THE UNIVERSITY OF CALGARY

The Classical and Quantum Complexity of the Goldreich-Levin Problem with Applications to Bit Commitment

by

Mark R. A. Adcock

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF Master of Science

DEPARTMENT OF Computer Science

CALGARY, ALBERTA

January, 2004

UNIVERSITY OF
CALGARY

Abstract

The classical Goldreich-Levin (G-L) theorem is presented as a block-box query problem, referred to herein as the G-L problem. The query complexity of this problem is bounded in both classical and quantum settings. The well-known upper bound of the classical G-L problem is analyzed in a pedagogical manner. A proof of the lower bound of the classical G-L problem is given using the techniques of classical information theory. This classical analysis is then extended to the realm of quantum computing by noting the similarity of the noiseless G-L problem to the inner-product query problem solved by the quantum circuit defined by Bernstein and Vazirani (B-V). An upper bound of the query complexity of the quantum G-L problem is proven by extension of the B-V circuit to incorporate noisy inner-product queries. The lower bound of the query complexity of the quantum G-L problem is proven by adapting the proof of the optimality of the quantum search algorithm to include modified inner-product queries.

Both the classical and quantum versions of the Goldreich-Levin theorem have cryptographic applications in the area of bit commitment. A discussion of the impossibility of unconditional quantum bit commitment is followed by the presentation of both classical and quantum bit commitment protocols that are based on the assumption of the existence of classical and quantum one-way permutations. The relative security of the classical and quantum protocols are compared where it is shown that the quantum version is quantitatively more secure.

Acknowledgements

Firstly I would like to give special thanks to my supervisor, Dr. R.E. Cleve, for his advice and guidance during the preparation of this thesis. Secondly I would like to express my gratitude to my employer, General Dynamics Canada, for their support. Finally, I wish to thank my family Lynn, Katie, Keith and Stephen for their perseverance through the many hours I have spent in study.

Table of Contents

List of Tables

List of Figures

Introduction

The bulk of this thesis deals with various aspects of the Goldreich-Levin (G-L) Theorem and its application to cryptography. The G-L Theorem is concerned with the reduction from the computational problem of inverting a one-way function to the problem of predicting a single bit—a so-called hard predicate—associated with that function. The G-L Theorem is presented as a black-box query (or oracle) problem that we refer to as the G-L problem. In our study of this problem, we have divided the thesis into four chapters. The first chapter deals with preliminaries — that is the background required to understand the mathematical and physical concepts inherent in the proofs. Those readers with the appropriate background can skim or skip this section in its entirety. We now provide a brief overview of each of the chapters.

In the first chapter, we begin with an overview of classical information theory, specifically focusing on the tools used in the proofs of the query complexity of the classical G-L problem. This includes a discussion of entropy and related quantities as well as a discussion of basic asymptotic notation. In our study of the mathematical theory of information, we realize that information must exist in some physical sense — either as a symbol written on a piece of paper or the physical position of a switch. The language of physics is used to describe the physical states inherent in our information. Physics went through a revolution when the quantum theory was proposed and later demonstrated to accurately describe the behaviour of things at atomic scales. The mathematics of the quantum theory is much richer than its classical counterpart, and we will see that quantum information is also correspondingly richer than its classical counterpart. We follow our discussion of classical information

theory with a brief introduction to quantum information and computing that we will use in our study of the query complexity of the quantum G-L problem. We conclude Chapter 1 with a brief overview of cryptography with the main focus on the concept of bit commitment, which motivates the need for a hard predicate and introduces the G-L Theorem. Bit commitment is a useful application of the G-L Theorem whose reduction provides a means for us to compare the relative security of classical and quantum bit commitment schemes.

In Chapter 2, we focus on the classical G-L Theorem. We expand on Goldreich and Levin's original proof and provide a detailed analysis of the upper bound of the query complexity of this problem. The query complexity of the G-L problem is dependent on the *advantage* of the oracle. We speak of the *noiseless* G-L problem when the advantage is such that the oracle always returns the correct answer. Our study of the upper bound is followed by an analysis employing the techniques of classical information theory to provide a lower bound.

In Chapter 3, we introduce the quantum circuit used by Bernstein and Vazirani (B-V) to solve the inner product (IP) query problem. We show how the solution given by this circuit solves a problem that is equivalent to the noiseless G-L problem. We then capitalize on this recognition to extend the B-V circuit's applicability to noisy IP queries whose probabilistic response relates to the classical theorem in a natural way. We use this circuit to derive an upper bound of the query complexity of the quantum G-L problem. We then adapt the proof of the optimality of the quantum search algorithm to include modified inner-product queries in order to lower bound the quantum G-L problem.

The final chapter focuses on applications of the classical and quantum G-L The-

orems in the area of bit commitment. The concept of a quantum one way permutation is introduced. This is in turn used to develop protocols for both bit and qubit-commitment schemes. The relative security of these quantum protocols are compared to the equivalent classical protocols.

Chapter 1 Preliminaries

1.0 Introduction

The material presented in later chapters of this thesis requires some understanding of classical information theory, complexity theory, quantum information theory and cryptography. The purpose of this chapter is to present an overview of these large bodies of knowledge with a focus on the key concepts employed in the proofs presented herein. The chapter begins with an introduction into the mathematical foundation of information theory, where we focus on various aspects of Shannon entropy. This is followed by a brief introduction into the key concepts of quantum computing where we introduce the definition of the qubit and the quantum circuit representation of quantum algorithms. This framework is then expanded upon to include some examples of quantum algorithms that find application later in the thesis. The chapter concludes with a brief overview of the science of cryptography in which we pay particular attention to the concept of bit commitment. The conflicting needs of effective bit commitment provides motivation for the G-L Theorem.

1.1 Classical Information Theory

Information theory, as a distinct branch of mathematics, began in 1948 when Claude Shannon published his landmark paper "A Mathematical Theory of Communication" [30]. In this section, we discuss the major concepts of what we will refer to as classical information theory as distinguished from quantum information theory.

1.1.1 Entropy

We begin by introducing the concept of *Shannon entropy*. We follow this with definitions of conditional entropy and mutual information. We give examples of specific calculations for clarity and, in some cases, perform some of the calculations that are applied later on in the thesis.

The Shannon entropy of a random variable, X, is a measure of the average uncertainty of the random variable. Alternatively it can be viewed as the amount of information gained, on average, when we learn the value of X. The Shannon entropy is written as a function of the probability distribution of X, $p_1, \ldots, p_n$, as

$$H(X) \equiv H(p_1, \ldots, p_n) \equiv -\sum_x p_x \log p_x. \tag{1.1}$$

Note that in this definition the logarithm indicated by 'log' is taken to the base 2. We will use this convention throughout this thesis, while 'ln' indicates a natural logarithm. There are other definitions of entropy such as *minimum* entropy and *maximum* entropy defined in the literature, but we drop all adjectives and refer to Shannon entropy simply as entropy for the remainder of this thesis.

It is instructive to specifically look at the entropy of a two-outcome random variable

$$H(p) \equiv -p \log p - (1-p) \log(1-p), \tag{1.2}$$

where p and $1-p$ are the probabilities of the two possible outcomes. This is sometimes referred to as the *binary entropy*. The graph of the function $H(p)$ is shown in Figure 1.1. The figure illustrates some of the basic properties of entropy — it equals 0 when $p = 0$ or $p = 1$ and it takes on its maximum value of 1 when $p = \frac{1}{2}$. Intuitively, this makes sense since when $p = 0$ or $p = 1$ there is no uncertainty, and the uncertainty

is maximum when $p = \frac{1}{2}$. For purposes later on in the thesis, we now derive an expression for $H(p)$ when p is just slightly less than $\frac{1}{2}$. Thus we set $p = \frac{1}{2} - \varepsilon$, and the binary entropy is expressed as

$$\begin{aligned} H(p) &= -(\tfrac{1}{2} - \varepsilon)\log(\tfrac{1}{2} - \varepsilon) - (\tfrac{1}{2} + \varepsilon)\log(\tfrac{1}{2} + \varepsilon) \\ &= \frac{1}{\ln 2}\left[\left(\tfrac{1}{2} - \varepsilon\right)\ln\left(\frac{2}{1-2\varepsilon}\right) + \left(\tfrac{1}{2} + \varepsilon\right)\ln\left(\frac{2}{1+2\varepsilon}\right)\right] \\ &= \frac{1}{\ln 2}\left[\left(\tfrac{1}{2} - \varepsilon\right)(\ln 2 - \ln(1-2\varepsilon)) + \left(\tfrac{1}{2} + \varepsilon\right)(\ln 2 - \ln(1+2\varepsilon))\right] \\ &\geq \frac{1}{\ln 2}\left[\left(\tfrac{1}{2} - \varepsilon\right)(\ln 2 + 2\varepsilon) + \left(\tfrac{1}{2} + \varepsilon\right)(\ln 2 - 2\varepsilon)\right] \\ &= 1 - \frac{4}{\ln 2}\varepsilon^2. \end{aligned} \tag{1.3}$$

Here we have used the relationship $\pm x \geq \ln(1 \pm x)$ to create a quadratic bound in ε for this entropy. For comparison with $H(p)$, this bound is also plotted in Figure 1.1. Before moving on to the entropy of more than just a single random variable, it is interesting to write down the entropy of a random variable that has a uniform distribution over 2^n outcomes. The entropy of such a random variable is

$$H(X) = -\sum_{i=1}^{2^n} p_i \log p_i = -\sum_{i=1}^{2^n} \frac{1}{2^n}\log\frac{1}{2^n} = \log 2^n = n. \tag{1.4}$$

It is stated, but not proven, that the number of bits used to describe the random variable is the maximum value that $H(X)$ can take. It is left to the reader to prove that this always occurs when the random variable is uniformly distributed.

So far we have defined the entropy of a single random variable. The *joint entropy* is a measure of the uncertainty of a bivariate distribution. It is defined as

$$H(X,Y) = -\sum_{y}\sum_{x} p(x,y)\log p(x,y). \tag{1.5}$$

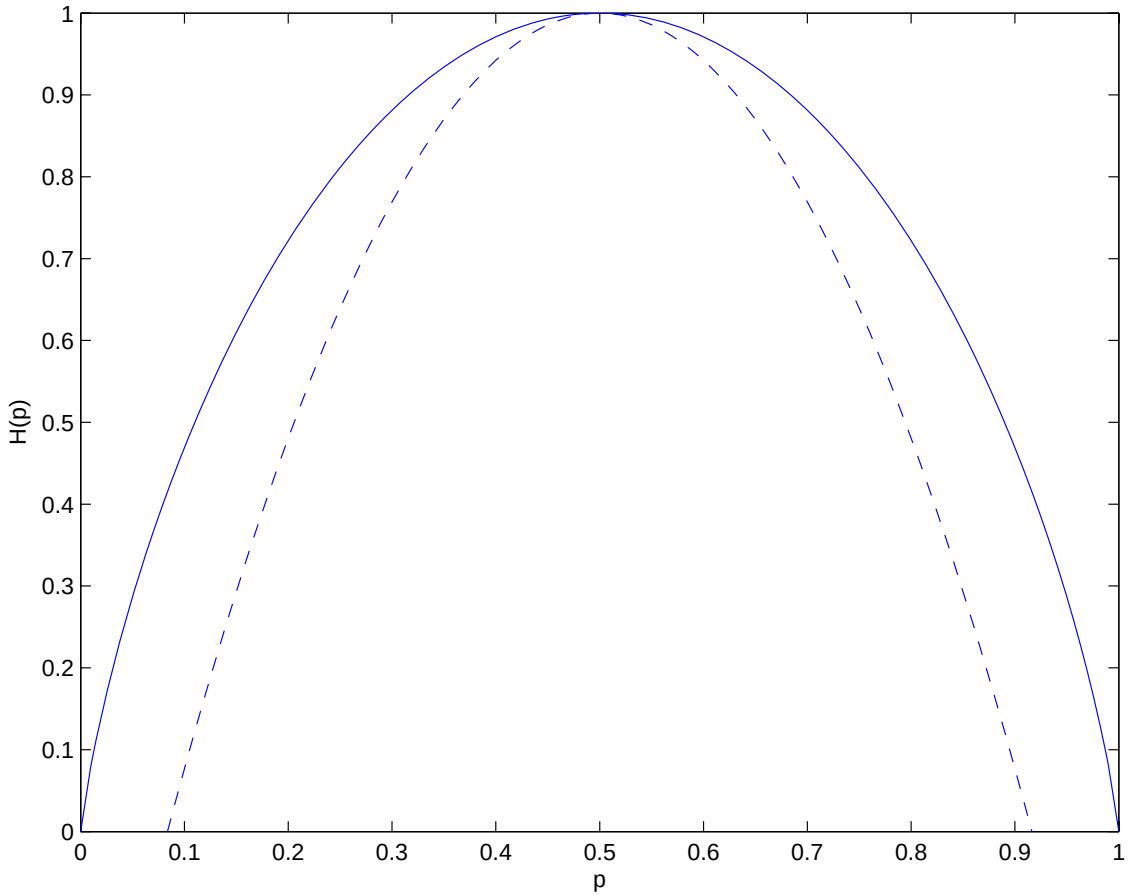

Figure 1.1: Plot of the binary entropy (solid line) and a quadratic approximation given in Equation 1.3 (dashed line).

The conditional entropy of a random variable given another random variable is the expected value of the entropies of the conditional probability distributions averaged over the conditioning random variable. It is defined as follows

$$H(Y|X) = \sum_x p(x) H\left(Y|X = x\right). \tag{1.6}$$

A very useful quantity is *the mutual information content of* X *and* Y, which is a measure of how much information X and Y have in common. If we add the information content of X, which is $H(X)$, to the information content of Y, information that

is common to both will have been counted twice in the sum. Subtracting the joint information of (X, Y), which is $H(X, Y)$, we obtain an expression for the *mutual information* of X and Y as

$$I(X : Y) = H(X) + H(Y) - H(X, Y). \tag{1.7}$$

Introducing the definition of conditional entropy given in Equation 1.6, Equation 1.7 is commonly rewritten as

$$I(X : Y) = H(X) - H(X|Y). \tag{1.8}$$

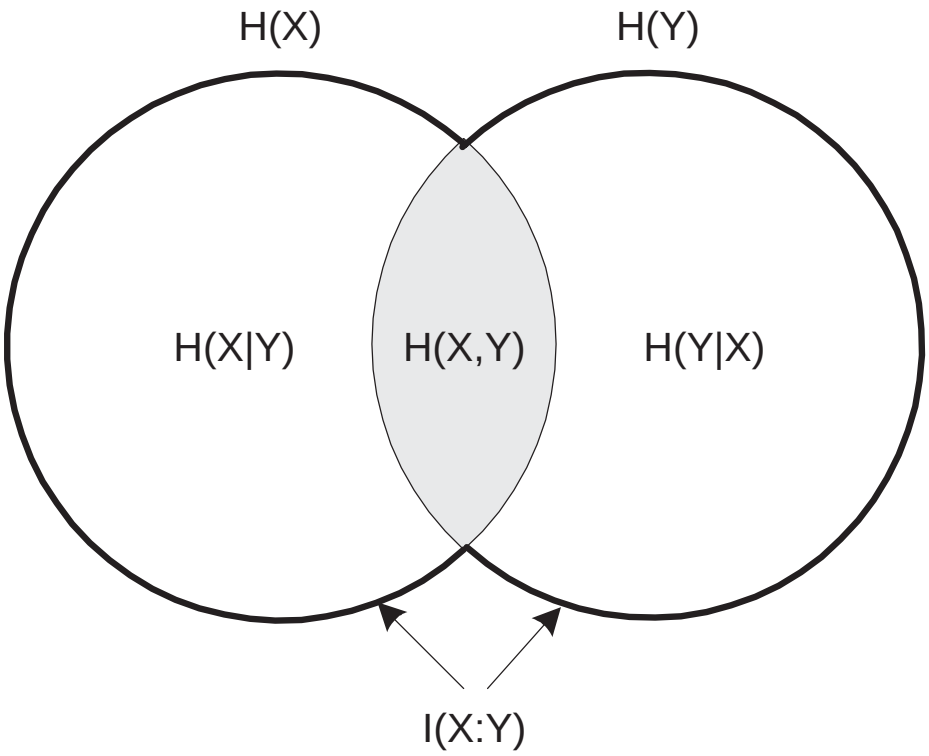

Figure 1.2: Entropy Venn diagram showing relationship between joint entropy $H(X, Y)$, conditional entropy $H(X|Y)$ and mutual information $I(X : Y)$.

These various entropy definitions are most easily remembered when they are expressed in the Venn diagram presented in Figure 1.2. We note the symmetry of the mutual information in its arguments and use this to write down

$$H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{1.9}$$

This property is useful in defining a chain rule for entropy. We write down the definition without proof (see [13, page 22] for a clear one):

$$H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1). \tag{1.10}$$

In Chapter 2 in our proof of the classical lower bound of the G-L Theorem, we need to develop an expression for how information about an unknown random variable is revealed by a succession of random variables that are the result of "noisy" responses to oracle queries. Equation 1.10 forms the basis for the desired expression. We first explore the simple case where $n = 3$, and in keeping with notation we use in Chapter 2, we relabel the random variables used in Equation 1.10 and write

$$\begin{aligned} H(A, Y_1, Y_2) &= H(A) + H(Y_1|A) + H(Y_2|Y_1, A) \\ &= H(Y_1) + H(Y_2|Y_1) + H(A|Y_1, Y_2). \end{aligned}$$

Solving this expression for $H(A|Y_1, Y_2)$, we have

$$\begin{aligned} H(A|Y_1, Y_2) &= H(A) + H(Y_1|A) + H(Y_2|Y_1, A) - H(Y_1) - H(Y_2|Y_1) \\ &= H(A) + \sum_{i=1}^{2} \left(H(Y_i|A, Y_{i-1}) - H(Y_i|Y_{i-1}) \right). \end{aligned} \tag{1.11}$$

In the last equality in Equation 1.11, we assume that $H(Y_1|Y_0) = H(Y_1)$ and $H(Y_1|A, Y_0) = H(Y_1|A)$. The quantity $H(A|Y_1, Y_2)$ is interpreted as the amount

of information we have about the random variable A given the information about the random variables Y_1 and Y_2, which are the result of making $m = 2$ queries to our "noisy" oracle. We readily extend Equation 1.11 to the general case where we make m queries

$$(A|Y_1, Y_2, \ldots, Y_m) = H(A) + \sum_{i=1}^{m} \left(H(Y_i|A, Y_1, \ldots, Y_{i-1}) - H(Y_i|Y_1, \ldots, Y_{i-1})\right). \tag{1.12}$$

We will make use of Equation1.12 in Chapter 2 where we wish to bound the number of queries we have to make to the "noisy" oracle to determine the unknown random variable A with reasonable probability.

We conclude this discussion of some of the properties of entropy by considering the entropy of a bi-level probability distribution where $2^{n/2}$ elements of $\{0,1\}^n$ each have probability $\delta/2^{n/2}$ and $2^n - 2^{n/2}$ elements each have probability $(1-\delta)/(2^n - 2^{n/2})$. The entropy of this distribution is

$$\begin{aligned}
H_{\text{bi-level}} &= H\ \underbrace{\tfrac{\delta}{2^{n/2}}, \ldots, \tfrac{\delta}{2^{n/2}}}_{2^{n/2}}, \underbrace{\tfrac{1-\delta}{2^n - 2^{n/2}}, \ldots, \tfrac{1-\delta}{2^n - 2^{n/2}}}_{2^n - 2^{n/2}} \\
&= -\sum_{i=1}^{2^{n/2}} \frac{\delta}{2^{n/2}} \log \frac{\delta}{2^{n/2}} - \sum_{i=2^n - 2^{n/2}}^{2^n} \frac{1-\delta}{2^n - 2^{n/2}} \log \frac{1-\delta}{2^n - 2^{n/2}} \\
&= -\delta \log \frac{\delta}{2^{n/2}} - (1-\delta) \log \frac{1-\delta}{2^n - 2^{n/2}} \\
&= -\delta \log \delta + \delta \log(2^{n/2}) - (1-\delta)\log(1-\delta) + (1-\delta)\log(2^n - 2^{n/2}) \\
&= H(\delta) + \delta \log(2^{n/2}) + (1-\delta)\log(2^n - 2^{n/2}).
\end{aligned} \tag{1.13}$$

The entropy of an arbitrary distribution having $2^{n/2}$ elements with probability p_i

and $2^n - 2^{n/2}$ elements with probability q_i is written

$$H\left(\underbrace{p_1, \ldots, p_{2^{n/2}}}_{2^{n/2}}, \underbrace{q_1, \ldots, q_{2^n - 2^{n/2}}}_{2^n - 2^{n/2}}\right) = -\sum_{i=1}^{2^{n/2}} p_i \log p_i - \sum_{i=1}^{2^n - 2^{n/2}} q_i \log q_i. \tag{1.14}$$

We have already mentioned that in Chapter 2 we bound the number of queries we need to make to a "noisy" oracle in order to determine the value of an unknown random variable. To assist in establishing this bound, we show that the entropy of the bi-level distribution given in Equation 1.13 upper bounds the entropy of any distribution with $\sum_i p_i = \delta$. It is of interest to note that this generalizes the spirit of Equation 1.4, where we have $H(X) \leq H(uniform) = n$. We thus make the claim that the entropy of the arbitrary distribution given by Equation 1.14 is at most equal to the entropy of the bi-level distribution. We prove the claim as follows:

$$\begin{aligned}
&H\left(\underbrace{\tfrac{\delta}{2^{n/2}}, \ldots, \tfrac{\delta}{2^{n/2}}}_{2^{n/2}}, \underbrace{\tfrac{1-\delta}{2^n - 2^{n/2}}, \ldots, \tfrac{1-\delta}{2^n - 2^{n/2}}}_{2^n - 2^{n/2}}\right) - H\left(\underbrace{p_1, \ldots, p_{2^{n/2}}}_{2^{n/2}}, \underbrace{q_1, \ldots, q_{2^n - 2^{n/2}}}_{2^n - 2^{n/2}}\right) \\
&= -\sum_{i=1}^{2^{n/2}} \frac{\delta}{2^{n/2}} \log \frac{\delta}{2^{n/2}} - \sum_{i=2^n - 2^{n/2}}^{2^n} \frac{1-\delta}{2^n - 2^{n/2}} \log \frac{1-\delta}{2^n - 2^{n/2}} \\
&\quad + \sum_{i=1}^{2^{n/2}} p_i \log p_i + \sum_{i=1}^{2^n - 2^{n/2}} q_i \log q_i \\
&= -\sum_{i=1}^{2^{n/2}} p_i \log \frac{\delta}{2^{n/2} p_i} - \sum_{i=1}^{2^n - 2^{n/2}} q_i \log \frac{1-\delta}{(2^n - 2^{n/2}) q_i} \\
&\geq -\sum_{i=1}^{2^{n/2}} p_i \left(1 - \frac{\delta}{2^{n/2} p_i}\right) - \sum_{i=1}^{2^n - 2^{n/2}} q_i \left(1 - \frac{1-\delta}{(2^n - 2^{n/2}) q_i}\right) \\
&= \sum_{i=1}^{2^{n/2}} p_i - \sum_{i=1}^{2^{n/2}} \frac{\delta}{2^{n/2}} + \sum_{i=1}^{2^n - 2^{n/2}} q_i - \sum_{i=1}^{2^n - 2^{n/2}} \frac{1-\delta}{2^n - 2^{n/2}} \\
&= 0.
\end{aligned} \tag{1.15}$$

We will now discuss some of the terminology of complexity theory that will be used

later in the thesis.

1.1.2 Complexity Theory

Complexity theory provides a methodology for analyzing the *computational complexity* of different computational problems. A computational problem's complexity is determined by the computational power needed to solve it. Computational complexity is often measured by two variables: T (for time complexity) and S (for space complexity, or memory requirements). Both T and S are commonly parameterized as functions of n, where n is the size of the input. In this thesis, we wish to bound the number of *oracle queries* required to solve the G-L problem. We are particularly interested in determining upper and lower bounds on the G-L query problem that are *asymptotically tight*. The bounds will "cap" the number of oracle queries we need to make and will be parameterized as a function of n, the number of bits in an unknown string. We will use the O ('big O') notation to set asymptotically tight *upper* bounds on the number of queries that a *particular* algorithm must make to determine the value of the string. We then use the Ω ('big Ω') notation to set asymptotically tight *lower* bounds on the number of queries that *any* algorithm must make to determine the value of the string. Also, in several definitions used later in the thesis, we make use of bounds that are not asymptotically tight. For these, we use the o ('little o') notation for non-asymptotically tight *upper* bounds and the ω ('little ω') notation for non-asymptotically tight *lower* bounds. We now provide definitions of each of these bounds along the lines given in [10] and [32], and for the purpose of cross reference, provide some examples.

The big O notation is particularly useful for studying the worst-case behaviour of

a specific algorithm. In this notation we only care about the term of the complexity function that grows the fastest. For example, if the time complexity of a given algorithm is $40n^4 + 30n^2 + 20\log^2 n + 10$, then the computational complexity is on the order of n^4 which is expressed $O(n^4)$. We formalize this notion in the following definition.

Definition 1 An *asymptotic upper bound* of a function $f(n)$ is $g(n)$ and is written $f(n) = O(g(n))$ provided the following conditions are met:

1. f and g are two functions $f, g : \mathbb{N} \to \mathbb{R}$.
2. positive integers c and n_0 exist so that for every integer $n \geq n_0$, $f(n) \leq cg(n)$.

We say that $g(n)$ is an *asymptotic upper bound* for $f(n)$, to emphasize that we are suppressing constant factors. Intuitively, $f(n) = O(g(n))$ means that f is less than or equal to g if we disregard differences up to constant factor.

When studying the behaviour of a class of algorithms, we wish to set *lower bounds* on the resources required by *any* member of that class. We use the Ω notation for this purpose. A formal definition of Ω notation is written along the lines of Definition 1.

Definition 2 An *asymptotic lower bound* of a function $f(n)$ is $g(n)$ and is written $f(n) = \Omega(g(n))$ provided the following conditions are met:

1. f and g are two functions $f, g : \mathbb{N} \to \mathbb{R}$.
2. positive integers c and n_0 exist so that for every integer $n \geq n_0$, $f(n) \geq cg(n)$.

In Chapters 2 and 3 of this thesis we will be comparing both upper and lower bounds of classical and quantum algorithms designed to to solve the G-L query

problem. It is interesting to note that while determining an upper bound requires just a specific algorithm be created and analyzed, determining a lower bound is usually more involved. This difficulty arises because the lower bound is often the result of some inherent constraint on the problem at hand that must be studied outside of the context of any particular algorithm. Once a lower bound has been proven for a given class of algorithms, no algorithm of that class can do better. Of course, an algorithm of a different class can do better as we will show when we provide lower bounds to both the classical and quantum G-L problem. Tools used to determine lower bounds include those of information theory. We now wish to introduce the concept of a *tight* bound. The bound on some problem A is said to be tight if A is both $O(f(n))$ and $\Omega(f(n))$. In this case we say that $A = \Theta(f(n))$. We now give definitions on non-asymptotic upper and lower bounds.

Another type of upper bound is one that is not asymptotically tight for which we use the o-notation.

Definition 3 A *non-asymptotic upper bound* of a function $f(n)$ is $g(n)$ and is written $f(n) = o(g(n))$ provided the following conditions are met:

1. f and g are two functions $f, g : \mathbb{N} \to \mathbb{R}$.
2. for *any* $c > 0$, there exists some $n_0 > 0$ such that for all $n \geq n_0$, $f(n) < cg(n)$.

The statement $f(n) = o(g(n))$ means $f(n)$ is "less than" any constant multiple of $g(n)$ for "big" values of n. We can thus think of $f(n)$ as "eventually" being trapped below all constant multiples of $g(n)$. The definitions of O-notation and o-notation are similar, but in o-notation the $f(n) < cg(n)$ holds for *all* constants $c > 0$. Perhaps the difference between the notations is most strikingly captured in the following. If

$f(n) = O(g(n))$, then assuming such a limit exists, $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c$; whereas if $f(n) = o(g(n))$, then assuming such a limit exists, $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$. In our final definition of this section, we define the non-asymptotically tight lower bound.

Definition 4 A *non-asymptotic lower bound* of a function $f(n)$ is $g(n)$ and is written $f(n) = \omega(g(n))$ provided the following conditions are met:

1. f and g are two functions $f, g : \mathbb{N} \to \mathbb{R}$.
2. for *all* $c > 0$, there exists some $n_0 > 0$ such that for all $n \geq n_0, f(n) > cg(n)$.

The statement $f(n) = \omega(g(n))$ means $f(n)$ is "greater than" any constant multiple of $g(n)$ when we ignore "small" values of n. We can thus think of $f(n)$ as "eventually" being trapped above any constant multiple of $g(n)$. Again, the definitions of Ω-notation and ω-notation are similar, and we capture the difference between the notations most strikingly in the following. If $f(n) = \Omega(g(n))$, then assuming such a limit exists, $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c$; whereas if $f(n) = \omega(g(n))$, then assuming such a limit exists, $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \infty$.

$f(n)$	Notation
$3n^3$	$= O(n^3)$
$3n^3$	$\neq o(n^3)$
n^{100}	$= n^{O(1)}$
$n^{\log\log n}$	$= n^{\omega(1)}$
n^n	$= n^{\omega(n)}$

Table 1.1: Some functions and their bounds

We shall use these bounds frequently throughout the thesis. In Table 1.1, we give some examples of functions and their membership or not in these sets. We now give in introduction to the aspects of quantum information theory that we will be making use of later in the thesis.

1.2 Quantum Information Theory

In order to study information in a quantum setting we need to expand several classical concepts into the quantum domain. In this section, we firstly expand our concept of a classical bit and introduce the concept of a quantum bit, which is termed a *qubit*. We introduce the idea of a quantum circuit as an analogy to a classical computer algorithm. Finally as preparation for proofs that will appear in Chapter 3, we introduce a simple quantum circuit in the study of Deutsch's algorithm and also study the slightly more involved Grover's algorithm.

1.2.1 Quantum bits

What is a qubit? Just as a classical bit has a *state*, which can be either 0 or 1, a qubit has two possible *basis* states $|0\rangle$ or $|1\rangle$. Here we are using the *Dirac* notation to describe a quantum mechanical state where the symbol '$|\ \rangle$' is termed a *ket* - the right syllable of bracket. The fundamental difference between bits and qubits is that qubits can be in a state *other* than $|0\rangle$ or $|1\rangle$. It is possible to form *superpositions* of states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{1.16}$$

The numbers α and β are complex numbers subject to the constraint that $|\alpha|^2+|\beta|^2 = 1$. Thus we can think of a qubit as a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as the *computational basis states*, and form an orthonormal basis for this vector space. When it comes to measuring a qubit, we get the result 0 with probability $|\alpha|^2$ and the result 1 with probability $|\beta|^2$. It is remarkable that we can not examine a qubit to determine its quantum state,

but nonetheless we can achieve powerful results with qubits.

Before going into how we can manipulate qubits, we wish to describe them in the langauge of linear algebra. The basis kets are written as the following two column vectors

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1.17}$$

Like a two-bit system, a two-qubit system has four possible basis states, which are written as follows

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}; |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{1.18}$$

These basis vectors are often expressed using the outer or *Kronecker* product notation

$$\begin{aligned} |00\rangle &= |0\rangle \otimes |0\rangle = |0\rangle^{\otimes 2} \\ &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \times 1 \\ 1 \times 0 \\ 0 \times 1 \\ 0 \times 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{1.19}$$

This notation is extended to define an n-qubit state, e.g.

$$|00\ldots0\rangle = |0\rangle^{\otimes n} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{1.20}$$

We now move on to define the quantum gates, which are used to manipulate qubits in order to compute with them.

1.2.2 Quantum gates

A classical computer is built around *logic gates*; in an analogous manner, a quantum computer is built from a *quantum circuit*, which contains elementary *quantum gates*. In this section, we introduce single and multiple qubit quantum gates. We state that a universal set of quantum gates can be constructed from some basic single and double qubit gates.

In a classical computer the non-trivial single bit logic gate is the NOT gate, whose operation is defined by its *truth table*. The 0 and 1 states are interchanged by this operation, in which $0 \mapsto 1$ and $1 \mapsto 0$. The analogous quantum NOT gate for a qubit acts linearly on a single qubit state

$$\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|1\rangle + \beta|0\rangle. \tag{1.21}$$

It is convenient to represent this quantum NOT gate, conventionally written X, as the two-by-two matrix

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.22}$$

As it turns out, all quantum gates acting on a single qubit can be described by *unitary* two-by-two matrices. Recall that the matrix U is unitary if $U^\dagger U = I$, where $U^\dagger$ is the complex conjugate transpose of U. There are many non-trivial single qubit

gates. One of the most important in quantum computation is the *Hadamard* gate,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.23}$$

This gate takes either of the computational basis states into an equal superposition of these states. The effect of the two-qubit gates, X and H is illustrated in Figure 1.3. Before moving on to discuss two qubit gates, we develop an expression for the

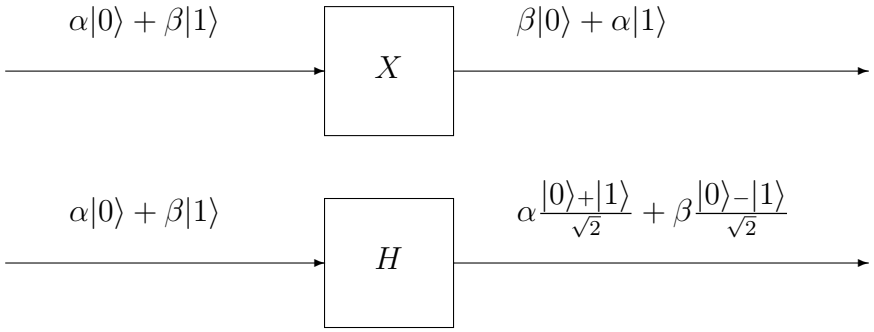

Figure 1.3: Single qubit logic gates

effect of n Hadamard gates operating on an n qubit state. Recall that we used the Kronecker product to express the two qubit basis states. If we wished to operate on each of the two component qubits, we would analogously express the four-by-four Hadamard matrix as

$$H^{\otimes 2} = H \otimes H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{1.24}$$

Of course we could extend this notation to n qubit states, but the effect of this Hadamard on a given input state can be much more succinctly expressed. To see how to proceed, we firstly explicitly write out the result for the $n = 2$ case:

$$\begin{aligned} H^{\otimes 2}|00\rangle &= \tfrac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ H^{\otimes 2}|01\rangle &= \tfrac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ H^{\otimes 2}|10\rangle &= \tfrac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) \\ H^{\otimes 2}|11\rangle &= \tfrac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle). \end{aligned} \tag{1.25}$$

There is an interesting pattern in how the plus and minus symbols are distributed in these four equations. We can capture this relationship by looking at the *inner product* of the input and output states. Defining $z \in \{0,1\}^2$ as the input states and $x \in \{0,1\}^2$ as the output states, we express the inner product between the two states as

$$z \cdot x = z^{(1)}x^{(1)} \oplus z^{(2)}x^{(2)}. \tag{1.26}$$

Here $z^{(i)}$ is the *ith* bit of the state $|z\rangle$ and the resulting inner product is a single bit. With this definition of the inner product we can rewrite the four equations 1.25 in the following nicely compact form

$$H^{\otimes 2}|z\rangle = \frac{1}{2} \sum_{x \in \{0,1\}^2} (-1)^{z \cdot x}|x\rangle. \tag{1.27}$$

The reader can readily verify the correctness of this expression. It is particularly satisfying how readily this representation extends to the n-qubit case:

$$H^{\otimes n}|z\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{z \cdot x}|x\rangle. \tag{1.28}$$

In designing quantum circuits to solve particular problems, it is often the case that the qubit registers are prepared in the n-qubit state $|00\ldots0\rangle$, and this state is then operated on by n Hadamard gates to produce an equal superposition of the 2^n basis states. Equation 1.28 with $|z\rangle = |00\ldots0\rangle$ thus appears often in the analysis of quantum circuits. The final component required to make a useful quantum circuit are the multiple qubit gates.

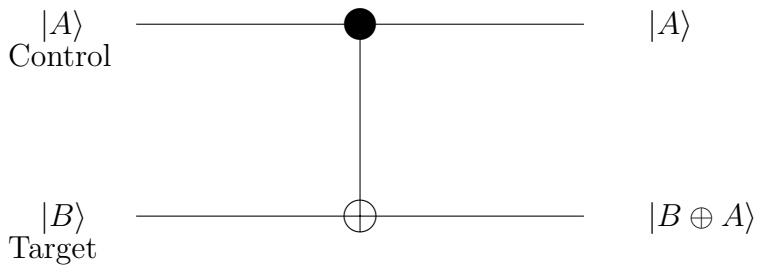

Figure 1.4: The CNOT Gate: an example two-qubit logic gate

The prototypical multiple qubit gate is the two qubit gate termed the controlled-NOT (CNOT) gate. Figure 1.4 is the standard quantum circuit representation of this gate. The upper qubit is referred to as the *control* qubit while the lower qubit is referred to as the *target* qubit. The effect of this gate is to flip the target qubit when the control qubit is 1. This effect is expressed in the following equations.

$$|00\rangle \to |00\rangle; \quad |01\rangle \to |01\rangle; \quad |10\rangle \to |11\rangle; \quad |11\rangle \to |10\rangle.$$

The CNOT has the following matrix representation

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.29}$$

Before moving on to discuss the use of these elementary quantum gates in quantum circuits, we introduce the semantics of quantum measurement.

1.2.3 Quantum measurement

It is postulated that closed quantum systems evolve according to unitary evolution. However, when a measurement is made this simple picture breaks down. To explain what happens, we give a very brief description of the effects of measurement on a quantum system. See [27, Pages 84-90] for a more complete discussion.

We begin by defining the measurement operator M_m, where the the index $m \in \{0,1\}^n$ refers to the measurement outcomes that may occur in an experiment involving an n-qubit system. The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I. \tag{1.30}$$

If the state of a quantum system is $|\psi\rangle$ before a measurement is made, then the probability that the result m occurs is

$$\Pr[m] = \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{1.31}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{1.32}$$

These two equations are understood by an example. Consider the measurement of a qubit in the computational basis. This is a measurement on a single qubit with two outcomes, which are defined by the two measurement operators

$$M_0 = |0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

and

$$M_1 = |1\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that $M_0 M_0^\dagger + M_1 M_1^\dagger = I$, which means that the probabilities sum to one as we would expect is required to completely describe the system. Continuing with our example suppose the state being measured is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, then the probability of measuring outcome 0 is

$$\Pr[m=0] = \langle\psi|M_0|\psi\rangle = |\alpha|^2. \tag{1.33}$$

Similarly $\Pr[m=1] = |\beta|^2$. The state after measurement in the two cases is thus

$$\begin{aligned} \frac{M_0|\psi\rangle}{|\alpha|} &= \frac{\alpha}{|\alpha|}|0\rangle = |0\rangle \\ \frac{M_1|\psi\rangle}{|\beta|} &= \frac{\beta}{|\beta|}|1\rangle = |1\rangle, \end{aligned} \tag{1.34}$$

where we have ignored the global phase factors $\alpha/|\alpha|$ and $\beta/|\beta|$. We see from this simple example that the measurement operator as formulated in Equation 1.31 and Equation 1.32 provides a consistent means for quantum measurement. The subject

of measurement of quantum systems is much richer, and the interested reader is directed to the referenced literature.

1.2.4 Quantum computation

An excellent example of a simple quantum circuit that shows the power of the quantum computer over the classical computer is known as Deutsch's Algorithm [16]. The Bernstein-Vazarani Algorithm, which appears later on in the thesis, is similar to this algorithm. Deutsch's Algorithm determines a certain *global* property of an unknown single-bit function $f : \{0, 1\} \rightarrow \{0, 1\}$. The property of interest is whether or not f is *balanced* or *constant*. On a single bit input, which can be either 0 or 1, there are four possible outputs that any function can have. We can characterize the four possible function categories as follows:

$$f_{00} : \begin{cases} f(0) = 0 \\ f(1) = 0 \end{cases} ; f_{01} : \begin{cases} f(0) = 0 \\ f(1) = 1 \end{cases} ; f_{10} : \begin{cases} f(0) = 1 \\ f(1) = 0 \end{cases} ; f_{11} : \begin{cases} f(0) = 1 \\ f(1) = 1 \end{cases} .$$

Note that f_{00} and f_{11} are said to be *constant* functions while f_{01} and f_{10} are said to be *balanced* functions for obvious reasons. The problem of determining whether f is balanced or constant is termed a *black-box* or *oracle* problem. In these types of problems, we are presented with a black box that computes f for us without revealing f. If we had only classical information, two queries to the oracle are required to determine if f is balanced or not. The remarkable feat that Deutsch's Algorithm accomplishes is that it can determine if the function is balanced or constant in just one application of the black box. We present the circuit in Figure 1.5 where we have marked the four states of interest. The effect of the box marked f in Figure 1.5 is to modulo-two sum f(upper qubit) onto the lower qubit. This action can also be

represented by the operator $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. Since $f : \{0,1\} \to \{0,1\}$, the matrix representation of U_f is a *permutation matrix*, which is always unitary. We now proceed with a step-by-step analysis of these states.

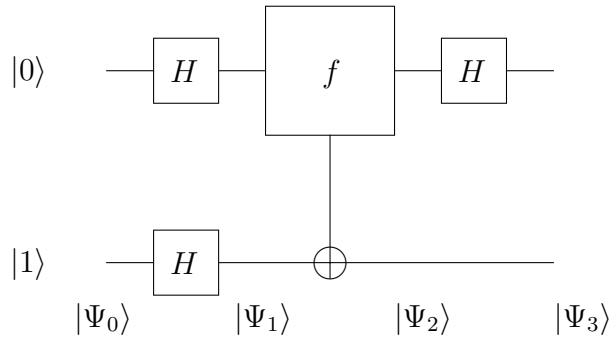

Figure 1.5: Quantum circuit implementing Deutsch's Algorithm

We begin by noting the input state is,

$$|\Psi_0\rangle = |0\rangle \otimes |1\rangle = |01\rangle.$$

Note that we will suppress the Kronecker operator at times for convenience. Application of the Hadamard gates leaves us with the superposition

$$\begin{aligned} |\Psi_1\rangle &= \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \\ &= \tfrac{1}{2}\left[|0\rangle(|0\rangle - |1\rangle) + |1\rangle(|0\rangle - |1\rangle)\right]. \end{aligned}$$

The next step is the core of the algorithm — the application of the black box U_f. This unitary operator acts in a similar manner to that depicted in Figure 1.4. That

is the value of f is modulo-two summed with the bottom qubit as follows

$$\begin{aligned} |\Psi_2\rangle &= \tfrac{1}{2}\left[|0\rangle(|0\oplus f(0)\rangle - |1\oplus f(0)\rangle) + |1\rangle(|0\oplus f(1)\rangle - |1\oplus f(1)\rangle)\right] \\ &= \tfrac{1}{2}\ (-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle) \\ &= \frac{1}{\sqrt{2}}\ (-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\ \ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{1.35}$$

Before applying the final Hadamard transformation, we note that Equation 1.35 has two distinct outcomes corresponding to the *constant* case, where $f(0) = f(1)$, and the *balanced* case, where $f(0) = f(1)$. We thus rewrite Equation 1.35

$$|\Psi_2\rangle = \begin{cases} \pm\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ if } f \text{ is constant} \\ \pm\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ if } f \text{ is balanced.} \end{cases}$$

The final Hadamard application reveals that the balanced and constant cases result in different states of the upper qubit since

$$|\Psi_3\rangle = \begin{cases} \pm|0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ if } f \text{ is constant} \\ \pm|1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \text{ if } f \text{ is balanced.} \end{cases}$$

We see that a single application of the circuit and a measurement of the upper qubit reveals whether f is constant or balanced. This is quite remarkable when compared to the classical case where two queries are required. Deutsch's problem is generalized to the n-bit case in the what is known as the Deutsch-Jozsa [17] problem. In the n-bit classical case, we need to make $2^{n-1} + 1$ queries (worst-case) whereas only one query suffices for the quantum case using the Deutsch-Jozsa algorithm. The Deutsch, Deutsch-Jozsa, Bernstein-Vazarani algorithms, along with Shor's famous factoring algorithm [31] are all applications of the quantum Fourier transform [27, page 37]. Another interesting type of quantum algorithm is known as Grover's search algorithm.

We will use Grover's algorithm [22], also called the quantum search algorithm, in our proof of the quantum version of the Goldreich-Levin theorem in Chapter 3. We give an introduction to the algorithm here. If there are $N = 2^n$ different elements in a list and we are required to search the list for a particular element, on a classical computer $\Omega(N)$ operations are required to find the element. Grover's algorithm provides significant speed up requiring only $O(\sqrt{N})$ operations to find the element. In Chapter 3 we will show that this bound is in fact optimal. We now describe Grover's algorithm and provide an outline of a proof of an upper bound of the algorithm.

In Grover's algorithm, we are given an oracle Γ with $f : \{1, 2, \ldots, N\} \to \{0, 1\}$ defined as

$$|x\rangle|q\rangle \xrightarrow{\Gamma} |x\rangle|q \oplus f(x)\rangle, \tag{1.36}$$

where x is the index list register and the oracle qubit $|q\rangle$ is flipped if $f(x) = 1$. By definition $f(x) = 1$ if x is a solution to the search problem. Just as we did in Deutsch's algorithm if we set the oracle qubit initially in the state $(|0\rangle - |1\rangle)/\sqrt{2}$, we can express the oracle more succinctly as

$$\begin{aligned} |x\rangle \quad \frac{|0\rangle - |1\rangle}{\sqrt{2}} \quad &\xrightarrow{\Gamma} \quad (-1)^{f(x)}|x\rangle \quad \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\ |x\rangle \quad &\xrightarrow{\Gamma} \quad (-1)^{f(x)}|x\rangle. \end{aligned} \tag{1.37}$$

In order to simplify the description, in the last step of the preceding we have adopted the convention of omitting the state of the oracle qubit because it remains unchanged. The oracle is said to *mark* the solutions to the search problem by shifting the phase of the solution. Grover's search algorithm involves the repeated application of the operator Γ and a second operator that we will now define.

The second operator involves the application of a conditional phase shift operator with every computational basis state except $|0\rangle$ receiving a phase shift of -1. This operator may be written $C_{Phase} = 2|0\rangle\langle 0| - I$. For clarity, we derive the matrix representation of this operator for the $n = 2$ case as

$$\begin{aligned} C_{Phase} &= 2|00\rangle\langle 00| - I \\ &= 2\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \end{aligned}$$

If we apply Hadamard operators to both the left and right sides of C_{Phase}, we can define

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I, \tag{1.38}$$

where the state $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$ is the equal superposition of all the basis states as defined in Equation 1.28 with $|z\rangle = |00\ldots0\rangle$. The action of this operator on an arbitrary state is to reflect it through the state $|\psi\rangle$. This can be readily seen by noting that

$$(2|\psi\rangle\langle\psi| - I)|a\rangle = 2p|\psi\rangle - |a\rangle, \tag{1.39}$$

where $p = \langle\psi|a\rangle$ is the *projection* of $|a\rangle$ onto $|\psi\rangle$. Thus we can view the action of the operator presented in Equation 1.39 as inverting the sign of the amount of $|a\rangle$ perpendicular to $|\psi\rangle$. This is a reflection of $|a\rangle$ through $|\psi\rangle$. This operator is combined with the oracle operator Γ, in the definition of the Grover iteration operator G, which is written

$$G = (2|\psi\rangle\langle\psi| - I)\Gamma. \tag{1.40}$$

Grover's algorithm is simply the repeated application of the operator G until the output state is nearly the solution state. We now describe how a single iteration of G gets us closer to the solution state. Consider a two dimensional vector space with

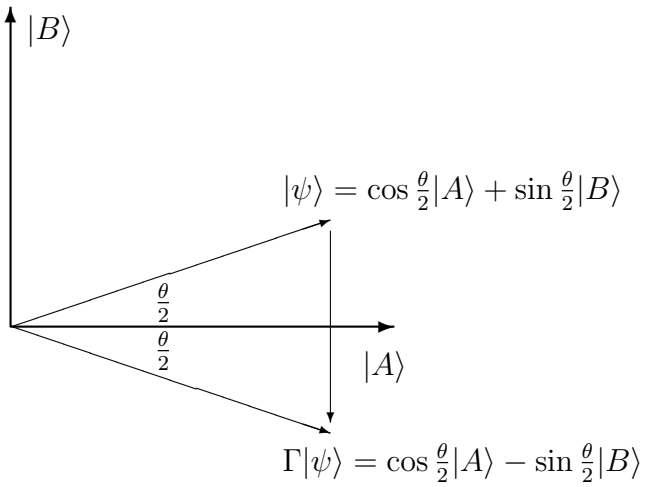

Figure 1.6: The action of the operator Γ is to reflect the state $|\psi\rangle$ about the state $|A\rangle$

two basis vectors, $|A\rangle$ and $|B\rangle$, which are non-solutions (bad) and solutions (good) respectively to the search problem. These states are defined as

$$\begin{aligned} |A\rangle &= \frac{1}{\sqrt{N-M}} \sum_{x \in bad} |x\rangle \\ |B\rangle &= \frac{1}{\sqrt{M}} \sum_{x \in good} |x\rangle. \end{aligned} \tag{1.41}$$

Here we are assuming a search space of N elements of which there are M solutions. It is apparent that $|A\rangle$ and $|B\rangle$ form an orthonormal basis. We can thus express the equal superposition state as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|A\rangle + \sqrt{\frac{M}{N}}|B\rangle \tag{1.42}$$

in this basis. We start Grover iteration on this state. We will now describe some pictures to get a feel for how the iteration works.

In order to understand the action of Grover iteration, we present two figures. In

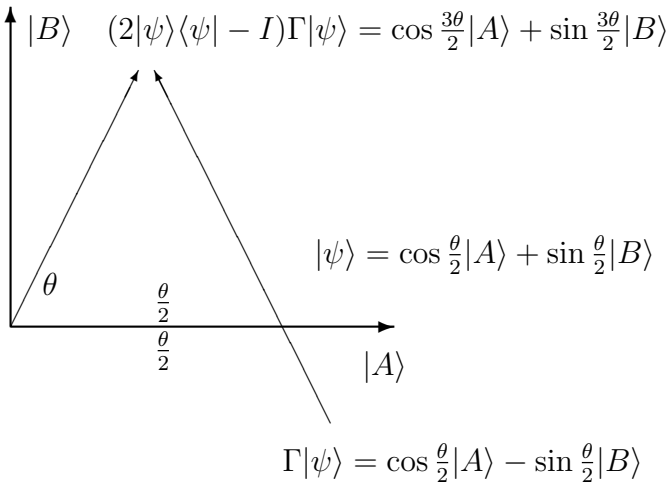

Figure 1.7: The action of the operator $(2|\psi\rangle\langle\psi| - I)$ is to reflect the state $\Gamma|\psi\rangle$ about the state $|\psi\rangle$

Figure 1.6, we show the action of the oracle Γ, which is a reflection about the $|A\rangle$ axis (Remember that this oracle gives a minus sign to solution vectors). In Figure 1.7, we show the action of the operator $2|\psi\rangle\langle\psi| - I$, which is also a reflection but this time about the starting state $|\psi\rangle$. Note that the net result of applying the operator G to the starting state $|\psi\rangle$ has been to move it closer to being parallel to the solution state $|B\rangle$. The state $G|\psi\rangle = \cos\frac{3\theta}{2}|A\rangle + \sin\frac{3\theta}{2}|B\rangle$ is θ radians closer to the state $|B\rangle$. Furthermore if we start with the state $G|\psi\rangle$, then the state $G^2|\psi\rangle$ will be 2θ radians closer to $|B\rangle$ than the starting state $|\psi\rangle$. (Note that $2\theta = \frac{\theta}{2} + \frac{3\theta}{2}$ as the operator in Equation 1.39 always reflects through the equal superposition state, $|\psi\rangle$). Thus k applications of the the operator G has the effect of rotating the state $|\psi\rangle$ $k\theta$ radians

towards $|B\rangle$. This continued application of G may be expressed as

$$G^k|\psi\rangle = \cos \quad \frac{2k+1}{2}\theta \quad |A\rangle + \sin \quad \frac{2k+1}{2}\theta \quad |B\rangle. \tag{1.43}$$

We will now upper bound the number of times we have to apply the operator until with high enough probability we have the solution.

Starting with the initial state of the system $|\psi\rangle = \quad \overline{\frac{N-M}{N}}|A\rangle + \quad \overline{\frac{M}{N}}|B\rangle$, we note that we need to rotate through arccos $\quad \overline{\frac{M}{N}}$ radians in order to move the system to $|B\rangle$. We thus define the number of Grover iterations

$$n_{GI} = CI\left(\frac{\arccos\sqrt{M/N}}{\theta}\right), \tag{1.44}$$

where $CI(x)$ is the *closest integer* to x. The arccos function provides us with a convenient way to upper bound the number of iterations. Noting that $n_{GI} \leq \left\lceil \frac{\pi}{2\theta} \right\rceil$ and $\frac{\theta}{2} \geq \sin\frac{\theta}{2} = \quad \overline{\frac{M}{N}}$, we have

$$n_{GI} \leq \left\lceil \frac{\pi}{4} \quad \overline{\frac{N}{M}} \right\rceil. \tag{1.45}$$

Thus we see that $n_{GI} = O \quad \overline{\frac{N}{M}}$. Note that n_{GI} depends on the *number* of solutions M, but not on the *identity* of these solutions. Provided we know M, the search algorithm is applicable as described. The interested reader is directed to [27, Section 6.3] for an explanation on how to remove the need for a knowledge of M. Thus for the case where there is a single solution only, we have

$$n_{GI} = O \quad \sqrt{N} \quad , \tag{1.46}$$

as we stated earlier. We will have need to call on Grover's search algorithm in Chapter 3 of this thesis and will make use of the result given in Equation 1.46. We now move on to discuss some of the concepts within cryptography that are pertinent to proofs in later chapters.

1.3 Overview of Cryptography

Cryptography is usually associated the study of how effectively information may be concealed from prying eyes during its transmission. Cryptographic protocols have also been developed for other purposes such as two distant and un-trusting parties wishing to share the result of a fair coin toss using digital communications only. *Bit commitment* is related to such a protocol. All cryptographic protocols may be analyzed using the techniques of information theory, complexity theory and probability theory. In this section, we briefly discuss the concept of absolute security and present some of the trade-offs between different techniques of more traditional cryptography in light of quantum information. We then turn to the concept of bit commitment and focus both on the quantification of its desirable characteristics and on actual implementations.

1.3.1 Cryptographic Algorithms and Protocols

In this section we first discuss, in very general terms, cryptographic algorithms, which are sometimes referred to as *ciphers*. We divide these algorithms into two categories *symmetric* and *asymmetric* algorithms. We give some examples and discuss the relative pros and cons of the two approaches. We then go onto to describe the motivation for the more esoteric cryptographic protocol of bit commitment. We provide some examples identifying deficiencies, which we will use to motivate the need for the G-L Theorem.

A cryptographic algorithm is the mathematical function used for encryption and decryption. The security of ciphertext transmission is based on a number of factors,

but for well-designed algorithms, of paramount importance is the *key*, denoted by k. Plaintext is denoted by m for message, and the ciphertext is denoted c. Both the encryption and decryption operations use the key, so the functions are expressed as

$$E_k(m) = c$$
$$D_k(c) = m. \tag{1.47}$$

There are two general types of key-based algorithms: symmetric and public-key. *Symmetric algorithms*, sometimes called conventional algorithms usually require that the sender and the receiver agree on a key before they can communicate securely. Note that this is a fundamental issue with this type of security algorithm. That is, in order to communicate securely, we must first communicate securely!

Public-key algorithms, also known as *asymmetric algorithms*, are designed so that the key used for encryption is different than the key used for decryption. Furthermore, the decryption key cannot (at least in a reasonable amount of time) be calculated from the encryption key. The algorithms are called "public-key" because the encryption key can be made public. We now discuss some of the issues of balancing these approaches against the need for quantifiable security.

The whole point of cryptography is to keep the plaintext secret from eavesdroppers. All algorithms exhibit varying degrees of resistance, so it is necessary to quantify the degree to which an algorithm can withstand an *attack* made by a cryptanalyst. Of paramount importance is the concept of *unconditional security*. An algorithm is unconditionally secure if, no matter how much ciphertext a cryptanalyst has, there is not enough information to recover the plaintext. Given unlimited computing resources, only a *one-time pad* is unbreakable. To understand why this

is so, we construct a one-time pad and perform a little analysis. We start with a message string $m \in \{0,1\}^n$ and a random key string, $k \in \{0,1\}^n$. By random we mean each key string occurs with probability $P_k = 1/2^n$. We construct the cipher text by modulo-two summing the key with the message such that

$$c = E_k(m) = m \oplus k. \tag{1.48}$$

Next we convince ourselves that the ciphertext has the same probability distribution as the key. Assuming P_m is the probability of a message string, we calculate the probability that the ciphertext is equal to any particular value as follows

$$\begin{aligned} \Pr[c = z] &= \sum_{\substack{k,m \in \{0,1\}^n \\ m \oplus k = z}} P_m \cdot P_k \\ &= \sum_{m} P_m \sum_{k = m \oplus z}^{k} \frac{1}{2^n} \\ &= \sum_{m} P_m \cdot \frac{1}{2^n} \\ &= \frac{1}{2^n}. \end{aligned} \tag{1.49}$$

From this we see that the ciphertext has the same probability distribution as the key, but note the dependence of this upon $P_k = 1/2^n$ *exactly*. If the outcome probability of a key differs at all from this value, information about the plaintext will be present in the ciphertext. This will occur if the key is generated by a source that is only *pseudo-random*, or if the number of message bits is greater than the number of key bits since a key bit would then have to be used multiple times in construction of the ciphertext. This latter fact highlights the impracticality of the one-time pad since a new key of length at least that of the message to be exchanged is required

for each secret. If a protocol that does not have a one-time pad at its core is used to secure communications, the eavesdropper with sufficient computational resources can recover the message. Recognition of the impracticality of key distributions in a classical setting has led to the development of ways to get around the problem. Public-key cryptography skirts the issue by using the concept of trap-door one-way functions to allow the encryption key to made public by the person wishing to receive a secure message while the decryption key is kept private. Quantum key distribution using a protocol like BB84 [14] solves the problem by exchanging the key over a quantum channel, in a manner that ensures that any eavesdropping can be detected. Discussion of these two very interesting parts of classical and quantum cryptography is beyond the scope of this thesis and the interested reader is directed to the references. We now turn our attention to bit commitment protocols.

Two parties, especially two *un-trusting* parties, may wish to do more than just communicate securely. Using digital communications, they may wish to compute a value such as the result of a fair coin toss, generate a shared random sequence, authenticate each other's identity or sign a contract. One of the key foundation protocols applicable to these needs is called *bit commitment*. This important protocol has many interesting applications as diverse and thought-provoking as *zero-knowledge proofs* [29, pages 101-109]. Again, detailed discussion of applications is beyond the scope of this thesis, and the reader is directed to the references for more information. We now focus on the problem of implementing bit commitment with quantifiable properties.

1.3.2 Bit Commitment

In this section we introduce the concept of bit commitment. We discuss several implementations of bit commitment protocols. We pay particular attention to bit commitment using one-way functions, which we use as a springboard into the Goldreich-Levin theorem.

Bit commitment can best be understood using a simple physical example. Suppose Alice wants to commit to a prediction of a particular event in the future (e.g., whether a particular stock will be a winner or not), but she does not want to reveal her prediction until sometime later. Bob, on the other hand, wants to make sure that Alice cannot change her mind after she has committed to her prediction. A physical implementation of this commitment might proceed as follows. Alice chooses a bit, 0 or 1, and writes it on a piece of paper, which she deposits in a locked box. She gives the box to Bob but keeps the key. She cannot change what she wrote, and without the key, Bob cannot open the box. But at some later point, Alice can give Bob the key and reveal her bit. This concrete example illustrates the two key requirements of an effective bit commitment scheme. The scheme must be both *concealing* and *binding*. We will now discuss how to implement schemes without the encumbrances of physical safes and keys.

Although bit commitment appears to be quite a simple concept, it is actually quite difficult to come up with robust digital schemes. The difficulty arises in trying to make the proposal simultaneously concealing *and* binding. In fact, *unconditional* bit commitment is actually not possible. The proof of the impossibility of unconditional *classical* bit commitment is beyond the scope of this thesis. (A sketch of the

proof of the impossibility of *unconditional* quantum bit commitment is presented in Chapter 3.) However, we will illustrate that finding a protocol with reasonably good concealing and binding properties is a challenge. Consider the following candidate for a bit commitment protocol. Suppose Alice wishes to commit to a bit, b. She selects a random n-bit string, r. She has a bit-commit function $Commit(r, b) = r + b$ (the addition is modulo 2^n), which gives as output an n-bit commitment string, c, which she sends to Bob. This protocol is illustrated in figure 1.8. It is assumed

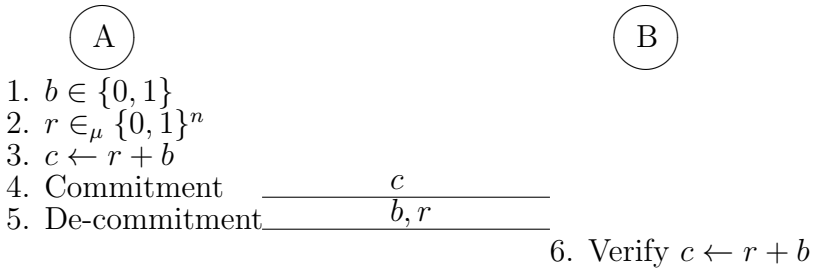

Figure 1.8: A naive bit commitment protocol

that Bob also knows the bit-commit function, $Commit(r, b)$. When Alice is ready to de-commit, she sends the bit, b, and the random n-bit string, r to Bob, who then verifies the bit that Alice committed to. On the face of it, this protocol appears to satisfy our requirements that the scheme be both concealing and binding. It is *perfectly* concealing in the sense that the commitment string, c is dependent on there being 2^n random strings. But is it binding? The answer is no because any particular commitment string could equally be a commitment to 0 or to 1 since Bob has no knowledge of which random string was selected. Thus a dishonest Alice can change her commitment at will. We could add more steps to this protocol, such as having Bob generate the string r and Alice encrypt the commitment with a one-time pad,

but we would still have a protocol with poor binding properties. We need another approach to find a bit commitment protocol with reasonable concealing *and* binding properties.

Other bit commitment protocols exist and of particular interest are those that use one-way functions. We are now going to define a special case of a one-way function. A one-way permutation, for which we will use the abbreviation OWP, is a length preserving one-way function having a unique inverse and is defined as follows.

Definition 5 A *one-way permutatin* $f : \{0,1\}^n \rightarrow \{0,1\}^n$ has the following properties:

1. Given x, $f(x)$ is computable in polynomial time.
2. Each $f(x)$ has a unique inverse.
3. Let A be any $t(n)$-time algorithm with success probability $\varepsilon(n)$. If x is randomly selected from all n-bit strings, that is $x \in_\mu \{0,1\}^n$, and $y \leftarrow f(x)$ and $\Pr[A(y) \in f^{-1}(y)] = \varepsilon(n)$, then $\frac{t(n)}{\varepsilon(n)} \in n^{\omega(1)}$.

The last statement in this definition means that the ratio of the time resources to any algorithm's success probability is at best *super-polynomial*. See Table 1.1 for some examples of functions that are super-polynomial. We can use a OWP to implement a bit commitment scheme that is *computationally* concealing and *perfectly* binding. We will first present another naive implementation, which is depicted in Figure 1.9. We see that this protocol is indeed perfectly binding in that once Alice has made the commitment, there is no way she can change it because the OWP is one-to-one. But how concealing is it? The critical part of this protocol is how the string r and the

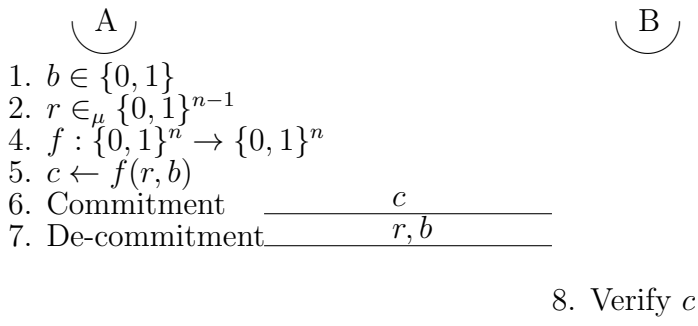

Figure 1.9: A naive bit commitment protocol based on a one-way permutation

bit b are combined into the n-bit argument of the OWP. Indeed depending on the specific combination, it is important to note that the protocol may not be concealing at all. This is because it may be computationally very easy for Bob to determine the single bit in the commitment string that the bears information about b. We note that the OWP appears to solve the binding problem, but it would nice to quantify the degree of concealment. Rather than mixing the commitment bit into the n-bit argument of the one way function, it would be nice to somehow define a single bit that is also *hidden* by the OWP. Ideally we would also like to quantify how hard it is for Bob to determine the value of the commitment given this *hard* bit and any other ancillary information necessary to make the commitment. For this we look to the concept of a *hard predicate* of a one-way permutation.

A hard predicate of a OWP is a single bit that is easy to determine given x but is hard to determine given $f(x)$. We give the following definition of a hard predicate.

Definition 6 A *hard predicate* of a one-way permutation $f : \{0,1\}^n \to \{0,1\}^n$ is a function $h : \{0,1\}^n \to \{0,1\}$ that has the following two properties:

1. h is computable in polynomial time.

2. Let A be any $t(n)$-time algorithm with success probability $\varepsilon(n)$. If $x \in_\mu \{0,1\}^n$, and $\Pr[A(f(x)) = h(x)] = \frac{1}{2} + \varepsilon(n)$, then $\frac{t(n)}{\varepsilon(n)} \in n^{\omega(1)}$.

The next question is how do we find a hard predicate of f. For this we turn to the *Goldreich-Levin* Theorem. This theorem provides a generic means for determining a hard predicate given any one-way permutation. In Chapter 2 we state the Goldreich-Levin Theorem and its adjunct, the Goldreich-Levin *black-box query problem*. We proceed to develop upper and lower bounds for the query problem in a classical setting. In Chapter 3, we repeat the analysis in a quantum setting. Finally in Chapter 4, we return to the analysis of both quantum and classical bit commitment protocols using the Goldreich-Levin Theorem.

Chapter 2 Classical Goldreich-Levin Theorem

2.0 Introduction

The Goldreich-Levin (G-L) Theorem [20] was first presented and an upper bound proven in 1989. An improvement to the original bound was discussed by O. Goldreich in 1999 [21]. In this chapter, we explore the *classical* G-L Theorem as distinguished from the *quantum* G-L Theorem, which we study in Chapter 3. Here, we give a detailed analysis of both the original and the improved classical upper bounds and prove the classical lower bound.

The context of the Goldreich-Levin (G-L) Theorem is to find a so-called hard predicate for a one-way length preserving function f that is a one-way permutation. Recall that as discussed in Chapter 1, a one-way permutation is loosely defined as a permutation that uniquely maps an n-bit string, x, into another n-bit string, $f(x)$, where it is computationally easy to perform the mapping but computationally difficult to invert the mapping. A hard predicate of a one-way permutation is a single bit, h, which is hidden by the permutation f in the following sense. Given x, computing $h(x)$ is computationally easy, but given $f(x)$, computing $h(x)$ is computationally difficult. Formal definitions of one way permutations and hard predicates are given in Chapter 1, Definitions 5 and 6. The G-L Theorem offers a generic means for constructing hard predicates given any one-way permutation. A key step in the reduction of the G-L Theorem is the solution of a black-box, or oracle, query problem referred to herein as the G-L problem.

The G-L problem is concerned with determining the number of times a two-

part oracle must be queried in order to determine an unknown n-bit string a. The two oracles are the inner-product oracle, IP, and the equivalence oracle EQ. The query problem is presented pictorially in Figure 2.10. The inner-product of two n-bit strings, a and x, is denoted $a \cdot x$, and is defined as

$$a \cdot x = a^{(1)}x^{(1)} + a^{(2)}x^{(2)} + \cdots + a^{(n)}x^{(n)}. \tag{2.50}$$

The additions here are modulo two, so the result of an IP query is a single bit. As depicted in Figure 2.10, the IP oracle either returns the correct result of the inner product of the string x with the string a or it returns the inverted result. The

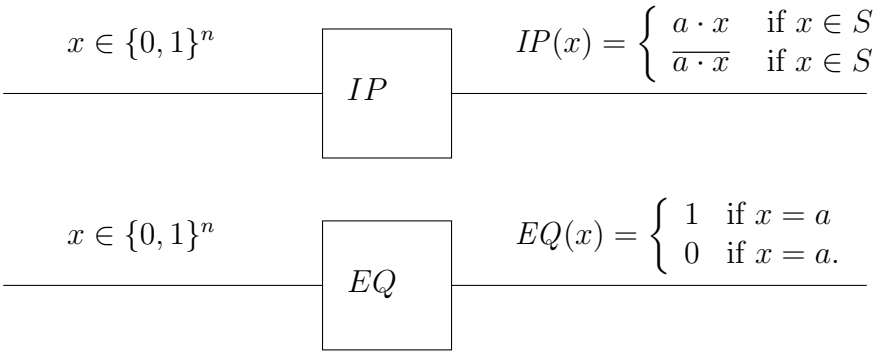

Figure 2.10: The Goldreich-Levin Query Problem

correctness of the result depends on whether the string x is a member of the set S or not. The set $S \subseteq \{0,1\}^n$ is chosen subject to the condition that $|S| = (\frac{1}{2} + \varepsilon)2^n$. The value of $\varepsilon > 0$ is termed the *advantage* of the IP oracle. The output of an inner-product query is thus only slightly correlated with $a \cdot x$ in the sense that

$$\Pr[IP(x) = a \cdot x] \geq \tfrac{1}{2} + \varepsilon. \tag{2.51}$$

The $EQ(x)$ oracle is used to determine if the input string, x, is equal to the unknown string a or not.

The focus of this chapter is to determine both an upper and a lower bound of the query complexity of the G-L problem. Firstly, we study the query complexity of the problem using a specially constructed algorithm that makes use of these two oracles. We bound the success probability of the algorithm, which enables us to provide an upper bound on the number of queries required to determine the correct value of the unknown string. Secondly, we provide a proof of the lower bound of the query complexity by analyzing the amount of information that is revealed in each IP query. We will discuss the generation and use of hard-core predicates in Chapter 4.

2.1 Upper Bounding the Classical G-L Problem

2.1.1 The Problem

The discussion presented in the following expands on the ideas given by M. Bellare in his 1999 manuscript [3]. In solving the G-L query problem, we are allowed to use any combination of the two oracles in our quest to determine a with the idea that we wish to obtain the value using the fewest total number of queries. Of course we could simply choose to use only EQ queries in which case we would determine the correct value of a on average after making approximately 2^{n-1} queries, but an exponential upper bound is not very interesting.

Can using the IP query reduce the number of queries? The answer is yes with the amount of the reduction dependent on the value of ε, the advantage of the IP

oracle. Consider the case where ε is equal to $\frac{1}{2}$; that is, the IP oracle always returns the correct value of the inner product. In this case we could adopt the strategy of determining a one bit at a time. We would achieve this by sequentially offering the basis strings, e_k, (those strings with a one in the k^{th} position and zeroes in all other positions) to the IP oracle. This strategy would result in the correct value of a after only n queries without even having to resort to the use of the EQ oracle. We see that in this exact case, using only IP queries we make exponentially fewer queries than we make by blindly using the EQ oracle. The analysis becomes more interesting in the case where ε is less than $\frac{1}{2}$. Before proceeding with the analysis of this case, we must consider what we mean by the statement that the IP oracle is returning the correct value of the inner product with probability $\frac{1}{2} + \varepsilon$. Simply stated, it means that there are a collection of good strings, for which the IP oracle returns the correct value of the inner product, and a collection of bad strings for which it returns the incorrect value. The ratio of good strings to the whole is $\frac{1}{2} + \varepsilon$; while the ratio of the bad strings to the whole is $\frac{1}{2} - \varepsilon$. When ε is less than $\frac{1}{2}$, the problem with the approach we outlined for the case where $\varepsilon = \frac{1}{2}$ is that it is possible that some (or all) of our basis strings could happen to be bad strings in which case the previous strategy would not work at all.

In order to get around the problem of our basis strings potentially being in the bad bunch, we compare two *self-correcting* strategies in two separate algorithms. We carefully construct these algorithms and determine bounds on their success probability in terms of the number of bits n and the advantage ε. We then use these bounds to estimate the query complexity of IP and EQ queries required to determine the correct value of the string a.

2.1.2 The High Advantage Case

The *high advantage* case is where $\varepsilon > \frac{1}{4}$. We begin by considering the case, where ε is very close to $\frac{1}{2}$, say $\frac{1}{2} - \delta$. With this definition, we can rewrite Equation 2.51 as

$$\Pr[IP(x) = a \cdot x] \geq 1 - \delta. \tag{2.52}$$

In order to develop a workable algorithm, we look toward the first self-correcting strategy to remove the possibility that our input strings are all in the bad bunch. This strategy reduces the worst case input string to the average case by invoking the IP oracle only on random points. We achieve this by relying on the linearity of the inner product, $a \cdot x = a \cdot (x \oplus r) \oplus a \cdot r$. Here r is a random string. An algorithmic strategy is to use

$$IP(x \oplus r) \oplus IP(r) \tag{2.53}$$

to approximate $IP(x)$.

We now introduce a simple algorithm, which we will refer to as *ExtractA*. This algorithm is so named because its aim is to *extract* the unknown string a and because it is the first of two such algorithms, *ExtractA* and *ExtractB*, that employ different self-correction strategies. Algorithm *ExtractA* is presented in the box denoted Algorithm 1. Note that our algorithm queries the IP oracle twice for each input string

1: Input: e_k
2: $r_k \overset{random}{\longleftarrow} \{0,1\}^n$
3: $bit_1 \longleftarrow IP(e_k \oplus r)$
4: $bit_2 \longleftarrow IP(r)$
5: Return $bit_1 \oplus bit_2$

Algorithm 1: Algorithm *ExtractA*

— once with a random string and once again with the same random string added

to the input string. The algorithm then returns the difference, which is a single bit having value 1 or 0. Although we will show that this is a workable strategy for the high advantage case, we will also show that using two *IP* queries for each input string severely limits the size of the advantage to which this strategy is applicable. There are two *IP* queries in the approximation given by Expression 2.53. For each individual query, the probability that the returned value is not equal to $a \cdot x$ is δ. Since our algorithm must make two queries to the *IP* oracle, we have a bit more work to bound its failure probability as follows:

$$\begin{aligned} \Pr_r[IP(x \oplus r) \oplus IP(r) = a \cdot x] &\leq \Pr_r[IP(x \oplus r) = a \cdot (x \oplus r) \vee IP(r) = a \cdot r] \\ &\leq \Pr_r[IP(x \oplus r) = a \cdot (x \oplus r)] + \Pr_r[IP(r) = a \cdot r] \\ &= \delta + \delta = 2\delta. \end{aligned}$$

Here, we have used the *union bound* to bound the probability of our algorithm failing. The success probability of algorithm *ExtractA* is thus

$$\Pr[ExtractA(e_k) = a \cdot e_k] \geq 1 - 2\delta. \tag{2.54}$$

Assuming the input, e_k, are again the n basis strings and a new random string, r, is drawn for each input, the probability that all n calls return the right answer is

$$(1 - 2\delta)^n \geq 1 - 2n\delta. \tag{2.55}$$

If we want to keep the error probability limited to $\frac{1}{2}$, then it is sufficient that $\delta \leq \frac{1}{4n}$. Thus we see that δ tends to 0 as n tends to infinity. If we keep within this very restrictive bound, our algorithm has an error probability of at most $\frac{1}{2}$, by making $2n$ queries to the *IP* oracle and 1 query to the *EQ* oracle; an error probability of

$\frac{1}{4}$ by making $4n$ queries to the *IP* oracle and 2 queries to the *EQ* oracle, and error probability γ by making $O\left(n \log \frac{1}{\gamma}\right)$ queries to the *IP* oracle and $O\left(\log \frac{1}{\gamma}\right)$ queries to the *EQ* oracle.

We conclude from the preceding analysis that we need to modify our strategy if we wish to work outside the restriction of $\delta \leq \frac{1}{4n}$. We note that with the current strategy, the success probability for each bit is only $1 - 2\delta$. The success probability for an n-bit string is this quantity raised to the n^{th} power, which rapidly goes to zero with increasing n. We can improve this situation through a technique sometimes called *probability amplification*. This involves making $m > 1$ calls to our algorithm with the same value of e_k, but with a new random sting r drawn each time. Since our algorithm returns a bit each time, we can sum the result of the m queries, and if the result is $\geq \frac{m}{2}$ conclude that the bit is a 1 otherwise it is 0. We present the improved algorithm *ExtractAI* in the box denoted Algorithm 2. We now proceed to bound the

```
1: Input: e_k, m
2: sum ⟵ 0
3: for i = 1 to m do
4:    sum ⟵ sum + ExtractA(e_k)
5: end for
6: if sum ≥ m/2 then
7:    y^(k) ⟵ 1
8: else
9:    y^(k) ⟵ 0
10: end if
11: Return y^(k) {This is an estimate of the kth bit of a}
```

Algorithm 2: Algorithm *ExtractAI*

success probability of this algorithm. We recognize that the **for** loop in algorithm *ExtractAI* is just a sequence of m Bernoulli trials with success probability p, where we use the convention that for a successful trial the returned bit is 1. Thus there are

two cases we have to consider — where $p \geq 1 - 2\delta$, which corresponds to the case $a \cdot e_k = 1$, or where $p \leq 2\delta$ which corresponds to the case $a \cdot e_k = 0$. Either way, the resulting probability distribution is Binomial, and we have to determine which of the two cases it is. Figure 2.11 is a graphical representation of the two possible distributions for $m = 20$ and $\delta = \frac{1}{8}$. Note that the two probability distributions show significant overlap in the region near $m/2$. If for the $a \cdot e_k = 0$ case, the sum happened to be $> m/2$, we would make the wrong conclusion about the value of the bit. Likewise for the $a \cdot e_k = 1$ case if the sum happened to be $< m/2$, we would make

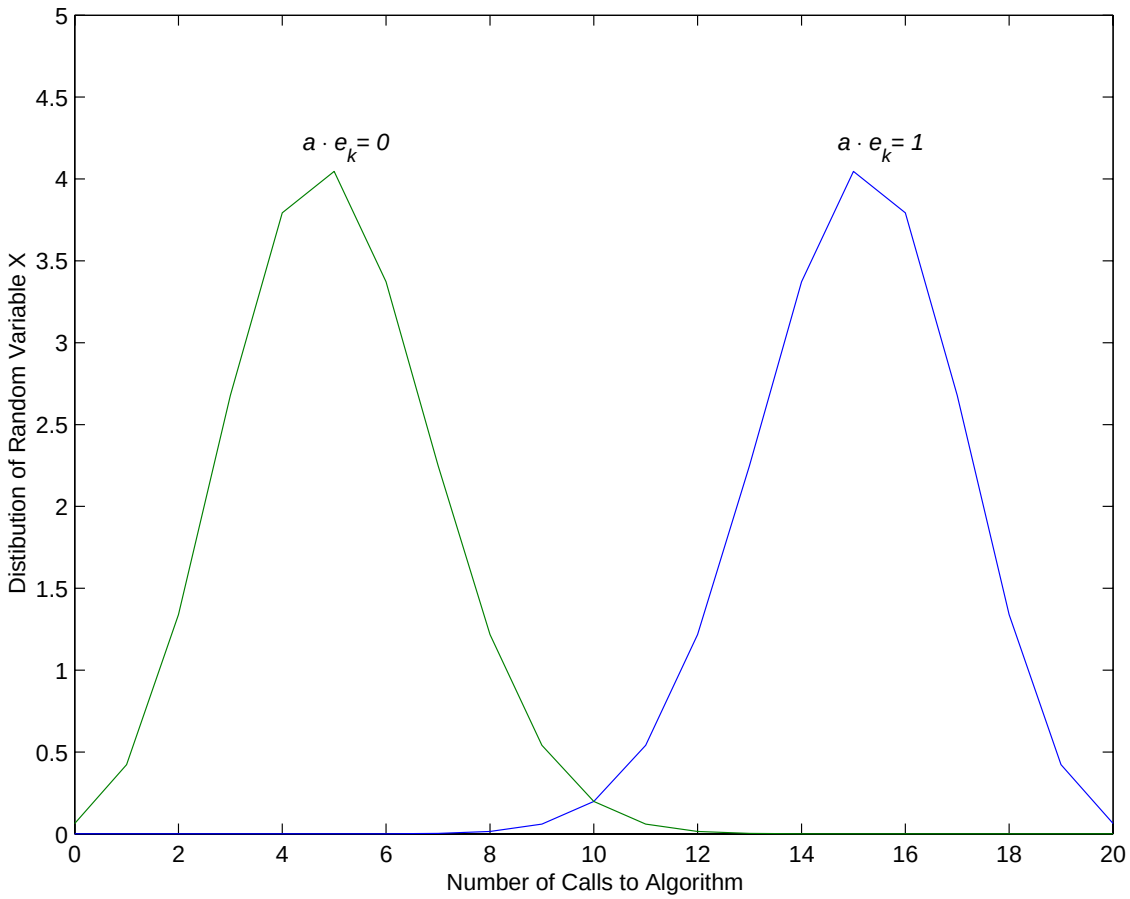

Figure 2.11: Binomial Distributions for $a \cdot e_k = 0$ and $a \cdot e_k = 1$.

the wrong conclusion. The size of these *tails* of the distributions offers us a means by which we can bound the error probability of our modified algorithm. We use *tail-inequalities* to bound this overlap probability. Our strategy will be to sum the result of m queries using the value $m/2$ as our decision threshold. This will improve matters so that rather than having the very restrictive bound of δ inversely varying with n, we can achieve acceptable success probability with a polynomial number of queries and $\delta < \frac{1}{4}$.

We begin the analysis by defining the random variable $X = X_1 + X_2 + \ldots + X_m$ as the variable *sum* in algorithm *ExtractAI*. The X_i are binomially distributed *independent* random variables, which allows us to apply the powerful *Chernoff* bound to the distribution's tail. However here we are going to apply the weaker *Chebychev* bound because it is also applicable to *pairwise independent* random variables, which we will be employing later when we solve the general case. The expected value and the variance of a binomially distributed random variable are

$$\begin{aligned} \mu(X) &= mp \\ Var(X) &= m(1-p)p. \end{aligned} \tag{2.56}$$

Chebychev's inequality states

$$\Pr\left[|X - \mu| > A\right] \leq \frac{Var(X)}{A^2}. \tag{2.57}$$

This inequality formalizes what we intuitively expect — that the probability that a particular value falls greater than a certain distance from the mean of the distribution diminishes as that distance increases. Now we have two cases to consider. Firstly when $e_k \cdot a = 1$, we have $p \geq 1 - 2\delta$, and we want to find the probability that $X < \frac{m}{2}$.

Secondly when $e_k \cdot a = 0$, we have $p \leq 2\delta$, and we want to find the probability that $X > \frac{m}{2}$. In both cases, it can be shown that the absolute distance from the mean can be expressed as

$$|X - \mu| > \frac{m}{2} - 2m\delta. \tag{2.58}$$

Combing the results of Equation 2.56 with Equation 2.57 and Equation 2.58 allows us to write

$$\Pr\left[|X - \mu| > \frac{m}{2} - 2m\delta\right] \leq \frac{m(1-2\delta)(2\delta)}{\frac{m}{2} - 2m\delta\ ^2}. \tag{2.59}$$

The preceding bounds the error probability of Algorithm *ExtractAI*, which outputs a single bit after a majority vote over the m queries. After simplification it is expressed as

$$\Pr\left[ExtractAI(e_k, m) = a \cdot \varepsilon_k\right] \leq \frac{(1-2\delta)(2\delta)}{m(\frac{1}{2} - 2\delta)^2}. \tag{2.60}$$

We note from this equation that there is a vertical asymptote at $\delta = \frac{1}{4}$. We next bound the probability of successfully recovering all n bits of the string a. We write this success probability as:

$$\begin{aligned} \Pr\left[(EAI(e_1), EAI(e_2), \ldots, EAI(e_n)) = a\right] &\geq 1 - \frac{(1-2\delta)(2\delta)}{m(\frac{1}{2} - 2\delta)^2}^{\ n} \\ &\geq 1 - \frac{n}{m}\frac{(1-2\delta)(2\delta)}{(\frac{1}{2} - 2\delta)^2}, \end{aligned} \tag{2.61}$$

where we have used the shorthand $EAI(e_k)$ for $ExtractAI(e_k, m)$. Finally bounding this success probability to a half, we have

$$\frac{1}{2} \leq \frac{n}{m}\frac{8\delta(1-2\delta)}{(1-4\delta)^2}. \tag{2.62}$$

This is > 0 when $0 < \delta < \frac{1}{4}$ and approaches $+\infty$ as $\delta \to \frac{1}{4}+$. It is instructive to express Equation 2.62 in terms of a new variable $\varepsilon_{\frac{1}{4}} = 1 - 4\delta$, which allows us to

bound the number of calls over which we make a majority vote as

$$m \leq \frac{2n \quad 1 - \varepsilon_{\frac{1}{4}}^2}{\varepsilon_{\frac{1}{4}}^2}. \tag{2.63}$$

We note that algorithm *ExtractAI* makes $2nm$ queries to the IP oracle. Thus for $\varepsilon_{\frac{1}{4}} > 0$, which corresponds to $\delta < \frac{1}{4}$, we provide an upper bound for the number of IP queries as

$$q_{IP} = O \quad \frac{n^2}{\varepsilon_{\frac{1}{4}}^2} \quad . \tag{2.64}$$

We conclude this sub-section noting that probability amplification must form a part of an effective algorithmic strategy, but we have to get past the "brick wall" of $\delta = \frac{1}{4}$. Our current self-correction strategy performs two IP queries to extract each bit before probability amplification. The two queries effectively doubles the error probability resulting in the "wall" at $\delta = \frac{1}{4}$ rather than at $\delta = \frac{1}{2}$ as we desire. We thus conclude that if we wish to solve for polynomially small ε, it may be more promising to make only one IP query to extract each bit.

2.1.3 General Case

The general case is when $0 < \varepsilon \leq \frac{1}{2}$. In the high advantage case we ran our algorithm over the n basis strings as input. We also relied on the linearity of the inner product function and used repeated runs to amplify probability. In the general case, where ε may be $\frac{1}{4}$ or less, we will again employ these tricks, but we also need an algorithm that with high enough probability determines $a \cdot e_k$ for a given e_k while making only *one* IP query. We will refer to this algorithm simply as Algorithm *ExtractB*. Before we state and prove a theorem that bounds the probability that *ExtractB* will return

the correct value of $a \cdot x$, we need to employ another clever trick. That is a method for producing M pairwise independent strings from $O(\log(M))$ random strings.

In order to explain this trick, we let m and M be integers with $M = 2^m - 1$. We define the set $[m] = \{1, \ldots, m\}$ and the listing $S_1, \ldots, S_M$ of all the non-empty subsets of [m] in the canonical order usually associated with counting binary numbers. It is helpful to think of each of the members of S_M, denoted S_i, as being the bit positions of those bits equal to 1 in the m-bit binary representation of the number i. For example, the usual 4-bit representation of the number 3 is 0011. Reading the first bit position from the right, we have the first and second bits equal to one as is indicated by $S_3 = \{1, 2\}$. This is not the only ordering we could imagine, but it is straightforward to write the first several S_i explicitly as

$$\begin{aligned}
S_1 &= \{1\} \\
S_2 &= \{2\} \\
S_3 &= \{1, 2\} \\
S_4 &= \{3\} \\
S_5 &= \{1, 3\} \\
S_6 &= \{2, 3\} \\
S_7 &= \{1, 2, 3\} \\
S_8 &= \{4\} \\
&\vdots \\
S_M &= \{1, 2, \ldots, m\}.
\end{aligned}$$

Now let $R = (r_1, \ldots, r_m)$ be an m-tuple of m random n-bit strings. We use the

definition of R and our listing of the M subsets of $[m]$ to construct the sequence $Q = (q_1, \ldots, q_M)$. Each of the q_i are formed as bitwise modulo-two sums of particular n-bit strings selected from R as

$$q_i = \bigoplus_{j \in S_i} r_j. \tag{2.65}$$

For clarity, we again write down several of the q_i explicitly as

$$\begin{aligned}
q_1 &= \bigoplus_{j \in S_1} r_j = r_1 \\
q_2 &= \bigoplus_{j \in S_2} r_j = r_2 \\
q_3 &= \bigoplus_{j \in S_3} r_j = r_1 \oplus r_2 \\
q_4 &= \bigoplus_{j \in S_4} r_j = r_3 \\
&\vdots \\
q_M &= \bigoplus_{j \in S_M} r_j = r_1 \oplus r_2 \oplus \cdots \oplus r_m.
\end{aligned}$$

It is important to note that whilst the r_k are independent random variables, the q_i are only *pairwise* independent since for every $i, j \in [M]$ with $i = j$ and $a, b \in \{0,1\}^n$

$$Pr[q_i = a \text{ and } q_j = b] = Pr[q_i = a] \cdot Pr[q_j = b]. \tag{2.66}$$

This can be understood by noting that if $q_i = q_j$, then there is some string r_k that belongs to one and not the other, and a modulo-two sum involving r_k is unpredictable from a sum not involving r_k. This means having the value of one of them does not help predict the value of another, but having the value of any two of them may help predict the value of the others. This pairwise independence is a crucial property of this construction that will be used to bound the query complexity.

Our new algorithm is also going to make one query $IP(x \oplus r)$, which is on a random point. Again, we need to find a way to modulo-two subtract the effect of adding a random string in order to approximate $IP(x)$. Recall that we achieved this in the high advantage case by making a second IP query as we did in Expression 2.53. To avoid making a second IP query here, we will achieve the same effect by defining a sequence of M single bits constructed using the inner product function. Our algorithm will then modulo-two subtract all possible values of these bits for each query on a random point. As an aside, we will also see that this modulo-two summing of all possible points will form the basis of *probability amplification* employed in algorithm *ExtractB*. We first define the sequence of m bits $B = (b_1, \ldots, b_m)$ as $b_k = a \cdot r_k$. We then define the sequence of M bits $D = (d_1, \ldots, d_M)$ with each of the bits calculated as

$$d_i = \bigoplus_{j \in S_i} b_j. \tag{2.67}$$

We write the first several d_i explicitly as

$$\begin{aligned}
d_1 &= \bigoplus_{j \in S_1} b_j = b_1 \\
d_2 &= \bigoplus_{j \in S_2} b_j = b_2 \\
d_3 &= \bigoplus_{j \in S_3} b_j = b_1 \oplus b_2 \\
&\vdots \\
d_M &= \bigoplus_{j \in S_M} b_i = b_1 \oplus b_2 \oplus \cdots \oplus b_m.
\end{aligned}$$

We again use the linearity of the inner product function to write

$$a \cdot (x \oplus \bigoplus_{j \in S_i} r_j) = (a \cdot x) \oplus \bigoplus_{j \in S_i} (a \cdot r_j) = (a \cdot x) \oplus \bigoplus_{j \in S_i} b_j. \tag{2.68}$$

Using Equation 2.67 and Equation 2.65, we rewrite Equation 2.68 as

$$a \cdot (x \oplus q_i) = (a \cdot x) \oplus d_i. \tag{2.69}$$

Finally Equation 2.69 is rearranged in order to approximate $IP(x)$ by the following quantity to be employed by our algorithm

$$IP(x \oplus q_i) \oplus d_i. \tag{2.70}$$

Armed with the Expression 2.70, we are now able to construct algorithm *ExtractB*. Once we have constructed and offered a step-by-step analysis of the algorithm, we state and prove a theorem bounding the algorithm's success probability. We now describe the intent of the algorithm.

We begin by noting that Expression 2.70 will form the heart of algorithm *ExtractB*. Comparing Expression 2.70 with Expression 2.53, which formed the heart of algorithm *ExtractA*, we note two crucial differences. Firstly in Expression 2.70, only one call is made to the *IP* oracle whereas two are made in Expression 2.53. This means that the error probability will not be doubled, which we will show allows algorithm *ExtractB* to work with polynomially small advantage rather than being limited to an advantage only polynomially smaller than $\frac{1}{4}$ as we saw in the high advantage case. Secondly, in Expression 2.70 we note that the d_i are not explicitly known. In algorithm *ExtractB*, we get around this problem by using a value of m sufficiently small to permit us to try all of the 2^m possible values of the d_i.

As with Algorithm *ExtractA*, Algorithm *ExtractB* performs a bit-by-bit extraction of the unknown string a. Once we think we have a candidate string, we offer it to the *EQ* oracle to see if it is the right one. We again perform probability amplification for each of the bits we extract, but rather than doing this over m random

strings and two IP queries as we did in algorithm *ExtractA*, we do it with M *pairwise* independent strings and only one IP query modulo two summed with all of the possible d_i bits. In the case of Algorithm *ExtractA*, we *chose* to bound the post-amplification error probability using Chebychev's inequality where the random variables were Binomially distributed. In the case of Algorithm *ExtractB*, we *must* bound the post-amplification error probability using Chebychev's inequality because the random variables are only pairwise independent. After we have described Algorithm *ExtractB*, we will state and prove a theorem of its success probability and consequently bound the number of queries for polynomially small ε.

```
1: Input: e_k, n, m, M
2: Initialize(e_k, n, M)
3: for l = 1 to M do
4:    Let B = b_1 ... b_m be the binary representation of l − 1
5:    for k = 1 to n do
6:       Amplify(B, T_k)
7:    end for
8:    y ⟵ y^(1) ... y^(n)
9:    if EQ(y) = 1 then
10:      a ⟵ y
11:   end if
12: end for
13: Return a
```

Algorithm 3: Algorithm *ExtractB*

The main part of Algorithm *ExtractB* is presented in the box denoted Algorithm 3, and its two subroutines, *Initialize* and *Amplify*, are presented in the boxes denoted Algorithm 4 and Algorithm 5 respectively. In the first line of Algorithm *ExtractB*, the subroutine *Initialize* is called. This subroutine returns the n by M *table* of the

1: Input: e_k, n, M
2: **for** $k = 1$ to m **do**
3: $\quad r_k \overset{random}{\longleftarrow} \{0,1\}^n$
4: **end for**
5: **for** $i = 1$ to M **do**
6: $\quad q_i \longleftarrow \bigoplus_{j \in S_i} r_j$
7: $\quad$ **for** $k = 1$ to n **do**
8: $\quad\quad T_{k,i} \longleftarrow IP(e_k \oplus q_i)$
9: $\quad$ **end for**
10: **end for**
11: Return T

Algorithm 4: Subroutine *Initialize*

results of the IP queries,

$$T = \begin{bmatrix} T_{1,1} & \cdots & T_{1,i} & \cdots & T_{1,M} \\ \vdots & & \vdots & & \vdots \\ T_{k,1} & \cdots & T_{k,i} & \cdots & T_{k,M} \\ \vdots & & \vdots & & \vdots \\ T_{n,1} & \cdots & T_{n,i} & \cdots & T_{n,M} \end{bmatrix}, \tag{2.71}$$

which will be used later on in the algorithm for probability amplification in a manner similar to what we did in the high advantage case. We also denote T_k as the M-component, k^{th} row of T. We generate the table T by making nM calls to the IP oracle as follows. For each of the n basis stings e_k, an IP query is made with e_k modulo two summed with one of the M pairwise independent strings, Q, generated from the m random n-bit stings, R, per Equation 2.65. Note that it is only in the subroutine *Initialize* that the IP oracle is called. It is also helpful to note that each T_k consists of the results of M IP queries made with the modulo-two sum of a *unique* e_k with each of the random strings Q. After this initialization step, in line 3

of Algorithm *ExtractB* we enter a loop in which we sequentially enumerate the M, m-bit strings, B. For each one of the m-bit strings and each of the n basis strings, we then call the subroutine *Amplify*. It is in this subroutine that we perform the probability amplification necessary to achieve an acceptable success probability.

1: Input: B, T_k
2: $sum \longleftarrow 0$
3: **for** $i = 1$ to M **do**
4: $\quad d_i \longleftarrow \bigoplus_{j \in S_i} b_j$
5: $\quad c_i \longleftarrow T_{k,i} \oplus d_i$
$\quad sum \longleftarrow sum + c_i$
6: **end for**
7: **if** $sum \geq \frac{M}{2}$ **then**
8: $\quad y^{(k)} \longleftarrow 1$
9: **else**
10: $\quad y^{(k)} \longleftarrow 0$
11: **end if**
12: Return $y^{(k)}$ {This is an estimate of the *kth* bit of a}

Algorithm 5: Subroutine *Amplify*

In the subroutine *Amplify*, the counter *sum* is used as the basis for making the majority vote decision in our estimate of each bit, $a \cdot e_k$. We begin the subroutine by setting this counter to 0. One of the inputs to the subroutine is an m-bit string $B = b_1 \ldots b_m$. We use Equation 2.67 to generate M pairwise independent bits from this input string. Each one of these M bits is then sequentially modulo-two summed to the corresponding table entry $T_{k,i}$, which removes the effect of making the IP query on a random point. The result of this operation is then arithmetically added to the counter *sum*. Once this operation has been performed M times, a majority vote decision is made by comparing the counter, *sum*, to $\frac{M}{2}$. If it is greater than this value, we estimate that the bit $a \cdot e_k$ is 1; otherwise it is 0. We then return this

estimate of the kth bit of $a \cdot x$ to the main algorithm.

Algorithm *ExtractB* thus performs a bit-by-bit extraction for all n bits by making n calls to the subroutine *Amplify* for each m-bit string, B. Once a candidate string has been returned by the subroutine, it is then offered to the EQ oracle to see if it is the correct one. We note the key role that the EQ oracle plays in this algorithm — without it there is little chance of success. This extraction continues for all possible 2^m values of the string B, and as a result, there will be a particular string B^* that satisfies the relation $b_j = a \cdot r_j$ for $j = 1, \ldots, m$. It is for this particular string that we have a chance of recovering the string a. To see this, consider that in the specific case where $i = 3$ in subroutine Amplify we have

$$\begin{aligned} T_{(k,3)} \oplus d_3 &= IP(e_k \oplus q_3) \oplus d_3 \\ &= IP(e_k \oplus r_1 \oplus r_2) \oplus (b_1 \oplus b_2) \\ &= IP(e_k \oplus r_1 \oplus r_2) \oplus (a \cdot r_1 \oplus a \cdot r_2). \end{aligned} \tag{2.72}$$

The preceding has two outcomes, which may be expressed as

$$T_{(k,3)} \oplus d_3 = \begin{cases} a \cdot e_k & \text{if } e_k \oplus r_1 \oplus r_2 \in S \\ \overline{a \cdot e_k} & \text{if } e_k \oplus r_1 \oplus r_2 \in S. \end{cases} \tag{2.73}$$

For the specific string B^*, Equation 2.72 and Equation 2.73 are true for all i. This observation will allow us to bound the success probability of the algorithm *ExtractB*, but first we state and prove a bound on subroutine *Amplify*'s failure probability.

Theorem 1 *Let $M = 2^m - 1$ and B^* be the m-bit string for which $b_j = a \cdot r_j$ for $j = 1, \ldots, m$. Also, let e_k be an n-bit string with a one in the k^{th} position and zeroes elsewhere, and let T_k be the M-component k^{th} row of T. Then for any e_k, we have* $\Pr[Amplify(B^*, T_k) = a \cdot e_k] \leq 1/M\varepsilon^2$

Proof: Our proof parallels that presented in the low advantage case. We begin by noting that the counter *sum* in subroutine *Amplify* is a random variable, which we rename Y for convenience and express as

$$Y = Y_1 + Y_2 + \ldots + Y_M. \tag{2.74}$$

We again have two cases to consider, $e_k \cdot a = 1$, for which we label the random variable $Y^{(1)}$, and $e_k \cdot a = 0$, for which we label the random variable $Y^{(0)}$. Given these definitions, the expectations for the individual events comprising $Y^{(1)}$ and $Y^{(0)}$ have the following forms

$$\begin{aligned} E\ \ Y_i^{(1)} &:= 1 \cdot \Pr\ \ Y_i^{(1)} = 1\ \ + 0 \cdot \Pr\ \ Y_i^{(1)} = 0 \\ &= \Pr\ \ Y_i^{(1)} = 1 \\ &= \tfrac{1}{2} + \varepsilon, \end{aligned} \tag{2.75}$$

and

$$\begin{aligned} E\ \ Y_i^{(0)} &:= 1 \cdot \Pr\ \ Y_i^{(0)} = 1\ \ + 0 \cdot \Pr\ \ Y_i^{(0)} = 0 \\ &= \Pr\ \ Y_i^{(0)} = 1 \\ &= \tfrac{1}{2} - \varepsilon. \end{aligned} \tag{2.76}$$

For both cases $Y_i = {Y_i}^2$, and the variance of Y_i is simply expressed as

$$\begin{aligned} Var[Y_i] &:= E[{Y_i}^2] - E[Y_i]^2 \\ &= E[Y_i](1 - E[Y_i]) \\ &= (\tfrac{1}{2} + \varepsilon)(\tfrac{1}{2} - \varepsilon) \\ &= \frac{1}{4} - \varepsilon^2. \end{aligned} \tag{2.77}$$

Note that we shall suppress the index that distinguishes the two cases whenever the results are identical. Using μ to denote $E[Y]$, the linearity of expectation tells us that

$$\begin{aligned} \mu^{(1)} &= M\ \left(\tfrac{1}{2}+\varepsilon\right)\ , \text{ and} \\ \mu^{(0)} &= M\ \left(\tfrac{1}{2}-\varepsilon\right)\ . \end{aligned} \tag{2.78}$$

We noted in Equation 2.66 that the $q_1, \ldots, q_M$ are pairwise independent random variables. We observe that since the Y_i are derived through the combination of modulo-two sums with the q_i and IP queries on a fixed set S, the Y_i are also pairwise independent. With this observation it can be shown [3, Lemma 5] that the variance of the sum, Y can be expressed

$$Var[Y_1 + \cdots + Y_M] = M \cdot Var[Y_i]. \tag{2.79}$$

We will again use Chebychev's inequality to bound the failure probability. We have two cases to consider. Firstly when $e_k \cdot a = 1$, we have $E\ \left[Y_i^{(1)}\right] \geq \frac{1}{2} + \varepsilon$, and we want to find the probability that $Y < \frac{M}{2}$. Secondly when $e_k \cdot a = 0$, we have $E\ \left[Y_i^{(0)}\right] \leq \frac{1}{2} - \varepsilon$, and we want to find the probability that $Y > \frac{M}{2}$. In both cases, it can be shown that the absolute distance from the mean can be expressed as

$$|Y - \mu| > M\varepsilon. \tag{2.80}$$

Using Chebychev's inequality, which is give in Equation 2.57 along with Equation 2.80 and Equation 2.79, we write

$$Pr\left[|Y - \mu| > M\varepsilon\right] \leq \frac{M\ \left(\frac{1}{4} - \varepsilon^2\right.}{\left(M\varepsilon\right)^2}. \tag{2.81}$$

The preceding bounds the error probability of subroutine *Amplify*. After simplification, it is expressed

$$\begin{aligned} \Pr[Amplify(B^*, T_k) = a \cdot e_k] &\leq \frac{1}{4M\varepsilon^2} - \frac{1}{M} \\ &\leq 1/M\varepsilon^2, \end{aligned} \tag{2.82}$$

which concludes the proof of Theorem 1. ∎

We now turn our attention to bounding the failure probability of algorithm *ExtractB*. As discussed previously due to the loop considering all possible values of B, we need only consider the case where $b_j = a \cdot r_j$ for $j = 1, \ldots, m$. In this case, we note that the algorithm *ExtractB* calls the subroutine *Amplify* a total of n times with n different values of e_k but always with the same values of $r_1, \ldots, r_m$ and $b_1 \ldots b_m$. The probability that all of these calls return the wrong answer is upper-bounded by n times the probability that the k^{th} call returns the wrong answer. We thus bound the failure probability of algorithm *ExtractB* as

$$\Pr[ExtractB = a] \leq n/M\varepsilon^2. \tag{2.83}$$

In order to get a success probability of $\frac{1}{2}$, it is sufficient to set $M = \frac{2n}{\varepsilon^2}$. We noted in our discussion of the algorithm *ExtractB* that a total of nM IP queries and M EQ queries were made. Denoting the number of IP queries made as q_{IP} and the number of EQ queries q_{EQ}, we express the query complexity of algorithm *ExtractB* as

$$\begin{aligned} q_{IP} &= O\left(\frac{n^2}{\varepsilon^2}\right) \\ q_{EQ} &= O\left(\frac{n}{\varepsilon^2}\right). \end{aligned} \tag{2.84}$$

Equation 2.84 is an upper bound of the query complexity of G-L problem, which is what we have been after, but it is not a *tight* bound. In the following section, we

show that we can achieve a tighter bound by incorporating linear block codes into our algorithm.

2.1.4 Improvement to General Case

The idea of using linear block codes to improve the bounds given in the previous section is discussed in high-level terms by O. Goldreich in his 1999 book, *Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness* [21]. We will now provide a detailed discussion of how to improve these results by incorporating a linear block code into our algorithm. This block code will be used to correct errors and reduce the number of queries we require in order to obtain a satisfactory success probability.

Linear block codes are an important class of error correcting codes, which are widely used in the reliable transmission of information over noisy communication channels. In communications theory an (n, k, t) binary block code that can correct any combination of up to t errors, requires a total of n bits be transmitted. Of these, there are k data bits and $n-k$ redundant bits. The term linear refers to the formation of a vector space by linear combinations of basis vectors. Thus we can state that the set of all n-bit vectors over the *finite-field* $GF(2)$ of k linearly-independent basis vectors $g_1, g_2, \ldots, g_k$ is a binary (n, k) linear block code C. If the g_i are arranged as rows of a $k \times n$ generator Matrix $\mathbf{G}$, an n-bit codeword c can be expressed as

$$c = [i_1, i_2, \ldots, i_k] \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_k \end{bmatrix} = i\mathbf{G}. \tag{2.85}$$

where i is a k-bit information vector.

Most of the block codes that have proven to be useful in practical applications belong to a class of codes called *cyclic* codes. Cyclic codes are easy to encode and many efficient decoding schemes have been defined. An illustrative example of cyclic codes are the *Bose-Chaudhuri-Hocqeunghem codes*, usually referred to as BCH codes. These codes form an infinite class of (n, k, t) cyclic block codes that have capabilities for *multiple-error detection and correction*. In reference [26, page 121] it is shown that for any positive integers m and $t < n/2$, there exists a binary BCH code with block length $n = 2^m - 1$ having no more than mt redundant bits. Each such code can correct up to t errors per n-bit codeword and is thus said to be a *t-error-correcting* code.

Another class of linear block codes are the *Justesen codes*. These codes have provably good asymptotic properties [23]. Asymptotically good codes exhibit the property that for $d \geq 2t + 1$ defined as the minimum distance of the code, the ratio d/n remains nonzero as the block length n tends to infinity. We shall make use of this property to reduce the number of IP queries required. In order to avoid symbol clashing, we will refer to an (n, k, t) code as a $(cn, n, \alpha cn)$ with n the number of data bits corresponding to the length of the unknown string a as previously defined. For the purposes of this thesis, we give the following alternate definition of an asymptotically good code.

Definition 7 For an *asymptotically good code*, there exists positive constants c' and α' with the following property: For all n, there exists a $c \leq c'$ and an $\alpha \geq \alpha'$ such that there exists an $(cn, n, \alpha cn)$ code.

This follows from [23]. An important point to note is that the error correcting capability is proportional to the total number of bits transmitted. It is of interest to note that with our new definition of n, the BCH codes appear to exhibit this desirable "asymptotic" property with $c' = 4$ and $\alpha' = 1/16$ despite the fact that strictly speaking, the BCH codes are *asymptotically bad* [26, page 136]. We can deduce this desirable property of the BCH codes from studying tables of *primitive BCH codes* [26, pp. 122-123]. Since the BCH codes are of such practical importance, proving that Definition 7 is true for the BCH codes may be of some interest.

We now show how incorporating an $(cn, n, \alpha cn)$ code leads to a reduction in the number of IP queries made by algorithm *ExtractB*. We redefine the $cn \times n$ generator matrix $\mathbf{G}$ as

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1cn} \\ g_{21} & g_{22} & \cdots & g_{2cn} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{ncn} \end{bmatrix}. \tag{2.86}$$

We express a cn-bit codeword, y, in terms of an n-bit information vector x and the generator matrix $\mathbf{G}$ as

$$[y_1, y_2, \ldots, y_{cn}] = [x_1, x_2, \ldots, x_n]\mathbf{G}. \tag{2.87}$$

Note that we used the index k in Section 2.1.1 to identify the basis strings e_k. Here we change this index to κ to avoid confusion with the use of k in the literature as the number of data bits in a linear block code. Denoting the cn, n-bit column vectors of the matrix $\mathbf{G}$ as z_κ and using Equation 2.85, we modify algorithm *ExtractB*. We thus present the modified algorithm, *ExtractBI* in the box marked Algorithm 6.

1: *Initialize*(z_k, cn, M)
2: **for** $l = 1$ to M **do**
3: Let $B = b_1 \dots b_m$ be the binary representation of $l - 1$
4: **for** $\kappa = 1$ to cn **do**
5: *Amplify*(B, T_k)
6: **end for**
7: $y_{\text{received}} \longleftarrow y^{(1)} \dots y^{(cn)}$
8: $y \longleftarrow Decode\ \ y_{\text{received}}$
9: $x \longleftarrow y \cdot \mathbf{G}^{-1}$
10: **if** $EQ(x) = 1$ **then**
11: $a \longleftarrow x$
12: **end if**
13: **end for**
14: Return a

Algorithm 6: Algorithm *ExtractBI*

In the previous section, we showed that it was sufficient to we set $M = 2n/\varepsilon^2$. We will now show that instead of making M depend on the number of bits and the advantage, we can make it depend on the advantage only. We set

$$M = \frac{\beta}{\varepsilon^2} \text{ with } \beta > 2. \tag{2.88}$$

Defining q as the error probability given by Equation 2.81, we have

$$q \leq \frac{1}{M\varepsilon^2} = \frac{1}{\beta}. \tag{2.89}$$

We assume we have an *efficient*, binary-cyclic code that corrects any combination of $t = \alpha cn$ or less errors. We run algorithm *ExtractBI*, which for each value of M makes cn calls to subroutine *Amplify*. Each of the calls are again made with T_κ, but T is constructed in subroutine *Initialize* on input $z_\kappa = (g_{1\kappa}, g_{2\kappa}, \dots, g_{n\kappa})$ rather than e_κ. We use the z_κ because we wish to perform a bit-by-bit extraction of y_a, which is the codeword of the n-bit string a. The z_κ are the appropriate input strings since

each of the bits of y_a are

$$y_a^{(\kappa)} = g_{1\kappa} \oplus g_{2\kappa} \oplus \ldots \oplus g_{n\kappa}. \tag{2.90}$$

Thus each of the $y^{(\kappa)}$ generated by the subroutine *Amplify* are estimates of the bits of the cn-bit codeword y_a. Once we have extracted all cn bits, we remove errors from the *received* code word using an efficient *Decode* algorithm — see for example the *Kasami* decoding algorithm [26, page 157]. Finally we calculate a candidate solution string x using $\mathbf{G}^{-1}$ and offer the result to the EQ oracle. We now bound the success probability of this algorithm.

What is the chance that the offering to the EQ oracle is incorrect? It is the same as the chance that we get more than αcn errors in the cn-bits of the string y_{received}. In order to determine this probability, we first recognize that the value of each of these cn bits is a random variable with success probability $p = 1 - q$. The calls to the subroutine *Amplify* thus constitute a sequence of Bernoulli trials. We use tails of the Binomial distribution to bound the probability of having more than αcn errors. Defining the random variable Y to be the number of successes in cn trials, we use the bound on the left tail of the binomial distribution given in [10, page 122] as follows

$$\begin{aligned} Pr[Y < (cn - \alpha cn)] &= \sum_{i=0}^{cn-\alpha cn-1} \binom{cn}{i} p^i q^{cn-i} \\ &< \frac{(1-\alpha)cnq}{cnp - (1-\alpha)cn} \binom{cn}{(1-\alpha)cn} p^{(1-\alpha)cn} q^{\alpha cn} \\ &\leq \frac{(1-\alpha)q}{(\alpha - q)} \\ &\leq \frac{q}{(\alpha - q)} \end{aligned} \tag{2.91}$$

Setting this probability to at most a $\frac{1}{2}$, and using Equation 2.89 and Equation 2.91

we have

$$\beta < \frac{3}{\alpha}. \tag{2.92}$$

Combining the results of Equation 2.88 and Equation 2.92, we have $M < 3/\alpha\varepsilon^2$. Finally noting that the algorithm *ExtractBI* makes a total of nM IP queries and M EQ queries, we bound the query complexity as

$$\begin{aligned} q_{IP} &= O\left(\frac{n}{\varepsilon^2}\right) \\ q_{EQ} &= O\left(\frac{1}{\varepsilon^2}\right) . \end{aligned} \tag{2.93}$$

Equation 2.93 is an upper bound of the query complexity of G-L problem. In the following section, we will demonstrate that these are essentially *tight* bounds by giving lower bounds to q_{IP} and q_{EQ}.

2.2 Lower Bounding the Classical G-L Problem

In the previous section we developed a rather complex algorithm that solves the G-L problem with $O(n/\varepsilon^2)$ IP queries and $O(1/\varepsilon^2)$ EQ queries. But is the algorithm optimal? If we restrict ourselves to classical information, is there some clever scheme that will solve it with fewer steps? In this section we answer these questions. We first provide a lower bound on the number of EQ queries required in the case where we can perform an unlimited number of IP queries. We follow this with a section where we use classical information theory to determine the minimum number of IP queries required to solve the problem whenever the number of EQ queries is less than $2^{n/2}$. We conclude that the aforementioned upper bound is indeed a *tight bound*.

2.2.1 Lower Bounding the Number of *EQ* Queries

We begin by considering the case where we need to determine the minimum number of EQ queries required regardless of the number of IP queries performed. Here we can imagine that IP queries are cheap, and we can use them freely in order to limit the number of EQ queries required. We now propose a theorem.

Theorem 2 *For $n = 2k$ and $\varepsilon = \frac{1}{2 \cdot 2^k}$, the instance of the G-L problem requires $\Omega(1/\varepsilon^2)$ EQ queries, regardless of how many IP queries are performed.*

The proof is quite complex. In upper-bounding the G-L problem, we discussed the use of linear block codes. We now look to a specific code, the *Hadamard Code*, to represent the situation after we have applied 2^n noisy IP queries. For $n = 2k$, a single Hadamard code word is $h_k : \{0,1\}^n \rightarrow \{0,1\}^{2^n}$ defined as $h_k(a)_x = a \cdot x$, where $a, x \in \{0,1\}^n$. Thus x is an index into the 2^n components of $h_k(a)$. For illustrative purposes, we construct the table of codewords for the $n = 2$ case in Table 2.2. From

a	$h_1(a)$
00	0000
01	0101
10	0011
11	0110

Table 2.2: Hadamard Code Words for the 2-bit case

Table 2.2 we then form and label the matrix H_1 of Hadamard code words as

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \tag{2.94}$$

The rows and columns of H_k can thus be expressed as $H_k(a, x) = a \cdot x$ for $a, x \in \{0,1\}^{2k}$. Note that the definition of the 4×4 matrix H_1 given here is slightly different from the 4×4 matrix $H^{\otimes 2}$ in Equation 1.24. In fact what we have here is equivalent to taking the log to base -1 of each of the elements of the usual $H^{\otimes 2}$. In this section alone, we will use the symbol H to represent this 0-1 version of H. Each of the codewords in Table 2.2 has a *Hamming distance* $\Delta = 2$ to each of the other codewords. It is easily shown that for the m-dimensional Hamming code, the Hamming distance to all other code words is $m/2$. A received codeword having $< m/4$ errors will thus uniquely decode to the correct codeword. Since the Hadamard code is formed by the same inner product operation as IP queries and the G-L problem has noisy IP queries, we are motivated to explore the error correcting capability of the Hadamard code in our quest to lower bound the EQ queries.

In the G-L problem, we have noisy IP queries. Again for illustrative purposes, consider the $n = 2$ case with $\varepsilon = \frac{1}{4}$ — that is, one of our strings is bad. Here we have $1 = \frac{m}{4}$, so we see that we are just over the "edge" of the ability of the code to uniquely recover a code word. As it turns out, when this code is just passed its ability to uniquely correct, we see an interesting and self-similar structure that we will exploit in our proof of Theorem 2. To further understand this, suppose that some $a \in \{0,1\}^2$ is encoded as $h_1(a)$, which is then sent through a noisy channel in which one bit is flipped. Note that the process of generating $h_1(a)$ would involve making four IP queries, and we would not know which one of the queries returned a "bad" result. However, if the resulting string is $z_1 = [0\,0\,0\,1]$, then what can be deduced about what a was? We present the Table 2.3 to answer this question. In the bottom row we have the received vector $z_1 = [0\,0\,0\,1]$, and in the rightmost column

we have the vector of Hamming distances $\Delta_1 = [1\,1\,1\,3]$ between z_1 and the rows of H_1. Each component of Δ_1 is formed from z_1 and each row of H_1 as

$$\Delta_{1,a} = \sum_{x \in \{0,1\}^2} (z_{1,x} \oplus h_1(a)_x). \tag{2.95}$$

Clearly, one can deduce that $a \in \{00, 01, 10\}$, but nothing else. We note that we now

a	$h(a)$	Δ_1
00	0000	1
01	0101	1
10	0011	1
11	0110	3
z_1	0001	

Table 2.3: Hamming Distances, Δ_1, to Noisy Code Word $z_1 = [0001]$

only have to perform at most 3 EQ queries instead of the at most 4 EQ queries that would be required if we had not at first performed our 4 "free" IP queries. Since this list of possible a strings for the $n = 2$ and $\varepsilon = \frac{1}{4}$ case is smaller than the complete list, we are motivated to determine how this list size grows as a function of n and ε. As we increase n, we note that the larger Hadamard matrices are constructed in a self similar manner, and we also construct our codewords in a self-similar manner. To get a feel for how the list grows, we construct a similar analysis for the $n = 4$ case, where 6 bits of $h_2(a)$ are flipped. If the result is the codeword $z_2 = [0001\ 0001\ 0001\ 1110]$, then there will be 10 possibilities for a. We see this by looking at Table 2.4 where we note that there are 10 possible code words at Hamming distance 6 away and 6 possible code words at Hamming distance 10 away. To summarize what we see in Tables 2.3 and 2.4, we have a number of "close" code words and a number of "far" codewords. We would like to determine formulas for both the number and the actual Hamming distance of these two types of codewords as functions of ε.

a	$h(a)$				Δ_2
0000	0000	0000	0000	0000	6
0001	0101	0101	0101	0101	6
0010	0011	0011	0011	0011	6
0011	0110	0110	0110	0110	10
0100	0000	1111	0000	1111	6
0101	0101	1010	0101	1010	6
0110	0011	0011	1100	1100	6
0111	0110	0110	1001	1001	10
1000	0000	0000	1111	1111	6
1001	0101	0101	1010	1010	6
1010	0011	0011	1100	1100	6
1011	0110	0110	1001	1001	10
1100	0000	1111	1111	0000	10
1101	0101	1010	1010	0101	10
1110	0011	1100	1100	0011	10
1111	0110	1001	1001	0110	6
z_2	0001	0001	0001	1110	

Table 2.4: Hamming Distances, Δ_2, to Noisy Code Word $z_2 = [0001\ 0001\ 0001\ 1110]$

We begin by defining formulas for two key quantities, which we denote as r_k and s_k. As we construct several lemmas that will support the proof of Theorem 2, we will see the quantities r_k and s_k appearing in several roles.

Lemma 2.1: If r_k and s_k satisfy the recurrence

$$\begin{aligned} r_{k+1} &= 3r_k + s_k \\ s_{k+1} &= 3s_k + r_k \end{aligned} \tag{2.96}$$

with boundary conditions $r_0 = 1$ and $s_0 = 0$, then $r_k = (4^k + 2^k)/2$ and $s_k = (4^k - 2^k)/2$.

Proof: The proof is a simple induction over k. We begin by noting that the formulas for r_k and s_k are true for the base case where $k = 0$ since

$$\begin{aligned} r_0 &= (4^0 + 2^0)/2, \text{and} \\ s_0 &= (4^0 - 2^0)/2. \end{aligned}$$

We now assume that $r_k = (4^k + 2^k)/2$ and $s_k = (4^k - 2^k)/2$ are true for an arbitrary case where we set $k = k_0$ and then calculate the formulas for r_{k_0+1} and s_{k_0+1} . Thus we have

$$\begin{aligned} r_{k_0+1} &= 3r_{k_0} + s_{k_0} \\ &= 3(4^{k_0} + 2^{k_0})/2 + (4^{k_0} - 2^{k_0})/2 \\ &= (4^{k_0+1} + 2^{k_0+1})/2. \end{aligned} \tag{2.97}$$

Similarly we have,

$$\begin{aligned} s_{k_0+1} &= 3s_{k_0} + r_{k_0} \\ &= 3(4^{k_0} - 2^{k_0})/2 + (4^{k_0} + 2^{k_0})/2 \\ &= (4^{k_0+1} - 2^{k_0+1})/2, \end{aligned} \tag{2.98}$$

which completes the proof. ∎

For reference we have written the first several terms of s_k and r_k in Table 2.5. As an

k	s_k	r_k	4^k
0	0	1	1
1	1	3	4
2	6	10	16
3	28	36	64

Table 2.5: The first few values of $r_k = (4^k + 2^k)/2$ and $s_k = (4^k - 2^k)/2$.

aside, it may be of interest to some readers to note that an alternate proof of Lemma 2.1 could be given by expressing the conditions of the Lemma as the following matrix equation

$$\begin{aligned} \begin{bmatrix} s_{k+1} \\ r_{k+1} \end{bmatrix} &= \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} s_k \\ r_k \end{bmatrix} \\ &= \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^{k+1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{aligned} \tag{2.99}$$

The square matrix in Equation 2.99 can be diagonalized as

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{2.100}$$

Finally solving for s_k and r_k, we have

$$\begin{bmatrix} s_k \\ r_k \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^k \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{(4^k - 2^k)}{2} \\ \frac{(4^k + 2^k)}{2} \end{bmatrix}. \tag{2.101}$$

In Tables 2.3 and 2.4, we introduced the noisy code words z_1 and z_2. We now wish to define a recurrence formula for z_k and to establish that the quantities r_k and s_k represent the numbers of zeroes and ones respectively in the codeword z_k.

Lemma 2.2: If the noisy codeword $z_k \in \{0,1\}^{4^k}$ satisfies the recurrence

$$z_{k+1} = z_k z_k z_k \overline{z}_k, \tag{2.102}$$

with boundary condition $z_0 = 0$, then there are $r_k = (4^k + 2^k)/2$ zeroes in z_k and $s_k = (4^k - 2^k)/2$ ones .

Proof: For the base case $k = 0$ it is clear since for $z_0 = 0$, we have $r_0 = 1$ and $s_0 = 0$. We assume that the statement of the lemma holds for a particular k, then

by the recurrence given in Equation 2.102, z_{k+1} consists of three copies of z_k each of which contains r_k zeroes and s_k ones and one copy of $\overline{z}_k$ which contains s_k zeroes and r_k ones. Thus we see that the numbers of zeroes and the numbers of ones satisfy the same recursion formulas that appear in Lemma 2.1. By Lemma 2.1, we conclude that there are $r_k = (4^k + 2^k)/2$ zeroes in z_k and $s_k = (4^k - 2^k)/2$ ones in z_k as required. ∎

The numbers r_k and s_k will now be shown to be relevant both to the number of potential solutions that are "close" to Hadamard code words and to the actual Hamming distance of these potential solutions. With reference to Table 2.5, it is interesting to note that s_1, r_1 and s_2, r_2 appear as both the "close" and the "far" Hamming distances, in the vectors Δ_1 and Δ_2 in Tables 2.3 and 2.4. The numbers also appear as the *quantities* of these terms in a complementary manner. We will exploit these facts in forming the central Lemma that we will use in our proof of Theorem 2.

Lemma 2.3: There are $(4^k+2^k)/2$ possible values of $h(a) \in \{0,1\}^{4^k}$ with Hamming distance $(4^k - 2^k)/2$ from z_k and $(4^k - 2^k)/2$ possible values of $h(a)$ with Hamming distance $(4^k + 2^k)/2$ from z_k.

Proof: We note the interesting relationship between Hamming distances, Δ_1 and Δ_2 and the code words z_1 and z_2 in Tables 2.3 and 2.4 respectively. We see that Δ_1 can be formed by replacing each zero in z_1 by s_1 and each one in z_1 by r_1. We also see that Δ_2 is formed similarly by replacing each zero in z_2 by s_2 and each one in z_2 by r_2. We thus define Δ_k^j as z_k with each zero replaced by s_j and each one replaced by r_j. Its complement $\overline{\Delta}_k^j$ is defined as z_k with each zero replaced by r_j and each one replaced by s_j. For example with this definition, we can write $\Delta_1^1 = [1\,1\,1\,3]$,

$\Delta_1^2 = [6\,6\,6\,10]$ and $\overline{\Delta}_2^2 = [10\,10\,10\,6\ 10\,10\,10\,6\ 10\,10\,10\,6\ 6\,6\,6\,10]$. We next define the quantity $\Gamma_k(w, M)$, which is a 4^k-element vector produced from the arguments w, a string of length 4^k, and M a $4^k \times 4^k$ matrix. The entry in component x of this vector is the Hamming distance between w and row x of M. Note that

$$\Gamma_k(z_k, H_k) = \left[\sum_{j=1}^{4^k} z_{k,j} \oplus H_k(1,j), \sum_{j=1}^{4^k} z_{k,j} \oplus H_k(2,j), \ldots, \sum_{j=1}^{4^k} z_{k,j} \oplus H_k(4^k,j) \right], \tag{2.103}$$

where $\Gamma_k(z_k, H_k)$ is a 4^k vector consisting of the 4^k Hamming distances between z_k and each of the 4^k rows of H_k.

Our strategy for proving Lemma 2.3 is as follows. We know that by the proof of Lemma 2.2 and by the definition of Δ_k^k, it consists of r_k entries having value s_k and s_k entries having value r_k. It is sufficient to show that these entries are Hamming distances. Thus, establishing that the equality

$$\Gamma_k(z_k, H_k) = \Delta_k^k \tag{2.104}$$

holds is sufficient to complete the proof. We prove this equality by induction over k. For the base case, we specifically choose $k = 1$ rather that $k = 0$ in order to explicitly write out a sample calculation of Equation 2.103. Thus using Equation

2.103 and our definitions of z_1 and H_1 we note that

$$\begin{aligned}\Gamma_1(z_1, H_1) &= \begin{bmatrix} 0 \oplus H_1(1,1) + 0 \oplus H_1(1,2) + 0 \oplus H_1(1,3) + 1 \oplus H_1(1,4) \\ 0 \oplus H_1(2,1) + 0 \oplus H_1(2,2) + 0 \oplus H_1(2,3) + 1 \oplus H_1(2,4) \\ 0 \oplus H_1(3,1) + 0 \oplus H_1(3,2) + 0 \oplus H_1(3,3) + 1 \oplus H_1(3,4) \\ 0 \oplus H_1(4,1) + 0 \oplus H_1(4,2) + 0 \oplus H_1(4,3) + 1 \oplus H_1(4,4) \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_1 \\ s_1 \\ r_1 \end{bmatrix} = \Delta_1^1. \end{aligned} \tag{2.105}$$

Similarly, we have

$$\Gamma_1(\overline{z}_1, H_1) = \Gamma_1(z_1, \overline{H}_1) = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_1 \\ r_1 \\ s_1 \end{bmatrix} = \overline{\Delta}_1^1. \tag{2.106}$$

The need for this latter equation will become apparent during the inductive step.

It is a little more involved to prove the general case, which is proven by induction on k_0. We first assume that $\Gamma_k(z_k, H_k) = \Delta_k^k$ for all k up to k_0. In the inductive step, we show that $\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1}) = \Delta_{k_0+1}^{k_0+1}$. Before proceeding to the inductive step, we need to make some further definitions. We have already defined z_{k+1} in Equation 2.102, we now give a recursive expression for H_{k+1} in terms of H_k in Equation 2.107. We note that H_{k+1} has four rows each containing either four copies of H_k (the first row), or two copies of H_k and $\overline{H}_k$, in different column positions of the other three

rows.

$$H_{k+1} = \begin{bmatrix} H_k & H_k & H_k & H_k \\ H_k & \overline{H}_k & H_k & \overline{H}_k \\ H_k & H_k & \overline{H}_k & \overline{H}_k \\ H_k & \overline{H}_k & \overline{H}_k & H_k \end{bmatrix}. \tag{2.107}$$

Since z_{k+1} also has four elements, formed from three copies of z_k and one copy of $\overline{z}_k$, we see that there are four separate Hamming distance vectors that we have to account for. By inspection of Equation 2.103, we can reduce these four cases to the following two cases:

$$\begin{aligned} \Gamma_k(z_k, H_k) &= \Gamma_k(\overline{z}_k, \overline{H}_k) \\ \overline{\Gamma}_k(z_k, H_k) &= \Gamma_k(\overline{z}_k, H_k) = \Gamma_k(z_k, \overline{H}_k). \end{aligned} \tag{2.108}$$

We pictorially represent the construction of all four cases in Figure 2.12.

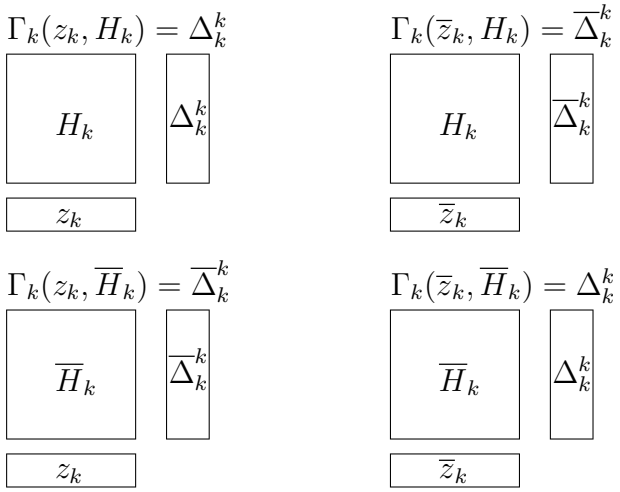

Figure 2.12: The columns Δ_k^k and $\overline{\Delta}_k^k$ represent vectors of Hamming distances between the noisy codeword z_k, its complement $\overline{z}_k$ and the matrix of Hadamard codewords H_k and its complement $\overline{H}_k$.

We are now ready to make the inductive step, which will actually require *two* steps — one for the *Hamming distances* and one for the *quantities* of these distances. We assume that $\Gamma_{k_0}(z_{k_0}, H_{k_0}) = \Delta_{k_0}^{k_0}$ *and* $\overline{\Gamma}_{k_0}(z_{k_0}, H_{k_0}) = \overline{\Delta}_{k_0}^{k_0}$ are true for some k_0. Note that we have shown both of these to be true for the base case in Equations 2.105 and 2.106. We now begin the *first* part of the inductive step. Using Equation 2.108 and the definitions of z_{k+1} and H_{k+1}, we write an expression for the first 4^{k_0} entries of $\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1})$, which is

$$3\Gamma_{k_0}(z_{k_0}, H_{k_0}) + \overline{\Gamma}_k(z_{k_0}, H_{k_0}) = 3\Delta_{k_0}^{k_0} + \overline{\Delta}_{k_0}^{k_0} = \Delta_{k_0}^{k_0+1}. \tag{2.109}$$

This equation has been formed from the four constituents of z_{k+1} and the four con-

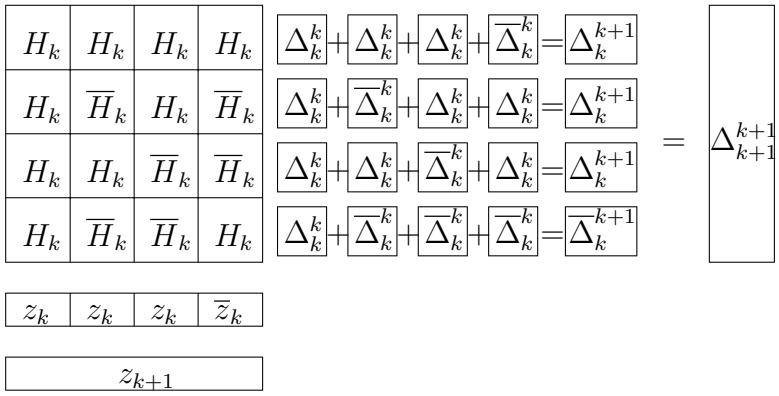

Figure 2.13: The column Δ_{k+1}^{k+1} represents the vector of Hamming distances between the noisy codeword z_{k+1} and the matrix of Hadamard codewords H_k. It is formed by summing and concatenating the columns Δ_k^k and $\overline{\Delta}_k^k$ formed in Figure 2.12.

stituents of the *first* row of H_{k+1} as depicted in Figure 2.13. The last equality of Equation 2.109 is true since each of the *Hamming distances* r_{k_0} and s_{k_0}, which are the constituents of $\Delta_{k_0}^{k_0}$ and $\overline{\Delta}_{k_0}^{k_0}$, are combined here in a manner consistent with the

recursion given in Equation 2.96. This results in the Hamming distances r_{k_0+1} and s_{k_0+1} consistent with our definition of $\Delta_{k_0}^{k_0+1}$. With reference to Figure 2.13 we see that the elements of Equation 2.109 also form the second 4^{k_0} entries and the third 4^{k_0} entries of $\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1})$. The final 4^{k_0} entries of $\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1})$ are

$$\Gamma_{k_0}(z_{k_0}, H_{k_0}) + 3\overline{\Gamma}_k(z_{k_0}, H_{k_0}) = \Delta_{k_0}^{k_0} + 3\overline{\Delta}_{k_0}^{k_0} = \overline{\Delta}_{k_0}^{k_0+1}. \tag{2.110}$$

We can now take the *second* part of the inductive step, where we want to establish the quantities of Hamming distances. In correspondence with the four rows of H_{k_0+1}, the first, second and third 4^{k_0} entries of $\Delta_{k_0+1}^{k_0+1}$ are as per Equation 2.109 and the fourth 4^{k_0} entries are as per Equation 2.110. We can thus write the expression

$$\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1}) = \begin{bmatrix} 3\Gamma_{k_0}(z_{k_0}, H_{k_0}) + \overline{\Gamma}_k(z_{k_0}, H_{k_0}) \\ 3\Gamma_{k_0}(z_{k_0}, H_{k_0}) + \overline{\Gamma}_k(z_{k_0}, H_{k_0}) \\ 3\Gamma_{k_0}(z_{k_0}, H_{k_0}) + \overline{\Gamma}_k(z_{k_0}, H_{k_0}) \\ \Gamma_{k_0}(z_{k_0}, H_{k_0}) + 3\overline{\Gamma}_k(z_{k_0}, H_{k_0}) \end{bmatrix}. \tag{2.111}$$

Since the *quantities* of s_{k_0+1} and r_{k_0+1} in $\Delta_{k_0}^{k_0+1}$ and $\overline{\Delta}_{k_0}^{k_0+1}$ are now combined in a manner consistent with the recursion given in Equation 2.102, we can express Equation 2.111 as

$$\Gamma_{k_0+1}(z_{k_0+1}, H_{k_0+1}) = \begin{bmatrix} \Delta_{k_0}^{k_0+1} \\ \Delta_{k_0}^{k_0+1} \\ \Delta_{k_0}^{k_0+1} \\ \overline{\Delta}_{k_0}^{k_0+1} \end{bmatrix} = \Delta_{k_0+1}^{k_0+1}, \tag{2.112}$$

thus completing the inductive proof of Equation 2.104.

We now use the definition of the vector Δ_k^k to complete the proof of Lemma 2.3. We have already established that there are r_k zeroes and s_k ones in z_k in our proof

of Lemma 2.2. In our definition of Δ_k^k, we replaced all the zeroes of z_k with s_k and all the ones of z_k with r_k resulting in a vector consisting of r_k values of Hamming distance s_k and s_k values of Hamming distance r_k. Thus there are $(4^k+2^k)/2$ possible values of $h(a) \in \{0,1\}^{4^k}$ with Hamming distance $(4^k-2^k)/2$ from z_k and $(4^k-2^k)/2$ possible values of $h(a)$ with Hamming distance $(4^k+2^k)/2$ from z_k as required. ∎

We are now ready to use the results of the proofs of Lemmas 2.1, 2.2 and 2.3 to prove Theorem 2.

Proof of Theorem 2: We define the IP oracle in terms of the noisy codeword z_k as $IP(x) = z_{k,x}$, where $z_{k,x}$ denotes the bit of z_k in component x. We want to establish that $IP(x) = z_{k,x}$ is correct on $(\frac{1}{2}+\varepsilon_k)\cdot 4^k$ inputs and incorrect on $(\frac{1}{2}-\varepsilon_k)\cdot 4^k$ inputs. We show this by noting that $IP(x)$ is correct for the "close" Hamming distances s_k. Thus we have

$$\begin{aligned} s_k &= (4^k - 2^k)/2 \\ &= \tfrac{1}{2} - \tfrac{1}{2\cdot 2^k} \cdot 4^k \\ &= (\tfrac{1}{2} - \varepsilon_k)\cdot 4^k, \end{aligned} \tag{2.113}$$

where we have used $\varepsilon_k = 1/(2\cdot 2^k)$ to establish this equality. We have already shown in the proof of Lemma 2.3 that the *quantity* of vectors having Hamming distance s_k is r_k. Since $s_k + r_k = 4^k$, we conclude that $IP(x)$ is *correct* on $r_k = (\frac{1}{2}+\varepsilon_k)\cdot 4^k$ inputs as required. Similarly we conclude that $IP(x)$ is *incorrect* on $s_k = (\frac{1}{2}-\varepsilon_k)\cdot 4^k$ inputs.

The quantity of vectors r_k having Hamming distance s_k constitutes the size of the list we have to search using EQ queries. We thus express r_k in terms of ε_k as

$$\begin{aligned} r_k &= (\tfrac{1}{2} + \varepsilon_k) \cdot 4^k \\ &= (\tfrac{1}{2} + \varepsilon_k) \cdot \tfrac{1}{4\varepsilon_k^2} \\ &= \frac{1}{(8\varepsilon_k^2)} + \frac{1}{(4\varepsilon_k)}. \end{aligned} \tag{2.114}$$

This means that, for $n = 2k$ and $\varepsilon_k = 1/(2 \cdot 2^k)$, if a is encoded into $h(a)$ and if there are fraction of $(\frac{1}{2} - \varepsilon_k)$ errors, which result in the string z_k, then there will be a number of values of $a \in \{0,1\}^{2k}$ that, when encoded as $h(a) \in \{0,1\}^{4^k}$, are consistent with z_k. The list of consistent values of a is of size $\frac{1}{(8\varepsilon_k^2)} + \frac{1}{(4\varepsilon_k)}$, which is $\Omega(1/\varepsilon^2)$ as required. ∎

The preceding proof settles the case where $n = 2k$ and $\varepsilon = 1/(2 \cdot 2^k)$ (which means that n and ε satisfy the relationship $\varepsilon = 1/(2 \cdot 2^{n/2})$). We claim that as long as $\varepsilon \geq 1/(2 \cdot 2^{n/2})$, the $\Omega(1/\varepsilon^2)$ EQ query lower bound holds. A rough sketch of the claim follows. We define a slightly modified noisy code word $z_{k,1}$, where in the last step of our recursive definition of z_k we substitute $z_{k+1} = z_k z_k z_k z_k$ for $z_{k+1} = z_k z_k z_k \overline{z}_k$. Given this substitution, we obtain the quantity of vectors $r_{k,1} = (4^{k-1} + 2^{k-1})/2$ having "close" Hamming distance $s_{k,1} = (4^{k-1} - 2^{k-1})/2$ to $z_{k,1}$ for a slightly larger $\varepsilon_{k,1} = 2/(2 \cdot 2^k)$. We note that $r_{k,1}$ is still of size $\Omega(1/\varepsilon_{k,1}^2)$. If we define $z_{k,m}$ as the noisy codeword where we continue this replacement of $z_{k+1} = z_k z_k z_k z_k$ for the last m steps of our construction (and still use Equation 2.102 for the first $k - m$ steps) , our list of "close" Hamming distance vectors grows like

$$r_{k,m} = (4^{k-m} + 2^{k-m})/2, \tag{2.115}$$

while the advantage grows like

$$\varepsilon_{k,m} = 2^m/(2 \cdot 2^k) = 1/(2 \cdot 2^{k-m}). \tag{2.116}$$

Inserting this expression into Equation 2.115, we have

$$r_{k,m} = \frac{1}{4 \cdot \varepsilon_{k,m}^2} + \frac{1}{2 \cdot \varepsilon_{k,m}} = \Omega \ \ \varepsilon_{k,m}^2 \ \ , \tag{2.117}$$

which completes the sketch. We will next use an information theoretical approach to lower bound the number of IP queries.

2.2.2 Lower Bounding the Number of *IP* Queries

We show that any classical algorithm solving the G-L problem with constant probability must make $\Omega(n/\varepsilon^2)$ IP queries (for a reasonable range of values of ε). This proof was first presented in [1].

Theorem 3 *Any classical probabilistic algorithm solving the G-L problem with success probability $\delta > 0$ requires either more than $2^{n/2}$ EQ queries or $\Omega(\delta n/\varepsilon^2)$ IP queries when $\varepsilon \geq \sqrt{n}2^{-n/3}$.*

Proof: The proof uses classical information theory, bounding the conditional mutual information about an unknown string that is revealed by each IP query, in conjunction with an analysis of the effect of EQ queries.

It is useful to consider an algorithm to be *successful* on a particular input if and only if it performs an EQ query whose output is 1 (at which point the value of a has been determined).

We begin by showing that it is sufficient to consider algorithms (formally, decision trees) that are in a convenient simple form. First, by a basic game-theoretic

argument [35], it suffices to consider deterministic algorithms, where their input data—embodied in the black-boxes for *IP* and *EQ* queries—may be generated in a probabilistic manner. Second, it can be assumed that all *EQ* queries occur only after all *IP* queries have been completed. To see why this is so, start with an algorithm that interleaves *IP* and *EQ* queries, and modify it as follows. Whenever an *EQ* query occurs before the end of the *IP* queries, the modified algorithm stores the value of the input to the query and proceeds as if the result were 0. Then, at the end of the *IP* queries, each such deferred *EQ* query is applied. The modified algorithm will behave consistently whenever the actual output of a deferred *EQ* query is 0, and also it will perform (albeit later) any *EQ* query where the output is 1. Henceforth, we consider only algorithms with the above simplifications.

Now we describe a probabilistic procedure for constructing the black boxes that perform *IP* and *EQ* queries. First, $a \in \{0,1\}^n$ is chosen randomly according to the uniform distribution. Then a set $S \subseteq \{0,1\}^n$ is chosen randomly, uniformly subject to the condition that $|S| = (\frac{1}{2} + \varepsilon)2^n$ (assuming that $\varepsilon 2^n$ is an integer). Then

$$IP(x) = \begin{cases} a \cdot x & \text{if } x \in S \\ \overline{a \cdot x} & \text{if } x \in S \end{cases} \tag{2.118}$$

and

$$EQ(x) = \begin{cases} 1 & \text{if } x = a \\ 0 & \text{if } x = a. \end{cases} \tag{2.119}$$

Consider an algorithm that makes m *IP* queries. If $m \geq \delta n/\varepsilon^2$ then the theorem is proven. Otherwise, since $\varepsilon \geq \sqrt{n}2^{-n/3}$, we have

$$m \;<\; \frac{\delta n}{\varepsilon^2} \leq \delta 2^{2n/3}. \tag{2.120}$$

We proceed by determining the amount of information about a that is conveyed by the application of m *IP* queries. Let A be the $\{0,1\}^n$-valued random variable corresponding to the probabilistic choice of $a \in \{0,1\}^n$, and let $Y_1, Y_2, \ldots, Y_m$ be the $\{0,1\}$-valued random variables corresponding to the respective outputs of the m *IP* queries. Let H be the Shannon entropy function (see, e.g., [13]). Then, for each $i \in \{1, 2, \ldots, m\}$,

$$H(A|Y_1, Y_2, \ldots, Y_i) = H(A|Y_1, \ldots, Y_{i-1}) - H(Y_i|Y_1, \ldots, Y_{i-1}) + H(Y_i|A, Y_1, \ldots, Y_{i-1}). \tag{2.121}$$

Combining the above equations yields

$$H(A|Y_1, Y_2, \ldots, Y_m) = H(A) + \sum_{i=1}^{m} \left(H(Y_i|A, Y_1, \ldots, Y_{i-1}) - H(Y_i|Y_1, \ldots, Y_{i-1}) \right). \tag{2.122}$$

(See Equation 1.11 and Equation 1.12 for a detailed derivation of this expression.) We shall now bound each term on the right side of Eq. 2.122. Since the *a priori* distribution of A is uniform, $H(A) = n$. Also, since the entropy of a single bit is at most 1, $H(Y_i|Y_1, \ldots, Y_{i-1}) \leq 1$ for all $i \in \{1, 2, \ldots, m\}$. Next, we show that, for all $i \in \{1, 2, \ldots, m\}$,

$$H(Y_i|A, Y_1, \ldots, Y_{i-1}) \;\geq\; 1 - (16/\ln 2)\varepsilon^2. \tag{2.123}$$

To establish Eq. 2.123, it is useful to view the set S as being generated during the execution of the *IP* queries as follows. Initially S is empty, and when the first *IP* query is performed on some input x, x is placed in S with probability $\frac{1}{2} + \varepsilon$ and in $\overline{S}$ with probability $\frac{1}{2} - \varepsilon$. The inputs to subsequent *IP* queries are also placed in either S or $\overline{S}$ with an appropriate probability, which depends on how the inputs to

previous queries are balanced between S and $\overline{S}$. After the execution of the first $i-1$ queries, the input to the i^{th} query is placed in S with probability

$$\frac{(\frac{1}{2}+\varepsilon)2^n - j}{2^n - (i-1)}, \tag{2.124}$$

where $j \in \{0, 1, \ldots, i-1\}$ is the number of previous inputs to queries that have been placed in S. Using Eq. 2.120, the above probability can be shown to lie between $\frac{1}{2} - 2\varepsilon$ and $\frac{1}{2} + 2\varepsilon$. It follows that

$$\begin{aligned} H(Y_i|A, Y_1, \ldots, Y_{i-1}) &\geq H(\tfrac{1}{2}+2\varepsilon, \tfrac{1}{2}-2\varepsilon) \\ &= -(\tfrac{1}{2}+2\varepsilon)\log(\tfrac{1}{2}+2\varepsilon) - (\tfrac{1}{2}-2\varepsilon)\log(\tfrac{1}{2}-2\varepsilon) \\ &\geq 1 - (16/\ln 2)\varepsilon^2, \end{aligned} \tag{2.125}$$

where we have used Equation 1.3 in the second inequality. This establishes Eq. 2.123. Now, substituting these inequalities into Eq. 2.122, we obtain

$$H(A|Y_1, \ldots, Y_m) \geq n - (16/\ln 2)m\varepsilon^2. \tag{2.126}$$

Intuitively, the *IP* queries yield information about the value of A in terms of their effect on the probability distribution of A conditioned on the values of $Y_1, \ldots, Y_m$. Eq. 2.126 lower-bounds the decrease in entropy that is possible.

From the conditions of the theorem, it can be assumed that, after the *IP* queries, $2^{n/2}$ *EQ* queries are performed. The algorithm succeeds with probability at least δ only if there exist $2^{n/2}$ elements of $\{0,1\}^n$ whose total probability (conditioned on $Y_1, \ldots, Y_m$) is at least δ. In Equation 1.15, we established that the maximum entropy that a distribution with this property can have is for a bi-level distribution, where $2^{n/2}$ elements of $\{0,1\}^n$ each have probability $\delta/2^{n/2}$ and $2^n - 2^{n/2}$ elements each

have probability $(1-\delta)/(2^n - 2^{n/2})$. Therefore,

$$\begin{aligned} H(A|Y_1,\ldots,Y_m) &\leq H\ \underbrace{\tfrac{\delta}{2^{n/2}},\ldots,\tfrac{\delta}{2^{n/2}}}_{2^{n/2}},\underbrace{\tfrac{1-\delta}{2^n-2^{n/2}},\ldots,\tfrac{1-\delta}{2^n-2^{n/2}}}_{2^n-2^{n/2}} \\ &= H(\delta,1-\delta)+\delta\log(2^{n/2})+(1-\delta)\log(2^n-2^{n/2}) \\ &< 1+\delta n/2+(1-\delta)n \\ &= n-\delta n/2+1. \end{aligned} \tag{2.127}$$

Combining Eq. 2.126 with Eq. 2.127, yields $m > (\ln 2)(\delta n - 2)/(32\varepsilon^2) \in \Omega(\delta n/\varepsilon^2)$, as required. ∎

In this chapter, we have established that the classical upper bound to the G-L problem requires $O\left(\frac{n}{\varepsilon^2}\right)$ IP queries and $O\left(\frac{1}{\varepsilon^2}\right)$ EQ queries. We have also established that any classical algorithm requires $\Omega\left(\frac{1}{\varepsilon^2}\right)$ EQ queries and $\Omega\left(\frac{n}{\varepsilon^2}\right)$ IP queries (under reasonable conditions of various parameters). We conclude that our bounds are indeed tight and that we are justified in stating that the total number of IP and EQ queries required to solve the classical G-L problem are

$$\begin{aligned} q_{IP} &= \Theta\left(n/\varepsilon^2\right), \\ q_{EQ} &= \Theta\left(1/\varepsilon^2\right). \end{aligned} \tag{2.128}$$

In the next chapter, we study and bound the query complexity of the quantum G-L problem.

Chapter 3 A Quantum Goldreich-Levin Theorem

3.0 Introduction

The classical G-L problem is based upon an inner product oracle. The Bernstein-Vazirani circuit also solves an inner product query problem although its discovery was unrelated to the analysis of hard predicates. In this chapter, we first analyze the query complexity of the Bernstein-Vazirani circuit. We compare the results with that of the noiseless classical G-L problem. We follow this with a detailed analysis of the same circuit but with the introduction of noise and show how this is readily interpreted as an upper bound of the general quantum G-L problem. The chapter concludes with a proof of the lower bound of query complexity of the quantum G-L problem.

3.1 The Bernstein-Vazirani Problem

The circuit presented by Bernstein-Vazirani in their 1993 paper [4] is an elegant example of a quantum circuit whose query complexity represents a polynomial speed up over its classical counterpart. An $n+1$ qubit state is presented to the input of the circuit. At the output of the circuit, the first n qubits are in the state $|a\rangle$. Measuring this state constitutes a single IP query and produces the bits of the unknown string a. The Bernstein-Vazirani circuit is presented in Figure 3.14. We will now demonstrate that the bits of a are recoverable in a single query by proceeding with a step-by-step analysis of the circuit using the numbered vertical lines in Figure 3.14 as reference.

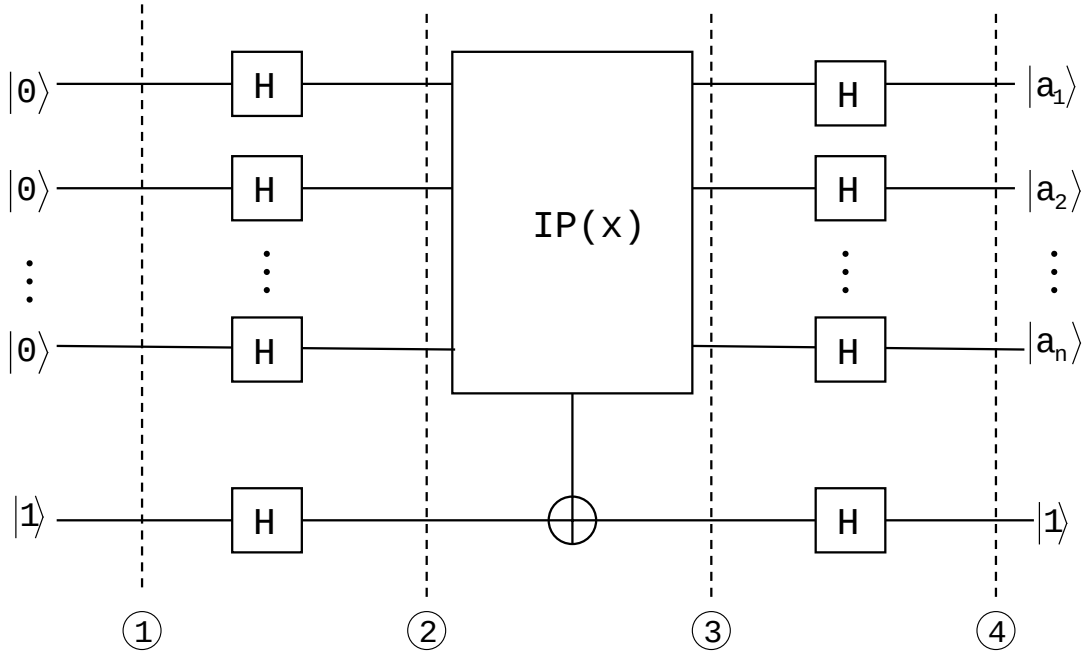

Figure 3.14: The Bernstein-Vazirani Circuit

1. The input state is the $n+1$ qubit basis state

$$|\Psi_1\rangle = |00...0\rangle|1\rangle.$$

2. After the application of the $n+1$ Hadamard transformations, we obtain an equal superposition of all the basis states

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}} \left(\sum_{x\in\{0,1\}^n} |x\rangle \right) \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right).$$

We have used $H^{\otimes n}|z\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{z\cdot x}|x\rangle$ and that the input state is $|z\rangle = |00\ldots0\rangle$ to express this result.

3. With $IP(x) = a \cdot x$, we modulo-two sum the result on to the bottom qubit to achieve the following result

$$|\Psi_3\rangle = \frac{1}{\sqrt{2^n}} \left(\sum_{x \in \{0,1\}^n} (-1)^{a \cdot x} |x\rangle \right) \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle \right).$$

This results from what is sometimes referred to as *phase kickback*. To appreciate this interesting effect, we note that $a \cdot x$ can have the value 0 or 1. For $a \cdot x = 1$, we have $|0 \oplus 1\rangle - |1 \oplus 1\rangle = (-1)^1(|0\rangle - |1\rangle)$, and for $a \cdot x = 0$, we have $|0 \oplus 0\rangle - |1 \oplus 0\rangle = (-1)^0(|0\rangle - |1\rangle)$. Since phase can be picked up for each of the 2^n values of x, we bring the phase factor $(-1)^{a \cdot x}$ into the sum.

4. Writing down the effect of the application of the final $n + 1$ Hadamard transformations is a little more involved. We begin by writing down

$$\begin{aligned}
|\Psi_4\rangle &= H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{a \cdot x} |x\rangle \right) |1\rangle \\
&= \frac{1}{2^n} \left(\sum_{x \in \{0,1\}^n} (-1)^{a \cdot x} \sum_{w \in \{0,1\}^n} (-1)^{x \cdot w} |w\rangle \right) |1\rangle \\
&= \left(\sum_{w \in \{0,1\}^n} \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (a \oplus w)} \right) |w\rangle \right) |1\rangle \\
&= |a\rangle |1\rangle.
\end{aligned}$$

The last step is achieved by noting that

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (a \oplus w)} = \begin{cases} 1 & \text{if } a = w \\ 0 & \text{if } a = w \end{cases},$$

since $a \oplus w = 0$ if and only if $a = w$.

We conclude that measuring the first n qubits of the B-V circuit gives us the n bits of the unknown string a with only one query. This compares with n queries in the *noiseless* classical case where the advantage, $\varepsilon = \frac{1}{2}$. This polynomial speed up of the noiseless quantum case over the noiseless classical case motivates us to study how the B-V circuit can be adapted to simulate noisy quantum IP queries. We will now present an upper-bound to the quantum G-L problem.

3.2 Upper-Bounding the Quantum Goldreich-Levin Theorem

In this section we adapt the B-V circuit to perform a *noisy* quantum inner product query. Here we think of "noisy" as pertaining to our definition of a unitary transformation that implements an imperfect quantum inner product theory. This is a different usage of the term "noisy" than the common usage, which pertains to quantum bit flips or phase flips to which error-correcting codes may be applied [27, Chapter 10]. In this section, we first provide definitions of *IP* and *EQ* queries in the quantum case in terms of unitary operations. We do this in a manner that is sufficiently general so that, whenever an *implementation* of a more general *IP* or *EQ* query is given as a general quantum circuit consisting of elementary quantum gates and measurements, a unitary query corresponding to our definition can be efficiently constructed from it.

Definition 8 A *quantum inner product* query (with bias ε) is a unitary transformation U_{IP} on $n+m$ qubits, or its inverse $U_{IP}^{\dagger}$, such that U_{IP} satisfies the following two properties:

1. If $x \in \{0,1\}^n$ is chosen randomly according to the uniform distribution and the last qubit of $U_{IP}|x\rangle|0^m\rangle$ is measured, yielding the value $w \in \{0,1\}$, then $\Pr[w = a \cdot x] \geq \frac{1}{2} + \varepsilon$.

2. For any $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, the state of the first n qubits of $U_{IP}|x\rangle|y\rangle$ is $|x\rangle$.

The first property captures the fact that, taking a query to be a suitable application of U_{IP} followed by a measurement of the last qubit, Eq. 2.51 is satisfied. Any implementation of a quantum circuit that produces an output that is $a \cdot x$ with probability on average $\frac{1}{2} + \varepsilon$ can be modified to consist of a unitary stage U_{IP} followed by a measurement of one qubit. The second property is for technical convenience, and any unitary operation without this property can be converted to one that has this property, by first producing a copy of the classical basis state $|x\rangle$. Moreover, given a circuit implementing U_{IP}, it is easy to construct a circuit implementing $U_{IP}^{\dagger}$.

Definition 9 A *quantum equivalence query* is the unitary operation U_{EQ} such that, for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$,

$$U_{EQ}|x\rangle|b\rangle = \begin{cases} |x\rangle|\overline{b}\rangle & \text{if } x = a \\ |x\rangle|b\rangle & \text{if } x = a, \end{cases} \tag{3.129}$$

where $\overline{b} = \neg b$.

For the quantum G-L problem, $a \in \{0,1\}^n$ and information about a is available only from quantum *IP* and *EQ* queries and the goal is to determine a. We can now state and prove the result about quantum algorithms for the G-L problem (which is similar to a result in [12] in a different context).

Theorem 4 *There exists a quantum algorithm solving the G-L problem with constant probability using $O(1/\varepsilon)$ U_{IP}, $U_{IP}^{\dagger}$ and U_{EQ} queries in total. Also, the number of auxiliary qubit operations used by the procedure is $O(n/\varepsilon)$.*

The proof is by a combination of two techniques: the algorithm in [4] for the exact case (i.e., when $\varepsilon = \frac{1}{2}$), which is shown to be adaptable to "noisy" data in [12] (with a slightly different noise model than the one that arises here); and amplitude amplification [8, 22, 9].

Since U_{IP} applied to $|x\rangle|y\rangle$ has no net effect on its first n input qubits, for each $x \in \{0,1\}^n$,

$$U_{IP}|x\rangle|0^m\rangle = |x\rangle \left(\alpha_x|v_x\rangle|a \cdot x\rangle + \beta_x|w_x\rangle|\overline{a \cdot x}\rangle\right), \tag{3.130}$$

where α_x and β_x are nonnegative real numbers, and $|v_x\rangle$ and $|w_x\rangle$ are $m-1$ qubit quantum states. If the last qubit of $U_{IP}|x\rangle|0^m\rangle$ is measured then the result is: $a \cdot x$ with probability α_x^2, and $\overline{a \cdot x}$ with probability β_x^2. Therefore, since, for a random uniformly distributed $x \in \{0,1\}^n$, measuring the last qubit of $U_{IP}|x\rangle|0^m\rangle$ yields $a \cdot x$ with probability at least $\frac{1}{2} + \varepsilon$, it follows that

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \alpha_x^2 \geq \frac{1}{2} + \varepsilon \tag{3.131}$$

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \beta_x^2 \leq \frac{1}{2} - \varepsilon. \tag{3.132}$$

Now, consider the quantum circuit C in Figure 3.15. We will begin by showing that $\langle a, 0^m, 1|C|0^n, 0^m, 0\rangle$ is real-valued and

$$\langle a, 0^m, 1|C|0^n, 0^m, 0\rangle \geq 2\varepsilon, \tag{3.133}$$

which intuitively can be viewed as an indication of the progress that C makes towards finding the string a. To establish Eq. 3.133, note that the operation C can be

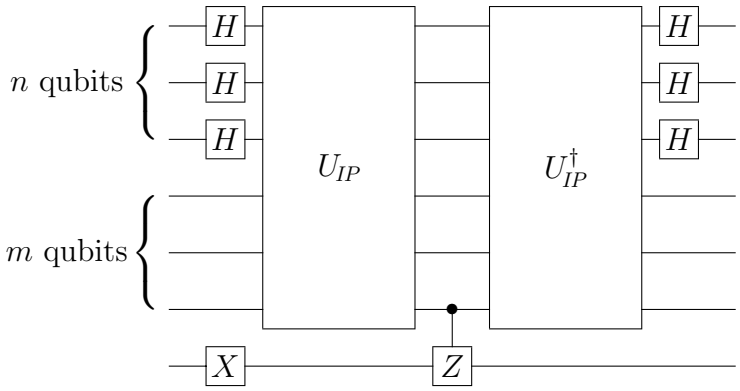

Figure 3.15: Quantum circuit C.

decomposed into the following five operations:

1. Operation C_1: Apply H to each of the first n qubits, and a NOT operation to the last qubit.

2. Operation C_2: Apply U_{IP} to the first $n+m$ qubits.

3. Operation C_3: Apply a controlled-Z to the last two qubits.

4. Operation C_4: Apply $U_{IP}^{\dagger}$ to the first $n+m$ qubits.

5. Operation C_5: Apply H to each of the first n qubits.

We note that since $\langle a, 0^m, 1|C|0^n, 0^m, 0\rangle = \langle a, 0^m, 1|C_5C_4C_3C_2C_1|0^n, 0^m, 0\rangle$, the quantity $\langle a, 0^m, 1|C|0^n, 0^m, 0\rangle$ is the inner product between state $C_3C_2C_1|0^n\rangle|0^m\rangle|0\rangle$ and

state $C_4^\dagger C_5^\dagger|a\rangle|0^m\rangle|1\rangle$. These states are

$$\begin{aligned}
C_3C_2C_1|0^n\rangle|0^m\rangle|0\rangle &= C_3C_2\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|0^m\rangle|1\rangle \\
&= C_3\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\left(\alpha_x|v_x\rangle|a\cdot x\rangle+\beta_x|w_x\rangle|\overline{a\cdot x}\rangle\right)|1\rangle \\
&= \tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\ \ \alpha_x(-1)^{a\cdot x}|v_x\rangle|a\cdot x\rangle+\beta_x(-1)^{\overline{a\cdot x}}|w_x\rangle|\overline{a\cdot x}\rangle\ \ |1\rangle \\
&= \tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{a\cdot x}|x\rangle\left(\alpha_x|v_x\rangle|a\cdot x\rangle-\beta_x|w_x\rangle|\overline{a\cdot x}\rangle\right)|1\rangle,
\end{aligned} \tag{3.134}$$

and

$$\begin{aligned}
C_4^\dagger C_5^\dagger|a\rangle|0^m\rangle|1\rangle &= C_4^\dagger\tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{a\cdot x}|x\rangle|0^m\rangle|1\rangle \\
&= \tfrac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{a\cdot x}|x\rangle\left(\alpha_x|v_x\rangle|a\cdot x\rangle+\beta_x|w_x\rangle|\overline{a\cdot x}\rangle\right)|1\rangle.
\end{aligned} \tag{3.135}$$

It follows from Eq. 3.134 and Eq. 3.135 (and using the fact that $\langle x|y\rangle = 0$ whenever $x = y$) that

$$\begin{aligned}
\langle a,0^m,1|C|0^n,0^m,0\rangle &= \tfrac{1}{2^n}\sum_{x\in\{0,1\}^n}\alpha_x^2-\beta_x^2 \\
&\geq (\tfrac{1}{2}+\varepsilon)-(\tfrac{1}{2}-\varepsilon) \\
&= 2\varepsilon,
\end{aligned} \tag{3.136}$$

which establishes Eq. 3.133.

Note that Eq. 3.133 implies that, if C is executed on input $|0^n\rangle|0^m\rangle|0\rangle$ $(= |0^n,0^m,0\rangle)$ and the result is measured in the classical basis, then the first n bits of the result will be a with probability at least $|\langle a,0^m,1|C|0^n,0^m,0\rangle|^2 \geq 4\varepsilon^2$. Therefore, if this process is repeated $O(1/\varepsilon^2)$ times, checking each result with an EQ query,

then a will be found with constant probability. A more efficient way of finding the value of a is to use *amplitude amplification* [8, 22, 9] using the transformation C and its inverse $C^\dagger$ in combination with EQ queries. The procedure is to compute for an appropriate value of k

$$(-CU_0C^\dagger U_{EQ})^k C|0^n, 0^m, 0\rangle \tag{3.137}$$

(where $U_0 = I - 2|0^n, 0^m, 0\rangle\langle 0^n, 0^m, 0|$), measure the state, and perform an EQ query on the result. Such a computation consists of $O(k)$ U_{IP}, $U_{IP}^\dagger$, and U_{EQ} queries. Since the number of solutions is known to be one, we set $k = \pi/(8\varepsilon)$ as discussed in our study of Grover's algorithm in section 1.2.4. As shown in [9], the expected total number of executions of C, $C^\dagger$, and U_{EQ} until a successful EQ query occurs is $O(1/\varepsilon)$. This implies that $O(1/\varepsilon)$ U_{IP}, $U_{IP}^\dagger$, and U_{EQ} are sufficient to succeed with constant probability. ∎

3.3 Lower Bounding the Quantum Goldreich-Levin Theorem

In the previous section we developed a straightforward quantum algorithm that solves the quantum G-L problem with $O(1/\varepsilon)$ IP queries and $O(1/\varepsilon)$ EQ queries. In this section we firstly determine the quantum lower bound on the number of EQ queries required by using both the result of the classical EQ lower bound and by evoking the well known lower bound on quantum search algorithms. We follow this with a section where we modify the proof of optimality of the quantum search algorithm in order to develop the lower bound on the number of IP queries. We conclude that the aforementioned upper bound is essentially a *tight bound*.

3.3.1 Equivalence of *EQ* Queries to Oracle Marking Queries

In the proof of the optimality of the search algorithm given in [2] and elsewhere, we are allowed to apply the marking oracle O_a which gives a phase shift of -1 to the solution $|a\rangle$ and leaves all other states invariant. This gate is interleaved with unitary operations $U_1, U_2, \ldots, U_k$ in constructing the proof. The optimality proof is directly applicable to the problem of lower bounding the number of EQ queries and with slight modification can be made applicable to the problem of lower bounding the number of IP queries. In both cases we need to show that the O_a oracle can be used to simulate the EQ oracle. We begin by noting that the EQ oracle given by Equation 3.129 can be written as

$$U_{EQ}|x\rangle|b\rangle = |x\rangle|b \oplus \delta_{x,a}\rangle, \tag{3.138}$$

which can also be nicely represented as the unitary matrix

$$U_{EQ} = \begin{bmatrix} [X]^{\delta_{x,a}} & 0 & \cdots & 0 \\ 0 & [X]^{\delta_{x,a}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & [X]^{\delta_{x,a}} \end{bmatrix}. \tag{3.139}$$

Assuming that the marking oracle only marks the *unique* state $|a\rangle$, this oracle can be written as

$$O_a|x\rangle = (-1)^{\delta_{x,a}}|x\rangle. \tag{3.140}$$

We define the *controlled* marking oracle as

$$\text{cont-}O_a|x\rangle|b\rangle = (-1)^{(\delta_{x,a})\cdot b}|x\rangle|b\rangle, \tag{3.141}$$

the circuit for which is presented in Figure 3.16. The cont-O_a operator can also be

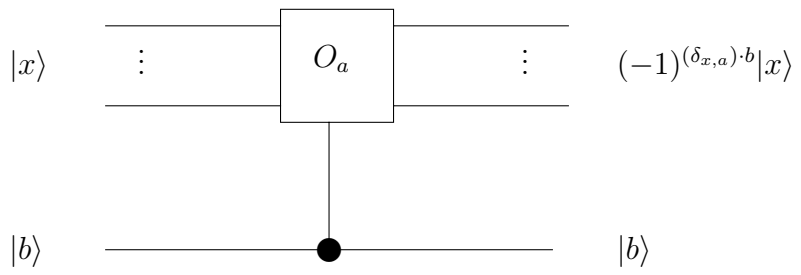

Figure 3.16: Circuit implementation of cont-$O_a|x\rangle|b\rangle$

nicely represented as the unitary matrix

$$\text{cont-}O_a = \begin{bmatrix} [Z]^{\delta_{x,a}} & 0 & \cdots & 0 \\ 0 & [Z]^{\delta_{x,a}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & [Z]^{\delta_{x,a}} \end{bmatrix}. \tag{3.142}$$

Multiplying Equation 3.142 on both sides by $I \otimes H$ we have

$$U_{EQ} = \begin{bmatrix} H[Z]^{\delta_{x,a}}H & 0 & \cdots & 0 \\ 0 & H[Z]^{\delta_{x,a}}H & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & H[Z]^{\delta_{x,a}}H \end{bmatrix}, \tag{3.143}$$

where we have used the relation $X = HZH$ to establish the equality with U_{EQ}. We present a circuit that is equivalent to Equation 3.138 in Figure 3.17. We thus conclude that we can directly simulate O_a queries with EQ queries and vice versa.

3.3.2 Lower Bounding the Number of *EQ* Queries

In Section 2.2.1 in our study of the classical case we showed that after performing an unlimited number of IP queries, we still have to search a list of length $\Omega(1/\varepsilon^2)$ using

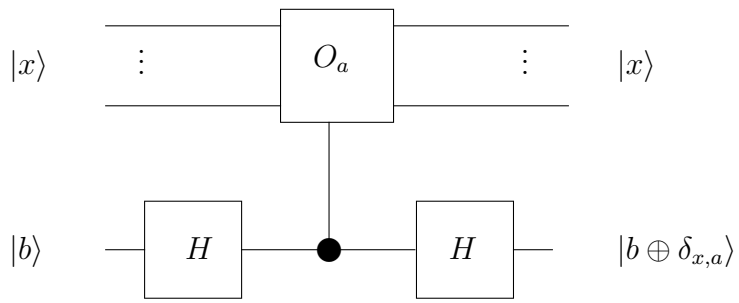

Figure 3.17: Circuit equivalent to $U_{EQ}|x\rangle|b\rangle$ constructed from a cont-O_a gate

EQ queries. In Section 1.2.3 in our discussion of quantum circuits, we showed that Grover's quantum search algorithm requires $O\left(\sqrt{N}\right)$ operations to search a list of N elements. It turns out that this upper bound is optimal, and one of the proofs is based on what is sometimes called the *Hybrid Argument*. A proof is not presented here, but the reader is directed to [27, pages 269-275] for an interesting discussion and proof of the optimality of the quantum search algorithm. Thus we can conclude that a circuit searching a list of N elements requires $\Omega\left(\sqrt{N}\right)$ operations. We now propose a theorem that lower bounds the number of quantum EQ queries, and give a proof sketch that makes use of the preceding.

Theorem 5 *For $\varepsilon \geq 1/(2 \cdot 2^{n/2})$, the instance of the quantum G-L problem with bias parameter ε requires $\Omega(1/\varepsilon)$ EQ queries regardless of how many IP queries are performed.*

Proof sketch: If we were to apply an unlimited number of noisy quantum IP queries, we would still need to apply the quantum EQ oracle a number of times in order to decide on the correct answer. As we demonstrated in Section 2.2, Theorem 2, there are $\Omega(1/\varepsilon^2)$ possible answers after applying an unlimited number of

IP queries. Intuitively this means that we must search a subspace whose size is $\Omega(1/\varepsilon^2)$. Accordingly by the optimality of the quantum search algorithm, we must use $\Omega(1/\varepsilon)$ EQ queries. Thus the upper bound on EQ queries presented in the algorithm developed in Section 3.2 is indeed a tight bound. ∎

3.3.3 Lower Bounding the Number of *IP* Queries

We are now going to modify the optimality proof to include two queries. The first query is the EQ query. The second query will be based on the operator A (and $A^\dagger$) parameterized by $p \in [1/2^n, 1]$. We define A as the unitary operation acting on n qubits such that, for all $y \in \{0,1\}^n$

$$A|y\rangle \quad = \quad \overline{1-p}|y\rangle + \mathbf{i}\sqrt{p}|y \oplus a\rangle. \tag{3.144}$$

We will later show that this operator is closely related to the IP oracle. In order to gain an appreciation for the form of the operator for the $n = 2$ case with $a = 01$ and $\sin\theta = \sqrt{p}$, we present its matrix representation as follows

$$A_2 = \begin{bmatrix} \cos\theta & \mathbf{i}\sin\theta & 0 & 0 \\ \mathbf{i}\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \mathbf{i}\sin\theta \\ 0 & 0 & \mathbf{i}\sin\theta & \cos\theta \end{bmatrix}. \tag{3.145}$$

Note that $|\langle a|A|0\rangle|^2 = p$. The second type of query is a *controlled-A* operation, denoted as cont-A, where cont-$A|y\rangle|b\rangle = (A^b|y\rangle)|b\rangle$, for all $y \in \{0,1\}^n$ and $b \in \{0,1\}$.

Consider the following *amplitude amplification* problem. There is an unknown $a \in \{0,1\}^n$ such that $a = 0$. Information about a is available by EQ, cont-A, and cont-$A^\dagger$ queries. The goal is to determine a. The well-known amplitude amplification

algorithm solves this problem using $O(1/\sqrt{p})$ EQ, cont-A, and cont-$A^\dagger$ queries. We first show that this is optimal in the number of cont-A, and cont-$A^\dagger$ queries.

Lemma 6 *The amplitude amplification problem requires* $\Omega\ \ 1/\sqrt{p}\ \ $ cont-A *or* cont-$A^\dagger$ *queries if the number of EQ queries is* $o(\sqrt{2^n})$.

Proof: This is straightforward to prove using the hybrid method. The proof is similar to the hybrid lower bound proof for searching with just EQ queries, except that cont-A and cont-$A^\dagger$ queries can be interleaved into the computation—and need to be accounted for.

The key idea is to bound the effect that each cont-A and cont-$A^\dagger$ query can have on a quantum state. The relevant result is that, for *any* quantum state $|\psi\rangle$,

$$|||\psi\rangle - \text{cont-}A|\psi\rangle|| \leq \quad \overline{2p}. \tag{3.146}$$

We can view the result of the operator cont-A on a state $|\psi\rangle$ as producing a new state with Euclidean distance $D = |||\psi\rangle - \text{cont-}A|\psi\rangle||$ from $|\psi\rangle$. Since the eigenvalues of $I - \text{cont-}A$ are either 0 or $1 - (\sqrt{1-p} + \mathbf{i}\quad \overline{p})$, the distance D is the same as the distance on the complex plane between 1 and $\sqrt{1-p} + \mathbf{i}\sqrt{p}$, which we present in Figure 3.18. We stress that this is the maximum effect that the operator cont-A can have on any state. Note that the size of p is greatly exaggerated in Figure 3.18 for the sake of clarity. We are interested in providing a bound to the distance D. By inspection we can use the Pythagorean theorem to write

$$\begin{aligned} D^2 &= p + \quad 1 - \quad \overline{1-p}\ ^2 \\ &= 2 - 2\quad \overline{1-p} \end{aligned} \tag{3.147}$$

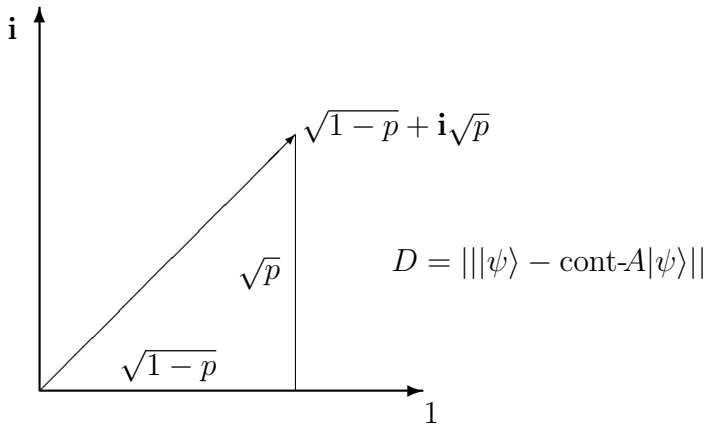

Figure 3.18: The maximum effect of the operator cont-A is represented in $\mathbb{C}$ as the distance between 1 and $\sqrt{1-p}+\mathbf{i}\sqrt{p}$

We note that for $0 \leq p \leq 1$, $\sqrt{1-p} \geq 1-p$. We use this to bound the distance given in 3.147 as

$$D \leq \quad \overline{2p}. \tag{3.148}$$

We now explain the use of this result in modifying the hybrid method. The hybrid approach works by replacing the *marking* oracle queries, O_a, with the identity operator. In our modification, we envision an arbitrary ordering of O_a, cont-A and cont-$A^\dagger$ queries. We define the state $|\psi_a^0\rangle$ as the state $|\psi_a^{k,l}\rangle$ where we have replaced k, EQ queries and l, cont-A (and cont-$A^\dagger$) queries with the identity operator. We are interested in bounding the average value of this distance $|||\psi_a^{k,l}\rangle - |\psi_a^0\rangle||$ over all $a \in \{0,1\}^n$. We define the maximum effect of the marking oracle at the *ith* step as $|||\psi_a^{i+1}\rangle - |\psi_a^i\rangle|| = 2\Delta_a^i$. We use this and our bound given in Equation 3.148 to write

$$|||\psi_a^{k,l}\rangle - |\psi_a^0\rangle|| \leq \sum_{i=0}^{k-1} 2\Delta_a^i + \sum_{i=k}^{k+l-1} \overline{2p}. \tag{3.149}$$

Note that we have used the triangle inequality in writing down Equation 3.149. We

now wish to obtain the *average* value of $|||\psi_a^{k,l}\rangle - |\psi_a^0\rangle||$ over all values of a. We express this average as

$$\frac{1}{N}|||\psi_a^{k+l}\rangle - |\psi_a^0\rangle|| \leq \frac{2}{N}\sum_{i=0}^{k-1}\left(\sum_{a\in\{0,1\}^n}\Delta_a^i\right) + \frac{1}{N}\sum_{i=k}^{k+l-1}\left(\sum_{a\in\{0,1\}^n}\overline{2p}\right) \quad (3.150)$$

From the Cauchy-Schwarz inequality, it is possible to prove that for any sequence of complex numbers c_i subject to $\sum_{i=1}^{N}|c_i|^2 = 1$, then $\sum_{i=1}^{N}|c_i| \leq \sqrt{N}$. The theorem can be applied to the first sum in Equation 3.150 since $\sum_{a\in\{0,1\}^n}(\Delta_a^i)^2 = 1$. Note that, as stated earlier, our amplitude amplification problem is not defined for $a = 0$. However, we can exclude $a = 0$ from the sum and not affect the inequality. We thus express Equation 3.150 as

$$\begin{aligned}\frac{1}{N}|||\psi_a^{k+l}\rangle - |\psi_a^0\rangle|| &\leq \frac{2}{N}\sum_{i=0}^{k-1}\left(\sum_{\substack{a\in\{0,1\}^n\\ a\neq 0}}\Delta_a^i\right) + \frac{1}{N}\sum_{i=k}^{k+l-1}\left(\sum_{\substack{a\in\{0,1\}^n\\ a\neq 0}}\overline{2p}\right)\\ &\leq \frac{2k}{\sqrt{N}} + l\ \overline{2p}. \end{aligned} \quad (3.151)$$

If we set $k \in o\left(\sqrt{N}\right)$ in Equation 3.151, then $l \in \Omega\left(1/\sqrt{p}\right)$ in order for $\frac{1}{N}|||\psi_a^{k+l}\rangle - |\psi_a^0\rangle||$ to be lower bounded by a positive constant. ∎

Next, we observe that a cont-A query can be used to simulate an IP query. The simulation is given by the circuit presented in Figure 3.19, where S is defined as $S|b\rangle = \mathbf{i}^b|b\rangle$, for $b \in \{0,1\}$.

Lemma 7 *If the last output qubit in the circuit presented in Figure 3.19 is measured then the probability that the outcome is $a \cdot x$ is $(1+\sqrt{p})/2$.*

Proof sketch: Let C denote the circuit presented in Figure 3.19. It is straightfor-

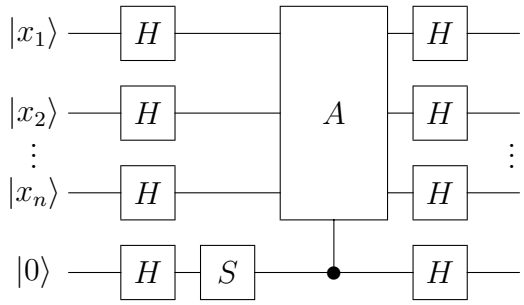

Figure 3.19: Simulating an IP query using a cont-A query. The last qubit, when measured, is biased towards $a \cdot x$.

ward to calculate that, for all $x \in \{0,1\}^n$,

$$\langle x, a \cdot x | C | x, 0 \rangle = \frac{1 + \sqrt{p} + \mathbf{i}(-1)^{a \cdot x}\sqrt{1-p}}{2}.$$

Taking the norm squared of this quantity yields $(1 + \sqrt{p})/2$. ∎

Using Lemma 7, any instance of the aforementioned amplitude amplification problem can be converted into an instance of the G-L problem. This means that the lower bound for amplitude amplification in Lemma 6 implies a corresponding lower bound for the G-L problem.

Theorem 8 *The G-L problem with bias parameter ε requires $\Omega(1/\varepsilon)$ IP queries if the number of EQ queries is $o(\sqrt{2^n})$.*

Proof sketch: The proof follows from the above reduction of the amplitude amplification problem to the G-L problem. Consider an instance of amplitude amplification with parameter $p = \varepsilon^2$. Suppose that the corresponding instance of the G-L problem, with IP queries defined as in Figure 3.19, can be solved with $o(1/\varepsilon)$ IP queries and $o(\sqrt{2^n})$ EQ queries. This algorithm would also solve the amplitude amplifi-

cation problem with $o(1/\sqrt{p})$ cont-A queries and $o(\sqrt{2^n})$ EQ queries, contradicting Lemma 6. ∎

3.4 Summary of Query Complexity of the G-L Problem

We now summarize the complexity bounds of both the classical and quantum G-L problems. In Table 3.6, we present the upper and lower classical bounds and the

Bound	Number of Queries	Notes
Classical Upper Bound	$O\ \frac{n}{\varepsilon^2}\ IP$ queries and $O\ \frac{1}{\varepsilon^2}\ EQ$ queries	Algorithm defined in Section 2.1 requires both queries.
Classical Lower Bound IP queries	$\Omega\ \frac{n}{\varepsilon^2}\ IP$ queries or $2^{\frac{n}{2}}\ EQ$ queries	True for $\varepsilon \geq \sqrt{n}2^{-n/3}$
Classical Lower Bound EQ queries	$\Omega\ \frac{1}{\varepsilon^2}\ EQ$ queries for unlimited IP queries	True for $\varepsilon \geq 1/\ \ 2 \cdot 2^{n/2}$
Quantum Upper Bound	$O\ \frac{1}{\varepsilon}\ IP$, $IP^\dagger$ and EQ queries	Algorithm defined in Section 3.2 requires all three queries.
Quantum Lower Bound IP queries	$\Omega\ \frac{1}{\varepsilon}\ IP$ queries or $\Omega(2^{n/2})$ EQ queries	True for $\varepsilon > 0$
Quantum Lower Bound EQ queries	$\Omega\ \frac{1}{\varepsilon}\ EQ$ queries for unlimited IP queries	True for $\varepsilon \geq 1/\ \ 2 \cdot 2^{n/2}$

Table 3.6: Summary of the query complexity of the classical and quantum G-L problems

upper and lower quantum bounds along with the constraints placed on them.

In the next chapter, we present classical and quantum bit commitment protocols whose security measures are quantified by the reduction of the computational problem of inverting classical and quantum one way functions to that of predicting a hard predicate of those functions. We generate hard predicates using the inner product of certain n-bit strings and use them in both classical and quantum protocols. In order to establish the reduction, we make use of fact that (under reasonable conditions of

various parameters) $\Omega\left(n/\varepsilon^2\right)$ IP queries are required to solve the classical G-L problem, and $\Omega\left(1/\varepsilon\right)$ IP queries are required to solve the quantum G-L problem. We are thus able to quantify the security of the bit commitment protocols in terms of the difficulty of inverting one-way functions. It is particularly satisfying to compare the *relative* security measures of the classical versus the quantum protocols, which result from the differing query complexities of the classical and quantum G-L problems.

Chapter 4 Cryptograhic Applications

4.0 Introduction

In this chapter, we discuss both classical and quantum bit commitment schemes based on the Goldreich-Levin Theorem. We begin by focusing on classical bit commitment. Here we expand on the concept of a *hard predicate* that we introduced in Chapter 1. We first give a definition of a hard predicate and then study its complexity by means of a reduction from the complexity of the problem of inverting a classical one-way permutation. We can view this complexity as a quantitative measure of the security of a bit commitment protocol based on the employment of this hard predicate. We conclude the classical discussions with a detailed description of a bit commitment protocol that employs a hard predicate derived from the classical Goldreich-Levin Theorem.

We follow this with a section on quantum bit commitment schemes. We begin with a brief history of quantum bit commitment and repeat a proof of the impossibility of *unconditional* quantum bit commitment. We then go on to discuss quantum bit and qubit commitment schemes based on the quantum version of the G-L theorem. We do this in a manner that parallels our discussion of the classical bit commitment scheme. We firstly quantify the complexity of the hard predicate, which allows us to compare the relative security of the classical and quantum schemes. We then present bit and qubit commitment schemes both of which employ a hard predicate derived from the quantum Goldreich-Levin Theorem.

In the final section, we note that bit (qubit) commitment schemes based on the

quantum G-L Theorem depend on the existence and implementation of a quantum one way permutation (QOWP). We conclude the chapter with a discussion of the current state of research into implementations of a QOWP.

4.1 Classical Bit Commitment based on G-L Theorem

In Chapter 1, we discussed bit commitment where we compared bit commitment using symmetric cryptography to bit commitment using one-way functions. We introduced the concept, and need, of a hard predicate of a one-way permutation. Here we are going to focus on classical bit-commitment using one-way permutations and hard predicates derived using the G-L theorem. We begin by slightly expanding the definition of a one-way permutation that we gave in Chapter 1.

Definition 10 Given a *one-way permutation* $f : \{0,1\}^n \to \{0,1\}^n$ as defined in Definition 5 in Chapter 1, let $\tilde{f}$ denote the permutation $\tilde{f} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ defined as $\tilde{f}(y,x) = (f(y),x)$. It is evident that $\tilde{f}$ is also a OWP.

With this definition of our OWP, we state that $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined as

$$h(y,x) = y \cdot x \tag{4.152}$$

is a hard predicate of $\tilde{f}$. Before discussing a bit-commitment protocol using the OWP $\tilde{f}$ and its hard predicate h, we will prove that h meets our definition (Chapter 1) and give the corresponding *dilution* of $\tilde{f}$.

4.1.1 Complexity of Classical Hard Predicates

In this section we investigate the complexity of the reduction of Goldreich and Levin from one-way permutations to hard predicates. With reference to Figure 4.20, we wish to show that any circuit, C, capable of *guessing* $z = a \cdot x$ given $f(a)$ and x, with reasonable success probability and polynomial size would also be able to invert $f(a)$ with reasonable success probability and polynomial size. This would contradict our definition of the one-way permutation, f, allowing us to conclude that no such circuit exists. More technically, we also provide a lower bound of the size of a circuit that predicts h from $\tilde{f}$. In Figure 4.20, we prepare the input to circuit C by starting with two n-bit strings a and x and subjecting them to the one-way permutation $\tilde{f}$ given in Definition 10. The quantities $f(a)$ and x are thus the inputs to our circuit

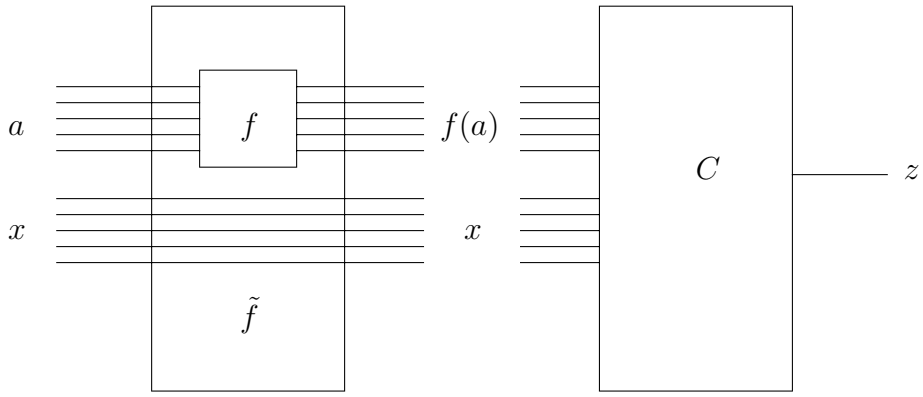

Figure 4.20: Circuit for predicting h

C, which returns the single bit z. What can we say about *size* of circuit C? In order to bound the answer to this question, we proceed with some definitions. The *size* of a classical circuit is understood to be relative to a suitable set of gates on one and

two bits.

Definition 11 A permutation $f : \{0,1\}^n \to \{0,1\}^n$ *is classically* (δ, T)*-hard to invert* if there is no classical circuit C of size T such that $\Pr_a[C(f(a)) = a] \geq \delta$.

Now the standard requirement for the hard-to-invert condition is that f is (δ, T)-hard to invert for all $\delta \in 1/n^{O(1)}$ and $T \in n^{O(1)}$. The idea behind a hard-predicate is to concentrate the information that a one-way function "hides" about its input into a single bit. Intuitively, $h : \{0,1\}^n \to \{0,1\}$ is a hard-predicate of f if, given $a \in \{0,1\}^n$, it is easy to compute $h(a)$; whereas, given $f(a)$ for randomly chosen $a \in \{0,1\}^n$, it is hard to predict the value of the bit $h(a)$ with probability significantly better than $\frac{1}{2}$.

Goldreich and Levin showed that if f is one-way then h is hard to predict from $\tilde{f}$. Instead of quantifying how well a circuit predicts h from $\tilde{f}$ as the amount by which $\Pr_{y,x}[C(g(y,x)) = h(y,x)]$ exceeds $\frac{1}{2}$, we adopt a related but slightly more complicated definition, that is suitable for our proof technique (we relate the two definitions in Lemma 9).

Definition 12 A circuit C (δ, ε)*-predicts* h *from* $\tilde{f}$ if

$$\Pr_y[\Pr_x[C(\tilde{f}(y,x)) = h(y,x)] \geq \tfrac{1}{2} + \varepsilon] \geq \delta. \tag{4.153}$$

To explain Eq. 4.153 in words, call $y \in \{0,1\}^n$ ε*-good* if $\Pr_x[C(\tilde{f}(y,x)) = h(y,x)] \geq \frac{1}{2} + \varepsilon$ for that value of y. Then Eq. 4.153 means $\Pr_y[y \text{ is } \varepsilon\text{-good}] \geq \delta$.

In order to help get an intuitive feeling of these two measures of predication, we present Figure 4.21, which is a contrived example of how a probability distribution $\varepsilon(y)$ might look for the $n = 7$ case. Note the global value of $\varepsilon = \mathrm{E}[\varepsilon_y]$ and region

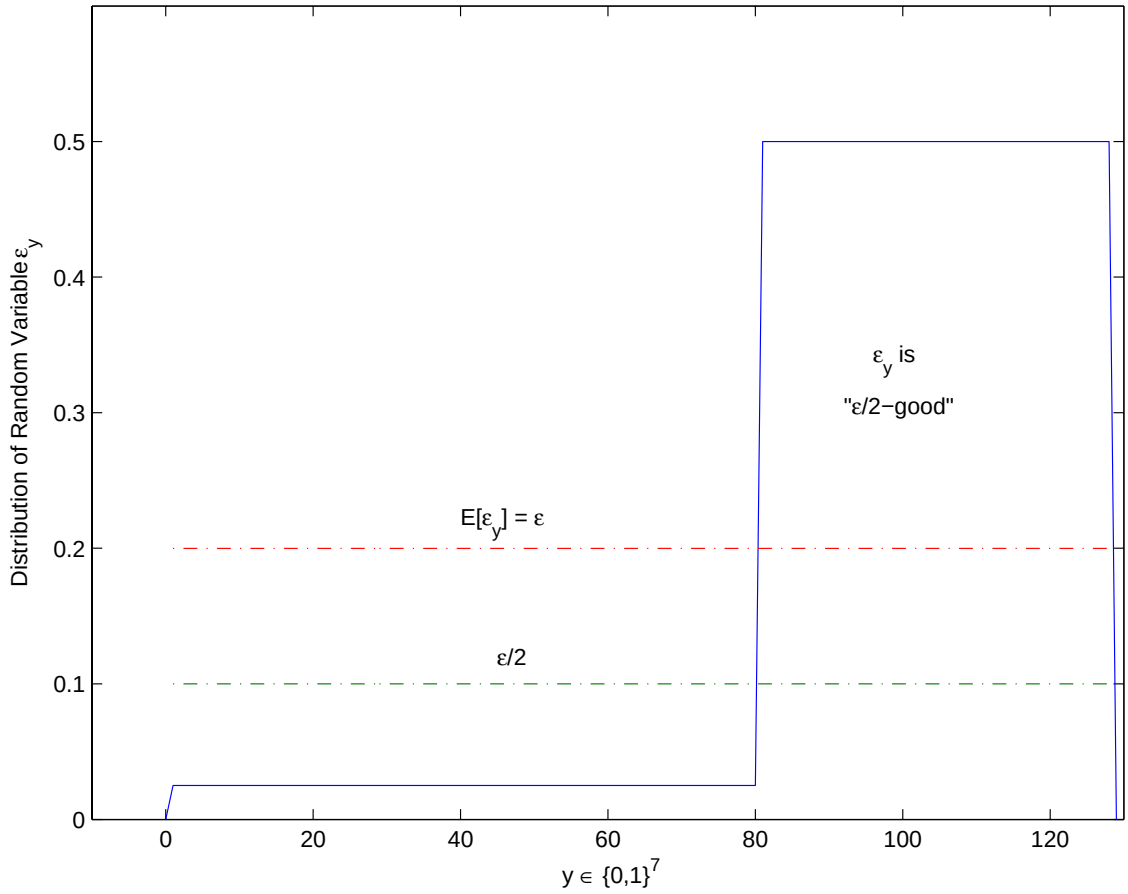

Figure 4.21: A contrived probability distribution ε_y showing the $\frac{\varepsilon}{2}$-good region.

where the y's are $\frac{\varepsilon}{2}$-good. The following lemma, which relates the two measures of prediction, is straightforward to prove by an averaging argument.

Lemma 9 *If* $\Pr_{y,x}[C(g(y,x)) = h(y,x)] \geq \frac{1}{2} + \varepsilon$ *then* G $(\varepsilon/(1-\varepsilon), \varepsilon/2)$*-predicts* h *from* $\tilde{f}$.

Proof: Let $p = \Pr[y \text{ is } \frac{\varepsilon}{2}\text{-good}]$. Then

$$\begin{aligned} \tfrac{1}{2} + \varepsilon &\leq \Pr[C(g(y,x)) = h(y,x) | y \text{ is } \tfrac{\varepsilon}{2}\text{-good}] \Pr[y \text{ is } \tfrac{\varepsilon}{2}\text{-good}] \\ &+ \Pr[C(g(y,x)) = h(y,x) | y \text{ is not } \tfrac{\varepsilon}{2}\text{-good}] \Pr[y \text{ is not } \tfrac{\varepsilon}{2}\text{-good}] \\ &< p + (\tfrac{1}{2} + \tfrac{\varepsilon}{2})(1-p), \end{aligned}$$

which implies $p > \varepsilon/(1-\varepsilon)$. ∎

Note in particular that, if $\Pr_{y,x}[C(g(y,x)) = h(y,x)] \geq \frac{1}{2} + 1/n^{O(1)}$ then C $(1/n^{O(1)}, 1/n^{O(1)})$-predicts h from g.

Theorem 10 *If $f : \{0,1\}^n \to \{0,1\}^n$ is classically $(\delta/2, T)$-hard to invert then any classical circuit that (δ, ε)-predicts h from $\tilde{f}$ must have size $\Omega(T\varepsilon^2/n)$.*

Proof sketch: The proof of this theorem is essentially a reduction from the problem of inverting f to the problem of (δ, ε)-predicting h. One begins by assuming that a circuit C of size $o(T\varepsilon^2/n)$ (δ, ε)-predicts h from g and then shows that, by making $O(n/\varepsilon^2)$ calls to both C and f (plus some additional computations), f can be inverted with probability $\delta/2$. The total running time of the inversion procedure is $o((n/\varepsilon^2)(T\varepsilon^2/n)) = o(T)$, contradicting the fact that f is $(\delta/2, T)$-hard to invert.

4.1.2 Protocol based on Classical G-L Theorem

As discussed in Chapter 2, the classical G-L Theorem and its attendant query problem are concerned with finding an unknown string based on information exposed by the response to inner-product queries. It should not be surprising then that a bit commitment protocol based on the G-L Theorem has at its core the inner-product based hard predicate expressed in Equation 4.152. The protocol is depicted in Figure

4.22. We assume that both Bob and Alice have access to a one-way permutation $f : \{0,1\}^n \to \{0,1\}^n$. First the *commitment* phase of the protocol is analyzed followed by the *de-commitment* phase. Commitment proceeds as follows.

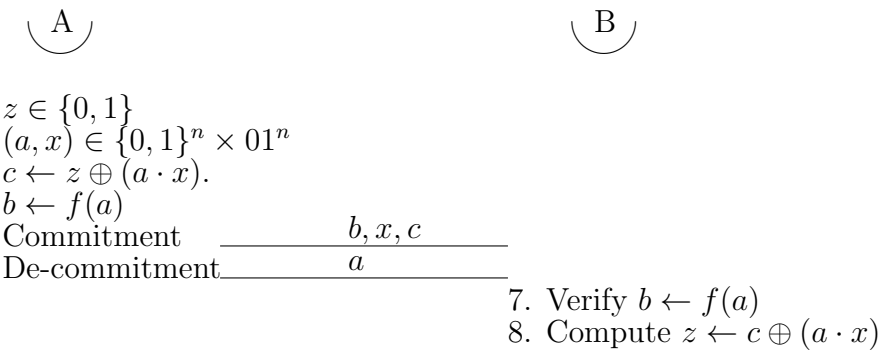

Figure 4.22: A bit commitment protocol based on classical G-L Theorem

1. Alice wishes to commit to bit $z \in \{0,1\}$.

2. Alice generates two random n-bit strings, a and x with $(a,x) \in \{0,1\}^n \times \{0,1\}^n$.

3. Alice computes a concealed bit using the inner product of her two strings, $c \leftarrow z \oplus (a \cdot x)$.

4. Alice computes the one-way permutation of one of the strings, $b \leftarrow f(a)$.

5. Alice sends her commitment, which consists of b, x, c to Bob.

The transmission from Alice is evidence of commitment. The commitment is *unconditionally binding* since there is no way Alice can change her commitment. The commitment is *computationally concealing* since as noted in Theorem 10, a classical circuit that (δ, ε)-predicts h from g must have size $\Omega(T\varepsilon^2/n)$. We started with a lower bound of $\Omega(T)$ for inverting the one way permutation f, but now we have

a weaker lower bound — by a *dilution* factor of $\Omega(n/\varepsilon^2)$ — for breaking the hard predicate. We use this factor as a measure to which this bit commitment protocol is *computationally concealing*. We note that b and x are n-bit strings, while c is a single bit. Thus, this protocol requires $O(n)$ bits for the commitment phase. The de-commitment proceeds when Alice decides it is time to reveal her bit.

1. Alice sends a, the second of the two random strings to Bob.

2. Bob checks the validity of this string by evaluating the one-way function: $b \leftarrow f(a)$.

3. If Bob is satisfied that Alice did not change her commitment (binding property of the protocol), he computes the bit Alice committed to: $z \leftarrow c \oplus (a \cdot x)$.

The de-commitment phase requires that n bits be exchanged. Before discussing the quantum versions of the hard predicate and the quantum bit commitment protocol, we discuss some of the history of quantum bit commitment and give a proof of the impossibility of unconditional bit commitment.

4.2 History of Quantum Bit Commitment

Over the years, various schemes have been proposed for implementing quantum bit commitment. One was even claimed to be unconditionally secure — the paper [6] appeared in FOCS'93 — but it turned out that the proof was wrong. In fact, not only was this scheme insecure, but it was subsequently proven that is impossible to implement unconditional bit commitment with quantum information. Proofs were

given by Mayers (1995/1997) [25] and by Lo and Chau (1996) [24]. The central idea of both proofs is based on the Schmidt decomposition of bi-partite quantum states.

Although it is impossible for a bit commitment protocol to be both perfectly concealing and perfectly binding, it is possible for it to be both partially concealing and partially binding. Spekkens and Rudolph [33] and others [7] have explored the trade-offs between the degree of bindingness and concealment that can be achieved simultaneously in any bit commitment protocol. In the following section, we give a proof sketch that addresses the non-existence of "perfect" bit commitment schemes. We then briefly explore an"almost" perfectly concealing scheme.

4.2.1 Impossibility of Quantum Bit Commitment

Suppose Alice and Bob share two quantum registers as depicted in figure 4.23. Pure states of these two registers together are elements of the Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$. We define the bases as $\mathcal{H}_A : \{|x\rangle : x \in \{0,1\}^n\}$ and $\mathcal{H}_B : \{|y\rangle : y \in \{0,1\}^m\}$. A basis for $\mathcal{H}_A \otimes \mathcal{H}_B$ is thus $\{|x\rangle|y\rangle : x \in \{0,1\}^n, y \in \{0,1\}^m\}$. Now Alice wishes to commit to a bit $b \in \{0,1\}$. She creates the state $|\psi_b\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ and sends the second register to Bob. Can this bit commitment scheme be simultaneously concealing and binding?

In order to show that it can not be, we consider the case where the commitment is perfectly concealing. That is where Bob can not distinguish between $|\psi_0\rangle$ and $|\psi_1\rangle$. If Bob decides to measure his register, the result is completely determined by $tr_{\mathcal{H}_A}|\psi_b\rangle\langle\psi_b|$ — that is by the partial trace achieved by tracing out register A. For the perfectly concealing case, the reduced density operators must be indistinguishable, and thus $tr_{\mathcal{H}_A}|\psi_0\rangle\langle\psi_0| = tr_{\mathcal{H}_A}|\psi_1\rangle\langle\psi_1|$. We now invoke the powerful technique of

Figure 4.23: Registers

Schmidt Decomposition. With this technique if we have $|\psi_b\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$, then it is possible to write $|\psi_b\rangle = \sum_{j=1}^{k} \sqrt{\rho_j}|\mu_j\rangle|\nu_j\rangle$ where $\{|\mu_1\rangle, \ldots, |\mu_k\}\rangle$ and $\{|\nu_1\rangle, \ldots, |\nu_k\}\rangle$ are orthonormal sets and $p_1, \ldots, p_k \in \{0, 1\}$ with $\sum_j p_j = 1$. Because the reduced density operators are indistinguishable, the Schmidt decompositions for Alice's two possible commitments, $|\psi_0\rangle$ and $|\psi_1\rangle$, may be written [27, page 110]

$$\begin{aligned} |\psi_0\rangle &= \sum_i \alpha_i |\gamma_i\rangle|\phi_i\rangle \\ |\psi_1\rangle &= \sum_i \alpha_i U(|\gamma_i\rangle)|\phi_i\rangle, \end{aligned} \tag{4.154}$$

where U is a unitary operator acting on A alone. In other words we can that say there exists a unitary operator acting on $\mathcal{H}_A$ such that

$$(U \otimes I)|\psi_0\rangle = |\psi_1\rangle, \tag{4.155}$$

which means that Alice can change her commitment at will. Since Alice can change her commitment without affecting Bob's register, we conclude that quantum bit commitment cannot be simultaneously concealing and binding.

We offer a concrete example for the two qubit case to illustrate the above concept.

Alice prepares the two states

$$\begin{aligned} |\psi_0\rangle &= \frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle \\ |\psi_1\rangle &= \frac{1}{\sqrt{2}}|+\rangle|0\rangle + \frac{1}{\sqrt{2}}|-\rangle|1\rangle. \end{aligned} \tag{4.156}$$

We also express these two states as the column vectors

$$|\psi_0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad |\psi_1\rangle = \tfrac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}. \tag{4.157}$$

She selects one of these two states and sends the second qubit to Bob. Now in order to convince ourselves that Bob can not distinguish which of these state his qubit represents, we will calculate the reduced density operator of $|\psi_0\rangle$ and $|\psi_1\rangle$ by tracing out the first qubit in each of the states of Equation 4.156.

$$\begin{aligned} \rho_0^A &= tr_{\mathcal{H}_A}|\psi_0\rangle = \tfrac{1}{2}|0\rangle\langle 0| + \tfrac{1}{2}|1\rangle\langle 1| \\ &= \tfrac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \end{aligned} \tag{4.158}$$

$$\begin{aligned} \rho_1^A &= tr_{\mathcal{H}_A}|\psi_1\rangle = \tfrac{1}{2}|+\rangle\langle +| + \tfrac{1}{2}|-\rangle\langle -| \\ &= \tfrac{1}{2}\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \end{aligned} \tag{4.159}$$

Since $\rho_0^A = \rho_1^A$, Bob is unable to distinguish $|\psi_0\rangle$ from $|\psi_1\rangle$, so this protocol is perfectly

concealing. However with $U = H$ in Equation 4.155, we express the operator

$$U \otimes I \;=\; H \otimes I = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}. \tag{4.160}$$

Applying this operator to the state $|0\rangle$, we have

$$\begin{aligned} (U \otimes I)|\psi_0\rangle \;&=\; \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ &=\; \tfrac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = |\psi_1\rangle. \end{aligned} \tag{4.161}$$

Thus we can clearly see in this example that Alice is able to change her commitment at will without Bob being aware.

So far we have considered the case where the protocol is *perfectly* concealing — that is where the two reduced density operators $\rho_0^A = \rho_1^A$. What happens if the protocol is "almost" perfectly concealing? We think of "almost" concealing as the probability of error (PE) of Bob guessing the state as being exponentially close to a half, which we write as

$$\left|\tfrac{1}{2} - PE\right| \leq 2^{-\alpha n}. \tag{4.162}$$

In [7] and [25] it is shown that under these conditions, there exist purifications $|\varphi_0\rangle$

and $|\varphi_1\rangle$ of the density operators ρ_0^A and ρ_1^A respectively that are very nearly the same state. Thus the square of the inner product between these states may be expressed

$$\langle\varphi_0|\varphi_1\rangle^2 \geq 1 - 2^{-\alpha n}. \tag{4.163}$$

We now show that Alice can "almost" perfectly cheat. If she wants to unveil $b = 0$, she maps $|\varphi_0\rangle$ into $|\psi_0\rangle$ and continues as if she has $b = 0$ in mind. Now if she wants to cheat and unveil $b = 1$, she executes on $|\varphi_0\rangle$ the unitary transformation F that would map $|\varphi_1\rangle$ into $|\psi_1\rangle$. She obtains the state $F|\varphi_0\rangle$. The square of the inner product between the desired state $F|\varphi_1\rangle$ and the actual state $F|\varphi_0\rangle$, is the same as given in Equation 4.163, which is exponentially close to 1. So, for all practical purposes, Alice can cheat as in the "perfect" case by applying this transformation F and then continuing as if she has $b = 1$ in mind. From this sketch, we conclude that quantum bit commitment is insecure. It is of interest to note that there is a trade-off between the degree of concealment of quantum bit commitment and how binding it is.

Since quantum information does not permit unconditional bit commitment, we look to computationally strong bit commitment schemes. Next we present a perfectly binding and computationally concealing bit commitment scheme based on the quantum Goldreich-Levin theorem.

4.3 Quantum Bit Commitment Based on G-L Theorem

Here we are going to focus on bit-commitment using quantum one-way permutations and *hard predicates* derived using the G-L theorem.

4.3.1 Security of Quantum G-L Based Bit Commitment

Our quantum version of the Goldreich-Levin Theorem is the following.

Theorem 11 *If $f : \{0,1\}^n \to \{0,1\}^n$ is quantumly $(\delta/2, T)$-hard to invert then any quantum circuit that (δ, ε)-predicts h from g must have size $\Omega(T\varepsilon)$.*

Proof: As in the classical case, the proof is essentially a reduction from the problem of inverting f to the problem of (δ, ε)-predicting h. Let $b = f(a)$ be an input instance—the goal is to determine a from b. We will show how to simulate *EQ* and *IP* queries in this setting and then apply the bounds in Theorem 8. It is easy to simulate an *EQ* query (relative to a) by making one call to f and checking if the result is b. Suppose that there exists a circuit G of size $o(T\varepsilon)$ that (δ, ε)-predicts h from $\tilde{f}$. Thus, $\Pr_y[\Pr_x[G(g(y,x)) = h(y,x)] \geq \frac{1}{2} + \varepsilon] \geq \delta$. Note that, with probability at least δ, a is ε-good, in the sense that $\Pr_x[G(g(a,x)) = h(a,x)] \geq \frac{1}{2} + \varepsilon$. When a is ε-good, computing $G(\tilde{f}(a,x)) = G(b,x)$ is simulating an *IP* query for x (relative to a). It follows from Theorem 8 that a can be computed with circuit-size $o((1/\varepsilon)(T\varepsilon)) = o(T)$ with success probability at least $\delta/2$ (where $1/2$ is the success probability of the algorithm that finds a when a is ε-good and δ is the probability that a is ε-good to begin with). This contradicts the $(\delta/2, T)$-hardness of inverting f, thus such a G cannot exist. ∎

4.3.2 Protocol Based on Quantum G-L Theorem

We begin by presenting a bit commitment protocol based on a quantum one-way permutation (QOWP). We assume that both Bob and Alice have access to the QOWP

$f : \{0,1\}^n \to \{0,1\}^n$. The protocol appears identical to the classical protocol presented in figure 4.22. In direct comparison to the classical case, here we again started with a lower bound of $\Omega(T)$ for inverting the *quantum* one way permutation f, but now the dilution factor for breaking the hard predicate is only $\Omega(1/\varepsilon)$ compared to $\Omega(n/\varepsilon^2)$ for the classical case. To get a feel for the relative difference between the best possible quantum and the best possible classical reductions, we offer the following example. Consider the case where $T = n^3$ and $\varepsilon = 1/n$. If we start with a classical one-way function that requires a computational cost of $\Omega(n^3)$ to invert and apply the Goldreich-Levin Theorem to construct a classical hard-predicate then the reduction implies only that the computational cost of predicting the predicate with probability $\frac{1}{2} + \frac{1}{n}$ is lower bounded only by a *constant*. However, if we start with a *quantum* one-way function that requires a computational cost of $\Omega(n^3)$ to invert and apply our quantum version of the G-L Theorem then the computational cost of predicting the predicate with probability $\frac{1}{2} + \frac{1}{n}$ is lower bounded by $\Omega(n^2)$. We

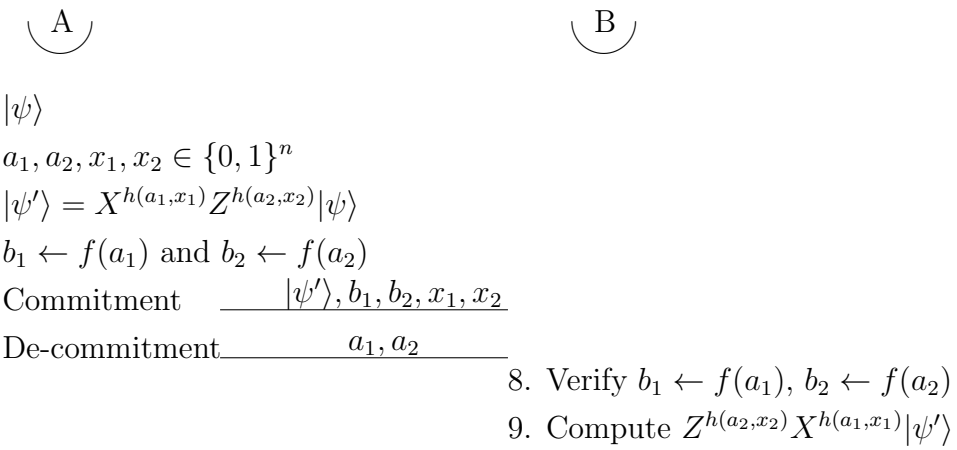

Figure 4.24: A qubit commitment protocol based on quantum G-L Theorem.

note that a bit commitment scheme the same as discussed in Section 4.1.2 could be

modified to incorporate a *quantum* one-way function. All other aspects of the protocol would be the same, except that the dilution factor is now only $\Omega(1/\varepsilon)$. Finally, and perhaps of more interest, we explain how a *qubit* commitment scheme can be constructed. First the *commitment* phase of the protocol is analyzed followed by the *de-commitment* phase. Commitment proceeds as follows.

1. Alice wishes to commit to the qubit $|\psi\rangle$.
2. Alice randomly chooses $a_1, a_2, x_1, x_2 \in \{0,1\}^n$.
3. Alice constructs the state $|\psi'\rangle = X^{h(a_1,x_1)} Z^{h(a_2,x_2)} |\psi\rangle$.
4. Alice computes the one-way permutation twice: $b_1 \leftarrow f(a_1)$ and $b_2 \leftarrow f(a_2)$.
5. Alice sends her commitment,which consists of $(|\psi'\rangle, b_1, b_2, x_1, x_2)$ to Bob.

Clearly, the scheme is perfectly binding. Intuitively, the scheme is computationally concealing, because $h(a_1, x_1)$ and $h(a_2, x_2)$ "look random" to Bob. We give a rough sketch of why we can make this conclusion in the following. If Bob can use his information to efficiently distinguish between the qubit that he receives from Alice in the commitment stage and a totally mixed state (density matrix $\frac{1}{2}I$) then this procedure can be adapted to distinguish between the pair of bits $r_1 = h(a_1, x_1)$ and $r_2 = h(a_2, x_2)$ and a pair of truly random bits, which would violate the result proven in Theorem 11. A quantum circuit that (δ, ε)-predicts h from g must have size $\Omega(T/\varepsilon)$. We started with a lower bound of $\Omega(T)$ for inverting the one way permutation f, but now we have a weaker lower bound — by a *dilution* factor of $\Omega(1/\varepsilon)$— for breaking the hard predicate. We note that the commitment requires that one qubit

and $O(n)$ bits be exchanged. The de-commitment proceeds when Alice decides it is time to reveal her bit.

1. Alice sends a_1, a_2 to Bob.

2. Bob checks if $f(a_1) = b_1$ and $f(a_2) = b_2$, rejecting if this is not the case. Otherwise, Bob accepts.

3. If Bob is satisfied that Alice did not change her commitment (binding property of the protocol), he computes the qubit Alice committed to: $Z^{h(a_2,x_2)}X^{h(a_1,x_1)}|\psi'\rangle$.

The de-commitment phase requires that $O(n)$ bits be exchanged. We note that in order to realize quantum bit and qubit protocols, we require a realizable quantum one-way function.

4.4 Quantum One-way Functions and Permutations

At the time of writing this thesis, little is known about quantum one-way functions and permutations. Other than to cite some references from the literature, further discussion is out of scope of this thesis.

In their 2000 paper, titled Quantum Public-Key Cryptosystems [28] by Okamato, Tanaka and Uchiyama discuss some interesting candidates.

Conclusion

We have proven *tight* bounds of the classical and quantum Goldreich-Levin (G-L) Problem to be $\Theta(n/\varepsilon^2)$ and $\Theta(1/\varepsilon)$ respectively. We have also presented unconditionally binding and computationally concealing bit and qubit commitment protocols based on hard predicates derived from the classical and quantum G-L Theorem. These protocols are dependent on the existence of classical and quantum one way permutations. Finally, we have shown that the difference between the classical and quantum bounds result in the quantum protocols having quantitatively better security than the classical protocols.

Although it appears that the door is closed and the story of the G-L Theorem is complete, this is not the case. We believe that if a simpler circuit relationship between A queries and IP queries could be developed rather than what we have shown in Chapter 3, a more elegant proof of the quantum lower bound of both EQ and IP queries could be determined. It would be of particular interest to see the quantum EQ lower bound proven without reference to the classical EQ lower bound. We also await any news on classical or quantum one way functions.

In parting we note that the analysis of the upper bound of the classical G-L Problem is much more involved than the analysis of the upper bound of the quantum G-L Problem. It appears that the inner product of two strings is one of those global properties that is particularly suitable to solution using the equal superposition of quantum states, which allows us to avoid the complexity of the classical analysis.

Bibliography

[1] M. Adcock and R. Cleve, "A quantum Goldreich-Levin theorem with cryptographic applications", *Proc. Symposium on Techincal Aspects of Computer Science (STACS 2002)*, pp. 323–334, 2002.

[2] A. Ambainis, M. Mosca, A. Tapp and R. de Wolf, "Private quantum channels", *Proc. 41st Ann. IEEE Symp. on Foundations of Computer Science (FOCS '00)*, pp. 547–553, 2000.

[3] M. Bellare, "The Goldreich-Levin Theorem", Manuscript, 1999.
(Available at `http://www-cse.ucsd.edu/users/mihir/`.)

[4] E. Bernstein and U. V. Vazirani, "Quantum complexity theory", *SIAM J. on Comput.*, Vol. 26, No. 5, pp. 1411–1473, 1997.

[5] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits", *SIAM J. on Comput.*, Vol. 13, No. 4, pp. 850–864, 1984.

[6] G. Brassard, C. Crpeau, R. Jozsa, and D. Langlois. "A quantum bit commitment scheme provably unbreakable by both parties" *In 34th Annual Symposium on Foundations of Computer Science*, pp. 362-371, November 1993.

[7] G. Brassard, C. Crépeau, D. Mayers and L. Salvail, "A brief review on the impossibility of quantum bit commitment", *Los Alamos preprint archive quant-ph/9712023*, December 1997.

[8] G. Brassard and P. Høyer, "An exact quantum polynomial-time algorithm for Simon's problem", *Proc. Fifth Israeli Symp. on Theory of Computing and Systems*, pp. 12–23, 1997.

[9] G. Brassard, P. Høyer, M. Mosca and A. Tapp, "Quantum amplitude amplification and estimation", To appear in *Quantum Computation and Quantum Information: A Millennium Volume*, AMS Contemporary Mathematics Volume. Available on the LANL preprint archive as quant-ph/0005055, 2000.

[10] T. H. Cormen, C. E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA 1990.

[11] H. F. Chau and H.-K. Lo, "One way functions in reversible computations", *Cryptologia*, Vol. 21, No. 2, pp. 139–148, 1997.

[12] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp, "Quantum entanglement and the communication complexity of the inner product function", *Proc. of the First NASA International Conf. on Quantum Computing and Quantum Communications*, Colin P. Williams (Ed.), Lecture Notes in Computer Science 1509, Springer-Verlag, pp. 61-74, 1999.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley and Sons, 1991.

[14] C. H. Bennett and G. Brassard, *Proc. IEEE International Conference on Computers, Systems, and Signal Processing, IEEE Press, Los Alamitos, Calif.*,A400 (1984), p. 175.

[15] C. Crépeau, F. Légaré and L. Salvail, "How to convert the flavor of a quantum bit commitment", to appear in *Advances in Cryptology — EUROCRYPT 2001.*

[16] D. Deutsch, "Quantum Theory, the Church-Turing principle and the Universal Quantum Computer", *Proceedings of the Royal Society (London)*, A400 (1985): 97-117.

[17] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation", *Proceedings of the Royal Society (London)*, A439 (1992): 553-558.

[18] P. Dumais, D. Mayers, and L. Salvail, "Perfectly concealing quantum bit commitment from any one-way permutation", *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel (Ed.), Lecture Notes in Computer Science 1807, Springer-Verlag, pp. 300–315, 2000.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[20] O. Goldreich and L. Levin, "Hard-core predicates for any one-way function", *Proc. 21th Ann. ACM Symp. on Theory of Computing (STOC '89)*, pp. 25–32, 1989.

[21] O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness*, Springer, 1999.

[22] L. K. Grover, "A fast quantum mechanical algorithm for database search", *Proc. 28th Ann. ACM Symp. on Theory of Computing (STOC '96)*, pp. 212–219, 1996.

[23] J. Justesen, "A class of constructive asymptotically good algebraic codes", *IEEE Trans. Inform. Theory*, 18:652-656, 1972.

[24] H.-K. Lo and H. F. Chau, "Is quantum bit commitment really possible?", *Phys. Rev. Lett.*, Vol. 78, No. 17, pp. 3410–3413, 1997.

[25] D. Mayers, "Unconditionally secure bit commitment is impossible", *Phys. Rev. Lett.*, Vol. 78, No. 17, pp. 3414–3417, 1997.

[26] A. Michelson and A. Levesque, *Error-Control Techniques for Digital Communication*, John Wiley and Sons, 1985.

[27] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[28] T. Okamato, K. Tanaka and S. Uchiyama "Quantum Public-Key Cryptosystems", *CRYPTO 2000*, LNCS 1880, pp. 147–165, 2000.

[29] B. Schneir, *Applied Cryptography, Second Edition*, John Wiley and Sons, New York, NY 1996.

[30] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, pp. 379–423, 1948.

[31] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. on Computing*, Vol. 26, No. 5, pp. 1484–1509, 1997.

[32] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, Boston, MA 1997.

[33] R. W. Spekkens and T. Rudolph, "Degrees of concealment and bindingness in quantum bit commitment protocols", *Phys. Rev. A*, Vol. 65, 012310, 2002.

[34] B. M. Terhal and J. A. Smolin, "Single quantum querying of a database", *Phys. Rev. A*, Vol. 58, No. 3, pp. 1822–1826, 1998.

[35] A. C.-C. Yao, "Lower bounds by probabilistic arguments", *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science (FOCS '83)*, pp. 420–428, 1983.