useful theorem regarding irreducible representations is the following:

Theorem 3.3 (Schur's Lemma)**.** *Let G be a finite group, V and W $\mathbb{C}$-vector spaces, $\rho\colon G \to \mathrm{GL}(V)$, $\sigma\colon G \to \mathrm{GL}(W)$ representations of G and $f \in \mathrm{Hom}_{\mathbb{C}}(V, W)$ such that $f \circ \rho_g = \sigma_g \circ f$, for all $g \in G$. Then $f \equiv 0$ or f is bijective (this means that ρ and σ are isomorphic). Also, for $\rho = \sigma$ we have $f = \lambda \cdot \mathrm{id}$ for some $\lambda \in \mathbb{C}$.*

Another important aspect when it comes to representations of a group, are *characters*.

Definition 3.4. Let G be any group. Every group homomorphism $\chi\colon G \to \mathbb{C}^\star$ is called a (multiplicative) character.

Definition 3.5. Let ρ be a representation of a group G. The *character* χ_ρ of the representation ρ is the group homomorphism $\chi_\rho\colon G \to \mathbb{C}^\star$, defined by

$$\chi_\rho(g) \coloneqq \mathrm{tr}(\rho_g)$$

As already mentioned, we want to solve the HSP for abelian groups. The irreducible representations of finite abelian groups turn out to be quite 'simple'.

Lemma 3.6. *Let G be a finite abelian group and $\rho\colon G \to \mathrm{GL}(V)$ an irreducible representation, then $\dim(V) = 1$.*

Proof. For $g \in G$ and for all $h \in G$ we have

$$\rho_h \circ \rho_g = \rho_{hg} = \rho_{gh} = \rho_g \circ \rho_h$$

because G is abelian. Hence, by Schur's Lemma $\rho_g(v) = \lambda v$ holds for all $g \in G, v \in V$ and some $\lambda \in \mathbb{C}$. So every linear subspace of V is G-invariant, and because ρ is irreducible, it must be $\dim(V) = 1$. □

This concludes our brief excursion into representation theory. From now on, let G denote a *finite abelian* group for which we will use additive notation. It is well known that we have

$$G \cong \bigoplus_{i=1}^{r} \mathbb{Z}/m_i\mathbb{Z}$$

for some $r, m_1, \ldots, m_r \in \mathbb{N}$. Without loss of generality, we may thus view elements of G as vectors of length r with the i-th entry from the set $\{0, \ldots, m_i - 1\}$. We will also assume that we know the numbers m_i, which is not trivial, but Cheung and Mosca have given an efficient quantum algorithm for that purpose in [CM01] (see also [Lom04, Thm. 5.23]). Our first goal is to generalise the quantum Fourier transform we have seen earlier to arbitrary finite abelian groups. The QFT that we already know will turn out to be the Fourier transform over the cyclic group of order $N = 2^n$.

We will now prove a nice relation between finite abelian groups and their characters.

Proposition 3.7. *Let $\widehat{G}$ denote the set of all characters of the finite abelian group G. Then $\widehat{G}$ is a group and $G \cong \widehat{G}$.*

Proof. At first we prove the statement for cyclic groups. Because G is a finite group, we have $|\chi(g)| = 1$ for all $g \in G$ and all characters χ. Moreover, if $C = \langle c \rangle$ is a cyclic group with generator c, it suffices to specify the value $\chi(c)$, which must be a $|C|$-th root of unity, and vice versa, every choice of a $|C|$-th root of unity defines a different character. This proves the statement for the cyclic case. Now consider an arbitrary finite abelian group G. We have

$$G \cong \bigoplus_{i=1}^{r} \mathbb{Z}/m_i\mathbb{Z}$$

so we regard any $g \in G$ as a vector of length r with integer entries in the range of $\{0, \ldots, m_i - 1\}$ in the i-th component. Let e_i be the vector which is one in the i-th component and zero otherwise for every $i \in \{1, \ldots, r\}$. Every character χ of G is completely determined by its values on the set $\{e_i\}$, because χ is a group homomorphism. In addition $\operatorname{ord}(\chi(e_i)) \mid m_i$, so we can write $\chi(e_i) = \omega_{m_i}^{h_i}$ for $\omega_{m_i} = \exp(2\pi i/m_i)$, a primitive m_i-th root of unity, and some integer $h_i \leq m_i$ for all $i \in \{1, \ldots, r\}$. Hence, we have a one-to-one correspondence between a character and a group element, namely $(h_1, \ldots, h_r)$. So

$$\varphi \colon G \to \widehat{G}, \quad h \mapsto \left(g \mapsto \chi_h(g) \coloneqq \prod_{i=1}^{r} \omega_{m_i}^{h_i g_i} \right) \tag{3.1}$$

is a bijection. From this we also see that $\widehat{G}$ forms a group with group operation defined via $\chi_{g_1}\chi_{g_2} \coloneqq \chi_{g_1+g_2}$. Moreover, $\varphi(0, \ldots, 0) = \mathrm{id}$ and $\varphi(g_1 + g_2) = \varphi(g_1)\varphi(g_2)$, so $G \cong \widehat{G}$ as required. □

Proposition 3.8. *Let G be a finite abelian group, $\chi \in \widehat{G}$ a character and $\chi_0 \equiv 1$ the identity character, then*

$$\sum_{g \in G} \chi(g) = \begin{cases} |G|, & \text{if } \chi = \chi_0, \\ 0, & \text{otherwise.} \end{cases}$$

Definition 3.9 (General QFT over finite abelian groups)**.** Let G be a finite abelian group and $\chi_g(h) \coloneqq \prod_{i=1}^{r} \omega_{m_i}^{h_i g_i}$ as in Equation (3.1). Then

$$\mathcal{F}_G \coloneqq \frac{1}{\sqrt{|G|}} \sum_{g,h \in G} \chi_g(h) \, |g\rangle \langle h| \tag{3.2}$$

is called the *quantum Fourier transform over the group G.*

Remark 3.10. Note that our previous definition in Equation (2.1) is just the special case of $G = \mathbb{Z}/N\mathbb{Z}$ with $N = 2^n$. So actually, we could have called it the cyclic quantum

Fourier transform. However, generally we omit that specification. We also note that

$$|0\rangle \overset{\mathcal{F}_G}{\longmapsto} \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \,.$$

So $\mathcal{F}_G$ creates a uniform superposition over all group elements. It is possible to generalise the Fourier transform even further and define it for every finite group. This can be achieved by

$$\hat{f}(\rho) = \sqrt{\frac{d_\rho}{N}} \sum_{g \in G} f(g)\rho(g),$$

where G is some finite group of order N, $f \colon G \to \mathbb{C}$ is any function and ρ is an irreducible representation of G with dimension d_ρ. Since we have seen in Lemma 3.6 that every representation of a finite abelian group is one-dimensional, this is compatible with Equation (3.2). For a more detailed approach on groups and their representations with regard to the hidden subgroup problem, see for example [NC00, Appendix 2].

Remark 3.11. Regarding complexity, note that $\mathcal{F}_G = \bigotimes_{i=1}^{r} \mathcal{F}_{\mathbb{Z}/m_i\mathbb{Z}}$, so $\mathcal{F}_G$ can be implemented efficiently because the quantum Fourier transform for cyclic groups can be implemented efficiently.

Next we will examine how the quantum Fourier transform acts on a superposition of states which all belong to a subgroup H of G since our goal is still to solve the hidden subgroup problem. In this regard, the so-called *orthogonal subgroup* will come in handy.

Definition 3.12. Let $H \subseteq G$ be a subgroup. Then the orthogonal subgroup is defined as

$$H^\perp := \{g \in G : \chi_g(h) = 1 \ \forall h \in H\}.$$

Regarding the orthogonal subgroup, one can show that

Theorem 3.13. *With the notation introduced above we have*

(i) $H^\perp$ *is a subgroup of* G,

(ii) $G/H \cong H^\perp$,

(iii) $(H^\perp)^\perp = H$.

Proof. See [Lom04, Thm. 3.1] □

The idea is now to give an algorithm that outputs a generating set for $H^\perp$ with high probability, and then use it to recover the original hidden subgroup H.

Lemma 3.14. *Let $|H\rangle$ and $|H^\perp\rangle$ denote the uniform superpositions over the subgroups $H, H^\perp \subset G$ respectively. Then*

$$|H\rangle \overset{\mathcal{F}_G}{\longmapsto} |H^\perp\rangle .$$

Proof. We observe that

$$\begin{aligned} |H\rangle = \frac{1}{\sqrt{|H|}} \sum_{h\in H} |h\rangle \overset{\mathcal{F}_G}{\longmapsto} &= \frac{1}{\sqrt{|G|}} \sum_{g,h'\in G} \chi_g(h') \, |g\rangle \langle h'| \left(\frac{1}{\sqrt{|H|}} \sum_{h\in H} |h\rangle \right) \\ &= \frac{1}{\sqrt{|G||H|}} \sum_{\substack{g\in G \\ h\in H}} \chi_g(h) \, |g\rangle \\ &= \frac{1}{\sqrt{|G||H|}} \sum_{g\in G} \left(\sum_{h\in H} \chi_g(h) \right) |g\rangle . \end{aligned}$$

The coefficient of $|g\rangle$ is $|H|$ if $g \in H^\perp$, and 0 otherwise by Proposition 3.8. Hence,

$$|H\rangle \overset{\mathcal{F}_G}{\longmapsto} \sqrt{\frac{|H|}{|G|}} \sum_{g\in H^\perp} |g\rangle = \frac{1}{\sqrt{|H^\perp|}} \sum_{g\in H^\perp} |g\rangle = |H^\perp\rangle .$$

Here we used $|H^\perp| = |G|/|H|$. This equation holds because of $G/H \cong H^\perp$ and we have proved the statement. □

Now we can finally present the algorithm that essentially solves the hidden subgroup problem in the finite abelian case. We will be working with two registers. In the first we will store the group elements of G, and in the second we will compute the function f which is given to us as an oracle.

Algorithm 1 Algorithm for the abelian hidden subgroup problem

INPUT: Two qubit registers initially in the state $|0\rangle\,|0\rangle$, an abelian group G and a black box oracle that computes a function f that separates cosets of a subgroup $H \subset G$.
OUTPUT: A uniformly distributed random element of $H^\perp$.

1: Apply the Fourier transform $\mathcal{F}_G$ to the first register. Afterwards compute f in the second register.
2: Perform a measurement of the second register.
3: Apply the Fourier transform once again to the first register.
4: Measure the first register and output the outcome.

Let us take a closer look at Algorithm 1. The initial state of the registers is $|0\rangle\,|0\rangle$ and the algorithm performs the following actions on them:

$$|0\rangle\,|0\rangle \overset{\mathcal{F}_G,f}{\longmapsto} \frac{1}{\sqrt{|G|}} \sum_{g\in G} |g\rangle\,|f(g)\rangle\,.$$

Say the outcome of the measurement is $f(t)$ for some $t \in G$, then the state collapses to

$$\frac{1}{\sqrt{|H|}} \sum_{h\in H} |t+h\rangle\,|f(t)\rangle\,.$$

Here we used that the function f separates cosets of the subgroup H. The content of the second register is not important for the next steps. So from now on, we omit it. Step 3 yields

$$\begin{aligned}\frac{1}{\sqrt{|H|}} \sum_{h\in H} |t+h\rangle \overset{\mathcal{F}_G}{\longmapsto} \quad & \frac{1}{\sqrt{|H|}} \sum_{h\in H} \frac{1}{\sqrt{|G|}} \sum_{g,h'\in G} \chi_g(h')\,|g\rangle\,\langle h'|t+h\rangle \\ &= \frac{1}{\sqrt{|H||G|}} \sum_{\substack{g\in G\\ h\in H}} \chi_g(t+h)\,|g\rangle \\ &= \frac{1}{\sqrt{|H^\perp|}} \sum_{g\in H^\perp} \chi_g(t)\,|g\rangle\,,\end{aligned}$$

where the last equality is obtained by a calculation, similar to the one we did in the previous lemma. At last we measure the first register and obtain a uniformly distributed random element of $H^\perp$ (recall that $|\chi_g(t)| = 1$).

This concludes the description of the algorithm. What is left to show is that, if we run this algorithm a polynomial number of times, we obtain a generating set for $H^\perp$, from which we have to reconstruct the original subgroup H.

The first problem is taken care of by the following proposition. For a proof see [Lom04, Thm. D.1].

Proposition 3.15. *Let G be a finite group, $k \in \mathbb{N}$ and $g_1, \ldots, g_{k+\log\lceil|G|\rceil}$ uniformly sampled elements of G, then*

$$\mathrm{Prob}[\langle g_1, \ldots, g_{k+\log\lceil|G|\rceil}\rangle = G] \geq 1 - \frac{1}{2^k}.$$

The second problem can be overcome by solving a system of modular linear equations. For a detailed discussion, see once again [Lom04, p. 23].

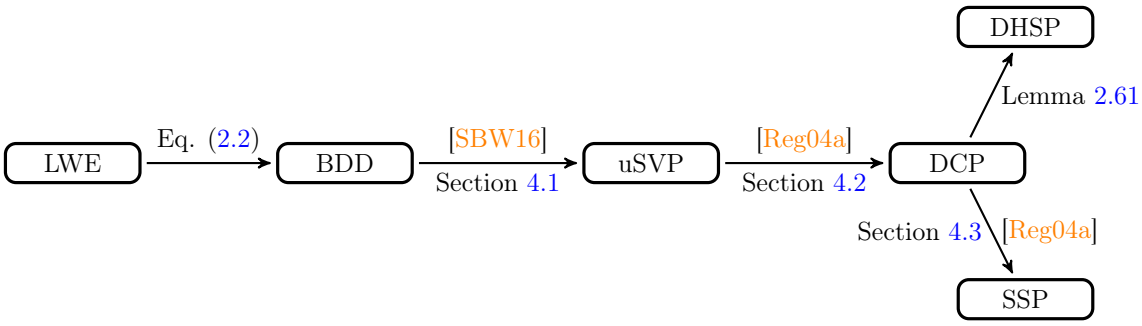
4 Some Reductions of Lattice Problems

In this chapter we want to present several reductions from lattice problems onto one another. The most important ones being

1. a reduction from the unique shortest vector problem (uSVP) to the dihedral coset problem (DCP), and therefore (see Lemma 2.61) also to the dihedral hidden subgroup problem (DHSP), and

2. a reduction from the dihedral coset problem to the *subset sum problem* (SSP) in Section 4.3.

As we have already mentioned, we can view instances of learning with errors (LWE) as instances of the bounded distance decoding problem (BDD) over certain lattices (see Remark 2.54 or later in more detail Section 6.1) which are constructed by a matrix consisting of LWE samples. Because there also exists a reduction from BDD to uSVP (see Section 4.1), we will have two different ways of solving LWE instances via algorithms for the dihedral hidden subgroup problem and the subset sum problem. The roadmap for this section is illustrated in the following diagram, where reductions are represented by arrows. The approximation factors of the corresponding problems are omitted for convenience, but will be specified in the respective sections.



4.1 A Reduction from BDD to uSVP

As already mentioned in Section 2.3.3, we can view an instance of (search) LWE as a BDD instance on the lattice defined in Equation (2.2). The best known BDD-to-uSVP reduction is currently a probabilistic polynomial-time reduction from $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$ where $\gamma > 1$, is polynomial in the lattice dimension n, and is due to Bai et al. ([SBW16]). The actual reduction is from $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$, but that is a mere technical detail (see e.g. [LM09] Section 3). Much the same as prior reductions from BDD to uSVP instances, an approach called *Kannan's embedding technique* is used to construct a uSVP instance of an $(n+1)$-dimensional lattice $\mathcal{L}'$ defined by the matrix

$$B' = \begin{pmatrix} B & t \\ 0 & kd_0 \end{pmatrix},$$

where B is the basis (matrix) of an n-dimensional lattice and t is the target vector of the $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ instance. The number d_0 is an approximation of $\mathrm{dist}(t, \mathcal{L}(B)) \coloneqq d$ satisfying $d_0 \in [d, d/(1-1/n)]$, which can be efficiently computed ([Bab86] or Remark 4.3). In the reduction of Bai et al. the factor k was chosen to be $1/(n-1)$. The key to the improved reduction is to choose the matrix B not just as the basis of the given BDD-lattice, as it was done in older reductions, but rather the basis of a sparsified sublattice, which still contains a closest vector to t, but no other close-by vectors. Ultimately, this results in an increased uSVP-gap $\lambda_2(\mathcal{L}')/\lambda_1(\mathcal{L}')$ and thus an improved reduction.

Let us begin by defining the lattice sparsification.

Definition 4.1. Let B be the basis of an n-dimensional BDD-lattice $\mathcal{L}$, p a positive prime integer and $z \in \mathbb{Z}_p^n$. Then we define the sparsification of $\mathcal{L}$ with respect to p and z as

$$\mathcal{L}_{p,z} \coloneqq \{x \in \mathcal{L} : \langle z, B^{-1}x \rangle = 0 \mod p\}.$$

In other words, the sparsified lattice is the set of all lattice points whose coordinate vectors, with respect to the basis B, are orthogonal to z over $\mathbb{Z}_p$.

We can efficiently compute a basis of the sparsified lattice as the following lemma shows.

Lemma 4.2. *There exists a polynomial-time algorithm which computes a basis $B_{p,z}$ of $\mathcal{L}_{p,z}$ when given a basis B of $\mathcal{L}$ as well as p and z.*

Proof. Let $b_1, \ldots, b_n$ denote the column vectors of B and $\bar{b}_1, \ldots, \bar{b}_n$ the column vectors of $B^{-T} = (B^T)^{-1}$. Note that we have $(\bar{b}_i)^T b_j = \delta_{ij}$. In the following we describe an algorithm that computes a basis of $\mathcal{L}_{p,z}$. If $z = 0$, the algorithm simply outputs $B_{p,z}$. If $z \neq 0$, the algorithm does the following. Assuming without loss of generality $z_n \neq 0$, the

algorithm computes B^{-T} and sets

$$\widehat{B} \coloneqq \left(\bar{b}_1, \ldots, \bar{b}_{n-1}, \tfrac{1}{p}\sum_{i=1}^{n} z_i \bar{b}_i\right).$$

Afterwards it computes and outputs $B_{p,z} = \widehat{B}^{-T}$. Since B^{-T} has full rank, so does $\widehat{B}$. To see that $\widehat{B}^{-T}$ is a basis of $\mathcal{L}_{p,z}$, consider $x = \widehat{B}^{-T} y$ for some $y \in \mathbb{Z}^n$. We have

$$\widehat{B}^T B = \begin{pmatrix} \bar{b}_1 \\ \vdots \\ \bar{b}_{n-1} \\ \frac{1}{p}\sum_{i=1}^{n} z_i \bar{b}_i \end{pmatrix} (b_1, \ldots, b_n) = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{p}z_1 & & \ldots & & \frac{1}{p}z_n \end{pmatrix}.$$

Computing $(\widehat{B}^T B)^{-1}$ yields

$$(\widehat{B}^T B)^{-1} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{z_1}{z_n} & & \ldots & -\frac{z_{n-1}}{z_n} & \frac{p}{z_n} \end{pmatrix},$$

which gives us

$$B^{-1}x = B^{-1}\widehat{B}^{-T}y = (\widehat{B}^T B)^{-1} y = \begin{pmatrix} y_1 \\ \vdots \\ y_{n-1} \\ -\sum_{i=1}^{n-1} \frac{z_i y_i}{z_n} + \frac{p y_n}{z_n} \end{pmatrix} =: v$$

and finally

$$\langle z, v \rangle = p y_n \equiv 0 \mod p.$$

We conclude the proof by remarking that the algorithm obviously runs in polynomial-time since the essential step is the computation of the inverse of a matrix. □

We will now present the algorithm that reduces $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$. The input is a basis B of the BDD lattice, a target vector t as well as two numbers d_0, l_0 whose roles we will discuss afterwards.

Remark 4.3. We will prove that the above algorithm works correctly if the input parameters d_0, l_0 satisfy $d_0 \in \mathrm{dist}(t, \mathcal{L}) \cdot [1, 1/(1-1/n))$ and $l_0 \in \lambda_1(\mathcal{L}) \cdot [1, 1/(1-1/n))$. Of course, we have to make sure that the reduction is called with such suitable values for l_0 and d_0. To do this, we first note that there exist polynomial-time algorithms that find $d_1 \in \mathrm{dist}(t, \mathcal{L}) \cdot [1, 2^{n/2})$ and $l_1 \in \lambda_1(\mathcal{L}) \cdot [1, 2^{n/2})$. For l_1, this follows from Proposition 2.33, and for d_1 this is due to [Bab86, Thm. 3.1].

Algorithm 2 $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$ reduction algorithm

INPUT: A basis B of an n-dimensional $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ lattice $\mathcal{L}$, a target vector t and two numbers d_0, l_0.
OUTPUT: A lattice vector $c \in \mathcal{L}$ with $\|c-t\| = \mathrm{dist}(t, \mathcal{L})$.

1: Compute the smallest prime p greater than $4\gamma n^2$.
2: Sample $z, u \in \mathbb{Z}_p^n$ uniformly at random and compute $w = B\bar{u} \in \mathcal{L}$ such that $u \equiv \bar{u} \mod p$ as well as $\|t + w\| \geq (n+1)l_0/\sqrt{2}$.
3: Compute a basis $B_{p,z}$ of the sparsified lattice $\mathcal{L}_{p,z}$.
4: Set $k = 1/(n-1)$ and define the $n+1$-dimensional lattice $\Lambda(B')$ via

$$B' = \begin{pmatrix} B_{p,z} & t+w \\ 0 & kd_0 \end{pmatrix}.$$

5: Run the $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$ solver with input B'. Let $s' = (s'_1, \ldots, s'_{n+1})$ be its output. Then output $(s'_1, \ldots, s'_n) + t$.

Let $x_i = (\frac{n}{n-1})^{i-1}$ for $i \in \mathbb{N}$. A simple calculation shows that

$$x_i = 2^{\frac{n}{2}} \Leftrightarrow i = \frac{n}{2\log\left(\frac{n}{n-1}\right)} + 1 =: C = \mathcal{O}(n^2).$$

Since this is polynomial in n, we can cover the interval $[1, 2^{n/2})$ by the polynomially sized set $\{x_i \cdot [1, 1/(1-1/n)) : 1 \leq i \leq \lceil C \rceil\}$. So calling the above algorithm with all values in the sets $\{d_0 x_i : 1 \leq i \leq \lceil C \rceil\}$ and $\{l_0 x_i : 1 \leq i \leq \lceil C \rceil\}$ for the parameters d_0 and l_0, respectively, and keeping a best solution among the returned ones assures that the presented algorithm will work correctly, once we have proved the correctness with the assumptions on d_0 and l_0.

We continue by analysing the algorithm step by step, concerning running time and correctness. In the first step, we calculate the smallest prime greater than $4\gamma n^2$. By the Bertrand-Chebyshev Theorem (for a proof see [AZQ10, Chapter 2]) we know that there is a prime p with $4\gamma n^2 < p < 8\gamma n^2 - 2$. So we can just test these (polynomially many) numbers, starting with $\lfloor 4\gamma n^2 + 1 \rfloor$, on primality using, for example the AKS primality test, which runs in time $\widetilde{\mathcal{O}}(\log(n)^6)$ ([LJP02]). One might wonder, why we compute the rather arbitrary looking vector w in the first step. We will address this issue right now by explaining the idea behind the lattice sparsification and why it increases the uSVP-gap.

The idea behind the sparsification is to remove all but one of the vectors inside a ball of radius $r = \lambda_1(\mathcal{L})/\sqrt{2}$ around the target vector t. This is done in two steps. The first one is to show that the number of lattice points in the ball of radius r around t, denoted by $\mathcal{B}(t, r)$, is polynomial in n (Lemma 4.4). In a second step we show that, with non-negligible probability, there is at most one vector in every ball around the multiples

of the shifted target vector that is kept in the sparsified lattice (Proposition 4.8). The shift is the vector w in the algorithm above. It is chosen at random since $u \in \mathbb{Z}_p^n$ is chosen uniformly at random, but we make sure that $t + w$ is a sufficiently long vector.

Lemma 4.4. *Let $r = \lambda_1(\mathcal{L})/\sqrt{2}$. For any n-dimensional lattice $\mathcal{L}$ and any target vector $t \in \mathbb{Q}^n$, we have*

$$\#(\mathcal{L} \cap \mathcal{B}(t, r)) \leq 2n.$$

Proof. If $n = 1$, then the statement is true since in any set of at least three distinct lattice vectors, two of them have distance at least $2\lambda_1(\mathcal{L}) > 2r$. Now assume that the statement is true for some $n \in \mathbb{N}$ and let $y_1, \ldots, y_N \in \mathbb{R}^{n+1}$ be distinct lattice vectors such that $\|y_i - t\| \leq \lambda_1(\mathcal{L})/\sqrt{2}$ for all i, and define $x_i \coloneqq y_i - t$. Assume without loss of generality that $x_N \neq 0$. We have to show that $N \leq 2(n+1)$. To do so, we consider the maps

$$x_i' \coloneqq f(x_i) \coloneqq \begin{cases} x_i, & \text{if } \langle x_i, x_N \rangle = \pm\|x_i\|\|x_N\|, \\ x_i - \frac{\langle x_i, x_N \rangle}{\|x_N\|^2} x_N, & \text{otherwise,} \end{cases}$$

and

$$x_i'' \coloneqq g(x_i') \coloneqq \begin{cases} 0, & \text{if } x_i' = 0, \\ \frac{r}{\|x_i'\|} x_i', & \text{otherwise.} \end{cases}$$

Note that $g(x_i') = g(x_j')$ if and only if $x_i' = k \cdot x_j'$ for some $k > 0$, and for every $x_i \neq 0$ we have $\|x_i''\| = r$. Since we want to use induction over n, the idea is to use the map $h \coloneqq g \circ f$ to adjust the x_i on an $(n+1)$-dimensional sphere with radius r and north pole $h(x_N) = x_N''$, such that all but at most two points get mapped to the n-dimensional equator, i.e. the subspace orthogonal to x_N''. Then by induction hypothesis we have $N \leq 2 + 2n$. We first observe that

$$2\langle x_i, x_j \rangle = \|x_i\|^2 + \|x_j\|^2 - \|x_i - x_j\|^2 \leq \frac{\lambda_1(\mathcal{L})^2}{2} + \frac{\lambda_1(\mathcal{L})^2}{2} - \lambda_1(\mathcal{L})^2 = 0.$$

holds for all $i \neq j$. Now, we will consider three cases.

(1) If $\langle x_i, x_N \rangle = \pm\|x_i\|\|x_N\|$ and $\langle x_j, x_N \rangle = \pm\|x_j\|\|x_N\|$, i.e. x_i and x_j are both collinear to x_N, then $\langle x_i', x_j' \rangle = \langle x_i, x_j \rangle$.

(2) If $\langle x_i, x_N \rangle \neq \pm\|x_i\|\|x_N\|$ and $\langle x_j, x_N \rangle = \pm\|x_j\|\|x_N\|$ or vice versa, we obtain that one of x_i' or x_j' – in this case x_i' – is orthogonal to x_N and the other is collinear to x_N.

(3) In the case, where $\langle x_i, x_N \rangle \neq \pm\|x_i\|\|x_N\|$ and $\langle x_j, x_N \rangle \neq \pm\|x_j\|\|x_N\|$, we assume that there is some $k > 0$ with $x_i' = k \cdot x_j'$. We can write $x_j = x_i + (1/k) \cdot v$ for some vector $v \in \mathbb{R}^{n+1}$ that, without loss of generality, satisfies $\langle v, x_N \rangle \leq 0$, since if the last inequality does not hold, we interchange x_i and x_j. By the definition of x_i' and

x'_j we obtain

$$x_i - \frac{\langle x_i, x_N\rangle}{\|x_N\|^2} x_N = k \cdot \left(x_i + \frac{1}{k} v - \frac{\langle x_i + \frac{1}{k} v, x_N\rangle}{\|x_N\|^2} x_N \right),$$

thus

$$v = (1-k)\left(x_i - \frac{\langle x_i, x_N\rangle}{\|x_N\|^2} x_N \right) + \frac{\langle v, x_N\rangle}{\|x_N\|^2} x_N$$

and therefore

$$\begin{aligned}
\langle x_i, x_j\rangle &= \left\langle x_i, x_i + \frac{1-k}{k} \cdot \left(x_i - \frac{\langle x_i, x_N\rangle}{\|x_N\|^2} x_N \right) + \frac{1}{k} \frac{\langle v, x_N\rangle}{\|x_N\|^2} x_N \right\rangle \\
&= \|x_i\|^2 + \frac{1-k}{k} \cdot \left(\|x_i\|^2 - \frac{\langle x_i, x_N\rangle^2}{\|x_N\|^2} \right) + \frac{1}{k} \frac{\langle v, x_N\rangle}{\|x_N\|^2} \langle x_i, x_N\rangle \\
&\geq \frac{1}{k}\|x_i\|^2 + \frac{1}{k} \frac{\langle v, x_N\rangle}{\|x_N\|^2} \langle x_i, x_N\rangle \\
&\geq \frac{1}{k}\|x_i\|^2 > 0.
\end{aligned}$$

This is a contradiction to $\langle x_i, x_j\rangle \leq 0$. Here the first inequality holds if and only if $k > 1$ but since k is positive by assumption we may as well without loss of generality assume $k > 1$, since either $k > 1$ or $1/k > 1$.

These observations together with the fact that there can only be at most two collinear lattice vectors inside a ball of radius $r = \lambda_1(\mathcal{L})/\sqrt{2}$ (analogously to the case $n = 1$) show, that there is a one-to-one correspondence between the x_i and the x''_i. Since at least $N-2$ points get mapped to the equator, i.e. $\langle x''_i, x''_N\rangle = 0$, the possible exceptions being $\pm x''_N$. We have $N - 2 \leq 2n$ by induction hypothesis, which proves the statement. □

Lemma 4.5. *For any lattice basis B of $\mathcal{L}$, integer q and lattice vectors x, y with $x \neq y$ and $\|x - y\| < q\lambda_1(\mathcal{L})$, the coordinate vectors of x and y with respect to B are different modulo q, i.e. $B^{-1}x \not\equiv B^{-1}y \bmod q$.*

Proof. Assume that $B^{-1}x \equiv B^{-1}y \bmod q$. This implies $x - y = q \cdot Bz$ for some vector $z \in \mathbb{Z}^n \setminus \{0\}$, hence $\|x - y\| \geq q\lambda_1(\mathcal{L})$, a contradiction. □

We will use the next lemma from [SD16], to make sure that there is only one (randomly shifted) vector in the ball around the (randomly shifted) target vector t, which is orthogonal to a uniform chosen z in $\mathbb{Z}_p^n$. Thus, a sparsification using z will ultimately result in a 'big gap' between the closest lattice vector to t and all other lattice vectors.

Lemma 4.6. *Let $t \in \mathbb{Q}^n$. For any prime number p, set of vectors $\{v_i\}_{1 \leq i \leq N} \subseteq \mathbb{Z}_p^n \setminus \{0\}$, and $t \notin \{v_i\}_{1 \leq i \leq N}$ we have*

$$\frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}} \leq \operatorname*{Prob}_{z,u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)} [\forall\ 1 \leq i \leq N : \langle z, t + u\rangle \equiv 0 \mod p,\ \langle z, v_i + u\rangle \not\equiv 0 \mod p\ ].$$

Proof. We introduce the events

$$A \coloneqq [\forall\ 1 \le i \le N : \langle z, t+u\rangle \equiv 0 \mod p,\ \langle z, v_i+u\rangle \not\equiv 0 \mod p],$$
$$B \coloneqq [\forall\ 1 \le i \le N,\ \forall k \in \mathbb{Z}_p \setminus \{1\} : t+u \neq k(v_i+u)].$$

Consider the sets

$$U \coloneqq \{x \in \mathbb{Z}_p^n : \langle x, t+u\rangle \equiv 0 \mod p\}, \quad V_i \coloneqq \{x \in \mathbb{Z}_p^n : \langle x, v_i+u\rangle \equiv 0 \mod p\}.$$

They are $(n-1)$-dimensional subspaces of $\mathbb{Z}_p^n$. Conditional on B, the subspaces U and V_i are distinct for every i, thus $U \cap V_i$ is an $(n-2)$-dimensional subspace with $|U \cap V_i| = p^{n-2}$. Let $V \coloneqq \bigcup_{i=1}^N V_i$, then

$$\operatorname*{Prob}_{z,u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)}[A|B] = \frac{|U \setminus V|}{|\,\mathbb{Z}_p^n\,|} \ge \frac{|U| - \sum_{i=1}^N |U \cap V_i|}{p^n} = \frac{1}{p} - \frac{N}{p^2}.$$

Note that

$$\begin{aligned}\operatorname*{Prob}_{u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)}[B^C] &= \operatorname*{Prob}_{u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)}[\exists\ 1 \le i \le N, k \in \mathbb{Z}_p \setminus \{1\} : k(v_i+u) = t+u] \\ &\le \sum_{i=1}^N \operatorname*{Prob}_{u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)}[\exists\ k \in \mathbb{Z}_p \setminus \{1\} : k(v_i+u) = t+u] \le \frac{N(p-1)}{p^n},\end{aligned}$$

where the last inequality follows because for any fixed i there are at most $p-1$ possible choices for u such that the equality $k(v_i+u) = t+u$ is satisfied since $k \in \mathbb{Z}_p \setminus \{1\}$. Summing up, we get

$$\begin{aligned}\operatorname{Prob}[A] &= \operatorname{Prob}[A|B]\operatorname{Prob}[B] + \operatorname{Prob}[A|B^C]\operatorname{Prob}[B^C] \\ &\ge \operatorname{Prob}[A|B]\left(1 - \frac{N(p-1)}{p^n}\right) \ge \frac{1}{p} - \frac{N}{p^2} - \frac{N(p-1)}{p^n} - \frac{N}{p^n} \\ &= \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}},\end{aligned}$$

where all probabilities are taken over the uniform and independent choices of u and z. □

Remark 4.7. In our case we have $N \le 2n$ as well as $4\gamma n^2 < p < 8\gamma n^2$. So the probability in Lemma 4.6 is non-negligible. By noting that it is at most $1/p + 1/p^n$, because $\langle z, t+u\rangle$ mod p is uniformly distributed on $\mathbb{Z}_p$ if $t+u \neq 0 \in \mathbb{Z}_p^n$, and that $\operatorname{Prob}[t+u=0] = 1/p^n$ for u chosen uniformly over $\mathbb{Z}_p^n$, we see that the lower bound is almost tight.

Proposition 4.8. *Let B be a basis of an n-dimensional lattice $\mathcal{L}$, t a target vector and $c \in \mathcal{L}$ with $\|c-t\| \le \lambda_1(\mathcal{L})/(\sqrt{2}\gamma) =: R$ for some $\gamma \ge 1$. Furthermore, let p be a prime number with $p \ge \gamma n + 1$. We have*

$$\operatorname*{Prob}_{z,u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)}\left[\begin{array}{c} c+w \in \mathcal{L}_{p,z} \cap \mathcal{B}(t+w, \gamma R) \\ \mathbb{Z}(c+w) \supseteq \mathcal{L}_{p,z} \cap \bigcup\limits_{1 \le i \le \lfloor \gamma n \rfloor} \mathcal{B}(i(t+w), \gamma R) \end{array}\right] \ge \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}}$$

with $N = \left|\mathcal{L} \cap \bigcup_{i=1}^{\lfloor \gamma n \rfloor} \mathcal{B}(i(t+w), \gamma R)\right|$. *If* $w \in \mathcal{L}$ *is chosen such that* $B^{-1}w \equiv u \bmod p$ *and* $\|t+w\| > \gamma R(n+1)$ *holds, then none of the multiples* $i(c+w)$ *lies in a ball centred at* $j(t+w)$ *with radius* γR *for* $i \neq j$, *i.e.*

$$i(c+w) \notin \bigcup_{\substack{1 \leq i,j \leq \lfloor \gamma n \rfloor \\ i \neq j}} \mathcal{B}(j(t+w), \gamma R)$$

holds for all $1 \leq i \leq \lfloor \gamma R \rfloor$.

Proof. Let $N_i \coloneqq |\,\mathcal{L} \cap \mathcal{B}(it, \gamma R) \setminus \{ic\}|$ and denote the lattice vectors inside a ball with radius γR around it without the shifted closest vector ic by $\{v_{ij}\}_{1 \leq j \leq N_i}$. We want to use Lemmata 4.5 and 4.6. For all $1 \leq i \leq \lfloor \gamma n \rfloor$ and $1 \leq j \leq N_i$ we observe that

$$\|ic - v_{ij}\| \leq i\|c-t\| + \|it - v_{ij}\| \leq \gamma n R + \gamma R = \frac{n+1}{\sqrt{2}} \lambda_1(\mathcal{L}) < p\lambda_1(\mathcal{L}).$$

For any vector x we denote its coordinate vector with respect to the basis B by $\tilde{x} \coloneqq B^{-1}x$. Lemma 4.5 gives us $i\tilde{c} \not\equiv \tilde{v}_{ij} \bmod p$. Note that $i < \gamma n + 1 \leq p$, thus i is invertible modulo p. We use Lemma 4.6 with $\tilde{c}$ and $\{\frac{1}{i}\tilde{v}_{ij}\}_{1 \leq i \leq \lfloor \gamma n \rfloor, 1 \leq j \leq N_i}$ and obtain

$$\operatorname*{Prob}_{z,u \leftarrow \mathcal{U}(\mathbb{Z}_p^n)} \left[\begin{array}{cc} \langle z, \tilde{c} + u \rangle \equiv 0 & \bmod p \\ \langle z, \tilde{v}_{ij} + iu \rangle \not\equiv 0 & \bmod p \ \forall i,j \end{array} \right] \geq \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}},$$

where we used that $\langle z, \tilde{v}_{ij} + iu \rangle \not\equiv 0 \bmod p$ if and only if $\langle z, \tilde{v}_{ij}/i + u \rangle \not\equiv 0 \bmod p$. To see that none of the multiples of the shifted closest vector ends up in another ball, we note:

$$\begin{aligned} \|i(c+w) - j(t+w)\| &= \|(j-i)(t+w) - i(c-t)\| \\ &\geq |j-i| \cdot \|t+w\| - |i| \cdot \|c-t\| \\ &> \gamma R(n+1) - \gamma n R = \gamma R, \end{aligned}$$

for $i \neq j$ and w chosen as in the assumption of the proposition. The last inequality is obtained by using the reverse triangle inequality. □

Remark 4.9. Since the input to the algorithm is a $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ lattice, we have $\|c-t\| \leq (1-1/n)\lambda_1(\mathcal{L})/(\sqrt{2}\gamma) < \lambda_1(\mathcal{L})/(\sqrt{2}\gamma)$, and since $p > 4\gamma n^2 > \gamma n + 1$, Proposition 4.8 can be applied to our reduction. In addition, we have $N \leq 2n \cdot \gamma n < p/2$ according to Lemma 4.4. The probability that the lattice sparsification process works can therefore be estimated as

$$\begin{aligned} \text{Prob}\,[\text{`Lattice sparsification works'}] &\geq \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}} \\ &> \frac{1}{p} - \frac{1}{2p} - \frac{1}{2p^{n-2}} \\ &\approx \frac{1}{2p} \geq \frac{1}{8\gamma n^2}. \end{aligned}$$

Therefore, we have proved the following: With non-negligible probability, the only vectors of the BDD lattice $\mathcal{L}$ belonging to the $\lfloor \gamma n \rfloor$ balls that are kept in the sparsified lattice are multiples of $c+w$. The first $\lfloor \gamma \rfloor$ vectors in the set $\{i(c+w)\}_{1 \le i \le \lfloor \gamma n \rfloor}$ are always kept since in this case $\|i(c+w) - i(t+w)\| \le \lambda_1(\mathcal{L})/\sqrt{2} = \gamma R$.

We will now analyse the $(n+1)$-dimensional lattice created in the second step of the algorithm.

Lemma 4.10. *Let $n \ge 2$, $\gamma \ge 1$, $t \in \mathbb{Q}^n$, $d := \operatorname{dist}(t, \mathcal{L})$, $c \in \mathcal{L}$ with $\|c - t\| = d$, $d_0 \in [d, d/(1-1/n)]$, and $k := 1/(n-1)$. Further, we let the vector w be chosen as in Proposition 4.8. We consider the $(n+1)$-dimensional lattice $\Lambda := \Lambda(B')$ defined by the basis*

$$B' = \begin{pmatrix} B_{p,z} & t+w \\ 0 & kd_0 \end{pmatrix}.$$

Then Λ satisfies $\lambda_2(\Lambda)/\lambda_1(\Lambda) \ge \gamma(1 + \Omega(1/n))$ and

$$s' = \begin{pmatrix} c-t \\ -kd_0 \end{pmatrix}$$

is a shortest vector of Λ, both with non-negligible probability (over the choices of u and z as in Proposition 4.8).

Proof. Every non-zero vector $v \in \Lambda(B')$ can be written as

$$v = \begin{pmatrix} b_v - m(t+w) \\ -mkd_0 \end{pmatrix}$$

for some $m \in \mathbb{Z}$, $0 \ne b_v \in \mathcal{L}_{p,z}$. Note that since $c+w \in \mathcal{L}_{p,z}$, taking $m = 1$ yields $s' \in \Lambda$. We want to establish lower bounds for the norm of such vectors v that are not parallel to s'. So from now on let $v \in \Lambda$ denote an arbitrary vector that is linearly independent from s', and without loss of generality we assume $m \ge 0$. First we note that $\|s'\|^2 = d^2 + k^2 d_0^2$. To prove the statement, we will consider three cases. If we assume $m = 0$, then

$$\|v\|^2 = \|b_v\|^2 \ge \lambda_1(\mathcal{L})^2 \ge \frac{2\gamma^2 d^2}{(1-\frac{1}{n})^2} \ge 2\gamma^2 d_0^2.$$

Here we used that $\mathcal{L}$ is a $\mathrm{BDD}_{(1-1/n/(\sqrt{2}\gamma))}$ lattice. Thus, we obtain

$$\begin{aligned} \frac{\|v\|^2}{\|s'\|^2} &\ge \frac{2\gamma^2 d_0^2}{d^2 + k^2 d_0^2} = \frac{2\gamma^2}{(\frac{d}{d_0})^2 + k^2} \\ &\ge \frac{2\gamma^2}{1+k^2} = \frac{2\gamma^2}{1 + \frac{1}{(n-1)^2}} =: C_1(n). \end{aligned}$$

In the second case we consider $1 \leq m \leq \gamma n$. By Proposition 4.8 we have

$$c + w \in \mathcal{L}_{p,z} \cap \bigcup_{i=1}^{\lfloor \gamma n \rfloor} \mathcal{B}\left(i(t+w), \frac{\lambda_1(\mathcal{L})}{\sqrt{2}}\right) \subseteq \mathbb{Z}(c+w)$$

with non-negligible probability. Since v is collinear to s' if and only if $b_v \in \mathbb{Z}(c+w)$, we obtain

$$\|v\|^2 = \|b_v - m(t+w)\|^2 + (md_0k)^2 \geq \frac{\lambda_1(\mathcal{L})^2}{2} + (md_0k)^2 \geq \left(\frac{d\gamma}{1-\frac{1}{n}}\right)^2 + (md_0k)^2$$

and thus

$$\begin{aligned}\frac{\|v\|^2}{\|s'\|^2} &\geq \frac{\left(\frac{d\gamma}{1-\frac{1}{n}}\right)^2 + (md_0k)^2}{d^2 + k^2 d_0^2} \geq \frac{\left(\frac{d\gamma}{1-\frac{1}{n}}\right)^2 + (mdk)^2}{d^2 + (\frac{d}{1-\frac{1}{n}})^2 k^2} \\ &= \frac{\gamma^2 + \frac{m^2}{n^2}}{(1-\frac{1}{n})^2 + \frac{1}{(n-1)^2}} \geq \frac{\gamma^2 + \frac{1}{n^2}}{(1-\frac{1}{n})^2 + \frac{1}{(n-1)^2}} =: C_2(n).\end{aligned}$$

Finally, we consider $m > \gamma n$. In this case we have

$$\frac{\|v\|^2}{\|s'\|^2} \geq \frac{(md_0k)^2}{d^2 + d_0^2 k^2} \geq \frac{(mdk)^2}{d^2 + (\frac{d}{1-\frac{1}{n}})^2 k^2} = \frac{m^2}{(n-1)^2 + (\frac{n}{n-1})^2} \geq \frac{\gamma^2}{(1-\frac{1}{n})^2 + \frac{1}{(n-1)^2}} =: C_3(n).$$

We combine the three lower bounds on $\|v\|^2/\|s'\|$ to complete the proof. We observe that

$$C_1(n) \geq \frac{2\gamma^2}{1+1} \geq 1 \quad \text{and} \quad C_3(n) \leq C_2(n).$$

Additionally, a quick calculation shows that $C_3(n) \geq C_1(n)$ if and only if $n \geq 2$. Thus

$$\frac{\|v\|}{\|s'\|} \geq 1$$

and to complete the proof, we note that for large enough n the uSVP-gapsatisfies

$$\frac{\lambda_2^2(\Lambda)}{\lambda_1^2(\Lambda)} \geq \min(C_1(n), C_2(n), C_3(n)) \geq \gamma^2 \left(1 + \Omega\left(\frac{1}{n}\right)\right).$$

□

Remark 4.11. For a nice geometrical illustration of the overall reduction see [SBW16] Figures 3 and 4.

According to Remark 4.3 and Proposition 4.8, the requirements of Lemma 4.10 are met with non-negligible probability. Thus, Lemma 4.10 yields that the $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$ solver outputs s' and Algorithm 2 finds the closest vector $c \in \mathcal{L}$ to t. Therefore, we have proved the following.

Theorem 4.12 (Reduction from BDD to uSVP)**.** *Let $1 \leq \gamma = \gamma(n) \leq poly(n)$. There is a probabilistic polynomial-time reduction from $BDD_{1/(\sqrt{2}\gamma)}$ to $uSVP_{\gamma(1+\Omega(1/n))}$.*

Remark 4.13. Since uSVP gets easier if the gap gets larger, we have in particular a reduction from $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ to uSVP_γ since any uSVP_γ solver can also solve $\mathrm{uSVP}_{\gamma(1+\Omega(1/n))}$.

4.2 A Reduction from uSVP to DHSP

In this section we want to present a reduction from uSVP to the dihedral hidden subgroup problem (DHSP), following the approach in [Reg04a]. More formally, the statement is as follows.

Theorem 4.14 (Corollary 1.2 in [Reg04a])**.** *If there exists a solution to the dihedral hidden subgroup problem (that samples cosets), then there exists a quantum algorithm that solves the* poly(n)*-unique shortest vector problem.*

We will prove Theorem 4.14 by showing the chain of reductions depicted in the following diagram.

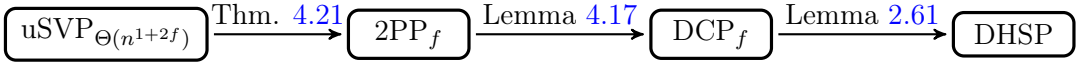


Figure 4.1: uSVP to DHSP reduction

Here, f is the failure parameter of DCP and 2PP is the abbreviation for the *two point problem* which is defined as follows.

Definition 4.15 (Two point problem ($\mathrm{2PP}_f$))**.** Let $M \in \mathbb{N}$ and f a failure parameter. We are given a polynomial (in $n \log M$) number of $1 + n\lceil \log M \rceil$ qubit registers. Each register is in the state

$$\frac{1}{\sqrt{2}}(|0, a\rangle + |1, a'\rangle)$$

with probability of at least $1 - 1/(n \log(2M))^f$, where $a, a' \in \{0, \ldots, M-1\}^n$ such that the difference $a - a'$ is fixed, and otherwise it is in an arbitrary state $|b, \overline{a}\rangle$ with $b \in \{0, 1\}$ and $\overline{a} \in \{0, \ldots, M-1\}^n$. The *two point problem* ($\mathrm{2PP}_f$) with failure parameter f is to find $a - a'$.

Remark 4.16. The dihedral coset problem and the two point problem appear to be very similar to each other. We say that an algorithm solves DCP_f if it outputs d with probability $\mathrm{poly}(1/\log N)$ in time $\mathrm{poly}(\log N)$. Analogously, an algorithm solves $\mathrm{2PP}_f$ if it outputs $a - a'$ with probability $\mathrm{poly}(1/(n \log M))$ in time $\mathrm{poly}(n \log M)$.

Lemma 4.17. *If there exists an algorithm that solves DCP with failure parameter f, then there exists an algorithm that solves 2PP with the same failure parameter.*

Proof. We claim that the following algorithm solves 2PP when given oracle access to a solver for DCP.

Algorithm 3 Reduction from 2PP to DCP

INPUT: A valid input to the two point problem and access to an algorithm that solves the dihedral coset problem.
OUTPUT: The solution to the two point problem

1: Transform the 2PP_f input registers into a valid input for DCP_f with $N = (2M)^n$. This is done by performing the action $(a_0, \ldots, a_{n-1}) \mapsto \sum_{i=0}^{n-1} a_i \cdot (2M)^i$ on the last $n\lceil \log M \rceil$ qubit registers.
2: Run the DCP solver with these transformed input registers. Let d denote the result.
3: Compute $\tilde{d} = d + M + 2M^2 + \ldots + 2^{n-1}M^n$.
4: **for** $i = 0$ to $n - 1$ **do**
5: Compute $d_i = \tilde{d} - M \mod (2M)$.
6: Set $\tilde{d} = (\tilde{d} - d_i - M)/(2M)$.
7: **end for**
8: Return $(d_0, \ldots, d_{n-1})$.

After applying the map

$$\varphi\colon \{0, \ldots, M-1\}^n \to \{0, \ldots, (2M)^n - 1\},$$
$$(a_0, \ldots, a_{n-1}) \mapsto \sum_{i=0}^{n-1} a_i \cdot (2M)^i$$

in the first step, we obtain registers of the form

$$\frac{1}{\sqrt{2}}(|0, \varphi(a)\rangle + |1, \varphi(a')\rangle)$$

with probability of at least $1 - 1/(n(\log 2M))^f$. We have

$$d \coloneqq \varphi(a) - \varphi(a') = a_0 - a_0' + (a_1 - a_1') \cdot 2M + \ldots + (a_{n-1} - a_{n-1}') \cdot (2M)^{n-1},$$

which is fixed because the difference $a - a'$ is fixed by assumption. Therefore, we have created a valid input for DCP with $N = (2M)^n$. We run the DCP algorithm, which outputs d with high probability. Next we produce

$$\begin{aligned}\tilde{d} &\coloneqq d + M + 2M^2 + \ldots + 2^{n-1}M^n \\ &= d_0 + M + (d_1 + M) \cdot 2M + \ldots + (d_{n-1} + M) \cdot (2M)^{n-1},\end{aligned}$$

where we set $d_i \coloneqq a_i - a'_i$. Because $-M + 1 \leq d_i \leq M - 1$ for all i, we obtain $1 \leq d_i + M \leq 2M - 1$, and we can extract the d_i successively from $\tilde{d}$. At first we compute $\tilde{d} \mod 2M$ which gives us d_0. Then we compute $\tilde{d_1} \coloneqq (\tilde{d} - d_0 - M)/(2M)$ from which we get d_1 by considering $\tilde{d_1}$ modulo $2M$. Continuing this way, we obtain all d_i. The algorithm outputs $(d_0, \ldots, d_{n-1})$, the solution to the two point problem. □

We will now come to the main result of this section: the reduction from uSVP to 2PP. First, we will explain the intuition and basic idea of the reduction and then present it more formally. The first step of the reduction is of course to create an input for the two point problem. Since we want to use an algorithm for 2PP to find a solution for uSVP, the hidden difference in the instance of 2PP should more or less be the shortest vector of the unique SVP lattice. We will create an instance of 2PP such that the fixed difference $a - a'$ is

$$\hat{u} \coloneqq \left(u_1, \ldots, \frac{u_{i_0} - m}{p}, \ldots, u_n\right),$$

where i_0 is some index, $p > n^{2+2f}$ is an integer prime number and $1 \leq m \leq p - 1$ such that $u_{i_0} \equiv m \mod p$. From this, we can extract the shortest unique vector $u \in \mathcal{L}$. To create a superposition of only two lattice points with fixed difference $\hat{u}$, we start with a uniform superposition over many lattice points. We partition the underlying space into cubes, such that there are at most two lattice points in each cube and if that is the case, their difference should be $\hat{u}$. The partition is obtained by computing some function F in a separate register and the measurement of this register will leave us with the desired superposition over two lattice points, hiding the vector $\hat{u}$, i.e. an input register for 2PP.

We will now present a description of a subroutine that creates input registers for the two point problem. The routine will start with two quantum registers. The first one contains one qubit and the second one $n\lceil \log M \rceil$ qubits, initially in the state $|0, 0\rangle$. The functions g, h and $F \coloneqq g \circ h$ are defined in the discussion subsequent to the subroutine which is as follows:

Algorithm 4 Subroutine: Creating input registers for the two point problem

INPUT: An LLL-reduced basis B of an n-dimensional $c_{\text{unq}} n^{1+2f}$-uSVP lattice $\mathcal{L}$, a prime $p > n^{2+2f}$, a positive integer m with $1 \leq m \leq p - 1$ that satisfies $u_1 \equiv m \mod p$ where u_1 is the first component of a shortest vector $u \in \mathcal{L}$, and an approximation l of the length of the shortest vector such that $\|u\| \leq l \leq 2\|u\|$.
OUTPUT: A valid input for the two point problem

1: Set $M = 2^{4n}$ and apply the Hadamard transform on $1 + n\lceil \log M \rceil$ qubits.
2: Compute the function $F = g \circ h$ in a third register and measure the result.
3: Output the first two registers.

Here, $c_{\text{unq}} > 0$ is some constant, which has to fulfil some requirements we specify later.

Let us take a closer look at what the above routine does exactly. We denote the LLL-reduced basis by $\{b_1, \dots, b_n\}$.

The registers are initially in the state $|0\rangle |0\rangle$. After the first step we have created the uniform superposition over all elements of the set $\mathcal{A} \coloneqq \{0,1\} \times \{0, \dots, M-1\}^n$. Now the registers are in the state

$$\frac{1}{\sqrt{2M^n}} \sum_{(t,a)\in\mathcal{A}} |t, a\rangle .$$

Since we want to hide $\hat{u} \in \mathcal{L}$, we need to map elements of $\mathcal{A}$ to lattice points. This is done by the function $h\colon \mathcal{A} \to \mathcal{L}$;

$$h((t,a)) \coloneqq (a_1 p + tm) b_1 + \sum_{i=2}^{n} a_i b_i.$$

As we already mentioned, we want to partition the space into cubes. We randomly translate them to make sure that they are not aligned with the lattice. Let $\{e_1, \dots, e_n\}$ denote the standard basis of $\mathbb{R}^n$. For $v = \sum_{i=1}^n v_i e_i \in \mathbb{R}^n$ we compute the function $g\colon \mathbb{R}^n \to \mathbb{Z}^n$;

$$g(v) \coloneqq \left(\left\lfloor \frac{v_1}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} - w_1 \right\rfloor, \dots, \left\lfloor \frac{v_n}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} - w_n \right\rfloor \right),$$

where the real numbers $w_1, \dots, w_n$, which are chosen uniformly at random from $[0,1)$, are responsible for the random translation of the cubes and c_{cube} is some constant. Note that $F \coloneqq g \circ h$ maps every point of $\mathcal{A}$ to some $q \in \mathbb{Z}^n$ which represents one cube in the partition of $\mathbb{R}^n$. After the routine has computed F in the second step and measured the result, say q, we have produced the state

$$\sum_{\substack{(t,a)\in\mathcal{A} \\ F(t,a)=q}} |t, a\rangle |q\rangle , \tag{4.1}$$

where the normalisation factors were omitted for convenience. What remains to show is that this really creates an input register for the two point problem, which hides $\hat{u}$ with probability at least $1 - 1/(n \log(2M))^f$. In the following we will assume $i_0 = 1$ without loss of generality.

Lemma 4.18. *In the setting of Algorithm 4 let $c_{\text{unq}} > 2c_{\text{cube}}$. For every $q \in \mathbb{Z}^n$ there exist at most two elements $(0,a), (1,a') \in \mathcal{A}$ that get mapped to q by F. If this is the case, then*

$$a' - a = \hat{u} = \left(\frac{u_1 - m}{p}, u_2, \dots, u_n \right).$$

Proof. Let $x, y \in \mathcal{L}$ be in the image of f, i.e. $x = f(t,a)$ and $y = f(s,a')$ for some $(t,a),(s,a') \in \mathcal{A}$, with $g(x) = g(y) = q$. Also let x_i, y_i denote their respective coordinates in the standard basis of $\mathbb{R}^n$ for $1 \leq i \leq n$. At first, we show that $x - y$ is a multiple of the shortest vector $u \in \mathcal{L}$. Assuming the contrary, we have

$$\|x - y\| > c_{\text{unq}} n^{1+2f} \|u\| \geq \frac{1}{2} l \cdot c_{\text{unq}} n^{1+2f},$$

since by assumption $\mathcal{L}$ is a $c_{\text{unq}} n^{1+2f}$-unique lattice and $2\|u\| \geq l$. This implies that $|x_i - y_i| > \frac{1}{2} l \cdot c_{\text{unq}} n^{\frac{1}{2}+2f}$ for at least one index i, because otherwise we would have

$$\|x - y\| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \leq \sqrt{n \cdot \left(\frac{1}{2} l \cdot c_{\text{unq}} n^{\frac{1}{2}+2f}\right)^2},$$

a contradiction. To see that this implies $g(x) \neq g(y)$, note that

$$\begin{aligned} 0 = g(x)_i - g(y)_i &= \left\lfloor \frac{x_i}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} - w_i \right\rfloor - \left\lfloor \frac{y_i}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} - w_i \right\rfloor \\ &\geq \frac{x_i - y_i}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} - 1 > 0, \end{aligned}$$

where we used $x_i \geq y_i$ without loss of generality. Hence, $x - y = k \cdot u$ for some $k \in \mathbb{Z} \setminus \{0\}$. The first coordinate of $x - y$ in the LLL-basis is

$$k u_1 \equiv km \equiv a_1 p + tm - (a_1' p + sm) \equiv (t - s)m \mod p.$$

So $t - s \equiv k \mod p$. If we assume $t = s$, we get $|k| \geq p$ and therefore

$$\|x - y\| \geq p\|u\| > n^{2+2f} \|u\| > n^{1+2f} c_{\text{unq}} \|u\| \geq l \cdot c_{\text{cube}} n^{1+2f}$$

for n large enough, which again implies $g(x) \neq g(y)$. This means we have, without loss of generality $t = 0$, and $s = 1$. It follows that $k \equiv 1 \mod p$ and even $k = 1$ because for $|k| > 1$ we would get the same contradiction as before. This completes the proof of the lemma sind $x_1 - y_1 = 1 \cdot u_1 = p(a_1 - a_1') + m$. □

We have seen that the state created in Equation (4.1) consists of at most two points, whose difference is $\hat{u}$. What remains to show is that the probability of the procedure not failing is high enough.

Lemma 4.19. *Let m be chosen as in Algorithm 4 and let $u = \sum_{i=1}^{n} u_i b_i$ denote the unique shortest vector. If we choose $(t,a) \in \mathcal{A} = \{0,1\} \times \{0,\ldots,M-1\}^n$ uniformly at random, then with probability of at least $1 - \frac{1}{(n \log(2M))^f}$ there exists $a' \in \{0,\ldots,M-1\}^n$ such that $F(1-t,a') = F(t,a)$.*

Proof. Let $q \in \mathbb{Z}^n$ represent any fixed cube of our partition and let Q be the random variable representing the measurement of $F(t,a)$ at the end of step (2) of Algorithm 4. Then

$$\text{Prob}[Q = q] = \underset{(t,a)\leftarrow U(\mathcal{A})}{\text{Prob}}[F(t,a) = q] = \frac{1}{2M^n}|\{(t,a) \in \mathcal{A} : F(t,a) = q\}|,$$

where the probability on the left is taken over all the randomness in Algorithm 4. So we may as well assume that t and a are chosen uniformly at random. We define

$$a' \coloneqq \left(a_1 \pm \frac{u_1 - m}{p}, a_2 \pm u_2, \ldots, a_n \pm u_n\right),$$

where the "+" corresponds to the case $t = 0$ and the "−" to the case $t = 1$.

There are two possible ways the procedure can fail to put out a valid input register for the two point problem. First, a' might not be in $\{0, \ldots, M-1\}^n$. We know from Lemma 2.34 that $|u_i| \leq 2^{2n}$. Thus, we obtain

$$\begin{aligned}\text{Prob}[a' \notin \{0,\ldots,M-1\}^n] &\leq \text{Prob}[\exists i : a_i < 2^{2n} \text{ or } a_i \geq M - 2^{2n}] \\ &= n \cdot \frac{2^{2n} + 2^{2n}}{M} = \frac{n2^{2n+1}}{M} \end{aligned} \tag{4.2}$$

because a was chosen uniformly at random. The second possibility for the procedure to fail is that $f(1-t,a')$ and $f(t,a)$ are located in different cubes. We observe that

$$\begin{aligned} f(1-t,a') - f(t,a) &= (a_1 p \pm (u_1 - m) + (1-t)m - (a_1 p + tm))b_1 + \sum_{i=2}^{n}(a_i \pm u_i - a_i)b_i \\ &= \pm u. \end{aligned}$$

Since $w_1, \ldots, w_n \in [0,1)$ are uniformly chosen, we have

$$\text{Prob}[F(1-t,a')_i \neq F(t,a)_i] \leq \frac{|\langle u, e_i\rangle|}{l \cdot c_{\text{cube}} n^{\frac{1}{2}+2f}} \leq \frac{|\langle u, e_i\rangle|}{\|u\| c_{\text{cube}} n^{\frac{1}{2}+2f}} \tag{4.3}$$

for every coordinate i. This implies

$$\text{Prob}[F(1-t,a') \neq F(t,a)] \leq \frac{\sum_{i=1}^{n}|\langle u, e_i\rangle|}{\|u\| c_{\text{cube}} n^{\frac{1}{2}+2f}} \leq \frac{1}{c_{\text{cube}} n^{2f}},$$

where we used sub-additivity and

$$\|u\|_1 = \sum_{i=1}^{n}|u_i| \cdot 1 \leq \left(\sum_{i=1}^{n}|u_i|^2\right)^{\frac{1}{2}} \cdot \left(\sum_{i=1}^{n}1\right)^{\frac{1}{2}} = \sqrt{n} \cdot \|u\|$$

by the Cauchy-Schwarz inequality. Summing up, we can bound the error probability of the procedure by

$$\text{Prob}[\text{created register is 'bad'}] \leq \frac{n2^{2n+1}}{M} + \frac{1}{c_{\text{cube}} n^{2f}} \leq \frac{1}{(n\log(2M))^f} \tag{4.4}$$

for c_{cube} large enough. Note that although $2c_{\text{cube}}$ is bounded by c_{unq}, we can consider an instance where c_{unq} and therefore also c_{cube} are large enough for the estimations to work since we are dealing with $\Theta(n^{1+2f})$-uSVP instances. The last inequality above can be seen by noting that

$$n2^{2n+1}c_{\text{cube}}n^{2f}(n\log(2M))^f + M(n\log(2M))^f \leq Mc_{\text{cube}}n^{2f}$$

for c_{cube} (and again n) sufficiently large, since we are interested in the asymptotic behaviour for $n \to \infty$. Note that $n\log(2M) = n\log(2) + 4n^2$, so (half) the term on the right side dominates the first summand because $n2^{2n+1}(n\log(2M))^f \leq M = 2^{4n}$ and for c_{cube} large enough it also dominates the second summand. □

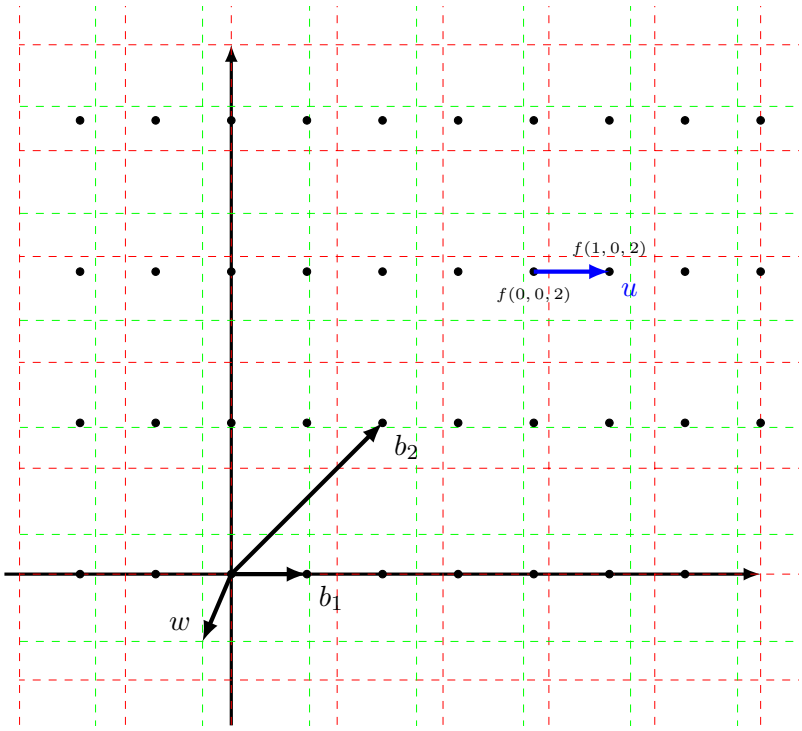


Figure 4.2: Illustration of the lattice partition into cubes

Example 4.20. We give an example of the described procedure above in Figure 4.2. We take $\mathcal{L} = \mathcal{L}(B)$ with the basis

$$\begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix},$$

so we have $n = 2$, $\lambda_1(\mathcal{L}) = 1$ and $\lambda_2(\mathcal{L}) = 2$. Assume we take $c_{\text{cube}} = 0.5$, $l = 2$ and $f = 0.003$. Since the shortest vector is $(1, 0)^T$, we take $i_0 = 1$, $p = 5$ and $m = 1$. In Figure 4.2 the lattice $\mathcal{L}$ is displayed together with two different partitions of the plane with edge length 1.42. The red one is without any translation and the green one with translation by the vector $w = (0.62, 0.11)^T$. Consider the lattice points $h(0, 0, 2) = (4, 4)^T$

and $h(1,0,2) = (5,4)^T$. We have $5/\sqrt{2} \approx 3.5355$ as well as $4/\sqrt{2} \approx 2.8284$. Hence the probability that $g(4,4)$ and $g(5,4)$ differ in the first coordinate is $1/\sqrt{2} \approx 0.7071$, where the probability is taken over the choice of $w_1 \in [0,1)$. This serves as an illustration for the estimation of the probability that $F(1-t,a')$ and $F(t,a)$ differ in the i-th coordinate, see Equation 4.3.

We have proved that the described routine indeed, with high probability, produces a 'good' input register for the two point problem, which hides the vector $\hat{u}$. Therefore, we can now extract the unique shortest vector $u \in \mathcal{L}$ by using the 2PP oracle.

Theorem 4.21 (Reduction from uSVP to 2PP)**.** *Let $p > n^{2+2f}$ be any fixed prime number. If there exists a solution to the two point problem with failure parameter $f > 0$, then there exists a quantum algorithm that, when called with the input*

- *a $(c_{\text{unq}} n^{1+2f})$-unique lattice $\mathcal{L}$ with shortest vector $u = \sum_{i=1}^{n} u_i b_i$ for some sufficiently large constant c_{unq},*
- *an integer index i_0,*
- *a positive integer m with $1 \leq m \leq p-1$ and $u_{i_0} \equiv m \bmod p$*
- *and an approximation of the length of the shortest vector l with $\|u\| \leq l \leq 2\|u\|$,*

returns u with the same probability as the success probability 2PP oracle.

Proof. To create an complete input for the 2PP oracle, we call Algorithm 4 to create an input register for the two point problem $\text{poly}(n \log M) = \text{poly}(n)$ times. Lemmata 4.18 and 4.19 assure that the input for the oracle, created this way, is indeed a valid input for the two point problem. Also, we have shown that the hidden difference in this 2PP instance is

$$a' - a = \left(u_1, \ldots, \frac{u_{i_0} - m}{p}, \ldots, u_n \right).$$

So if we call the 2PP oracle with the correct parameters l, m, i_0, the output will be $a' - a$ and we can extract u from this vector. What remains to show is how to find the correct values for the three parameters. The solution is quite simple – we just try all possible values. From Proposition 2.33 we know that $\|b_1\| \leq 2^{(n-1)/2} \|b_1^*\|$. Thus we can just divide $\|b_1\|$ by increasing powers of 2 and we know that one of the $(n-1)/2$ values for l will be as required. Because $u \in \mathcal{L}$ is the shortest vector, we know that $u/p \notin \mathcal{L}$. This implies the existence of some index i_0 such that $u_{i_0} \not\equiv 0 \bmod p$. With $1 \leq i_0 \leq n$ and $1 \leq m \leq p-1$, we have a total of $(p-1)n(n-1)/2 = \mathcal{O}(n^{4+2f})$ possible values for the parameters. We call the procedure with each of these values and, for each output of our 2PP oracle, we compute the respective candidate for the shortest vector. By keeping only the shortest of the vectors produced this way and discarding all non-lattice vectors, we find the solution to the uSVP instance. □

We can now prove the claimed reduction from uSVP to DHSP.

Proof of Theorem 4.14. The proof is just a combination of the several reductions in this section shown in Theorem 4.21 and Lemmata 4.17 and 2.61, see Figure 4.1. □

Remark 4.22. The uSVP to 2PP reduction, which we have presented, can be improved such that the uSVP parameter is not $\Theta(n^{1+2f})$, but rather $\Theta(n^{1/2+2f})$. Regev gives an algorithm that partitiones the underlying space with balls instead of cubes. The improved algorithm has many similarities with the one described above. In particular, the important Lemmata 4.18 and 4.19 are almost identical to the ones we need if the partition is made with balls. However, some additional, technical lemmata are needed. The details can be found in [Reg04a] Section 3.3.

4.3 A Reduction from DCP to the Subset Sum Problem

In this section we will yet present another reduction from the dihedral coset problem. This time we reduce DCP to the so-called *subset sum problem*(SSP). In contrast to the rather simple reduction from DCP to DHSP (Lemma 2.61), the reduction to SSP is more complicated and requires some preliminary lemmata before we can describe the quantum reduction. The approach we will take is again taken from [Reg04a].

Definition 4.23 (Subset sum problem (SSP), decision version)**.** Given a set of integers $A = \{a_1, \ldots, a_r\}$, where $r \in \mathbb{N}$, and a target $t \in \mathbb{N}$, the *decisional subset sum problem* (SSP) is to decide whether there exists a subset $B \subseteq \{1, \ldots, r\}$ such that

$$\sum_{i \in B} a_i = t.$$

The search version of SSP would be to find a subset such that the respective numbers a_i sum up to t.

Remark 4.24. It is a well-known fact that the subset sum problem is NP-complete.

Remark 4.25. Obviously, being able to solve the search version efficiently also means that we are able to solve the decision version efficiently. The opposite direction is also true. Given an oracle that solves the decision version of SSP, we can use it to tell us whether there is a subset of $\{a_1, \ldots, a_{r-1}\}$ that sums to t. If that is the case, then we do not need a_r and proceed with $\{a_1, \ldots, a_{r-2}\}$. If there is no proper subset that sums to t then this tells us that we need a_r. Then we set $t_1 = t - a_r$ and call the oracle with the set $\{a_1, \ldots, a_{r-1}\}$ and the new target value t_1. This procedure recursively finds a subset that sums up to t.

Another variant of SSP, which is the important one for us in this section, is the following.

Definition 4.26 (Subset sum problem (SSP) with a modulus)**.** Given a set of integers $A = \{a_1, \ldots, a_r\}$, where $r \in \mathbb{N}$, a target $t \in \mathbb{N}$, and a modulus $N \in N$, the *subset sum problem* (SSP) with modulus N is to find a subset $B \subseteq \{1, \ldots, r\}$ such that

$$\sum_{i \in B} a_i \equiv t \mod N. \tag{4.5}$$

We call an input (A, t, N) legal if there exists at least one subset B such that the above equation holds. Although it does not make a huge difference, we assume the input numbers in A to be 'reduced mod N' by which we mean that they are in the range $0 \leq a_i \leq N - 1$ for all i.

Remark 4.27. For the rest of the section we will refer to the subset sum problem with a modulus by the abbreviation SSP. We will only consider SSP instances with $r = \log N + c_r$ for the size of the input sequence of numbers A, with a constant c_r.

The main theorem we want to prove in this section is the following.

Theorem 4.28 (Reduction from DCP to SSP, Thm. 1.3 in [Reg04a])**.** *If there exists an algorithm that solves* $1/\text{poly}(\log N)$ *of the legal inputs to the subset sum problem with modulus* N*, then there exists a quantum algorithm that solves the dihedral coset problem with failure parameter* $f = 1$*.*

Remark 4.29. Since DCP gets easier for larger values of the failure parameter, because the probability that registers are 'bad' is lower, Theorem 4.28 actually gives us a reduction from DCP_f, with $f \geq 1$ to SSP.

We will now describe the basic idea of the DCP to SSP reduction. We assume that we have access to a solver $\mathcal{S}$ for the subset sum problem that can find a solution for a $1/(\log^{c_s} N)$ fraction of the legal inputs, where $c_s \geq 0$ is some constant depending on the solver. Lemma 4.30 shows that, for a randomly chosen input set A to the subset sum problem, at least half of the inputs is legal. Therefore, $\mathcal{S}$ answers a non-negligible fraction of *all* SSP inputs. Also we can assume that the output of $\mathcal{S}$ is always correct since it can be easily checked if the output satisfies Equation (4.5). If the solver is not able to find a solution to SSP, it outputs an error. In the first part of the reduction, Lemma 4.34, we describe two almost identical quantum routines R_1, R_2 which receive as input the qubit registers from the dihedral coset problem with failure parameter 1 as well as q-matching m. A q-matching is a function that divides some set into two parts and every element of one subset has distance q from its 'match' in the other subset. The output of R_1 and R_2 is either 0 or 1 with probabilities which depend on q and d. Recall that d is the 'shift' that we need to find to solve DCP. Moreover, if the matching has a certain 'nice' property, the routines R_1 and R_2 have a high enough success probability, see Lemma 4.33.

In the second part of the reduction (Lemma 4.37) we will use the routines R_1, R_2 to construct a third routine that finds an estimation of a multiple kd of d that has distance of at most $N/(\log^{c_m+1} N)$ from kd modulo N. We will then use this third routine iteratively to find kd mod N, from which we can extract d, the solution to the dihedral coset problem.

In the upcoming quantum reduction the input set A and the target number t to the subset sum problem will be obtained by the measurement of some qubits and are therefore randomly generated. In fact, A will be uniformly distributed. Naturally, the question arises whether there are enough, i.e. a non-negligible fraction, legal inputs for SSP, when the input parameters are chosen randomly. This is answered by the following lemma.

Lemma 4.30. *Let $r, N \in \mathbb{N}$, $N \geq 2$ and $c_r \geq 4$. For any fixed $t \in \{0, \dots, N-1\}$ and $(a_1, \dots, a_r) = A \subseteq \{0, \dots, N-1\}^r$ chosen uniformly at random, the probability that there is a legal input to the subset sum problem is at least $\frac{1}{2}$.*

Proof. We define the Bernoulli random variables

$$X_b \colon \{0, \dots N-1\}^r \to \{0,1\}, \quad a \mapsto \begin{cases} 1, & \text{if} \quad \sum_{i=1}^r a_i b_i \equiv t \mod N, \\ 0, & \text{otherwise,} \end{cases}$$

for each $b \in \{0,1\}^r$, $b \neq 0$. We have

$$\mathrm{E}[X_b] = \frac{1}{N} \quad \text{and} \quad \mathrm{Var}(X_b) = \mathrm{E}\big[X_b^2\big] - (\mathrm{E}[X_b])^2 = \frac{1}{N} - \frac{1}{N^2} < \frac{1}{N}$$

since $\sum_{i=1}^r a_i b_i$ is uniformly distributed on $\{0 \dots, N-1\}$ for every $b \neq 0$, because each a_i is uniformly drawn from $\{0, \dots N-1\}$ by assumption. This is clear for any b with exactly one 1 and follows via induction for all other $b \in \{0,1\}^r$. The random variables X_b are also pairwise independent. To see this, let $b \neq b'$ and assume that b and b' differ on exactly one coordinate, say $b_1 \neq b'_1$. Then without loss of generality $b_1 = 1, b'_1 = 0$. We get for $k, l \in \{0,1\}$:

$$\begin{aligned}
\mathrm{Prob}_{a_1,\dots,a_r}[X_b = k,\ X_{b'} = l] &= \mathrm{E}_{a_1,\dots,a_r}\big[\mathbb{1}_{[X_b=k,\ X_{b'}=l]}\big] \\
&= \mathrm{E}_{a_2,\dots,a_r}\big[\mathrm{E}_{a_1}\big[\mathbb{1}_{[X_b=k,\ X_{b'}=l]}\big]\big] \\
&= \mathrm{E}_{a_2,\dots,a_r}[\mathrm{Prob}_{a_1}[X_b = k,\ X_{b'} = l]] \\
&= \mathrm{E}_{a_2,\dots,a_r}\left[\frac{(N-1)^{1-k}}{N} \cdot \mathbb{1}_{[X_{b'}=l]}\right] \\
&= \mathrm{Prob}_{a_1,\dots,a_r}[X_b = k] \cdot \mathrm{Prob}_{a_1,\dots,a_r}[X_{b'} = l]
\end{aligned}$$

where we have used that, since $b'_1 = 0$, the value of a_1 for the outcome of $X_{b'}$ is irrelevant, and that

$$\mathrm{Prob}_{a_1}[X_b = k] = \frac{(N-1)^{1-k}}{N}$$

for any $a_2, \dots, a_r$. For arbitrary $b \neq b'$ we can make an analogous argument for every coordinate. We let the random variable

$$Y \coloneqq \sum_{\substack{b \in \{0,1\}^r \\ b \neq 0^r}} X_b$$

denote the number of legal inputs for any $a \in \{0, \dots, N-1\}^r$. We want to bound the probability of the event that there are no legal inputs, i.e. $[Y = 0]$. We get

$$\begin{aligned}
\text{Prob}[Y = 0] &\leq \text{Prob}\left[Y < \frac{1}{2} \cdot \frac{2^r - 1}{N}\right] = \text{Prob}\left[Y - \text{E}[Y] < -\frac{1}{2}\text{E}[Y]\right] \\
&= \text{Prob}\left[\text{E}[Y] - Y > \frac{1}{2}\text{E}[Y]\right] \leq \text{Prob}\left[|\text{E}[Y] - Y| > \frac{1}{2}\text{E}[Y]\right] \\
&\leq 4 \cdot \frac{\text{Var}(Y)}{(\text{E}[Y])^2} = 4 \cdot \frac{\text{Var}(X_b)N^2}{2^r - 1} < 4 \cdot \frac{N}{2^r - 1} \leq \frac{8}{2^{c_r}} \leq \frac{1}{2},
\end{aligned}$$

where we used $r = \log N + c_r$ as well as the Chebyshev bound (Proposition 2.38) and

$$\text{E}[Y] = \frac{2^r - 1}{N}, \quad \text{Var}(Y) = (2^r - 1)\text{Var}(X_b),$$

which holds since the random variables X_b are pairwise independent. □

Next we will give the formal definition of a matching, which will be important in the description of the different routines in the quantum reduction.

Definition 4.31. A function $m \colon D \subseteq \{0, \dots, N-1\} \to \{0, \dots, N-1\}$ is called a matching if $m(i) \neq i$ and $m(m(i)) = i$ for all $i \in D$ such that m is also defined on $m(i)$. If, in addition, $|m(i) - i| = q$ holds for all $i \in D$ then m is called a q-matching. Also, we define the sets

$$P_1(m) \coloneqq \{i \in D : m(i) > i\} \quad \text{and} \quad P_2(m) \coloneqq \{i \in D : m(i) < i\}.$$

Later we will be interested in matchings that have a certain property which is important for the success probability of the routines in the quantum reduction.

Definition 4.32. Let $c_m > 0$ be a constant and define

$$\overline{T}_{\text{error}}(A) \coloneqq \{t \in \{0, \dots, N-1\} : \mathcal{S}(A, t) \neq \text{error}\}$$

A matching m with domain D is called *good* if

$$\text{Prob}_A\left[\left|\{i \in D : i, m(i) \in \overline{T}_{\text{error}}(A)\}\right| \geq \frac{N}{\log^{c_m} N}\right] \geq \frac{1}{\log^{c_m} N}.$$

Lemma 4.33. *For any integer $q < \frac{N}{\log^{c_m} N}$ we define q-matchings via*

$$m_{q,1}(t) := \begin{cases} t+q, & \text{if } t \bmod 2q < q,\ t+q < N, \\ t-q, & \text{if } t \bmod 2q \geq q,\ t-q \geq 0. \end{cases}$$

$$m_{q,2}(t) := \begin{cases} t-q, & \text{if } t \bmod 2q < q,\ t-q \geq 0, \\ t+q, & \text{if } t \bmod 2q \geq q,\ t+q < N. \end{cases}$$

Let $c_m > 3c_s$ be a constant, then at least one of the $2\log^{c_m} N$ matchings in the set

$$\mathcal{M}_q := \{m_{q,1} \ldots, m_{\log^{c_m} N \cdot q,1}, m_{q,2} \ldots, m_{\log^{c_m} N \cdot q,2}\}$$

is a good matching.

The proof of Lemma 4.33 is rather technical but not very complicated and can be found in [Reg04a, Lemma 4.5]. We begin with the first part of the quantum reduction.

Lemma 4.34 (First part of the quantum reduction)**.** *Given $r = \log N + c_r$ input registers for the dihedral coset problem with failure parameter $f = 1$, access to an SSP solver $\mathcal{S}$ and a q-matching m, there exist routines R_1, R_2 which either fail or output one bit. In the latter case, the probability of the bit being 1 is $\frac{1}{2} - \frac{1}{2}\cos\left(2\pi q \frac{d}{N}\right)$ for R_1 and $\frac{1}{2} + \frac{1}{2}\sin\left(2\pi q \frac{d}{N}\right)$ for R_2. If the input matching was good, the success probability of the routines is $\Omega(\frac{1}{\log^{c_m} N})$.*

Proof. Recall that the input registers for DCP are either in the state

$$\frac{1}{\sqrt{2}}(|0, x\rangle + |1, x + d \bmod N\rangle),$$

which we called good, or in the state $|b, x\rangle$ with $b \in \{0, 1\}$ and $x \in \{0, \ldots, N-1\}$, which we called bad. The routines R_1 and R_2 will need a total of $(1 + \lceil \log N \rceil)r + r + 1$ qubit registers. The first registers are the DCP input registers. Each of these consists of $1 + \lceil \log N \rceil$ qubits and we require r of them. We assume without loss of generality that the first s of the DCP input registers are bad. The other registers are initialised as $|0, 0\rangle$. The routines begin by applying the quantum Fourier transform to the second register of the DCP input registers. This results in

$$\frac{1}{\sqrt{2N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{kx}{N}} |0, k\rangle + \frac{1}{\sqrt{2N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{k(x+d)}{N}} |1, k\rangle = \frac{1}{\sqrt{2N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{kx}{N}} \left(|0\rangle + e^{2\pi i \frac{id}{N}} |1\rangle\right) |k\rangle$$

for a good register and in

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{kx}{N}} |b\rangle |k\rangle$$

for a bad one.

Keep in mind that this step is performed on all r DCP input registers. The next step is to measure the second register from the DCP input registers. Let $A = (a_1, \dots a_r)$ be the result. Note that A is drawn uniformly from $\{0, \dots, N-1\}^r$, so we can use it as an input for the SSP solver $\mathcal{S}$ later. After the measurement, the whole register is in the state

$$\frac{1}{\sqrt{2^{r-s}}} \left[\bigotimes_{j=1}^{s} \left[e^{2\pi i \frac{a_j x_j}{N}} |b_j\rangle |a_j\rangle \right] \bigotimes_{j=s+1}^{r} \left[e^{2\pi i \frac{a_j x_j}{N}} \left(|0\rangle + e^{2\pi i \frac{a_j d}{N}} |1\rangle \right) |a_j\rangle \right] \right] |0,0\rangle .$$

If we omit the global phase $\prod_{j=1}^{r} \exp(2\pi i a_j x_j / N)$ and the $r\lceil \log N \rceil$ fixed qubits, our remaining $2r+1$ registers are in the state

$$\frac{1}{\sqrt{2^{r-s}}} \left[\bigotimes_{j=1}^{s} |b_i\rangle \bigotimes_{j=s+1}^{r} \left(|0\rangle + e^{2\pi i \frac{a_j d}{N}} |1\rangle \right) \right] |0,0\rangle . \tag{4.6}$$

We will refer to the first of these three registers above (in square brackets) as α and to the other two registers as β and γ. Then, we introduce the notation $t_\alpha \coloneqq \sum_{l=1}^{r} \alpha_l a_l$ for $\alpha \in \{0,1\}^r$ as well as

$$\begin{aligned} M &\coloneqq \{\alpha \in \{0,1\}^r : \alpha_l = b_l,\ 1 \le l \le s\}, \\ L &\coloneqq \{\alpha \in M : t_\alpha \in P_1(m),\ \mathcal{S}(A, t_\alpha) = \alpha,\ \mathcal{S}(A, m(t_\alpha)) \neq \text{error}\}, \\ R &\coloneqq \{\alpha \in M : t_\alpha \in P_2(m),\ \mathcal{S}(A, t_\alpha) = \alpha,\ \mathcal{S}(A, m(t_\alpha)) \neq \text{error}\}. \end{aligned}$$

We perform the following actions on the last two qubit registers containing r and one qubit respectively:

if $\alpha \in M \setminus (L \cup R)$

then exit

else if $\alpha \in L$

then $\beta \leftarrow \alpha$, $\gamma \leftarrow 1$

else if $\alpha \in R$

then $\beta \leftarrow \mathcal{S}(A, m(t_\alpha))$, $\gamma \leftarrow 1$

Note, that this requires exactly one evaluation of t_α, one evaluation of the matching m, and one oracle call to $\mathcal{S}$. After applying the actions above on the state in Equation (4.6), the resulting state is:

$$\frac{1}{\sqrt{2^{r-s}}} \left(\sum_{\substack{\alpha \in M \\ \alpha \notin L \cup R}} e^{2\pi i t_\alpha \frac{d}{N}} |\alpha, 0, 0\rangle + \sum_{\alpha \in L} e^{2\pi i t_\alpha \frac{d}{N}} |\alpha, \alpha, 1\rangle + \sum_{\alpha \in R} e^{2\pi i t_\alpha \frac{d}{N}} |\alpha, \mathcal{S}(A, m(t_\alpha)), 1\rangle \right) .$$

Note that we have multiplied the whole state by the fixed phase $\exp(2\pi i \sum_{i=1}^{s} a_i b_i)$ to make the notation more convenient. Using the 1:1-correspondence between the sets $P_1(m)$ and $P_2(m)$ induced by the matching m, we rewrite the last sum over R into a sum over L. This way we obtain

$$\sum_{\alpha \in L} \left(e^{2\pi i \langle \mathcal{S}(A, m(t_\alpha)), a \rangle \frac{d}{N}} \, |\mathcal{S}(A, m(t_\alpha)), \alpha, 1\rangle \right),$$

which together with the first two sums results in

$$\frac{1}{\sqrt{2^{r-s}}} \left(\sum_{\substack{\alpha \in M \\ \alpha \notin L \cup R}} e^{2\pi i t_\alpha \frac{d}{N}} \, |\alpha, 0, 0\rangle + \sum_{\alpha \in L} e^{2\pi i t_\alpha \frac{d}{N}} \left(|\alpha\rangle + e^{2\pi i q \frac{d}{N}} \, |\mathcal{S}(A, m(t_\alpha))\rangle \right) |\alpha, 1\rangle \right).$$

We also used $\langle \mathcal{S}(A, m(t_\alpha)), a \rangle - t_\alpha = q$ here. The next step in the routines is to measure the registers β and γ. If we measure $\gamma = 0$, the routines output an error. Otherwise the resulting state, omitting the measured qubits and renaming the basis states, is

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i q \frac{d}{N}} \, |1\rangle \right).$$

Now, we come to the difference in the two routines. Routine R_1 applies the Hadamard transform on this state and routine R_2 applies the transformation given by the unitary matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

and afterwards the Hadamard transform. The resulting states are

$$|\psi_1\rangle = \frac{1}{2} \left(\left(1 + e^{2\pi i q \frac{d}{N}} \right) |0\rangle + \left(1 - e^{2\pi i q \frac{d}{N}} \right) |1\rangle \right)$$
$$|\psi_2\rangle = \frac{1}{2} \left(\left(1 + e^{\frac{\pi i}{2} + 2\pi i q \frac{d}{N}} \right) |0\rangle + \left(1 - e^{\frac{\pi i}{2} + 2\pi i q \frac{d}{N}} \right) |1\rangle \right)$$

A simple calculation similar to the one in Example 2.20 shows that the probabilities for a measurement of 1 are as required. So all that remains to show is a high success probability if the matching m is good. The routines succeed if we measure $\gamma = 1$, the probability for that to happen is $|L \cup R|/2^{r-s}$. Let S denote the random variable that counts the number of bad registers. We have

$$\text{Prob}[S = 0] \geq \left(1 - \frac{1}{\log N} \right)^r = \left(1 - \frac{1}{\log N} \right)^{\log N + c_r} = \Omega(1)$$

since $(1 - 1/\log N)^{\log N + c_r} \xrightarrow{N \to \infty} e^{-1}$. We also note that if $S = 0$, the sets

$$L \cup R = \{\alpha \in \{0,1\}^r : t_\alpha \in P_1(m) \cup P_2(m),\ \mathcal{S}(A, t_\alpha) = \alpha,\ \mathcal{S}(A, m(t_\alpha)) \neq \text{error}\}$$

and

$$C \coloneqq \{i \in D : i, m(i) \in \overline{T}_{\text{error}}(A)\}$$

have the same cardinality since every $\alpha \in L \cup R$ gives us a $t_\alpha \in C$ and vice versa. If m was a good matching, we know that, with non-negligible probability, $|C|$ is at least $N/\log^{c_m} N$ which gives us

$$\frac{|L \cup R|}{2^r} \geq \frac{N}{2^{\log N + c_r} \log^{c_m} N} = \Omega\left(\frac{1}{\log^{c_m} N}\right)$$

and finally

$$\text{Prob}[\gamma = 1] \geq \text{Prob}[S = 0] \cdot \text{Prob}[\gamma = 1 | S = 0] = \Omega(1) \cdot \Omega\left(\frac{1}{\log^{c_m} N}\right),$$

which concludes the proof. □

Remark 4.35. Note that the assumption $f = 1$ on the failure parameter is only used to give a lower bound on the probability that none of the input registers are bad. Since this estimation works as well for $f \geq 1$, the routines R_1 and R_2 succeed also for DCP inputs with failure parameter $f \geq 1$ with non-negligible probability.

In the second part of the reduction we will need a little technical lemma.

Lemma 4.36. *Let* $0 < \varepsilon \leq 1/4$ *and* $x, y \in \mathbb{R}$ *with*

$$|x - \cos(\varphi)| \leq \varepsilon, \quad \text{and} \quad |y - \sin(\varphi)| \leq \varepsilon.$$

Then we can compute $\varphi \bmod 2\pi$ *up to an additive error of* 8ε.

Proof. Assume $x \geq 0$. Then

$$\begin{aligned}\left|\frac{y}{1+x} - \frac{\sin(\varphi)}{1+\cos(\varphi)}\right| &= \left|\frac{y - \sin(\varphi) + (y - \sin(\varphi))\cos(\varphi) + (\cos(\varphi) - x)\sin(\varphi)}{(1+x)(1+\cos(\varphi))}\right| \\ &\leq \frac{3\varepsilon}{(1+x)(1+\cos(\varphi))}.\end{aligned}$$

We have $x - \cos(\varphi) = r$ for some $r \in \mathbb{R}$ with $0 \leq |r| \leq \varepsilon$. If $r \geq 0$, we can estimate

$$\frac{3\varepsilon}{(1+x)(1+\cos(\varphi))} \leq \frac{3\varepsilon}{(1+x)(1+x-\varepsilon)} \leq \frac{3\varepsilon}{1-\varepsilon} \leq 4\varepsilon.$$

For $r < 0$ we can do a similar calculation. Using $\sin(2\varphi) = (1 + \cos(2\varphi))\tan(\varphi)$ and the mean value theorem, we get

$$\arctan\left(\frac{y}{1+x}\right) - \frac{\varphi}{2} = \frac{1}{\zeta^2 + 1}\left(\frac{y}{1+x} - \frac{\sin(\varphi)}{1+\cos(\varphi)}\right)$$

for some appropriate ζ, and therefore

$$\left|2\arctan\left(\frac{y}{1+x}\right)-\varphi\right|\leq 8\varepsilon.$$

If we have $x<0$, we instead use $2\operatorname{arccot}(y/(1-x))$ to estimate φ and proceed analogously. □

Lemma 4.37 (Second part of the quantum reduction)**.** *Assume we have access to the routines R_1, R_2 from Lemma 4.34. Then there exists a routine R that finds for any given integer $q<\frac{N}{\log^{c_m} N}$ a multiple $q'\in\{q,\ldots,\log^{c_m} N\cdot q\}$ of q and an estimate r such that*

$$r\in\left[q'd-\frac{N}{\log^{c_m+1} N}, q'd+\frac{N}{\log^{c_m+1} N}\right] \mod N.$$

holds with probability exponentially close to 1.

Proof. Consider the set of matchings $\mathcal{M}_q$ from Lemma 4.33. For any q'-matching m from this set we do the following. With the matching m as input (and the required DCP registers) we call routines R_1, R_2 each $\log^{3c_m+4} N$ times. If the number of successful calls to one of the routines is less than $\log^{2c_m+3} N$ we output an error. At first we show that this happens only with negligible probability. If we go through all the matchings in $\mathcal{M}_q$, which is still efficient since the size of $\mathcal{M}_q$ is polynomial in $\log N$, we are guaranteed to find at least one good matching. According to Lemma 4.34 the success probability of R_1 and R_2 with a good matching is at least $c_g\frac{1}{\log^{c_m} N}$ for a constant $c_g>0$. The probability that routine R fails with a good matching is at most

$$\left(1-c_g\frac{1}{\log^{c_m} N}\right)^{\log^{c_m+1} N},$$

since $\log^{c_m+1} N$ calls to the subroutines have to fail in order for R to fail. This probability, however, is exponentially small. So we assume that the number of successful calls to the routines, say $\mathcal{R}_1, \mathcal{R}_2$ respectively, are both at least $\log^{2c_m+3} N$. We denote the averages of their respective outputs as x and y. Let us think of the i-th outcome of routines R_1 and R_2 as Bernoulli random variables $R_{1,i}$ and $R_{2,i}$ and let

$$X=\frac{1}{\mathcal{R}_1}\sum_{i=1}^{\mathcal{R}_1} R_{1,i},\quad Y=\frac{1}{\mathcal{R}_2}\sum_{i=1}^{\mathcal{R}_2} R_{2,i}.$$

The respective random variables in the two families of random variables $(R_{1,i})_{1\leq i\leq\mathcal{R}_1}$ and $(R_{2,i})_{1\leq i\leq\mathcal{R}_2}$ are independent and identically distributed. Using the Chernoff-Hoeffding bound (Prop. 2.37), we get

$$\begin{aligned}&\operatorname{Prob}_{(R_{1,i})_{1\leq i\leq\mathcal{R}_1}}\left[\left|X-\left(\frac{1}{2}-\frac{1}{2}\cos(2\pi q'\frac{d}{N})\right)\right|>\frac{1}{c_e\log^{c_m+1} N}\right]\\ &\leq 2e^{-2\frac{\mathcal{R}_1}{c_e^2\log^{2c_m+2} N}}\leq 2e^{-2\frac{\log N}{c_e^2}}.\end{aligned}$$

Since we assumed that more than $\log^{2c_m+3} N$ calls to R_1 were successful. Note that this probability is exponentially small (in $\log N$). Hence, with probability exponentially close to 1, $x' \coloneqq 1 - 2x$ is an approximation of $\cos\left(2\pi q' \frac{d}{N}\right)$ up to an additive error of at most $2/(c_e \log^{c_m+1} N)$. A similar bound holds for y and therefore $y' \coloneqq 2y - 1$ is an approximation of $\sin\left(2\pi q' \frac{d}{N}\right)$ with the same bound for the additive error as x'. For c_e large enough, to make sure that the additive error is not bigger than 2π, this leads to an estimation of $q'd \bmod N$ with an additive error of at most $N/\log^{c_m+1} N$ by using Lemma 4.36 and multiplying everything with $\frac{N}{2\pi}$. □

To complete the description of the quantum reduction from DCP to SSP, we prove Theorem 4.28 by showing how to find the shift d in the good DCP registers.

Proof of Theorem 4.28. We claim that Algorithm 5 solves the dihedral coset problem with failure parameter $f \geq 1$ by making use of an oracle for the subset sum problem (in the subroutines R_1, R_2).

Algorithm 5 DCP_1 to SSP reduction

INPUT: Access to subroutine R from Lemma 4.37
OUTPUT: The hidden shift d in the DCP input registers, i.e the solution to DCP.

1: Call R with input $q = 1$. Let $(\hat{r}, \hat{q})$ be the output and set $q_1 = 1$, $r_1 = \hat{r}$.
2: **while** $q_i < \frac{4N}{\log^{c_m+1} N}$ **do**
3: Call R with input $q = 2q_i\hat{q}$. Let (r', q') be the output.
4: Set $q_{i+1} = \frac{q'}{\hat{q}}$, $a' = \left\lfloor \left(\frac{q_{i+1}}{q_i} r_i - r'\right)/N \right\rceil$ and $r_{i+1} = N \cdot a' + r'$.
5: **end while**
6: Compute $\hat{d} = \left\lfloor \frac{r_i}{q_i} \right\rceil$.
7: Output $d = \hat{d}\hat{q}^{-1}$.

We analyse the algorithm and prove its correctness. Let $d' \coloneqq d\hat{q}$. We note that after the initial call to routine R, the output r_1 satisfies

$$r_1 \in \left[d' - \frac{N}{\log^{c_m+1} N}, d' + \frac{N}{\log^{c_m+1} N} \right] \mod N.$$

We assume that we have an estimate r_i in the i-th step such that

$$r_i \in \left[q_i d' - \frac{N}{\log^{c_m+1} N}, q_i d' + \frac{N}{\log^{c_m+1} N} \right] \mod q_i N.$$

Note that this is fulfilled in the first step since $q_1 = 1$. Then we also have

$$\frac{q_{i+1}}{q_i} r_i \in \left[q_{i+1} d' - \frac{2N}{\log N}, q_{i+1} d' + \frac{2N}{\log N} \right] \mod q_{i+1} N$$

since $q_{i+1}/q_i \leq 2\log^{c_m} N$, and because of $q' \in \{2q_i\hat{q}, \ldots, 2\log^{c_m} N \cdot q_i\hat{q}\}$. The output of the call to routine R in the i-th step gives us

$$r' \in \left[q_{i+1}d' - \frac{N}{\log^{c_m+1} N}, q_{i+1}d' + \frac{N}{\log^{c_m+1} N}\right] \mod N.$$

The idea behind computing r_{i+1} the way we do is to obtain an estimate that satisfies

$$r_{i+1} \in \left[q_i d' - \frac{N}{\log^{c_m+1} N}, q_i d' + \frac{N}{\log^{c_m+1} N}\right] \mod q_{i+1}N.$$

To see that this indeed works out, we write

$$r' = q_{i+1}d' + \varepsilon_1 + Na_1 \text{ and } \frac{q_{i+1}}{q_i}r_i = q_{i+1}d' + \varepsilon_2 + q_{i+1}Na_2,$$

for some integers a_1, a_2 and $\varepsilon_1 \in [-\frac{N}{\log^{c_m+1} N}, \frac{N}{\log^{c_m+1} N}]$ and $\varepsilon_2 \in [-\frac{2N}{\log N}, \frac{2N}{\log N}]$. First we observe that for $N > 64$ we have

$$|\varepsilon_2 - \varepsilon_1| \leq \frac{N}{\log^{c_m+1} N} + \frac{2N}{\log N} < \frac{N}{2}.$$

Thus,

$$a' = \lfloor ((a_2q_{i+1} - a_1)N + \varepsilon_2 - \varepsilon_1)/N \rceil = a_2q_{i+1} - a_1.$$

Therefore,

$$r_{i+1} = N \cdot a' + r' = N(a_2q_{i+1} - a_1) + q_{i+1}d' + \varepsilon_1 + Na_1 \equiv q_{i+1}d' + \varepsilon_1 \mod q_{i+1}N$$

as required. Since q_i at least doubles in each step, the while loop is left after a polynomial number of $\log N - \log\log^{c_m+1} N + 3 = \mathcal{O}(\log N)$ steps. We leave the while loop when $q_i \geq 4N/(\log^{c_m+1} N)$, and in this case we have

$$\frac{r_i}{q_i} \in \left[d' - \frac{N}{q_i \log^{c_m+1} N}, d' + \frac{N}{q_i \log^{c_m+1} N}\right] \subseteq \left[d' - \frac{1}{4}, d' + \frac{1}{4}\right] \mod N.$$

Since this is an interval of width $1/2$, rounding to the nearest integer yields $d' = d\hat{q}$. □

Remark 4.38. It may be seen that the DCP_1 to SSP reduction requires time $\mathcal{O}(\log^{4c_m+7} N)$ together with $\mathcal{O}(\log^{4c_m+5} N)$ oracle calls and we need to store $\mathcal{O}(\log^{3c_m+6} N)$ qubits. We require $\mathcal{O}(\log^{3c_m+4} N)$ classical space, while the success probability is exponentially close to 1.

5 Algorithms for DHSP and SSP

In this chapter we will take a look at some quantum algorithms from the literature for solving DHSP and SSP. In Section 5.1 we consider an algorithm for DHSP due to Kuperberg ([Kup05]) and in Section 5.2 an algorithm for solving SSP due to Bernstein et al. ([BJLM13]). For a more detailed discussion of the algorithms as well as an in-depth analysis of their complexity we refer to the original papers ([Kup05],[Kup11],[Reg04b] and [BJLM13]).

5.1 DHSP Algorithms

In this section we will review some quantum algorithms for solving the dihedral hidden subgroup problem. At first we will discuss two 'older' algorithms due to Kuperberg ([Kup05]) and Regev ([Reg04b]). We will not analyse these two algorithms in detail, since they are generalised and outperformed by another DHSP algorithm, also due to Kuperberg ([Kup11]). However, the basic ideas behind all three algorithms are similar and discussing the first two serves as a good introduction to the topic.

5.1.1 Kuperberg's First Algorithm and Regev's Improvement

All three algorithms are concerned with the task of finding some kind of 'hidden shift', which is a bit string on $\lceil \log N \rceil$ bits. As usual for notational convenience, we will consider the case where N is a power of 2. The task is to solve the hidden subgroup problem in the dihedral group D_N, defined in Section 2.4. Recall that the hard cases of DHSP are the ones where the hidden subgroup H is of the form $H = \{(0,0),(1,d)\}$. The hidden bit string we have to find to solve DHSP in this case is d. One idea that all three DHSP algorithms share is that it suffices to determine the parity of d. This is due to the fact that once we know the least significant bit of d, we can consider another DHSP instance with a group isomorphic to $D_{N/2}$ and another hiding function $f'(s,r)$ defined as $f(s,2r)$ if d was even and $f(s,2r+1)$ if d was odd. This functions hide the subgroup $\{(0,0),(1,d/2)\}$ or $\{(0,0),(1,(d-1)/2)\}$, respectively. Proceeding inductively, we can determine the whole string d. Another similarity of the three algorithms is that they all

aim to create a qubit state of the form

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^d |1\rangle). \tag{5.1}$$

A measurement in the basis $|\pm\rangle$ (recall Example 2.20) reveals the parity of d. In order to create the state in Equation (5.1), the algorithms perform different kinds of combinational operations on qubit states of the form

$$|\psi_k\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i dk/N} |1\rangle). \tag{5.2}$$

In the description of Kuperberg's second algorithm, these states will be generalised to the so-called *phase vectors*, but for now we will be dealing with these states, which can be produced in the following way. Recall that in an instance of HSP we assume to have access to a unitary operator U_f that computes the hiding function f. The following steps show how we can produce a state of the form (5.2):

$$\begin{aligned} |0,0\rangle |0\rangle &\xmapsto{\mathcal{F}_{D_N}} \frac{1}{\sqrt{2N}} \sum_{\substack{b\in\{0,1\} \\ x\in\{0,\dots,N-1\}}} |b,x\rangle |0\rangle \\ &\xmapsto{U_f} \frac{1}{\sqrt{2N}} \sum_{\substack{b\in\{0,1\} \\ x\in\{0,\dots,N-1\}}} |b,x\rangle |f(b,x)\rangle . \end{aligned}$$

A measurement of the third register with result, say, y leaves the first two register in a state

$$\frac{1}{\sqrt{2}} |0,y\rangle + |1,y+d \mod N\rangle \xmapsto{\mathcal{F}_{\mathbb{Z}/N\mathbb{Z}}} \frac{1}{\sqrt{2N}} \sum_{j=0}^{N-1} e^{2\pi i y j/N} |0,j\rangle + \sum_{j=0}^{N-1} e^{2\pi i (y+d) j/N} |1,j\rangle .$$

Then a measurement of the second register with result, say, k produces the desired state $|\psi_k\rangle$. Note that the value k is known to us since we just measured it. We can think of this value as a label for the qubit state $|\psi_k\rangle$, thus the goal of the algorithms becomes the creation of a qubit state whose label has all but the leftmost bit of this label zeroed out. Now, we will describe briefly the way Kuperberg's first algorithm, as well as Regev's algorithm, achieves that goal.

In Kuperberg's first algorithm, a chain of routines is used to combine states of the form (5.2) in such a way, that the combined state has the label $k_1 - k_2$ with a probability of $1/2$. Each routine would get qubit states from his predecessor such that the labels of these states have already a number of bits zeroed out. The routine saves all the qubits and their respective labels in a pile and for each new incoming qubit, it checks whether the new label coincides with one already in the pile on the last, say m, bits that have not been zeroed out so far. If this is the case, the routine combines the two qubits and obtains with probability $1/2$ a new qubit whose label has m additional bits zeroed out.

Provided the decomposition $n = m^2 + 1$ it can be shown that this leads to a running time of $2^{\mathcal{O}(\sqrt{n})} = 2^{\mathcal{O}(\sqrt{\log N})}$. Thus, Kuperberg's first algorithm was the first one solving DHSP on a quantum computer in subexponential time. However, each of the m routines has to wait for $\mathcal{O}(2^m)$ qubits entering its pile on average. Thus, the algorithm does also require $2^{\mathcal{O}(\sqrt{\log N})}$ space.

The major improvement by Regev was to reduce the required space to a polynomial amount (in $\log N$), while obtaining an only slightly worse running time of $2^{\mathcal{O}(\sqrt{\log N \log\log N})}$. Regev achieves this by introducing a new technique that allows his algorithm to zero out blocks of bits without having to wait for a match to occur in the bits of the incoming labels. Since this technique is also adopted by Kuperberg in his later algorithm, we will talk about it briefly. Regev decomposes n to $n = 1 + ml$ with $m = \mathcal{O}(\sqrt{n/\log n})$ and $l = \mathcal{O}(\sqrt{n \log n})$. Instead of waiting and comparing labels, he just tensors $\mathcal{O}(l)$ qubits together and computes the function $\langle \vec{b}, \vec{k} \bmod 2^l \rangle$ in an ancilla register. Here $\vec{k}$ is the vector consisting of the $\mathcal{O}(l)$ labels and $\vec{b}$ is an arbitrary bit string of $\mathcal{O}(l)$ bits. A measurement of the ancilla register leaves us with a superposition over all the bit strings such that $\langle \vec{b}, \vec{k} \bmod 2^l \rangle$ coincides with the measurement, say z. The main cost of Regev's algorithm is to compute the set of all these 'good' bit strings. This is done by brute force, thus resulting in the slightly higher running time of $2^{\mathcal{O}(l)} = 2^{\mathcal{O}(\sqrt{n \log n})}$. Finally, a projective measurement on the subspace spanned by two (arbitrary) of these 'good' bit strings is performed to produce a qubit state of the form

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i d \langle \vec{b^2} - \vec{b^1}, \vec{k} \rangle / N} |1\rangle)$$

after renaming the basis states. The label of this state is $\langle \vec{b^2} - \vec{b^1}, \vec{k} \rangle$ and, clearly, its last l significant bits are all 0. By changing the utilised modulus, a similar combination can be performed on other l-bit blocks, e.g. 2^{2l} to zero out the second l-bit block from the right. The presented ideas, introduced in [Reg04b], are picked up and generalised by Kuperberg in his second DHSP algorithm.

5.1.2 Kuperberg's Second Algorithm for DHSP

Now, we will review Kuperberg's second algorithm for DHSP. The two papers by Kuperberg which provide more details than our summary are [Kup05] and [Kup11].

Recall that for $N \in \mathbb{N}$ the dihedral group D_N is given as the semi-direct product

$$D_N = \mathbb{Z}/2\mathbb{Z} \ltimes \mathbb{Z}/N\mathbb{Z}.$$

In a hard instance of DHSP we are given a function h that hides cosets of some subgroup $H \subset D_N$ which is of the form $\{(0,0),(0,s)\}$. However, in this section we will use the equivalent notation that was introduced in Definition 2.56. So from now on we write $H = \langle yx^s \rangle$ for the hidden subgroup. By considering the maps

$$f(a) \coloneqq h(x^a) \quad \text{and} \quad g(a) \coloneqq h(yx^a),$$

we see that DHSP is actually equivalent to the so-called *abelian hidden shift problem*, i.e given an abelian group A (here we have $A = \mathbb{Z}/N\mathbb{Z}$) and two injective functions f, g with $f(a) = g(a + s)$ for all $a \in A$, the task is to find s. For more information and details see [Kup05] Section 6. The situation that we assume is the following: We are given elements $r_j \in \mathbb{Z}/N\mathbb{Z}$ ($r_0 = 0$) for $0 \leq j \leq l-1$ for some $l \in \mathbb{N}$ and a set of injective functions $\{f_j\}$

$$f_j \colon \mathbb{Z}/N\mathbb{Z} \to X$$

satisfying

$$f_0(a + r_j s) = f_j(a)$$

for all j and all $a \in \mathbb{Z}/N\mathbb{Z}$, for a unique s. Here X is just some arbitrary (unstructured) set. In the case $l = 2$, this is equivalent to r_1 being invertible modulo N. Note that the abelian hidden shift problem is just the special case $l = 2$, $r_1 = 1$. We will also assume that we have access to an oracle U_f that performs the action

$$|j, a, 0\rangle \overset{U_f}{\longmapsto} |j, a, f_j(a)\rangle .$$

Let $|J\rangle$ and $|A\rangle$ denote the uniform superpositions over the sets $J = \{0, \ldots, l-1\}$ and $A = \mathbb{Z}/N\mathbb{Z}$. For simplicity, we also assume $N = 2^n$. The first thing Kuperberg's algorithm does is to compute a set of many so-called *phase vectors*, which is done in the following way. First we use the oracle to apply U_f. This yields

$$|J\rangle |A\rangle |0\rangle \overset{U_f}{\longmapsto} \frac{1}{\sqrt{lN}} \sum_{j,a} |j, a, f_j(a)\rangle .$$

Measuring and discarding the third register with result say $f_0(t)$ leads to

$$\frac{1}{\sqrt{l}} \sum_j |j, t - r_j s\rangle \overset{\mathcal{F}_{\mathbb{Z}/N\mathbb{Z}}}{\longmapsto} \frac{1}{\sqrt{l}} \sum_j |j\rangle \frac{1}{\sqrt{N}} \sum_k e^{2\pi i (t - r_j s)k/N} |k\rangle$$

and a measurement of the second register with result say b yields the state

$$|\psi\rangle = \frac{1}{\sqrt{l}} \sum_j e^{-2\pi i r_j s b/N} |j\rangle .$$

Definition 5.1. Let $l \in \mathbb{N}$, $N = 2^n$ for some $n \in \mathbb{N}$. A quantum state of the form

$$|\psi\rangle = \frac{1}{\sqrt{l}} \sum_{j=0}^{l-1} e^{2\pi i s b(j)/2^n} |j\rangle \tag{5.3}$$

is called a *phase vector* of length l. The coefficients $b(j)$ are called *phase multiplier functions*. If $m \leq n$, such that 2^m divides $b(j_1) - b(j_2)$ for all $j_1 \neq j_2$ then the state in Equation (5.3) can be written as

$$|\psi\rangle = \frac{1}{\sqrt{l}} \sum_{j=0}^{l-1} e^{2\pi i s b(j)/2^h} |j\rangle$$

with $h = n - m$. The number h is called a *height* of the phase vector.

Remark 5.2. Note that, since we do not assign the smallest possible height to a phase vector, we can view phase vectors of height h as well as phase vectors of height h' for any $h \leq h' \leq n$.

Kuperberg's algorithm uses the so-called *collimation* algorithm to produce more favourable phase vectors.

Algorithm 6 Collimation algorithm for phase vectors

INPUT: Phase vectors $|\psi_1\rangle, \ldots, |\psi_r\rangle$ of height h with lengths $l_1, \ldots, l_r$ and a collimation parameter m.
OUTPUT: A phase vector $|\Psi\rangle$ of height $h - m$ and length $l' \approx 2^{-m} \prod_{i=1}^{r} l_r$.

1: Compute $|\psi'\rangle = |\psi_1\rangle \otimes \ldots \otimes |\psi_r\rangle$.
2: Compute $b(\vec{j}) \coloneqq \sum_{j=1}^{r} b_j \bmod 2^m$ in another m qubit register, measure it and save the outcome, say c.
3: Compute the set of indices $\widehat{J} \subseteq J_1 \times \ldots \times J_r$ such that $b(\vec{j}) = c$ for $\vec{j} \in \widehat{J}$, set $l' = |\widehat{J}|$.
4: Renumber the resulting vector to form $|\Psi\rangle$ of length l' with indexing set $\{0, \ldots, l'-1\}$.

Remark 5.3. The explicit process of renumbering is done in the following way: We pick a bijection $\pi\colon \widehat{J} \to \{0, \ldots, l' - 1\}$ (Kuperberg uses the special permutation that also sorts the new phase multiplier table containing the values $b(j_1) + b(j_2)$ by the m lowest bits, see the proof in [Kup11, Prop. 4.2]) and apply the subunitary operator $U_\pi\colon \mathbb{C}^{l_1} \otimes \ldots \otimes \mathbb{C}^{l_r} \to \mathbb{C}^{l'}$ that annihilates all vectors orthogonal to $\mathbb{C}[\widehat{J}] \subseteq \mathbb{C}[J_1 \times \ldots \times J_r]$ and acts as π on all vectors in $\mathbb{C}[\widehat{J}]$. Here we denote by $\mathbb{C}[X]$ the Hilbert space with an orthonormal basis indexed by elements of some finite set X. In the case of X being a group G, $\mathbb{C}[G]$ is called the *group algebra*. The estimation of the length of the new phase vector $|\Psi\rangle$ is done heuristically by assuming that $b(\vec{j})$ is uniformly distributed on $\{0, \ldots, 2^m - 1\}$.

Regarding the complexity of the collimation algorithm we have the following results.

Proposition 5.4 (Proposition 4.2 in [Kup11])**.** *Let $r = 2$, $l_{\max} \coloneqq \max(l_1, l_2, l')$ with l' as the length of the collimated phase vector $|\Psi\rangle$. Then Algorithm 6 requires*

- $\widetilde{\mathcal{O}}(l_{\max})$ *classical time,*
- $\mathcal{O}(l_{\max} h)$ *classical space,*
- $\mathcal{O}(l_{\max} \max(m, \log l_{\max}))$ *classical space with quantum access,*
- $\operatorname{poly}(\log l_{\max})$ *quantum time and*
- $\mathcal{O}(\log l_{\max})$ *quantum space.*

We can now give a complete description of Kuperberg's Collimation sieve algorithm:

Algorithm 7 Kuperberg's collimation sieve algorithm

INPUT: A desired height h, a collimation parameter $m = m(h)$, a branching parameter $r = r(h)$, a starting minimum length l_0 and an oracle that computes U_f.
OUTPUT: A phase vector of height h.

1: **if** $h = n$ **then**
2: Extract phase vectors $|\psi_1\rangle, \ldots, |\psi_s\rangle$, as described in the beginning, and tensor them together such that the length of $|\Psi\rangle = |\psi_1, \ldots, \psi_s\rangle$ is at least l_0 and output $|\Psi\rangle$.
3: **else**
4: Obtain recursively and sequentially phase vectors $|\psi_1\rangle, \ldots, |\psi_r\rangle$ of height $h + m$ and collimate them with Algorithm 6 to produce a phase vector $|\Psi\rangle$ of height h and return this vector.
5: **end if**

When Algorithm 7 is run with $h = 1$, it produces the state

$$|\psi\rangle = \frac{1}{\sqrt{l}} \sum_{j=0}^{l-1} (-1)^{b(j)s} |j\rangle .$$

We compute a non-empty maximal set $X \subseteq \{0, \ldots, l-1\}$, such that $b(j)$ takes only the values 0 and 1 equally often for $j \in X$ and perform a partial measurement to determine whether $|\psi\rangle \in \mathbb{C}[X]$. If no such set X exists or $|\psi\rangle \notin \mathbb{C}[X]$, then we run Algorithm 7 again. However, if $|\psi\rangle \in \mathbb{C}[X]$, then $|\psi\rangle$ contains at least one qubit factor of the form $(|0\rangle + (-1)^s |1\rangle)/\sqrt{2}$ and a measurement in the Hadamard basis $|\pm\rangle$ yields the parity of the hidden shift s. We can now inductively compute the whole string s by noting that D_N contains the two subgroups

$$U_0 = \langle x^2, y \rangle \quad \text{and} \quad U_1 = \langle x^2, yx \rangle,$$

which are both isomorphic to $D_{N/2}$ and fulfil $H \subseteq U_b$ if and only if $b \equiv s \bmod 2$. This concludes the description of the algorithm.

Remark 5.5. Regarding time and space complexity, a heuristic analysis Algorithm 7 shows that its running time can be estimated as $\widetilde{\mathcal{O}}(2^{\sqrt{2 \log N}})$, see [Kup11] Section 4.5. Regarding the parameters m and r of Algorithm 7, we remark that increasing m saves quantum time at the cost of classical time and space, whereas increasing r saves quantum space at the cost of quantum time. For more details in that regard we refer to Sections 1, 2, 4 of [Kup11].

Since we are interested in a special application of this algorithm, we will also describe a slightly modified version that will be suitable for solving DHSP. Recall that for DHSP we only assume the existence of two hidden shifts f_0 and f_1. For notational convenience we assume $\log N = n = m^2 + 1$. The modified algorithm works as follows.

Algorithm 8 Modified version of Kuperberg's Algorithm for DHSP

INPUT: A number $m = \mathcal{O}(\sqrt{\log N})$, $2^{\mathcal{O}(\sqrt{\log N})}$ DHSP input registers, access to a subroutine that implements Algorithm 6.
OUTPUT: The parity of d with constant probability.

1: First stage: Form two phase vectors of length $l_0 = 2^{m+1}$, each from $m+1$ of the input qubits.
2: Call Algorithm 6 to produce another phase vector with output length l_1.
3: Divide the index set of the output vector from the previous step into segments of length l_0 and a left over segment of length l_2.
4: Perform a partial measurement corresponding to this partition into segments.
5: **if** the length of the measured segment is l_2 **then**
6: Discard the phase vector.
7: **else**
8: Store the produced phase vector of length l_0.
9: **end if**
10: Intermediate stages: Use the phase vectors produced in the previous stages to obtain recursively and sequentially phase vectors whose phase multiplier functions are more and more aligned by using the same procedure as described in the first stage.
11: Last stage: After aligning the last m bits of the phase multipliers and producing a phase vector of length l_1 using Algorithm 6, divide the index set into segments of two elements each (and one left over segment if l_1 is odd) and perform a partial measurement corresponding to this partition.
12: **if** the produced phase vector is of the form (5.1) **then**
13: Perform a measurement in the Hadamard basis to find the parity of d.
14: **else**
15: Restart from the beginning.
16: **end if**

Proposition 5.6. *Algorithm 8 requires quantum time and classical space* $2^{\mathcal{O}(\sqrt{\log N})}$ *and quantum space* $\mathcal{O}(\log \log N)$ *and has a non-negligible success probability.*

Proof. See [Kup11, Prop. 4.5]. □

5.2 A Quantum Algorithm for SSP

In this section we will review an algorithm for the subset sum problem to capitalise on Theorem 4.28. The algorithm which is due to Bernstein et al. is an improvement of an algorithm by Howgrave-Graham and Joux ([HGJ10]) and runs in time $\mathcal{O}(2^{(0.241\ldots+o(1))n})$. It combines different techniques for solving the subset sum problem, such as 'Left-right split', 'quantum walks', 'moduli' and 'representations'. We will briefly review these differ-

ent ideas and give an overview of their combination in order to obtain the complete algorithm afterwards. For a more detailed approach, we refer to the original paper [BJLM13].

Recall the search version of SSP: Given a sequence of integers $A = (a_1, \dots a_n)$ for $n \in \mathbb{N}$ and a target value t, the goal is to find a set of indices $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} a_i = t.$$

Remark 5.7. A variant of SSP we already know is the introduction of a modulus (recall Definition 4.26). Another variation of SSP is the following: We are given an additional natural number $1 \leq w \leq n$ and are asked to find solutions, i.e. an index set I, such that $|I| = w$. This problem is called the subset sum problem with weight w. Of course, we can also mix these two SSP variations and, indeed, we will do so later on.

Let $\Sigma \colon \mathcal{P}(\{1, \dots, n\}) \to \mathbb{N}$ denote the function that maps the index set I to the value $\sum_{i \in I} a_i$. We can reformulate the subset sum problem as the task to find a root of the function $\Sigma - t$. Note that we could use Grover's algorithm ([Gro96]) to find a root of this function, using about $2^{n/2}$ quantum evaluations of $\Sigma - t$. However, this is not optimal.

Now, we will briefly review the different techniques mentioned above. Throughout this section, we divide n by different numbers and assume that the result is an integer. This is purely for notational convenience. Everything we do can be generalised for arbitrary $n \in \mathbb{N}$.

The left-right split: To find an indexing set $I \subseteq \{1, \dots, n\}$ with $\Sigma(I) = t$, enumerate all sets $I_1 \subseteq \{1, \dots, n/2\}$ and compute the first list $L_1 \coloneqq \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, \dots, n/2\}\}$. Afterwards, we enumerate all sets $I_2 \subseteq \{n/2 + 1, \dots n\}$ and compute the second list $L_2 \coloneqq \{(t - \Sigma(I_2), I_2) : I_2 \subseteq \{n/2 + 1, \dots, n\}\}$. After the computation of each $t - \Sigma(I_2)$, we check for a collision with elements of L_1 in the first coordinate via binary search. If there is a collision, we output $I = I_1 \cup I_2$ and terminate. Otherwise there is no solution to the subset sum problem.

A quantum variant of this procedure is the following. This time consider the first list $L_1' \coloneqq \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, \dots, n/3\}\}$ and define the function

$$f \colon \mathcal{P}(\{n/3+1, \dots, n\}) \to \{0, 1\}, \quad f(I_2) \coloneqq \begin{cases} 0, & \text{if } t - \Sigma(I_2) \text{ is a first coordinate in } L_1', \\ 1, & \text{otherwise.} \end{cases}$$

Since binary search in L_1' takes time $\mathcal{O}(n)$, so does one evaluation of f. We can use Grover's algorithm to find a root of f. This leads to costs of $\mathcal{O}(n2^{n/3})$ in time and space for the quantum version in contrast to $\mathcal{O}(n2^{n/2})$ for the classical version.

Quantum walks: To find a unique collision of some function f with inputs of b-bits, i.e. the only two values $x \neq y$ with $f(x) = f(y)$, we search through the Johnson graph of r subsets of the set of b-bit strings. Here $r \approx 2^{2b/3}$ is an algorithm parameter

and the Johnson graph is the graph whose vertices are the subsets of all b-bit strings with cardinality r. The edges of the Johnson graph are between subsets which differ in exactly one element. A quantum walk algorithm due to Ambainis ([Amb07]), which acts on states of the form $|S\rangle|f(S)\rangle|T\rangle|f(T)\rangle$, where S and T are adjacent vertices of the Johnson graph and $f(S), f(T)$ are the multisets of the images of S and T under f, can be used to find the unique collison with $\mathcal{O}(2^{2b/3})$ evaluations of f. An important aspect in this context is the way we store the subsets of cardinality r as well as the multiset of their images. Bernstein et al. propose a radix tree for that purpose. As for Grover's algorithm, there are also generalisations for the cases when f has a known number of $k > 1$ collisons or an arbitrary unknown number of collisions. For more details we refer to [BJLM13] Section 3 and [Amb07].

Moduli: In Section 4.3, we have already encountered SSP with a modulus, recall Definition 4.26. We can modify the left-right split approach by choosing a positive integer modulus $M \approx 2^{n/4}$, some $m \in \{0, \ldots, M-1\}$ and computing

$$L_1^M := \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, \ldots, n/2\},\ \Sigma(I_1) \equiv m \mod M\}.$$

This is an instance of a subset sum problem of size $n/2$ with modulus M. This new subset sum instance will be handled by solving the family of subset sum problems (without a modulus!) given by the same input set $A = (a_1, \ldots, a_r)$ (recall that we assume that $0 \leq a_i \leq M-1$ for all i) and the different targets $m, m+M, \ldots, m+(n/2-1)M$. These SSP instances will be handled by a suitable subroutine, e.g. the original left-right split. Note that the subroutine has to find all solutions to the respective SSP instances to compute L_1^M. So the number of solutions must be added to the costs of the subroutine. However, since the a_i are assumed to be uniformly random, we have $2^{n/2}/M \approx 2^{n/4}$ solutions for $\Sigma(I_1) \equiv m \bmod M$ for every m on average. Since the running time of the left-right split algorithm is also $\mathcal{O}(n2^{n/4})$ in this case, we have $\mathcal{O}(n2^{n/4})$ for the subroutine. The computation of

$$L_2^M := \{(\Sigma(I_2), I_2) : I_2 \subseteq \{n/2, \ldots, n\},\ \Sigma(I_2) \equiv t - m \mod M\}$$

is done similarly. Again, we store L_1^M in a sorted table and for each element in L_2^M we check whether $t - \Sigma(I_2)$ appears in that table. If we find no collision, we just try a different value for m, until in the worst case all $m \in \{0, \ldots, M-1\}$ are exhausted, which produces a total cost of $\mathcal{O}(n2^{n/2})$.

Representations: Representation in this context refers to a decomposition of an (index) set and should not be confused with representations of groups. In the original left-right split we partitioned some index set I into two index sets, where we split the original set of indices in a left and a right half. However, we could also split the index set I into two parts I_1, I_2, where $|I_1| = |I_2| = |I|/2$ in a different way. The main advantage of this is that there are more possibilities for us to decompose the set I, and since we do not care how we decompose the set I, as long as we find it, this will improve the probability of the algorithm being successful.

The central idea of the improved algorithm by Bernstein et al. is the following. Since SSP can be thought of as a collision finding problem, as our previous discussion illustrates, they start with a classical collision finding algorithm for a function f that computes all possible values of f. They introduce an algorithm parameter r and adapt the algorithm such that it only computes $f(S)$ for a subset S of all inputs such that $|S| = r$. This requires less time, but results in a lower success probability. Then Bernstein et al. apply the quantum walk technique to walk through adjacent vertices in the Johnson graph of subsets of size r. The required number of quantum walk steps depends on the success probability of the previous algorithm. For this approach, the computation of the 'lower probability algorithm' has to be expressed in a data structure that allows efficient movement between adjacent sets. As already mentioned, they choose to store sets in augmented radix trees.

We will now give a description of the modified Howgrave-Graham-Joux algorithm, combining all ideas presented so far. For notational convenience, we assume that we are only searching for solutions of the SSP with weight $|I| = n/2$. Other weights can be handled by adjusting the set sizes in the following algorithm appropriately and unknown weights are handled by trying all possible $\mathcal{O}(n)$ weights.

Howgrave-Graham-Joux algorithm (unmodified version):

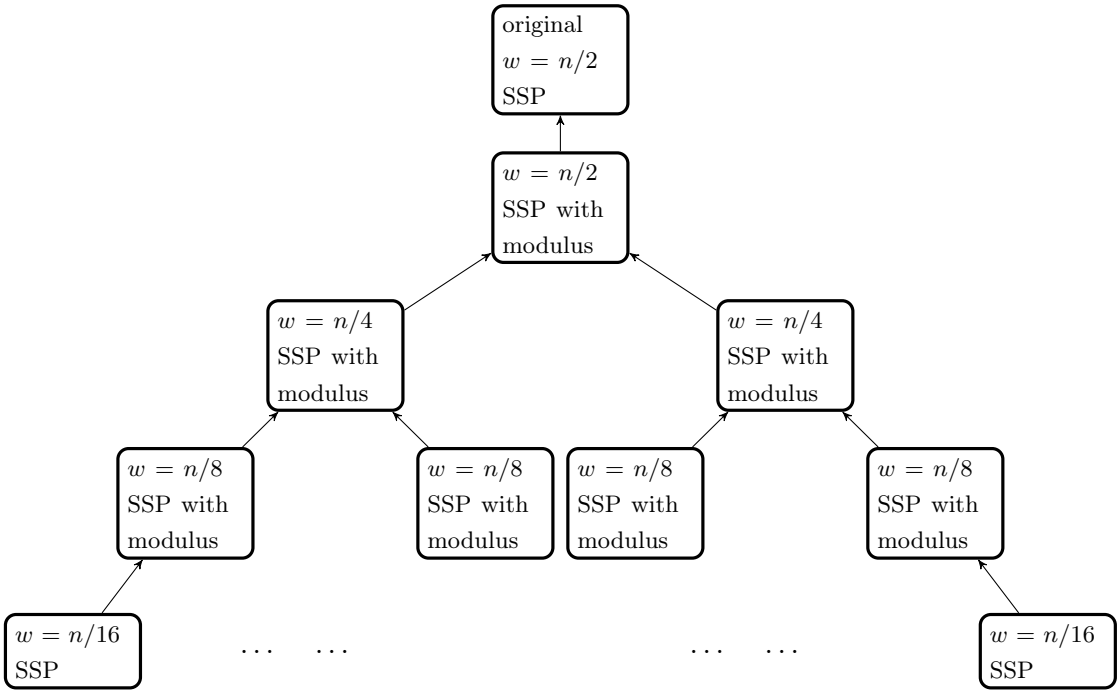


Figure 5.1: Tree structure of the Howgrave-Graham-Joux algorithm

In the first level of the Howgrave-Graham-Joux algorithm (see Figure 5.1) a modulus $M_1 \approx 2^{n/2}$ and a value $m_1 \in \{0, \ldots, M_1 - 1\}$ are picked, transforming the original

weight-$n/2$ SSP into one with a modulus. The goal is to compute the sets

$$L_1^{M_1} \coloneqq \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, \dots, n\},\ |I_1| = n/4,\ \Sigma(I_1) \equiv m_1 \mod M_1\}.$$

and

$$L_2^{M_1} \coloneqq \{(\Sigma(I_2), I_2) : I_2 \subseteq \{1, \dots, n\},\ |I_2| = n/4,\ \Sigma(I_2) \equiv t - m_1 \mod M_1\}.$$

Then we check for collisions $\Sigma(I_1) = t - \Sigma(I_2)$ that also satisfy $I_1 \cap I_2 = \emptyset$ since we do not perform the standard left-right split but rather use the representation technique. Since $a_1, \dots a_n$ are uniformly random and the number of possible decompositions of I in the sets $I_1, I_2 \subseteq \{1, \dots, n\}$ with $|I_1| = |I_2| = n/4$ is $\binom{n/2}{n/4} \approx 2^{n/2}$, we expect that for each choice of m_1 we find at least one element in $L_1^{M_1}$ such that I_1 is part of such a decomposition of I with high probability. For the approximation of the binomial coefficient we used the asymptotic approximation $\log \binom{n}{k} \approx nH(k/n)$, derived from Stirling's formula, where $H(\alpha) = -\alpha \log(\alpha) - (1-\alpha)\log(1-\alpha)$ is the binary entropy of $\alpha \in (0, 1)$. The remaining task is to compute the sets $L_1^{M_1}$ and $L_2^{M_1}$, which are weight-$n/4$ SSPs with moduli (second level of the tree). They are solved analogously by picking another modulus $M_2 \approx 2^{n/4}$, dividing M_1, producing four weight-$n/8$ SSPs with moduli. Each of these is finally solved by using the standard left-right split method resulting in eight weight-$n/16$ SSPs as the leaves of the tree. It can be shown that the success probability of one iteration of this algorithm is inverse polynomial in n. For this statement, a more detailed explanation and overview of the Howgrave-Graham-Joux algorithm and its organisation as a tree, we refer to the original paper [BJLM13, pp. 13-15].

The modified version: Bernstein et al. modify the algorithm above in the following way. They introduce a parameter $r \leq \binom{n/2}{n/16}$ that controls the number of randomly selected weight-$n/16$ subsets in the lowermost stage of the algorithm. In the extreme case $r = \binom{n/2}{n/16}$, this produces the original algorithm and for smaller values of r the success probability drops by a factor of $(r/\binom{n/2}{n/16})^8$, which is compensated by a quantum walk consisting of $\mathcal{O}(\sqrt{r}(\binom{n/2}{n/16}/r)^4)$ steps to assure that the algorithm has a constant success probability, see [Amb07]. In [BJLM13] it is stated that the optimal choice of the parameter r is on the scale of $\binom{n/2}{n/16}^{4/4.5}$ leading to a computational cost of $2^{(0.241\dots+o(1))n}$. Here, polynomial factors, such as the ones produced by binary search in previous steps, are suppressed. In the following we will always work with this running time, but have in mind that it actually is $\mathrm{poly}(n) \cdot 2^{(0.241\dots+o(1))n}$ for a rather small polynomial factor.

6 Solving LWE Instances via Reductions to SSP and DHSP

Now, we will use the quantum reductions given in Chapter 4 together with the algorithms presented in Chapter 5 in order to construct two approaches for solving the learning with errors problem (LWE). In the following, we will analyse the asymptotical running time, space requirements and success probabilities of the algorithms we obtain by combining the various quantum reductions we presented together with the algorithms for solving DHSP from Kuperberg and SSP from Bernstein et al. (BJLM). We begin with showing how to view an instance of LWE as an instance of the bounded distance decoding problem (BDD). Afterwards, we apply the reductions from Chapter 4 to transform the original LWE problem into an instance of the dihedral coset problem (DCP), or the subset sum problem (SSP), which we will finally solve with the algorithms from Chapter 5. This way, we will obtain two possible ways for solving LWE, which we can then compare to other quantum algorithms from the literature that solve other lattice problems that are assumed to be hard on quantum computers, e.g. the shortest vector problem (SVP). In the end we will also review some details in our chain of reductions that crucially impact the running time of the overall algorithms. By highlighting these, we will see where there is room for possible improvements in both approaches presented in this thesis.

6.1 Viewing LWE as BDD

It was already mentioned earlier, see Remark 2.54, that it is possible to think of LWE as an instance of BDD. We will now explain this idea in more detail.

Recall from Section 2.3.3 (Def. 2.50) that the output of the LWE distribution has the form $(a, b) = (a, \langle s, a\rangle + e \mod q)$ for some positive integer modulus q, a uniformly chosen $a \in \mathbb{Z}_q^n$, some secret $s \in \mathbb{Z}_q^n$, and some error term $e \in \mathbb{Z}_q$, chosen according to some error distribution χ on $\mathbb{Z}_q^n$, usually some discrete Gaussian. Assuming we are given m LWE-samples of the above form, where m is usually taken to be polynomial in n, we can group all of these samples together to form

$$\mathbf{b} = A^T s + \mathbf{e},$$

where $A \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{b}, \mathbf{e} \in \mathbb{Z}_q^m$ are the vectors formed by the b_i, e_i from the m LWE

samples, respectively. We define the following lattice

$$\mathcal{L} := \Lambda_q(A) := \{A^T s : s \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m \tag{6.1}$$

and remark that finding the closest vector of $\mathcal{L}$ to the target vector $\mathbf{b}$ yields the solution s to the LWE-search version. The lattice $\mathcal{L}$ is an example of a so-called 'q-ary lattice' by which we mean that $\mathcal{L}$ fulfils $q\mathbb{Z}^m \subseteq \mathcal{L}$, i.e. the membership of a vector x is determined by its value modulo q.

Definition 6.1. Let $n \leq m$, $A \in \mathbb{Z}^{n\times m}$ and $q \in \mathbb{N}$. We define

$$\Lambda_q(A) := \{y \in \mathbb{Z}^m \,|\exists s \in \mathbb{Z}^n : y \equiv A^T s \mod q\},$$
$$\Lambda_q^{\perp}(A) := \{y \in \mathbb{Z}^m \,|Ay \equiv 0 \mod q\}.$$

Later, we will be interested in the determinant of lattices of the above form. However, we cannot use A to compute the determinant because A is not quadratic. One way to solve this problem is given by the following algorithm (see [HW11, Prop. 1.12.]).

Algorithm 9 q-ary lattice base algorithm

INPUT: A matrix $A \in \mathbb{Z}_q^{n\times m}$ with $\operatorname{rank}(A) = n$ $(n \leq m)$.
OUTPUT: A lower-triangular, full-rank matrix $B \in \mathbb{Z}^{n\times n}$ that satisfies $B\cdot\mathbb{Z}^n \cong A\cdot\mathbb{Z}^m$.

1: **for** $j = 1$ to n **do**
2: Compute $d = \gcd(a_{jj}, \ldots, a_{jm})$.
3: Compute $d_j, \ldots, d_m \in \mathbb{Z}$ such that $\sum_{i=j}^m d_i a_{ji} = d$.
4: Set $\vec{a}_j = \sum_{i=j}^m d_i \vec{a}_i$ (here $\vec{a}_j$ denotes the j-th column of A)
5: **for** $k = j+1$ to m **do**
6: Set $\vec{a}_k = \vec{a}_k - \frac{a_{jk}}{d}\vec{a}_j$.
7: **end for**
8: **end for**
9: Return $B = (\vec{a}_1, \ldots, \vec{a}_n)$.

Lemma 6.2. *The lattices $\Lambda_q(A)$ and $\Lambda_q^{\perp}(A)$ have dimension m and satisfy $\Lambda_q^{\perp}(A) = q \cdot \Lambda_q(A)^\star$ as well as $\Lambda_q(A) = q \cdot \Lambda_q^{\perp}(A)^\star$.*

Proof. See [HW11, Prop. 1.13.] □

To determine the approximation factor when viewing LWE as an instance of BDD, we will need a reasonable approximation of the length of the shortest vector of the lattice defined in Equation (6.1). We will use a heuristic, known as the *Gaussian heuristic,* which approximates the number of lattice points in a set S by the quotient of the volume of the set and the determinant of the lattice, i.e.

$$|S \cap \mathcal{L}| \approx \frac{\operatorname{vol}(S)}{\det(\mathcal{L})}. \tag{6.2}$$

It is easy to use the Gaussian heuristic to estimate the length of the shortest vector. Let us take the set S to be an m-dimensional ball that contains exactly one lattice vector, and let r denote its radius. The volume of an m-dimensional ball of radius r is given as

$$\mathrm{vol}(S) = r^m \frac{\pi^{m/2}}{\Gamma(m/2+1)},$$

where Γ denotes the Gamma function. Using the Gaussian heuristic, we estimate the length of the shortest vector by

$$\lambda_1(\mathcal{L}) \approx \frac{\Gamma(m/2+1)^{1/m}}{\sqrt{\pi}} \det(\mathcal{L})^{1/m}.$$

The next step for us is to compute $\det(\Lambda_q(A))$.

Lemma 6.3. *If $A \in \mathbb{Z}_q^{n\times m}$ with $n \leq m$ and $\mathrm{rank}(A) = n$ modulo q, then*

$$\det(\Lambda_q^{\perp}(A)) = q^n \quad \textit{and} \quad \det(\Lambda_q(A)) = q^{m-n}.$$

Proof. See [HW11, Fact 2.3] □

Now, we can give the final estimate on the length of the shortest vector in the lattice given in Equation (6.1), using Stirling's approximation $\Gamma(z+1) \approx \sqrt{2\pi z}(\frac{z}{e})^z$, as

$$\lambda_1(\mathcal{L}) \approx \pi^{1/2m}\sqrt{\frac{m}{2\pi e}} q^{1-\frac{n}{m}} \approx \sqrt{\frac{m}{2\pi e}} q^{1-\frac{n}{m}}. \tag{6.3}$$

The final step to regard LWE as a BDD instance is to establish the approximation factor γ. Keep in mind that the 'LWE-lattice', defined in Equation (6.1), is an m-dimensional lattice according to Lemma 6.2. Recall from Section 2.3.3 that in an instance of BDD_γ we are asked to find the closest lattice point to a given target vector $\mathbf{t} \in \mathbb{R}^n$ assuming that $\mathrm{dist}(\mathbf{t}, \mathcal{L}) \leq \gamma(m)\lambda_1(\mathcal{L})$. Since the target vector is of the form $\mathbf{t} = A^T s + \mathbf{e}$, we have $\mathrm{dist}(\mathbf{t}, \mathcal{L}) = \|\mathbf{e}\|$, where $\mathbf{e}$ is drawn from the discrete Gaussian $D_{\mathbb{Z}^m,\beta}$ with mean 0 and standard deviation $\beta/\sqrt{2\pi}$, which can be realised by drawing the coordinates independently from $D_{\mathbb{Z},\beta}$ (Def. 2.39). Therefore we need an upper bound on the length of a vector, sampled according to $D_{\mathbb{Z}^m,\beta}$. In Definition 2.41 we introduced Micciancio's and Regev's smoothing parameter and remarked that if the parameter of a discrete Gaussian on a lattice is at least the smoothing parameter, it behaves like a continuous Gaussian. The following calculation may therefore serve as a motivation for an upper bound on the error. Recall that for the discrete Gaussian from Definition 2.39 we have $s = \sigma\sqrt{2\pi}$. If

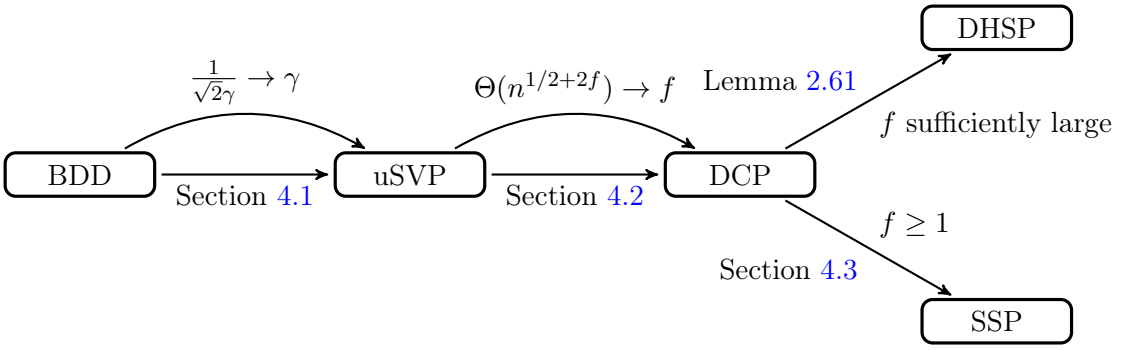


Figure 6.1: Path of reductions with parameters.

the error term is drawn from the normal distribution $\mathcal{N}(0, s/\sqrt{2\pi})$ we compute

$$\begin{aligned}
&\text{Prob}_{\mathbf{e}\leftarrow\mathcal{N}(0,s/\sqrt{2\pi})}\left[\|\mathbf{e}\| \geq s\sqrt{m}\right] = 2\int_{s\sqrt{m}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x \\
&\leq \frac{2}{\sqrt{2\pi}} \int_{s\sqrt{m}}^{\infty} \frac{x}{s\sqrt{m}\sigma} e^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x \\
&= -\frac{1}{\sqrt{m\pi}} e^{-\frac{x^2}{2\sigma^2}}\Big|_{s\sqrt{m}}^{\infty} = \frac{1}{\sqrt{m\pi}} e^{-\pi m}.
\end{aligned}$$

Thus, with probability exponentially close to 1, we have $\|\mathbf{e}\| \leq s\sqrt{m}$. For the discrete distribution, which is used to sample the error terms in the LWE instances, we have a similar result due to Banaszczyk.

Lemma 6.4 (Lemma 1.5(i) in [Ban93]). *Let $\mathcal{L}$ be a lattice, $s > 0$ and let $\mathcal{B}_d(0, \sqrt{d}s)$ denote the (closed) Euclidean ball of dimension d, centred at the origin with radius $\sqrt{d}s$. Let further $D_{\mathcal{L},s}$ be defined as in 2.39. Then*

$$D_{\mathcal{L},s}(\mathcal{L} \setminus \mathcal{B}_d(0, \sqrt{d}s)) \leq \frac{1}{2^d}.$$

In other words: The mass given to lattice points of norm greater than $\sqrt{d}s$ by the discrete Gaussian $D_{\mathcal{L},s}$ is exponentially small.

It was mentioned earlier that in most LWE applications we take the error distribution χ to be Ψ_β, as previously defined in Definition 2.50. Lemma 6.4 allows us to estimate $\text{dist}(\mathbf{t}, \mathcal{L}) \leq \beta\sqrt{m}$ with probability exponentially close to 1.

At this point we want to remind the reader of the path of reductions, together with the respective parameters of the problems, that were established in the previous sections. They are summarised in Figure 6.1.

Since we established a reduction from $\mathrm{BDD}_{1/\sqrt{2}\gamma}$ to uSVP_γ in Section 4.1, we want to bound $\mathrm{dist}(\mathbf{t}, \mathcal{L})$ by $\lambda_1(\mathcal{L})/(\sqrt{2}\gamma)$. This leads to

$$\beta\sqrt{m} \overset{!}{=} \frac{1}{\sqrt{2}\gamma}\lambda_1(\mathcal{L}) \overset{(6.3)}{\approx} \frac{1}{\sqrt{2}\gamma}\sqrt{\frac{m}{2\pi e}}q^{1-\frac{n}{m}}.$$

Therefore, we established a reduction from LWE to uSVP_γ with approximation factor

$$\gamma \approx \frac{q^{1-\frac{n}{m}}}{2\sqrt{\pi e}\beta}. \tag{6.4}$$

Remark 6.5. We keep in mind that there is still some randomness involved. For example, the matrix A, obtained from the LWE samples, is required to have full rank in $\mathbb{Z}_q$. However, since we have access to essentially arbitrary many LWE samples, this is a minor issue. Also, the upper bound on the length of the error vectors holds 'only' with probability exponentially close to 1. At last we mention that the approximation of $\lambda_1(\mathcal{L})$ is only a heuristic, but it seems to be a relatively good one in the case of random lattices ([MR09, Chapter 3]).

The reduction from $\mathrm{uSVP}_{\Theta(n^{1/2+2f})}$ to DCP_f together with the decomposition $\beta = \alpha q$ from the parameter of the LWE-error-distribution leads to

$$\alpha = C \cdot \frac{q^{-\frac{n}{m}}}{n^{1/2+2f}} \tag{6.5}$$

for some constant $C \in \mathbb{R}$.

Remark 6.6. Regev's hardness result from [Reg09] requires $\alpha q > 2\sqrt{n}$.

6.2 Combining the Reductions

To analyse the reductions from LWE to DHSP or SSP, we sum up the running time, space requirements and success probabilities of the various reductions discussed in the previous sections in Table 6.1 and 6.2. Note that we use n on the next page to refer to the dimension of the original BDD-lattice because we did so in the previous chapters, where the reductions were presented. Since the LWE-lattice defined in Equation (6.1) is actually a lattice of dimension $m \geq n$ (the number of LWE samples), according to Lemma 6.2, we will replace every n by an m when we analyse the actual reductions from LWE to SSP and DHSP later.

Table 6.1: Analysis of the individual reduction steps

Reduction	Time	Space	Success probability
DCP_f to DHSP [Reg04a]	no additional time	no additional space	$\left(\frac{(\log N)^f-1}{(\log N)^f}\right)^{\mathfrak{I}_{\mathrm{DHSP}}}$
DCP_1 to SSP [Reg04a]	$\mathcal{O}(\log^{4c_m+7} N)+\mathcal{O}(\log^{4c_m+5} N)\,\mathrm{oc}$	$\mathcal{O}(\log^{3c_m+6} N)$	exponentially close to 1
$\mathrm{2PP}_f$ to DCP_f [Reg04a]	$\mathcal{O}(1)+1\,\mathrm{oc}$	$\mathrm{poly}(n\log M)\cdot\mathcal{O}(n\log M)$	deterministic reduction
$\mathrm{uSVP}_{\Theta(n^{1/2+2f})}$ to $\mathrm{2PP}_f$ [Reg04a]	$LLL+(\mathcal{I}(n\log M)\cdot\mathcal{O}(1)+1\,\mathrm{oc})\cdot\mathcal{O}(n^{4+2f})$	$\mathcal{I}(n\log M)+\mathcal{O}(n^{4+2f})$	deterministic reduction
$\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ to uSVP_γ [SBW16]	$LLL+\mathcal{O}(n^4)\cdot(\widetilde{\mathcal{O}}(n^2)+\mathcal{O}(n^{2.373})+1\,\mathrm{oc})$	$\mathcal{O}(n^2\log(\gamma n^2))$	$\approx\frac{1}{8\gamma n^2}$

Table 6.2: Analysis of the utilised solver algorithms

Algorithm	Time	Space	Success probability	Required input registers
Kuperberg-DHSP2 [Kup11]	$2^{\mathcal{O}(\sqrt{\log N})}$	$2^{\mathcal{O}(\sqrt{\log N})}$ classical and $\mathcal{O}(\log\log N)$ quantum	$\mathcal{O}(1)$	$2^{\mathcal{O}(\sqrt{\log N\log\log N})}$
BJLM-SSP [BJLM13]	$\widetilde{\mathcal{O}}(2^{(0.241\ldots+o(1))r})$	$\widetilde{\mathcal{O}}(2^{(0.241\ldots+o(1))r})$	exponentially close to 1	$\mathcal{O}(\log N)$

The $\mathfrak{I}_{\text{DHSP}}$ given in Table 6.1 is the number of required 'good' input registers for the DHSP solver, M is the quantity introduced in Algorithm 4 ($M = 2^{4n}$), the abbreviation LLL stands for the running time of the LLL-algorithm (usually something along the lines of $\mathcal{O}(n^6 \log^3(\|B\|_\infty))$), $\mathcal{I}(n \log M)$ refers to the required number of input registers for the 2PP-solver, $N = (2M)^n$ (see Lemma 4.17) refers to the dihedral group D_N of order $2N$, f is the 2PP or DCP failure parameter, r is the number of elements used as input for SSP, c_m is the matching constant, required to fulfil $c_m > 3c_s$, and the abbreviation 'oc' stands for oracle call(s). Note that c_s is a constant used to describe what fraction of all legal inputs an SSP solver could handle. It can be taken to be zero if an SSP solver can find the solution for any legal SSP input. This is the case for the algorithm, considered in Section 5.2. We note that in the analysis of the two algorithms in Table 6.2 polynomial factors were suppressed since they are dominated by the exponential terms.

We can combine the reductions to produce a reduction from LWE, which we learned to think of as an BDD instance, to DHSP as well as to SSP. Since the chain of reductions is identical in both cases until the last reduction from DCP is made, we obtain for the two reductions the following results:

- LWE to SSP general time:

$$T_{\text{SSP,gen}} = 2LLL + (\widetilde{\mathcal{O}}(m^2) + \mathcal{O}(m^{2.373})) \cdot \mathcal{O}(m^4) + \mathcal{O}(m^{8+2}) \cdot \left[\mathcal{I}(m \log M) + \mathcal{O}(1) + \mathcal{O}(\log^{4c_m+7} N) + \mathcal{O}(\log^{4c_m+5} N) \cdot \text{solver time}\right]$$

- LWE to SSP time concrete (only dominant summands):

$$\begin{aligned} T_{\text{SSP}} &= \mathcal{O}(m^{24}) + \mathcal{O}(m^{20}) \cdot \widetilde{\mathcal{O}}(2^{(0.241\ldots+o(1))\cdot(4m^2+m)}) \\ &= \widetilde{\mathcal{O}}(2^{(0.241\ldots+o(1))\cdot(4m^2+m)}) \end{aligned}$$

- LWE to SSP space: $\widetilde{\mathcal{O}}(2^{(0.241\ldots+o(1))\cdot(4m^2+m)})$

- LWE to SSP success probability:

$$p_{\text{success,SSP}} \approx \frac{2\sqrt{\pi e}\beta}{8m^2 q^{-\frac{n}{m}}} = \frac{2\sqrt{\pi e}\alpha q^{\frac{n}{m}}}{8m^2}$$

- LWE to DHSP general time:

$$\begin{aligned} T_{\text{DHSP,gen}} = 2LLL &+ (\widetilde{\mathcal{O}}(m^2) + \mathcal{O}(m^{2.373})) \cdot \mathcal{O}(m^4) \\ &+ \mathcal{O}(m^{8+2f}) \cdot [\mathcal{I}(m \log M) + \mathcal{O}(1) + \text{solver time}] \end{aligned}$$

- LWE to DHSP time concrete (only dominant summands):

$$\begin{aligned} T_{\text{DHSP}} &= \mathcal{O}(m^{8+2f}) 2^{\mathcal{O}(m\sqrt{\log m})} + \mathcal{O}(m^{8+2f}) 2^{\mathcal{O}(m)} \\ &= \widetilde{\mathcal{O}}(2^{\mathcal{O}(m\sqrt{\log m})}) \end{aligned}$$

- LWE to DHSP space: $2^{\mathcal{O}(m)}$ classical and $\mathcal{O}(\log m)$ quantum.
- LWE to DHSP success probability:

$$p_{\text{success,DHSP}} \approx \frac{2\sqrt{\pi e}\alpha q^{\frac{n}{m}}}{8m^2}\left(\frac{(\log N)^f - 1}{(\log N)^f}\right)^{\mathfrak{I}_{\text{DHSP}}}$$

Here we took into account that the reductions considered earlier gave us $M = 2^{4m}$ and $N = (2M)^m$. Also, we choose $c_m = c_s = 0$ since the SSP solver is not restricted to only a fraction of the legal inputs.

At first, it seems odd that the time and space requirements in both cases worsen with a greater number of samples, since more LWE-samples, i.e more information about the secret, should not make it harder for us to solve LWE. However, we have to keep in mind that the LWE-lattice defined in Equation (6.1) has dimension m. Therefore, we have to figure out what an appropriate quantity for the number of LWE-samples is. We require the matrix A whose columns consist of the vectors $a_i \in \mathbb{Z}_q^n$, which are chosen uniformly at random, to have full rank with high probability, while keeping m as small as possible.

Lemma 6.7. *Let $n \in \mathbb{N}$, $q \in \mathbb{Z}$ a prime number and $A \in \mathbb{Z}_q^{n\times n}$ a matrix, where every entry is chosen uniformly at random over $\mathbb{Z}_q$. Then we have*

$$\text{Prob}[A \text{ has full rank}] \geq \left(1 - \frac{1}{q}\right)^n.$$

Proof. We prove the statement via induction over n. For $n = 1$ it is true, since for a prime number q the only element not invertible in $\mathbb{Z}_q$ is 0. Assume that the statement is true for $n - 1 \in \mathbb{N}$, $n \geq 2$. Let $(a_{i,j})_{1\leq i,j,\leq n} = A \in \mathbb{Z}_q^{n\times n}$ denote a matrix whose entries have been chosen uniformly over $\mathbb{Z}_q$, except for the entry $a_{n,n}$, which is still empty. Let $M_{i,j}$ denote the minor matrix of A, i.e. the submatrix obtained by deleting the i-th row and the j-th column. The Laplace expansion of $\det(A)$ is given as

$$\det(A) = \sum_{i=1}^{n}(-1)^{i+n}a_{i,n}\det(M_{i,n}).$$

Therefore, we have

$$\det(A) = 0 \iff a_{n,n}\det(M_{n,n}) = -\sum_{i=1}^{n-1}(-1)^{i+n}a_{i,n}\det(M_{i,n}).$$

This yields

$$\begin{aligned}\text{Prob}[\text{rank}(A) = n] &\geq \text{Prob}[\text{rank}(A) = n|\,\text{rank}(M_{n,n}) = n-1]\cdot\text{Prob}[\text{rank}(M_{n,n}) = n-1]\\ &= \left(1-\frac{1}{q}\right)\cdot\left(1-\frac{1}{q}\right)^{n-1}.\end{aligned}$$

□

Since the LWE modulus q is polynomial in n (in Regev's public-key cryptosystem based on LWE it is between n^2 and $2n^2$), Lemma 6.7 tells us that we can take the number of LWE-samples to be equal to n and still have a high probability that A has full rank if q is prime. Thus, we obtain, in the above summarisation of the presented reductions, that running time, required space and success probability all depend primarily on the main input parameter n of the LWE problem.

Another thing worth mentioning at this point is the failure parameter f of DCP. Although the size of f is not of great importance regarding the running time and space requirements since the exponential running time of the best currently known DHSP and SSP algorithms dominates the polynomial factors, depending on the failure parameter. Nevertheless, the failure parameter is important since it determines which uSVP instances we are able to solve, when we make use of the reductions to DHSP and SSP. Thus, f ultimately determines what LWE instances we are able to solve. In Section 6.1 we saw that the relation between the LWE parameters can be stated as

$$n = \left(C \cdot \frac{1}{q \cdot \alpha} \right)^{\frac{1}{1/2+2f}}, \tag{6.6}$$

when we take the number of LWE samples to be n, for some constant $C > 0$. Thus, knowing the failure parameter tells us what instances of LWE with main input parameter n we can hope to solve. The presented reduction from LWE to SSP requires $f \geq 1$. For the LWE to DHSP reduction the situation is more complicated. It is not clear whether – or to what extend – solvers for DHSP like the algorithms by Kuperberg and Regev are able to handle 'bad' DCP input registers. Therefore, we may require all DCP input registers to be 'good' in order to be able to solve DCP with a DHSP solver. Let us assume we want all DCP input registers to be 'good' with a probability of at least ε for some $0 < \varepsilon < 1$, i.e.

$$\begin{aligned} &\text{Prob[all DHSP input registers are 'good']} \geq \left(1 - \frac{1}{\log^f N}\right)^{\mathfrak{I}_{\text{DHSP}}} \overset{!}{\geq} \varepsilon \\ \iff\ & 1 - \varepsilon^{1/\mathfrak{I}_{\text{DHSP}}} \geq \frac{1}{\log^f N} \\ \iff\ & f \geq \frac{-\log(1 - \varepsilon^{1/\mathfrak{I}_{\text{DHSP}}})}{\log\log N} =: \Phi(N, \mathfrak{I}_{\text{DHSP}}, \varepsilon), \end{aligned} \tag{6.7}$$

where $\mathfrak{I}_{\text{DHSP}}$ denotes the number of required DHSP input registers.

Lemma 6.8. *Let $0 < \varepsilon < 1$. If the number of required* DHSP *input registers $\mathfrak{I}_{\text{DHSP}}$ is polynomial in $\log N$ with degree $k \geq 1$, then*

$$\lim_{N \to \infty} \Phi(N, \mathfrak{I}_{\text{DHSP}}, \varepsilon) = k.$$

Proof. For notational convenience we take $\mathfrak{I}_{\mathrm{DHSP}}(x) = x^k$ for some $k \in \mathbb{N}$. By using L'Hôpital's rule we obtain

$$\lim_{x\to\infty} -\frac{\log(1-\varepsilon^{1/x^k})}{\log(x)} = \lim_{x\to\infty} -\frac{\ln(\varepsilon)\cdot k\cdot \varepsilon^{1/x^k}}{x^k\cdot(1-\varepsilon^{1/x^k})}.$$

Since

$$\begin{aligned}\lim_{x\to\infty} x^k\cdot(1-\varepsilon^{1/x^k}) &= \lim_{x\to\infty} \frac{(1-\varepsilon^{1/x^k})}{\frac{1}{x^k}}\\ &= \lim_{x\to\infty} \frac{k\cdot x^{-k-1}\cdot\ln(\varepsilon)\cdot\varepsilon^{1/x^k}}{-k\cdot x^{-k-1}}\\ &= -\ln(\varepsilon)\end{aligned}$$

again by L'Hôpital's rule, we have that

$$\lim_{N\to\infty} \Phi(N,\mathfrak{I}_{\mathrm{DHSP}},\varepsilon) = k,$$

for $\mathfrak{I}_{\mathrm{DHSP}}(\log N) = \log^k N$ and for $\mathfrak{I}_{\mathrm{DHSP}}(\log N) = \mathcal{O}(\log^k N)$ the statement follows by an analogous calculation. □

Remark 6.9. Unfortunately, Kuperberg's algorithm requires $\mathfrak{I}_{\mathrm{DHSP}} = 2^{\mathcal{O}(\sqrt{\log N \log\log N})}$ input registers. In this case we have $\Phi(N,\mathfrak{I}_{\mathrm{DHSP}},\varepsilon) \xrightarrow{N\to\infty} \infty$ for every $0<\varepsilon<1$. For concrete applications the value of $\Phi(N,\mathfrak{I}_{\mathrm{DHSP}},\varepsilon)$, i.e. the lower bound for the failure parameter, has to be calculated for the given ε and N. Keep in mind that we have the relation $\log N = 4n^2 + n$, where n is the main security parameter of the LWE instance in question.

7 Room for Improvements and Conclusions

In this last chapter we will highlight and discuss some parts of the chains of reductions, where an improvement would significantly effect the required time at the end. We also briefly compare our results with algorithms that are used to solve other hard lattice problems.

7.1 Room for Improvements

As one can see from the previous section, it does not seem to be a very practical idea to solve LWE instances with reductions to SSP and DHSP, at least for now. The overall required time is exponential in the LWE-lattice-dimension m, which is not surprising. However, the actual size of the exponents is rather bad. The problems start to arise once we choose to reduce uSVP to DCP. The reason for the big expressions in the exponent is that the dihedral group underlying the instance of DCP, created in the reduction from 2PP, has a really high cardinality. In Lemma 4.17 we create an instance of DCP with dihedral group D_N such that $N = (2M)^n$, and in Algorithm 4 we take $M = 2^{4n}$, which leads to $\log N = 4n^2 + n = \mathcal{O}(n^2)$. The reason for the choice of M lies in Lemma 4.19 in the estimation given in Equation (4.2), which is needed to provide a bound for the probability that the created 2PP input registers are 'bad', see Inequality (4.4). The choice of M is, somewhat surprisingly, motivated by the estimation $|u_i| \leq 2^{2n}$ on the coordinates of the unqiue shortest vector in the LLL-reduced basis, which was provided by Lemma 2.34. Therefore, a tighter bound on the $|u_i|$ would result in a smaller dihedral group, thus improving the reduction given in [Reg04a]. It might be possible to achieve such a smaller bound by using other basis reduction algorithms than LLL, for example the BKZ (Blockwise Korkine-Zolotarev) basis reduction. However, in contrast to the LLL algorithm, there are no known algorithms that compute a BKZ-reduced basis in polynomial-time. Therefore, it would be necessary to make a trade-off between smaller bounds on the coefficients of the shortest vector on the one hand and the running time of a BKZ implementation on the other hand (there is a rather new implementation called BKZ 2.0 due to Chen and Nguyen, see [CN11]). If we assume that we could obtain a smaller bound on the $|u_i|$, say something like $|u_i| \leq 2^{\tau(n)}$ for some $\tau(n) = o(n)$, Lemma 4.19 would go through with $M = 2^{2\tau(n)}$ resulting in $\log N = \mathcal{O}(\tau(n) \cdot n)$.

Therefore, we would obtain an improved reduction from LWE to SSP, which requires $2^{\mathcal{O}(\tau(m)\cdot m)}$ time and space. Analogously, the LWE to DHSP reduction would be improved to require only $\approx 2^{\mathcal{O}(\sqrt{\tau(m)\cdot m})}$ time and classical space as well as $\approx \mathcal{O}(\log(\tau(m)\cdot m))$ quantum space.

However it is unlikely that we could hope for a bound $|u_i| \leq \text{poly}(n)$, since this would leave us with only $2n \cdot \text{poly}(n) - 1$ possible choices for the coefficients of u, and we could brute-force our way to a solution of uSVP.

Another alternative worth considering is to improve the reduction from 2PP to DCP itself to decrease the size of the dihedral group. Until today, there are no published reductions from 2PP to DCP that produce a dihedral group significantly smaller then $D_{2^{\mathcal{O}(n^2)}}$.

However, in an unpublished article ([LBF+13]) Li et al. claim to have established a reduction with dihedral group $D_{n^{13n\log n}}$, which would immediately improve the time and space requirements of the SSP reduction to $\approx 2^{\mathcal{O}(m\log m)}$. For the DHSP reduction we would obtain $\approx 2^{\mathcal{O}(\sqrt{m\log m})}$ for time and classical space as well as $\approx \mathcal{O}(\log(m\log m))$ quantum space this way.

7.2 Conclusions

In this thesis, we approached the learning with errors problem by reducing it to the dihedral hidden subgroup problem and the subset sum problem. This way we aimed to solve LWE-instances with the best currently known quantum algorithms for these problems. First, we presented several (quantum) polynomial-time reductions from the literature and then combined them to establish a reduction from LWE, which can be considered as an instance of the bounded distance decoding problem, to the dihedral coset problem. Then, we reduced DCP to the 'final' problems DHSP and SSP. We analysed the time and space requirements of solving LWE-instances with quantum algorithms that solve DHSP and SSP as well as their success probability. Unfortunately, for the SSP route, the results we obtained for the time and (classical) space requirements are superexponential in the number of LWE samples m which itself is at least as large as the dimension n of the LWE secret. For the DHSP route the space and time costs are exponential in m in addition to some polynomial factors in m. We compare these results to the running times for other lattice problems that are considered to be hard such as the shortest vector problem or its variant SVP_δ. For these two problems, it was shown in [LMVDP15] that there exist algorithms that make use of quantum search algorithms, similar to Grover's Algorithm ([Gro96]), in order to find a shortest vector in a lattice of dimension n, provably in time $2^{1.799n+o(n)}$ and heuristically in time $2^{0.286n+o(n)}$. Also, SVP_δ can be solved provably in time $2^{0.603n+o_\delta(n)}$ for any n-dimensional lattice. While against that background the running time of $2^{(0.241\ldots+o(1))n}$ of the SSP algorithm by Bernstein et al. ([BJLM13]) looked promising at first, the DCP to SSP reduction

produced an instance of SSP of input size $\mathcal{O}(\log N) = 4n^2 + n$ that turned out to be too large to be practical, again because of the huge N, i.e. the size of the dihedral group.

Also, we pointed out possible improvements in the chain of reductions presented in this thesis, for example the possibility of a better pre-processing of the involved uSVP-lattice basis to reduce the size of the coefficients of the unique shortest vector. This could lead to a reduced size of the dihedral group underlying the created DCP instance. Another approach would be an improved reduction from the two point problem to DCP, which reduces the cardinality of the involved dihedral group. We showed how these hypothetical improvements would affect the results of an approach similar to the one taken in this thesis.

In summary, tackling the learning with errors problem with a reduction to DHSP looks more promising than with a reduction to SSP. Attempting to solve LWE instances with an embedding approach, which would essentially be a reduction from LWE (as BDD) to uSVP without the last reduction to DCP (and from there to DHSP and SSP), as it was done and analysed in [AFG13], is likely to be a better approach until we can handle the size of the dihedral group in the DCP instance better.

Lastly, it may be possible that the additional structure inherent to the ring variant of LWE introduced by Regev et al. in ([LPR10]) may be exploited to establish a reduction from RLWE to DCP, similar to the one discussed in this thesis, with a smaller underlying dihedral group. This may be a reasonable objective for further research in this area.

It seems, however, unlikely that there will be an efficient quantum algorithm for the learning with errors problem, since lattice based cryptography seems to be somewhat resilient against the capabilities of quantum computers. Then again, the reduction of various lattice problems that are considered to be hard to the dihedral hidden subgroup problem together with the ability of quantum computers to solve the hidden subgroup problem efficiently in the abelian case, may suggest that there could be an efficient quantum algorithm one day.

Bibliography

[AFG13] Albrecht, Martin R. ; Fitzpatrick, Robert ; Göpfert, Florian: On the efficacy of solving LWE by reduction to unique-SVP. In: *International Conference on Information Security and Cryptology* Springer, 2013, 293–310

[Amb07] Ambainis, Andris: Quantum walk algorithm for element distinctness. In: *SIAM Journal on Computing* 37 (2007), Nr. 1, 210–239. http://epubs.siam.org/doi/pdf/10.1137/S0097539705447311

[AZQ10] Aigner, Martin ; Ziegler, Günter M ; Quarteroni, Alfio: *Proofs from the Book*. Bd. 274. Springer, 2010

[Bab86] Babai, László: On Lovász' lattice reduction and the nearest lattice point problem. In: *Combinatorica* 6 (1986), Nr. 1, 1–13. http://www.csie.nuk.edu.tw/~cychen/Lattices/On%20lovasz%20lattice%20reduction%20and%20the%20nearest%20lattice%20point%20problem.pdf

[Ban93] Banaszczyk, Wojciech: New bounds in some transference theorems in the geometry of numbers. In: *Mathematische Annalen* 296 (1993), Nr. 1, 625–635. https://pdfs.semanticscholar.org/d784/91c564d3be43072ee6a6f337ee2b8e2b427e.pdf

[BG14] Bai, Shi ; Galbraith, Steven D.: An improved compression technique for signatures based on learning with errors. In: *Cryptographers' Track at the RSA Conference* Springer, 2014, 28–47

[BJLM13] Bernstein, Daniel J. ; Jeffery, Stacey ; Lange, Tanja ; Meurer, Alexander: Quantum algorithms for the subset-sum problem. In: *International Workshop on Post-Quantum Cryptography* Springer, 2013, S. 16–33

[BKW03] Blum, Avrim ; Kalai, Adam ; Wasserman, Hal: Noise-tolerant learning, the parity problem, and the statistical query model. In: *Journal of the ACM (JACM)* 50 (2003), Nr. 4, 506–519. https://arxiv.org/pdf/cs/0010022.pdf

[CM01] Cheung, Kevin K. ; Mosca, Michele: Decomposing finite abelian

groups. In: *arXiv preprint cs/0101004* (2001). https://arxiv.org/pdf/cs/0101004.pdf

[CN11] Chen, Yuanmi ; Nguyen, Phong Q.: BKZ 2.0: Better lattice security estimates. In: *International Conference on the Theory and Application of Cryptology and Information Security* Springer, 2011

[Con09] Conrad, Keith: Dihedral Groups II. In: *Internet Online Book* (2009), 3–6. http://www.math.uconn.edu/~kconrad/blurbs/grouptheory/dihedral2.pdf

[DDLL13] Ducas, Léo ; Durmus, Alain ; Lepoint, Tancrède ; Lyubashevsky, Vadim: Lattice signatures and bimodal gaussians. Version: 2013. https://www.researchgate.net/profile/Leo_Ducas/publication/265809389_Lattice_Signatures_and_Bimodal_Gaussians/links/578cf91208ae59aa66815032.pdf. In: *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, 40–56

[DXL12] Ding, Jintai ; Xie, Xiang ; Lin, Xiaodong: A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. In: *IACR Cryptology EPrint Archive* 2012 (2012), 688. https://eprint.iacr.org/2012/688.pdf

[EH00] Ettinger, Mark ; Høyer, Peter: On quantum algorithms for noncommutative hidden subgroups. In: *Advances in Applied Mathematics* 25 (2000), Nr. 3, 239–251. https://arxiv.org/pdf/quant-ph/9807029.pdf

[EJ16] Espitau, Thomas ; Joux, Antoine: Adaptive precision LLL and Potential-LLL reductions with Interval arithmetic. In: *IACR Cryptology ePrint Archive* 2016 (2016), 528. https://eprint.iacr.org/2016/528.pdf

[FH13] Fulton, William ; Harris, Joe: *Representation theory: a first course*. Bd. 129. Springer Science & Business Media, 2013

[Gro96] Grover, Lov K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* ACM, 1996, 212–219

[HGJ10] Howgrave-Graham, Nick ; Joux, Antoine: New generic algorithms for hard knapsacks. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques* Springer, 2010, 235–256

[Hoe63] Hoeffding, Wassily: Probability inequalities for sums of bounded random variables. In: *Journal of the American statistical association* 58 (1963), Nr. 301, S. 13–30

[HW11] Hüttenhain, Jesko ; Wallenborn, Lars: Topics in Post-Quantum Cryptography. (2011). https://pdfs.semanticscholar.org/902b/a86fcd727d326f2fcdf60ddbf3d13163f7c8.pdf

[Kal83] Kaltofen, Erich: On the complexity of finding short vectors in integer lattices. In: *European Conference on Computer Algebra* Springer, 1983, 236–244

[Kle13] Klenke, Achim: *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013

[Kup05] Kuperberg, Greg: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: *SIAM Journal on Computing* 35 (2005), Nr. 1, 170–188. http://epubs.siam.org/doi/pdf/10.1137/S0097539703436345

[Kup11] Kuperberg, Greg: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: *arXiv preprint arXiv:1112.3333* (2011). https://arxiv.org/pdf/1112.3333v1.pdf

[LBF+13] Li, Fada ; Bao, Wansu ; Fu, Xiangqun ; Zhang, Yuchao ; Li, Tan: A reduction from LWE problem to dihedral coset problem. In: *arXiv preprint arXiv:1305.3769* (2013). https://arxiv.org/pdf/1305.3769.pdf

[LJP02] Lenstra Jr, Hendrik W. ; Pomerance, Carl: Primality testing with Gaussian periods. In: *FSTTCS*, 2002, 1

[LLL82] Lenstra, Arjen K. ; Lenstra, Hendrik W. ; Lovász, László: Factoring polynomials with rational coefficients. In: *Mathematische Annalen* 261 (1982), Nr. 4, 515–534. https://infoscience.epfl.ch/record/164484/files/nscan4.PDF

[LM09] Lyubashevsky, Vadim ; Micciancio, Daniele: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. Version: 2009. http://cseweb.ucsd.edu/~daniele/papers/uSVP-BDD.pdf. In: *Advances in Cryptology-CRYPTO 2009*. Springer, 2009, 577–594

[LMVDP15] Laarhoven, Thijs ; Mosca, Michele ; Van De Pol, Joop: Finding shortest lattice vectors faster using quantum search. In: *Designs, Codes and Cryptography* 77 (2015), Nr. 2-3, 375–400. https://eprint.iacr.org/2014/907.pdf

[Lom04] Lomont, Chris: The hidden subgroup problem-review and open problems. In: *arXiv preprint quant-ph/0411037* (2004). https://arxiv.org/pdf/quant-ph/0411037.pdf

[LPR10] Lyubashevsky, Vadim ; Peikert, Chris ; Regev, Oded: On ideal lattices and learning with errors over rings. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques* Springer, 2010, 1–23

[Mic11] Micciancio, Daniele: The geometry of lattice cryptography. Version: 2011. https://cseweb.ucsd.edu/~daniele/papers/FOSAD11.pdf. In: *Foundations of security analysis and design VI*. Springer, 2011, 185–210

[Mil08] Milne, James S.: *Algebraic number theory*. JS Milne, 2008 http://www.jmilne.org/math/CourseNotes/ANTc.pdf

[MR07] Micciancio, Daniele ; Regev, Oded: Worst-case to average-case reductions based on Gaussian measures. In: *SIAM Journal on Computing* 37 (2007), Nr. 1, 267–302. http://epubs.siam.org/doi/pdf/10.1137/S0097539705447360

[MR09] Micciancio, Daniele ; Regev, Oded: Lattice-based cryptography. Version: 2009. https://cseweb.ucsd.edu/~daniele/papers/PostQuantum.pdf. In: *Post-quantum cryptography*. Springer, 2009, 147–191

[NC00] Nielsen, Michael A. ; Chuang, Isaac L.: *Quantum computation and Quantum information*. Cambridge University Press India, 2000

[Neu92] Neukirch, Jürgen: *Algebraische Zahlentheorie*. Springer, 1992

[Neu13] Neukirch, Jürgen: *Algebraic number theory*. Bd. 322. Springer Science & Business Media, 2013

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

[Pei14] Peikert, Chris: Lattice cryptography for the internet. In: *International Workshop on Post-Quantum Cryptography* Springer, 2014, 197–219

[Pei16] Peikert, Chris: A Decade of Lattice Cryptography. (2016). https://eprint.iacr.org/2015/939.pdf

[Reg04a] Regev, Oded: Quantum computation and lattice problems. In: *SIAM Journal on Computing* 33 (2004), Nr. 3, 738–760. http://epubs.siam.org/doi/pdf/10.1137/S0097539703440678

[Reg04b] Regev, Oded: A subexponential time algorithm for the dihedral hid-

den subgroup problem with polynomial space. In: *arXiv preprint quant-ph/0406151* (2004). https://arxiv.org/pdf/quant-ph/0406151.pdf

[Reg05] Regev, Oded: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. (2005). https://users-cs.au.dk/~stm/local-cache/regev-on-lattices.pdf

[Reg09] Regev, Oded: On lattices, learning with errors, random linear codes, and cryptography. In: *Journal of the ACM (JACM)* 56 (2009), Nr. 6, 34. http://www.cims.nyu.edu/~regev/papers/qcrypto.pdf

[Reg10] Regev, Oded: The learning with errors problem. In: *Invited survey in CCC* (2010), 15. http://www.cims.nyu.edu/~regev/papers/lwesurvey.pdf

[SBW16] Shi Bai, Damien S. ; Wen, Weiqiang: Improved Reduction from the Bounded Distance Decoding Problem to the Unqiue Shortest Vector Problem in Lattices. (2016). http://perso.ens-lyon.fr/damien.stehle/downloads/bddusvp.pdf

[SD16] Stephens-Davidowitz, Noah: Discrete Gaussian sampling reduces to CVP and SVP. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* Society for Industrial and Applied Mathematics, 2016, 1748–1764

[Sho94] Shor, Peter W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* IEEE, 1994, 124–134

[Sho99] Shor, Peter W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In: *SIAM review* 41 (1999), Nr. 2, 303–332. https://arxiv.org/pdf/quant-ph/9508027.pdf