Classical Verification and Blind Delegation of Quantum Computations

by

Urmila M. Mahadev

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Umesh Vazirani, Chair
Professor Satish Rao
Professor Nikhil Srivastava

Spring 2018

Classical Verification and Blind Delegation of Quantum Computations

Abstract

Classical Verification and Blind Delegation of Quantum Computations

by

Urmila M. Mahadev

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Umesh Vazirani, Chair

In this dissertation, we solve two open questions. First, can the output of a quantum computation be verified classically? We give the first protocol for provable classical verification of efficient quantum computations, depending only on the assumption that the learning with errors problem is post-quantum secure.

The second question, which is related to verifiability and is often referred to as blind computation, asks the following: can a classical client delegate a desired quantum computation to a remote quantum server while hiding all data from the server? This is especially relevant to proposals for quantum computing in the cloud. For classical computations, this task is achieved by the celebrated result of fully homomorphic encryption ([23]). We prove an analogous result for quantum computations by showing that certain classical homomorphic encryption schemes, when used in a different manner, are able to homomorphically evaluate quantum circuits.

While we use entirely different techniques to construct the verification and homomorphic encryption protocols, they both rely on the same underlying cryptographic primitive of trapdoor claw-free functions.

Contents

Acknowledgments

I have been very happy throughout my time in graduate school, both to be at UC Berkeley and to be working on the questions in this thesis. I owe thanks to many people, starting with my advisor, Umesh Vazirani. There are many things to thank him for: for presenting the problem of verifiability in a particularly nice and approachable way, for allowing me to continue working on this problem while making very little progress, and for encouraging me and offering me chocolate at the right times. There were never any logistical concerns and I was free to spend all my time thinking about research, which may have been a bad thing given how hard it will be to leave Berkeley. I can't imagine a better graduate school experience.

Thanks to Dorit Aharonov for many visits to Israel where I learned new techniques, how to express my ideas, and was encouraged to continue working on the problems I cared most about. Zeph Landau has been a constant source of support, humor, and different perspectives. I thank Thomas Vidick for many useful conversations and feedback (and for spending a lot of time parsing hard to read drafts). I am grateful to the researchers I have worked with over the years and those who hosted me for productive visits, including Michael Ben-Or, Zvika Brakerski, Paul Christiano, Sanjam Garg, Stacey Jeffery, Iordanis Kerenidis, Mario Szegedy, and Ronald de Wolf. Going back a bit further, I thank Len Adleman and Manoj Gopalkrishnan for getting me involved in research in the first place.

The theory group at Berkeley has been an amazing place to be a graduate student and there were several people who made my days very entertaining, including Antonio, Ma'ayan, Siu Man, James, Anindya, Sara, Seung Woo, Piyush, and Greg. The faculty members created a wonderful culture and I thank Satish Rao in particular for many interesting conversations.

I thank my family: my parents for their unwavering support and for working so hard to make my life easy, my brother and sister for keeping me focused on the bigger picture by repeatedly asking me how my degree could help me change light bulbs or make the Christmas lights work properly, and Tuhin for teaching me the key to success on analysis tests.

Chapter 1

Introduction

The exponential power of quantum computing brings with it a major challenge: how can the output of such a computation be verified classically? The laws of quantum mechanics appear to conspire to make this task particularly daunting, by severely limiting the amount of classical information which can be obtained from a quantum system via measurement. Besides being a fundamental practical consideration, the verification of quantum computations has connections to deep philosophical questions about the viability of the scientific method (predict and test) in the context of quantum mechanics ([2]). The issue is as follows: how can one test a hypothesis about a theory in which predicting the results of an experiment take exponential time? The verification of a quantum computation through interaction provides a potential way forward.

Analogously to the result $\mathsf{IP} = \mathsf{PSPACE}$ ([45]), is it possible for a prover who is restricted to efficient quantum machines (a BQP prover) to convince a classical polynomial time verifier (a BPP machine) about membership in any BQP language? This question was first raised by Daniel Gottesman in 2004 ([28]). In the absence of any techniques for tackling this question, two weaker formulations were considered. In the first, it was shown that if the verifier had access to a small quantum computer, verification of all efficient quantum computations was possible ([13], [21], [1], [3]). The second formulation considered a classical polynomial time verifier interacting with two entangled, non communicating quantum provers (rather than just one machine), and showed that in this setting it was possible to verify the result of an arbitrary quantum computation ([44]). Although both lines of work initiated extensive research efforts, the question of classical verification by interaction with a single quantum computer has remained elusive.

In this dissertation, we answer this question affirmatively: we show that a classical polynomial time verifier (a BPP machine) can interact with an efficient quantum prover (a BQP machine) in order to verify BQP computations. We rely on the additional assumption that the verifier may use post-quantum cryptography that the BQP prover cannot break. More specifically, we rely on quantum secure classical encryption schemes, such as those based on the learning with errors problem ([43]). These schemes can be used to encrypt classical bits (or quantum states) in a way in which an efficient quantum machine cannot extract any

information.

This brings up a challenge related to verifiability, which is often referred to as blind computation: can a classical client delegate a desired quantum computation to a remote quantum server while hiding all data from the server? Quantum secure classical encryption schemes do not immediately answer this question; they provide a way of hiding data, but not of computing on the data. This question is particularly relevant to proposals for quantum computing in the cloud.

The classical analogue of this task, in which the client is a weak classical machine and the server is more powerful, was solved in 2009 with the celebrated construction of homomorphic encryption schemes ([23]). Unfortunately, these schemes are built only to handle computations on the encrypted data involving classical operations; the prospect of applying computations in superposition over encrypted bits seems to be much more difficult. This difficulty arises from the fact that all classical homomorphic encryption schemes require (for security) that each bit has many possible different encryptions. This property appears to preclude the quantum property of interference: interference requires that elements of the superposition representing the same bit string, but with opposite amplitudes, cancel out. If these elements have different encryptions, interference cannot happen, thereby preventing one of the key advantages of quantum algorithms.

Due to these obstacles, the question of quantum homomorphic encryption was weakened by allowing a quantum client ([14]). This variant has been well studied in recent years ([14], [39], [47], [33], [38], [20]), but the model has a number of shortcomings. The principal issue is that the quantum client relies on quantum encryption keys, which are not reusable; the client must generate fresh keys each time he wishes to delegate a quantum computation. Taken in conjunction with the fact that existing protocols ([20]) require the client to generate a number of quantum keys proportional to the size of the quantum circuit being applied, this means that the client's total work is as large as actually running the desired quantum computation himself.

The related question of blind quantum computation predated the question of quantum homomorphic encryption and has also been extensively studied, beginning with [15]. Blind quantum computation and quantum homomorphic encryption have the same goal of carrying out a computation on encrypted data, but blind computation allows multiple rounds of interaction between the client and server, while homomorphic encryption allows only one round of interaction. Even in the weaker model of blind computation, a quantum client has been a necessity so far (at a minimum, the client must be able to prepare certain single qubit states [13]).

In this dissertation, we return to the original question of quantum homomorphic encryption by providing (in Chapter 5) a homomorphic encryption scheme for quantum computations with a classical client. To do this, we show that certain classical homomorphic encryption schemes can be lifted to the quantum setting; they can be used in a different way to allow for homomorphic evaluation of quantum circuits. It follows that all properties of classical homomorphic encryption schemes (such as reusability of keys and circular security) also hold in the quantum setting. This scheme is the first (and currently only) to allow

blind quantum computation between a classical client and a quantum server.[1] Note that the homomorphic encryption protocol does not imply the verifiability protocol, as it is currently unknown whether blind computation implies verifiable computation in the setting of a classical verifier and a quantum prover. In Chapter 2, we discuss the relationship between blind and verifiable computation in detail, and provide an introduction to techniques used in blind and verifiable computing along the way.

Both results rely heavily on a classical cryptographic primitive called a trapdoor claw-free function, which is a function f which is two to one, easy to invert with access to a trapdoor, and for which it is computationally difficult to find any pair of preimages with the same image. Such a pair of preimages is called a claw, hence the name claw-free. These functions are particularly useful in the quantum setting, due to the fact that a quantum machine can create a uniform superposition over a random claw (x_0, x_1): $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$. This superposition can be used to obtain information which is conjectured to be hard to obtain classically: the quantum machine can obtain a string $d \neq 0$ such that $d \cdot (x_0 \oplus x_1) = 0$. In [11], this advantage was introduced and used to show how to generate information theoretic randomness from a single quantum device. In this thesis, we describe this advantage in detail in Chapter 4 and show how to use it to achieve both homomorphic encryption (Chapter 5) and verifiability (Chapter 6) of quantum computations using only a classical computer.

For the homomorphic encryption result, we begin with the fact that blindly computing a quantum circuit boils down to the ability of a quantum server to perform a CNOT gate (a reversible XOR gate) controlled by classically encrypted data: we need a method in which a quantum server holding a classically encrypted bit s can apply CNOT^s to a quantum state (i.e. apply the CNOT gate if and only if $s = 1$). This method must not leak information about the bit s. This observation is not new to this thesis, and is in fact quite well known ([14], [20]). However, all previous results proposed quantum functions to enable the server to perform such an operation (hence the need for a quantum client).

Here we propose a classical function, which we call an encrypted CNOT operation. Our encrypted CNOT operation relies on a trapdoor claw-free function for which each claw hides the bit s. We show that an approximation of such a function can be built given a classical encryption of the bit s as long as the classical encryption scheme satisfies several straightforward requirements; we call such schemes *quantum capable* encryption schemes. Next, by combining two existing homomorphic encryption schemes based on learning with errors ([24], [23]), we build a quantum capable homomorphic encryption scheme. The main result of Chapter 5 is as follows (stated formally in Theorem 5.4.2 and Theorem 5.5.2):

Theorem 1.0.1 (Informal) *Under the assumption that the learning with errors problem with superpolynomial noise ratio is computationally intractable for an efficient quantum ma-*

[1] There have been two papers ([34], [18]) proposing delegated blind quantum computation protocols between a classical client and a quantum server. Both of these results differ from ours in that they do not claim security (i.e. blindness) against a malicious quantum server for the delegation of general quantum computations. Moreover, both results require interaction.

chine, there exists a quantum leveled fully homomorphic encryption scheme with classical keys.

Our scheme shares the same properties as classical homomorphic encryption schemes, including key reusability and the fact that the amount of work performed by the client is proportional to the depth of the delegated circuit (rather than its size). Moreover, as in classical homomorphic encryption schemes, this overhead can be made independent of the circuit size with stronger assumptions regarding the circular security of the classical encryption scheme.

In the verifiability setting, we use trapdoor claw-free functions to allow the classical verifier to force the quantum prover to encode a quantum state (chosen by the prover) in a way which is unknown to the prover. By strengthening the claw-free property in two different ways, this encoding can be used to randomize any operator applied by the prover which is a deviation from the strategy of an honest prover, rendering the deviation of the prover essentially useless. We provide the definition and construction of an extended trapdoor claw-free family which satisfies the strengthened properties. This family is an extension of the family given in [11]. Like the construction in [11], our construction relies on the computational assumption that a BQP machine cannot solve the learning with errors problem with superpolynomial noise ratio ([43]). The main result of Chapter 6 is stated informally as follows (stated formally as Theorem 6.7.6):

Theorem 1.0.2 *Assuming the existence of an extended trapdoor claw-free family, all decision problems which can be efficiently computed in quantum polynomial time (the class BQP) can be verified by an efficient classical machine through interaction (formally, BQP= QPIP$_0$).*

In Section 6.8, we provide a learning with errors based construction of an extended trapdoor family, providing a proof of the following theorem (stated formally as Theorem 6.8.1 in Section 6.8):

Theorem 1.0.3 *Under the assumption that the learning with errors problem with superpolynomial noise ratio is computationally intractable for an efficient quantum machine, there exists an extended trapdoor claw-free family.*

Chapter 2

Blind versus Verifiable Quantum Computing

The goal of this chapter is to discuss the difficulties in extending blind computation to verifiable computation. While moving towards this goal, we also aim to provide an introduction to some of the key methods used in blind and verifiable computing in order to make them more accessible in later chapters. Throughout this section, we introduce preliminaries and techniques as they are required. For a more thorough treatment of the preliminaries, refer to Chapter 3.

As a rough outline of this chapter, we will begin by providing a particularly simple blind quantum computation scheme[1] (in the setting of a limited quantum client and a quantum server). We then discuss the obstacles to making this scheme verifiable, and what these obstacles imply for the case of extending classical client blind quantum computing to a scheme for classical verification of quantum computations.

2.1 Blind Quantum Computation with the Pauli One Time Pad

We begin with a simple encryption scheme for quantum states which is the analogue of the classical one time pad. Recall the one time pad method of classical encryption: to encrypt a string m, it is XORed with a random string r. Just as l classical bits suffice to hide an l bit string m, $2l$ classical bits suffice to hide an l bit quantum state $|\psi\rangle$. This is done by using the quantum version of the one time pad, called the *Pauli one time pad* (introduced in [5]). To define this encryption scheme, we will require the X, Y and Z Pauli operators, which are defined as follows:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, Y = iXZ \tag{2.1}$$

[1] To the best of our knowledge, this scheme has not been directly described in another paper, but it is not original work: it was implicit in the protocol in [1], [3].

The Pauli operators anti commute: $ZX = -XZ$. The l-qubits Pauli group consists of all elements of the form $P = P_1 \otimes P_2 \otimes \ldots \otimes P_l$ where $P_i \in \{\mathcal{I}, X, Y, Z\}$, together with the multiplicative factors -1 and $\pm i$. We will use a subset of this group, which we denote as $\mathbb{P}_l$, which includes all operators $P = P_1 \otimes P_2 \otimes \ldots \otimes P_l$ but not the multiplicative factors.

An l qubit quantum state $|\psi\rangle$ is one time padded by choosing $z, x \in \{0,1\}^l$ at random and applying $Z^z X^x$ to $|\psi\rangle$, creating $Z^z X^x |\psi\rangle$. The bit strings z, x are called the *Pauli keys* and are retained by the party who performs the encryption. A nice property is that, in the case that $|\psi\rangle$ is a standard basis state $|m\rangle$, the quantum one time pad is the same as the classical one time pad:

$$Z^z X^x |m\rangle \langle m| X^x Z^z = |m \oplus x\rangle \langle m \oplus x| \tag{2.2}$$

The key property of the Pauli one time pad used in blind computing is the fact that it can be used hide a quantum state entirely: to a third party with no knowledge of the Pauli keys, a Pauli one time padded quantum state is equal to the maximally mixed state (the identity $\mathcal{I}$), as seen in the following lemma (which is stated formally as Lemma 5.1.1 and proven in Section 5.2.3).

Lemma 2.1.1 ***Pauli Mixing*** *For a matrix ρ on l qubits*

$$\frac{1}{2^{2l}} \sum_{z,x \in \{0,1\}^l} Z^z X^x \rho (Z^z X^x)^\dagger = \frac{1}{2^l} \mathcal{I}^{\otimes l}$$

So far, we have seen that the Pauli one time pad can be used to hide quantum states. Now we will show how to use the Pauli one time pad for a blind computing protocol. Recall that the goal is for the *client* to delegate a quantum computation (specified by a unitary V) on a quantum input state $|\psi\rangle$ to the *server*, who should learn nothing about the client's data (the state $|\psi\rangle$). We can assume without loss of generality that V is public knowledge and does not need to be hidden (as is the case with the universal circuit).

To begin, the client hides the input state $|\psi\rangle$ by using the Pauli one time pad to create $Z^z X^x |\psi\rangle$. The client sends $Z^z X^x |\psi\rangle$ to the server and stores $z, x \in \{0,1\}^l$ as his Pauli keys. We will show that for all efficiently computable circuits V, the blind computing protocol we will describe allows the following mapping (the first line corresponds to the client's classical Pauli keys and the second to the server's encrypted quantum state):

$$(z, x) \to (z', x') \tag{2.3}$$

$$Z^z X^x |\psi\rangle \to Z^{z'} X^{x'} V |\psi\rangle \tag{2.4}$$

To put it in words, at the end of the protocol, the client should hold the Pauli keys z', x' and the server should hold $Z^{z'} X^{x'} V |\psi\rangle$.

For this section, we will require a few preliminaries about quantum gates (for more detail see Section 3.3.1). First, the Clifford group $\mathfrak{C}_l$ is a finite subgroup of unitaries, with the key property that it maps the Pauli group $\mathbb{P}_l$ to itself, up to a phase $\alpha \in \{\pm 1, \pm i\}$:

$$\forall C \in \mathfrak{C}_l, P \in \mathbb{P}_l : \ \alpha C P C^\dagger \in \mathbb{P}_l$$

We will also use the fact that a universal gate set for quantum computation consists of the Clifford group in addition to the Toffoli gate T, which is the quantum multiplication gate: for $a, b, c \in \{0, 1\}$, T maps $|a, b, c\rangle$ to $|a, b, c \oplus ab\rangle$.

2.1.1 Blind Application of Pauli and Clifford Gates

To achieve our goal, we simply need to show that the transformations in (2.3) can be computed if V is a Clifford gate or a Toffoli gate. To provide intuition, we also include the case in which V is a Pauli gate $Z^a X^b$. In this case, the server does not need to do anything, and the client updates his Pauli keys from (z, x) to $(z' = z \oplus a, x' = x \oplus b)$. The client's update effectively applies $Z^a X^b$ to the server's one time padded state, since

$$Z^z X^x |\psi\rangle = Z^{z' \oplus a} X^{x' \oplus b} |\psi\rangle = Z^{z'} X^{x'} Z^a X^b |\psi\rangle \tag{2.5}$$

The equality follows up to a global phase since Pauli operators anti commute.

We continue to the case in which V is equal to a Clifford gate C. Clifford gates are applied by two parallel computations: a Pauli key update by the client and a quantum operation by the server. The server applies the gate C to his one time padded state, resulting in $CZ^z X^x |\psi\rangle$. We now take advantage of the fact that the Clifford group preserves the Pauli group by conjugation: for all z, x, there exist z', x' such that

$$CZ^z X^x |\psi\rangle = Z^{z'} X^{x'} C |\psi\rangle \tag{2.6}$$

To complete the Clifford application, the client updates his Pauli keys (z, x) to (z', x').

2.1.2 Blind Application of the Toffoli Gate

The Toffoli gate is more complicated. It cannot be applied in parallel by the client and server, as was done for Clifford gates. This is because it does not preserve Pauli operators by conjugation; applying a Toffoli directly to a one time padded state yields:

$$TZ^z X^x |\psi\rangle = T(Z^z X^x) T^\dagger T |\psi\rangle \tag{2.7}$$

The correction $T(Z^z X^x)T^\dagger$ is not a Pauli operator, as was the case for Clifford operators; it is instead a Clifford operator which is dependent on the one time pad z, x. Since the correction is a Clifford gate and not a Pauli gate, it cannot be removed by a simple Pauli key update by the client. It follows that the server must participate in removing the correction. On the other hand, we must also ensure that the server does not know z, x: recall that z, x are the Pauli keys hiding the state $|\psi\rangle$, and therefore maintaining blindness.

Due to this difficulty, the server must use a different method to apply the Toffoli gate. Luckily, there is a method called *computation by teleportation* ([29]) in which the Toffoli gate can be applied using only Clifford gates (which are independent of the Pauli keys) and measurements, at the cost of requiring an extra state called a *magic state*. We first describe how this process works, and then plug it in to our blind computation protocol.

The following text is depicted in Figure 2.1. To compute a Toffoli gate by teleportation, the following magic state is required ($|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is a maximally entangled state called a Bell pair):

$$(T \otimes \mathcal{I}) \left|\Phi^+\right\rangle^{\otimes 3} \tag{2.8}$$

The qubits of the Bell pair have been arranged so that T acts on the second qubit of each pair. The goal is to use (2.8) to apply the Toffoli gate to a 3 qubit state $|\psi\rangle$. The first step is to perform teleportation of $|\psi\rangle$, using the first half of (2.8). We will not cover teleportation ([50]) here - see lecture notes ([48]) for a simple description of this primitive. The only difference of this setting in comparison to standard teleportation is that there is a Toffoli gate applied to the second half of the Bell pair; this means that the resulting state after teleportation will also have a Toffoli gate applied. If the measurement results resulting from the teleportation procedure are $a, b \in \{0,1\}^3$, the resulting state is $TX^bZ^a |\psi\rangle$.

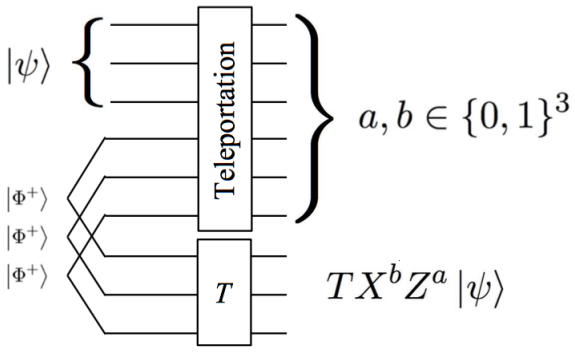

Figure 2.1: Computation by teleportation

To complete the Toffoli application, the operator $T(X^bZ^a)^\dagger T^\dagger$ is applied (this operator is a Clifford, as mentioned earlier). The result is $T |\psi\rangle$. Note that all operators applied in this process are either Clifford gates or measurements.

We now revert back to our blind computation protocol. Since the computation by teleportation process required only Clifford gates and measurements, it can be carried out on top of one time padded states as follows. First, the client one time pads the magic state from (2.8) using Pauli keys $\hat{z}_1, \hat{z}_2, \hat{x}_1, \hat{x}_2 \in \{0,1\}^3$ and sends the resulting state to the server. The server begins with the input

$$Z^z X^x |\psi\rangle \otimes (Z^{\hat{z}_1} X^{\hat{x}_1} \otimes Z^{\hat{z}_2} X^{\hat{x}_2} T) \left|\Phi^+\right\rangle^{\otimes 3} \tag{2.9}$$

Since the above state is one time padded, performing the teleportation process (which consists of a Clifford operator followed by measurement) is done as described in Section 2.1.1: the server performs the Clifford operator and measures (obtaining $a', b' \in \{0,1\}^3$) and the client updates his Pauli keys accordingly.

The measurement results a', b' come from a one time padded state; the one time pad needs to be removed in order to obtain a, b in Figure 2.1. The server needs to know a, b in order to apply the Clifford correction operator $T(X^b Z^a)^\dagger T^\dagger$. To assist him, the client sends him the one time pad needed to obtain a, b from a', b'. The server sends the client a', b', and the client and server can then jointly apply the correction operator (the client updates his Pauli keys and the server applies the correction to the state).

We now describe why the revealed one time pad does not compromise blindness. It suffices to show that the one time pad z, x is not leaked; then the state $|\psi\rangle$ remains hidden. To prove this, we will rely on the following observation: one one time pad is enough to fully hide a Bell pair; i.e., the following state is maximally mixed when randomized over $z', x' \in \{0, 1\}$:

$$(Z^{z'} X^{x'} \otimes \mathcal{I}) \left|\Phi^+\right\rangle \tag{2.10}$$

This is because of the entanglement of a Bell pair: if only one qubit of a Bell pair is considered at a time, it is maximally mixed. Randomizing one qubit of a Bell pair separates this qubit from the other (by turning it into a maximally mixed state), with the result that the other qubit also becomes maximally mixed.

This observation tells us that the two one time pads in (2.9) are redundant: one (say $\hat{z}_2, \hat{x}_2$) can be used to turn the three Bell pairs into a maximally mixed state, and the other $(\hat{z}_1, \hat{x}_1)$ is now independent randomness, entirely hidden from the server. Due to the teleportation process, the revealed one time pad is a linear combination of $z, x, \hat{z}_1, \hat{x}_1$. Since $\hat{z}_1, \hat{x}_1$ are now independently random bits, their randomness hides z, x from the server, completing the proof sketch of blindness (for a full proof, this argument is applied inductively for each magic state).

Note that this protocol can be optimized: the magic state can be reduced to 1 qubit (if we use the $\frac{\pi}{8}$ gate rather than the Toffoli gate as the non Clifford gate). We have chosen the above protocol due to its simplicity; it will be useful as we continue through this chapter.

2.1.3 Improving the Blind Computation Scheme

The key weakness of the above protocol is that the client must be quantum, and must create some number of quantum states corresponding to the size of the circuit. Although the client only requires a constant size quantum register, he is still running a polynomial size quantum computation, and it is not optimal that the client has to do so each time he wishes to delegate a quantum computation. Ideally, the client would be classical.

Chapter 5 of this thesis is dedicated to showing how to remove the need for a quantum client. To do so, we first convert the information theoretic Pauli one time pad encryption into a computationally secure encryption, as introduced in [14] and used in [20]. This is done as follows. Recall that throughout the blind computing protocol, the client holds the Pauli keys (z, x) and the server holds the encrypted state $Z^z X^x |\psi\rangle$. To convert to a computational scheme, the client simply encrypts his classical keys (z, x) (using a classical encryption scheme) and sends the encryption $\mathrm{Enc}(z, x)$ to the server at the start of the

protocol. Recall that to apply a gate V in the blind computation protocol, each party performs the following transformation (given in (2.3)):

$$(z, x) \to (z', x') \tag{2.11}$$
$$Z^z X^x |\psi\rangle \to Z^{z'} X^{x'} V |\psi\rangle \tag{2.12}$$

Since the server now holds the encrypted Pauli keys, the following transformation replaces the above expression:

$$\mathrm{Enc}(z, x) \to \mathrm{Enc}(z', x') \tag{2.13}$$
$$Z^z X^x |\psi\rangle \to Z^{z'} X^{x'} V |\psi\rangle \tag{2.14}$$

For this to be possible, the client must encode his Pauli keys using a classical encryption scheme which allows the server to apply an arbitrary classical circuit to the Pauli keys. This requirement is satisfied if the client uses a classical homomorphic encryption scheme. By definition, a homomorphic encryption scheme is a scheme which allows application of arbitrary classical circuits to the encrypted bits; i.e., for an efficiently computable circuit C, the server is able to compute the following mapping:

$$\mathrm{Enc}(x) \to \mathrm{Enc}(C(x)) \tag{2.15}$$

Next, recall the difficulty in applying the Toffoli gate. The server could apply the Toffoli gate directly to a one time padded state, but this would result in:

$$T Z^z X^x |\psi\rangle = T(Z^z X^x) T^\dagger T |\psi\rangle \tag{2.16}$$

Since the server did not know z, x, he could not apply the correction $T(Z^z X^x)T^\dagger$. In Chapter 5, we show that if the classical homomorphic encryption scheme used to encrypt the Pauli keys has certain properties, the server is able to use the encryptions of z, x to apply the operator $T(Z^z X^x)T^\dagger$, up to another Pauli correction. More precisely, if the server holds $\mathrm{Enc}(z, x)$ and a state $|\phi\rangle$, he can create

$$Z^{z'} X^{x'} T(Z^z X^x) T^\dagger |\phi\rangle \tag{2.17}$$

as well as $\mathrm{Enc}(z', x')$. The key is that the server can perform this transformation *on his own*; he needs no additional help from the client, either in the form of a magic state or in terms of classical interaction. The result is a 1 round protocol in which the client only needs to send classical messages to the server (in the case that the input state is a standard basis state[2]). If the input state is an arbitrary quantum state $|\psi\rangle$, the client only needs to send $Z^z X^x |\psi\rangle$ (and classical encryptions).

[2]Recall from (2.2) that if $|\psi\rangle$ is a standard basis state $|m\rangle$, the one time pad acts as the classical XOR operation, and the encryption of $|m\rangle$ is therefore a classical string.

2.2 Verifiable Computation from Blind Computation

In blind computation, the only concern is privacy: the server should not be able to learn anything about the computation. In verifiable computation, the concern is verifiability: the client (referred to as the verifier in this setting) should be able to check whether the server (the prover) is performing the requested computation. In this chapter, we focus on the following question: can we build a verifiable protocol from a blind protocol?

We show in this chapter that blind computing can be extended to verifiable computing, but only if the encryption scheme underlying the blind computation is *non-malleable*: it must be difficult to transform a valid encryption into another valid encryption which encrypts a different value[3]. This property may seem to be incompatible with the fact that we are running a blind computing protocol; after all, the prover must also be able to compute on the ciphertexts. The key is to combine the two properties: the prover should be able to compute, but only if instructed to do so by the verifier.

In the classical setting (described below), non-malleability is enforced by a simple check on the verifier's part and yields a verifiable scheme. In the quantum setting, this simple check is no longer possible. However, in the case that the verifier is quantum, non-malleability can be enforced and used in a different manner. Unfortunately, even these modified techniques seem unlikely to work in the case of a classical verifier delegating a quantum computation. To see this, we begin by describing the classical method and proceed to showing why these techniques break down in the quantum setting.

2.2.1 Classical Extension of Blind Computation to Verification

In this setting, the verifier would like to verifiably delegate the computation of a function $F : \{0,1\}^n \to \{0,1\}$ on an input $x \in \{0,1\}^n$ to the prover, meaning the verifier should accept iff the prover returns $F(x)$. The protocol described below was given in [16]. We assume the verifier can compute $F(0^n)$ and would like the server to compute $F(x)$, for some input $x \in \{0,1\}^n \setminus \{0^n\}$. Recall from Section 2.1.3 that a classical homomorphic encryption scheme is an encryption scheme which allows gates to be applied to the encrypted bits. The verifier encodes both 0 and x using a classical homomorphic encryption scheme and sends the encryptions to the prover in a random order. The prover is instructed to apply the function F homomorphically to each input encryption. Upon receiving the encrypted results, the verifier decrypts, checks if the result corresponding to $F(0)$ is correct, and accepts the output of $F(x)$ if so.

This protocol is not sound. The key reason is that the prover can homomorphically compute *any* function which results in the correct decrypted output for $F(0)$; there is nothing forcing him to apply the function F. This class of functions includes functions which behave as F on input 0 but compute other functions on all other inputs. More precisely, consider

[3]The standard definition of non-malleability is that it must be difficult to transform an encryption into an encryption of a related value. Our notion of non-malleability is slightly stronger and is required for the purpose of extending blind computation to verifiable computation.

the function F', which is defined as follows: $F'(0) = 0$ and for all $x \in \{0,1\} \setminus \{0^n\}$, $F'(x) = F(x) \oplus 1$.

The result of applying F' homomorphically to the encrypted inputs sent by the verifier is a correct decrypted answer in the case that the input is 0^n, and the incorrect decrypted answer in the case that the input is x. This type of cheating is frequently referred to as *under the hood cheating*: the prover is cheating by using a function which can distinguish between the two inputs, but the prover himself cannot distinguish, since he is computing homomorphically.

In the classical word, under the hood cheating can be avoided by enforcing non-malleability. As stated at the start of this section, this means that a valid encryption cannot be converted to another valid encryption which encrypts a different bit. In this setting, this means that since the server is able to compute the honest answers (by computing the function F homomorphically), he cannot compute alternate valid encryptions by computing a function other than F. If he could, this would imply that the server could break the non-malleability of the encryption scheme.

To enforce non-malleability, the verifier alters his behavior slightly. For the input 0^n, the verifier computes the final ciphertext $c = \mathrm{Enc}(F(0^n))$ that the prover should have obtained. Instead of decrypting the ciphertexts sent to him by the prover and checking if the output for $F(0^n)$ is correct, he checks if the prover's ciphertext output for this case is c. This is clearly enforcing non-malleability for the input 0^n. To see that this is also enforcing non-malleability for the input x, note that the two inputs (0^n and x) are computationally indistinguishable to the prover; he receives only their encryptions. It follows that if the prover could cheat on input x but not 0^n, he could distinguish between the two inputs, completing the soundness proof of the verification protocol.

2.2.2 Obstacles in Extending Blind Quantum Computation to Verification

In this section, we discuss the obstacles which arise in attempting to extend blind quantum computation schemes to verifiable schemes. First observe that there is a major difference between the blind quantum computing protocol in Section 2.1 and a classical homomorphic encryption scheme: the verifier in the quantum setting has more power, in the form of his role in updating the Pauli keys. For each gate applied, the verifier must do his part; therefore, if the prover tries to deviate from the protocol, the gates he applies will not have the intended effect. Is this enough control to provide non-malleability, therefore preventing under the hood attacks?

It turns out that a particularly simple cheating prover can succeed in under the hood cheating. To do so, the prover behaves almost exactly as the honest prover, except he applies a Pauli operator of his choice immediately prior to measurement (in the Toffoli teleportation process in Section 2.1.2). We show below that even such a simple deviation results in under the hood cheating. It follows that in order to ensure verifiability, we would have to ensure

non-malleability of measurement results.

Unfortunately, the same trick used in Section 2.2.1 to enforce non-malleability cannot be used with respect to the blind quantum computing protocol in Section 2.1: in this setting, it does not suffice for the verifier to simply generate the outputs of an honest prover on his own and check that the prover reports the same outputs. This is because these outputs include the measurement results (from the Toffoli teleportation process in Section 2.1.2), which consist of independent, uniformly random bits each time the protocol is performed honestly. If the verifier tried to run this process on his own to obtain the measurement results, and compared his results to the honest prover, the results would almost always differ. To get around this difficulty, we show that non-malleability can be used in a different way, but the drawback of this method is that it requires a quantum verifier. Finally, we discuss the implications of these observations to the setting in which the verifier is classical and the prover is quantum.

2.2.2.1 Under the Hood Attacks

We now show that in the protocol presented in Section 2.1, a prover can cheat under the hood by applying a Pauli operator of his choice prior to reporting measurement results (in the Toffoli teleportation process in Section 2.1.2). For convenience, assume the circuit to be applied consists of only one Toffoli gate and the 3 qubit state $|\psi\rangle$ as input. Assume the prover behaves honestly, except he applies a Pauli operator $X^{\hat{x}_1} \otimes X^{\hat{x}_2}$ ($\hat{x}_1, \hat{x}_2 \in \{0,1\}^3$) immediately prior to measurement. The prover then reports his measurement results to the verifier, who responds with the bit strings $a, b \in \{0,1\}^3$. This deviation results in the replacement of Figure 2.1 with the following figure, which illustrates that shifting the measurement results by $\hat{x}_1, \hat{x}_2$ has the same effect as shifting the correction operator applied to the state $|\psi\rangle$:

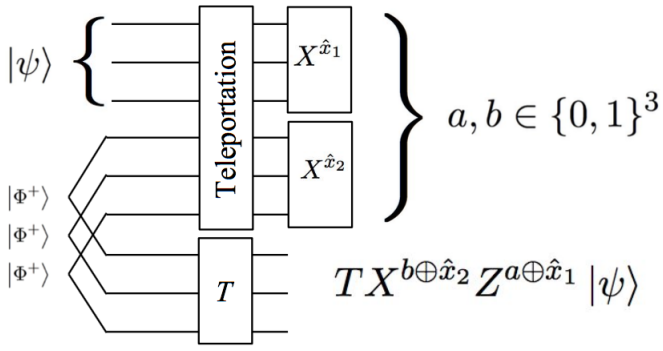

Figure 2.2: Cheating in computation by teleportation

Due to the anti commutation property of Pauli operators, the output state of Figure 2.2 is equal to:

$$TX^{b\oplus\hat{x}_2}Z^{a\oplus\hat{x}_1}|\psi\rangle = TX^bZ^aX^{\hat{x}_2}Z^{\hat{x}_1}|\psi\rangle = TX^bZ^a|\phi\rangle \tag{2.18}$$

Next, the verifier and prover jointly apply the correction operator $TX^bZ^aT^\dagger$. This operator is meant to complete the computation by teleportation process for the Toffoli operation and it does so, but with respect to the state $|\phi\rangle = X^{\hat{x}_2}Z^{\hat{x}_1}|\psi\rangle$ rather than the state $|\psi\rangle$, resulting in:

$$T|\phi\rangle = TX^{\hat{x}_2}Z^{\hat{x}_1}|\psi\rangle = TX^{\hat{x}_2}Z^{\hat{x}_1}T^\dagger T|\psi\rangle \tag{2.19}$$

The prover has succeeded in applying a Clifford attack $TX^{\hat{x}_1}Z^{\hat{x}_2}T^\dagger$ directly to the state $T|\psi\rangle$ which would have resulted from the honest behavior. This is an under the hood cheating attack, in which the server has control of the circuit applied to the unencrypted state, but does not learn anything about the computation. This attack increases in complexity as the rounds of the protocol progress; we do not know how to argue that the attack is less general than an arbitrary BQP circuit. It follows that altering the measurement results, even by just a Pauli operator, is an extremely powerful attack which we do not yet know how to handle.

2.2.2.2 Malleability in the Quantum Setting

The property of the blind quantum computing protocol in Section 2.1 which allows for under the hood cheating is that the Pauli one time pad encryption scheme is malleable: all measurement results must be accepted by the verifier, since an honest measurement result is distributed uniformly at random. The clear solution is to use a non-malleable encryption (rather than the Pauli one time pad) for all quantum states sent by the verifier to the prover.

Recall the notion of non-malleability: it must be difficult to map a valid encryption to another valid encryption.[4] If all states received by the prover are encrypted under a non-malleable encryption, the prover will have no choice but to behave as directed by the verifier. Put in another way, encrypting the input states of the prover with a non-malleable quantum encryption scheme ensures that the space of measurements resulting from the Toffoli teleportation protocol will become sparse (and unknown to the prover), therefore becoming easy for the verifier to check.

The papers [1]/[3] provide an example of the solution described above: the authors develop a verifiable protocol between a verifier (with a small quantum computer) and a quantum prover by using the quantum signed polynomial code in addition to the Pauli one time pad. To encode, the verifier first applies the signed polynomial code (the verifier must be quantum) and then applies the Pauli one time pad. Blind computation can be performed in almost the same manner as with the one time pad (the prover applies the gates on top of the encoding, and the verifier updates the keys). To prove security, the authors show that the combination of the quantum signed polynomial code and the Pauli one time pad forms a quantum authentication scheme (in other words, it is non-malleable).

[4] In the quantum setting, non-malleability is often referred to in terms of quantum authentication schemes, which achieve exactly the same guarantee.

2.2.2.3 Classical Client Blind Quantum Computation to Verification

So far, we have seen that extending blind computation to verifiable computation can be accomplished under certain conditions by relying on non-malleability. In the case of classical computations, it is possible since the outputs are deterministic, and in the case of quantum computations, it is possible by relying on a quantum verifier to perform a non-malleable quantum encoding. We now show that these conditions are not satisfied in the case of blind quantum computation with a classical client/verifier; in this setting, we do not know how to extend blind quantum computation to verifiable computation. We then briefly discuss the different path we took to develop a protocol which can classically verify quantum computations.

In the case that the verifier is classical but is delegating a quantum computation, as in the homomorphic encryption scheme provided in Chapter 5, it turns out that under the hood cheating is still possible. To understand this at a high level, we begin by recalling (from Section 2.1.3) that the scheme in Chapter 5 follows the same rough outline of the blind quantum computation protocol given in this chapter (in Section 2.1). The difference is that, in Chapter 5, the server is provided with encryptions of the Pauli keys and can use these to perform the Toffoli correction on his own; he does not require the assistance of a magic state. However, as we will see in Chapter 5, the server's procedure to apply the Toffoli correction still produces measurement results drawn uniformly at random, and under the hood cheating is again possible by deviating on these measurement results.

In this setting, the methods discussed above which rely on non-malleability are no longer viable solutions to avoid under the hood cheating. The method outlined in Section 2.2.1, in which the verifier computed the output of an honest prover, cannot be used for two reasons. First, the verifier is classical, and therefore cannot compute the output of an honest prover. Second, even if the verifier was quantum, the same issue stated at the start of Section 2.2.2 exists: the outputs of an honest prover are distributed uniformly at random, and therefore cannot be predicted by the verifier.

The method of using non-malleable quantum encryption schemes outlined in Section 2.2.2.2, in the case of a quantum verifier, also cannot be used. For such schemes to serve their purpose, it seems crucial that the verifier performs the encoding, and not the prover. It may seem that the verifier could instead delegate the application of the encoding to the prover (by using blind computing). However, in this case the verifier would have to be assured that the encoding is applied correctly by the prover in order to use the non-malleability guarantees later on. This is difficult for the same reason that it is difficult to verify a blind quantum computation without using a non-malleable quantum encryption scheme.

Enforcing the Existence of Qubits There seems to be an inherent reason why the previously described techniques are difficult to extend to the setting of a classical verifier/quantum prover. This is because the first step of non-malleability (and in turn, verification) is to ensure that the prover is respecting some type of qubit structure; when the prover is asked to apply different operations on the same qubit, he must actually be applying these operations

to the same qubit, rather than applying each operation to a different part of his space. In this sense, verification must certify that the prover has a well defined space of qubits and is using this space as requested by the verifier.

In previous verification protocols, the structure of the prover's space was enforced in one of two ways. In the case that the verifier has a small quantum computer, as discussed in Section 2.2.2.2, he can simply send encoded qubits to the prover. The non-malleable encoding ensures that the prover is in fact applying the operations requested to the specified qubits. If the verifier is classical, but has access to two quantum computers, the task of enforcing qubit structure is much more involved, but still possible ([44]). In order to do so, the verifier plays a game (such as the CHSH game [17]) with the two provers, in which quantum provers have a distinct advantage over classical provers. If the provers win the game with the optimal advantage, it is possible to fully characterize the space of the two provers: it can be shown that such provers share Bell pairs (maximally entangled states) and are performing the operations requested by the verifier on these Bell pairs. This characterization of the space of the provers allows the verifier to force the provers to carry out a computation.

In the setting of a classical verifier and a single quantum prover, we do not have the option of using previous techniques. We rely on quantum secure cryptography in order to develop an entirely new method in which a classical verifier can enforce that the quantum prover holds qubits and is behaving as requested on them. In more detail, we construct a classical encryption scheme which allows blind delegation of standard/Hadamard basis measurements: given an encryption of the bit 0 (resp. 1), the prover can apply a standard (resp. Hadamard) basis measurement to a state of his choice, without learning which measurement he is applying. We show that our encryption scheme can be used to enforce the existence of a well defined qubit on which the prover performs the measurement (either standard or Hadamard) requested by the verifier.

This proof of existence rests on the fact that the prover does not fully understand the cryptography used by the verifier, and any attempt at deviating from the protocol will be randomized in a way which is unknown to him, rendering such deviations ineffective. The randomization is derived from particularly strong security guarantees of our classical encryption scheme. Note that the existence of a well defined qubit rules out under the hood cheating: the prover must measure the same qubit, regardless of whether he is directed to perform a standard or Hadamard basis measurement.

In conclusion, we were not able to extend our homomorphic encryption protocol (Chapter 5) to a verifiable protocol. Instead, we build our verification protocol (Chapter 6) by developing a new classical encryption scheme with strong security guarantees. We leave as a (rather difficult) open question whether blind computation can be converted to verifiable computation in the setting of a classical verifier/ quantum prover.

Chapter 3

Preliminaries

Throughout this thesis, we borrow notation and definitions from [1], [3] and [11]. Parts of the following sections are also taken from these sources.

3.1 Notation

For all $q \in \mathbb{N}$ we let $\mathbb{Z}_q$ denote the ring of integers modulo q. We represent elements in $\mathbb{Z}_q$ using numbers in the range $(-\frac{q}{2}, \frac{q}{2}] \cap \mathbb{Z}$. We denote by $[x]_q$ the unique integer y s.t. $y = x \pmod q$ and $y \in (-\frac{q}{2}, \frac{q}{2}]$. For $x \in \mathbb{Z}_q$ we define $|x| = |[x]_q|$. For a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we write $\|\mathbf{u}\|_\infty \leq \beta$ if each entry u_i in $\mathbf{u}$ satisfies $|u_i| \leq \beta$. Similarly, for a matrix $U \in \mathbb{Z}_q^{n \times m}$, we write $\|U\|_\infty \leq \beta$ if each entry $u_{i,j}$ in U satisfies $|u_{i,j}| \leq \beta$. When considering an $s \in \{0,1\}^n$ we sometimes also think of s as an element of $\mathbb{Z}_q^n$, in which case we write it as $\mathbf{s}$.

We use the terminology of polynomially bounded, super-polynomial, and negligible functions. A function $n : \mathbb{N} \to \mathbb{R}_+$ is *polynomially bounded* if there exists a polynomial p such that $n(\lambda) \leq p(\lambda)$ for all $\lambda \in \mathbb{N}$. A function $n : \mathbb{N} \to \mathbb{R}_+$ is *negligible* (resp. *super-polynomial*) if for every polynomial p, $p(\lambda)n(\lambda) \to_{\lambda \to \infty} 0$ (resp. $n(\lambda)/p(\lambda) \to_{\lambda \to \infty} \infty$).

We generally use the letter D to denote a distribution over a finite domain X, and f for a density on X, i.e. a function $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$. We often use the distribution and its density interchangeably. We write U for the uniform distribution. We write $x \leftarrow D$ to indicate that x is sampled from distribution D, and $x \leftarrow_U X$ to indicate that x is sampled uniformly from the set X. We write $\mathcal{D}_X$ for the set of all densities on X. For any $f \in \mathcal{D}_X$, $\textsc{Supp}(f)$ denotes the support of f,

$$\textsc{Supp}(f) \;=\; \big\{x \in X \mid f(x) > 0\big\} \;.$$

For two densities f_1 and f_2 over the same finite domain X, the Hellinger distance between f_1 and f_2 is

$$H^2(f_1, f_2) \;=\; 1 - \sum_{x \in X} \sqrt{f_1(x) f_2(x)} \;. \tag{3.1}$$

and the total variation distance between f_1 and f_2 is:

$$\|f_1 - f_2\|_{TV} = \frac{1}{2}\sum_{x\in X} |f_1(x) - f_2(x)| . \tag{3.2}$$

The following lemma will be useful:

Lemma 3.1.1 *Let D_0, D_1 be distributions over a finite domain X. Let $X' \subseteq X$. Then:*

$$\left| \Pr_{x\leftarrow D_0}[x \in X'] - \Pr_{x\leftarrow D_1}[x \in X'] \right| \leq \|D_0 - D_1\|_{TV} \tag{3.3}$$

We require the following definition:

Definition 3.1.2 ***Computational Indistinguishability of Distributions*** *Two families of distributions $\{D_{0,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\{D_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ (indexed by the security parameter λ) are computationally indistinguishable if for all quantum polynomial-time attackers $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$*

$$\left| \Pr_{x\leftarrow D_{0,\lambda}}[\mathcal{A}(x) = 0] - \Pr_{x\leftarrow D_{1,\lambda}}[\mathcal{A}(x) = 0] \right| \leq \mu(\lambda) . \tag{3.4}$$

3.2 Learning with Errors and Discrete Gaussians

This background section on the learning with errors problem is taken directly from [11]. For a positive real B and positive integers q, the truncated discrete Gaussian distribution over $\mathbb{Z}_q$ with parameter B is supported on $\{x \in \mathbb{Z}_q : \|x\| \leq B\}$ and has density

$$D_{\mathbb{Z}_q,B}(x) = \frac{e^{\frac{-\pi\|x\|^2}{B^2}}}{\sum\limits_{x\in\mathbb{Z}_q,\, \|x\|\leq B} e^{\frac{-\pi\|x\|^2}{B^2}}} . \tag{3.5}$$

We note that for any $B > 0$, the truncated and non-truncated distributions have statistical distance that is exponentially small in B [6, Lemma 1.5]. For a positive integer m, the truncated discrete Gaussian distribution over $\mathbb{Z}_q^m$ with parameter B is supported on $\{x \in \mathbb{Z}_q^m : \|x\| \leq B\sqrt{m}\}$ and has density

$$\forall x = (x_1, \ldots, x_m) \in \mathbb{Z}_q^m , \qquad D_{\mathbb{Z}_q^m,B}(x) = D_{\mathbb{Z}_q,B}(x_1)\cdots D_{\mathbb{Z}_q,B}(x_m) . \tag{3.6}$$

Lemma 3.2.1 *Let B be a positive real number and q, m be positive integers. Let $\mathbf{e} \in \mathbb{Z}_q^m$. The Hellinger distance between the distribution $D = D_{\mathbb{Z}_q^m,B}$ and the shifted distribution $D+\mathbf{e}$ satisfies*

$$H^2(D, D+\mathbf{e}) \leq 1 - e^{\frac{-2\pi\sqrt{m}\|\mathbf{e}\|}{B}} , \tag{3.7}$$

and the statistical distance between the two distributions satisfies

$$\left\|D - (D+\mathbf{e})\right\|_{TV}^2 \leq 2\left(1 - e^{\frac{-2\pi\sqrt{m}\|\mathbf{e}\|}{B}}\right) . \tag{3.8}$$

Definition 3.2.2 *For a security parameter λ, let $n, m, q \in \mathbb{N}$ be integer functions of λ. Let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The* $\mathrm{LWE}_{n,m,q,\chi}$ *problem is to distinguish between the distributions $(\mathbf{A}, \mathbf{As}+\mathbf{e} \pmod q)$ and $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A}$ is uniformly random in $\mathbb{Z}_q^{n\times m}$, $\mathbf{s}$ is a uniformly random row vector in $\mathbb{Z}_q^n$, $\mathbf{e}$ is a row vector drawn at random from the distribution χ^m, and $\mathbf{u}$ is a uniformly random vector in $\mathbb{Z}_q^m$. Often we consider the hardness of solving* LWE *for any function m such that m is at most a polynomial in $n \log q$. This problem is denoted* $\mathrm{LWE}_{n,q,\chi}$*. When we write that we make the* $\mathrm{LWE}_{n,q,\chi}$ *assumption, our assumption is that no quantum polynomial-time procedure can solve the* $\mathrm{LWE}_{n,q,\chi}$ *problem with more than a negligible advantage in λ.*

As shown in [43, 42], for any $\alpha > 0$ such that $\sigma = \alpha q \geq 2\sqrt{n}$ the $\mathrm{LWE}_{n,q,D_{\mathbb{Z}_q,\sigma}}$ problem, where $D_{\mathbb{Z}_q,\sigma}$ is the discrete Gaussian distribution, is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \widetilde{O}(n/\alpha)$ in *worst case* dimension n lattices. This is proven using a quantum reduction. Classical reductions (to a slightly different problem) exist as well [41, 12] but with somewhat worse parameters. The best known (classical or quantum) algorithm for these problems run in time $2^{\widetilde{O}(n/\log\gamma)}$. For our construction we assume hardness of the problem against a quantum polynomial-time adversary in the case that γ is a super polynomial function in n. This is a commonly used assumption in cryptography (for e.g. homomorphic encryption schemes such as [24]).

We use two additional properties of the LWE problem. The first is that it is possible to generate LWE samples $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ such that there is a trapdoor allowing recovery of $\mathbf{s}$ from the samples.

Theorem 3.2.3 (Theorem 5.1 in [35]) *Let $n, m \geq 1$ and $q \geq 2$ be such that $m = \Omega(n \log q)$. There is an efficient randomized algorithm* $\textsc{GenTrap}(1^n, 1^m, q)$ *that returns a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and a trapdoor $t_{\mathbf{A}}$ such that the distribution of $\mathbf{A}$ is negligibly (in n) close to the uniform distribution. Moreover, there is an efficient algorithm* Invert *that, on input $\mathbf{A}, t_{\mathbf{A}}$ and $\mathbf{As}+\mathbf{e}$ where $\|\mathbf{e}\| \leq q/(C_T\sqrt{n \log q})$ and C_T is a universal constant, returns $\mathbf{s}$ and $\mathbf{e}$ with overwhelming probability over $(\mathbf{A}, t_{\mathbf{A}}) \leftarrow$* GenTrap*.*

The second property is the existence of a "lossy mode" for LWE. The following definition is Definition 3.1 in [4].

Definition 3.2.4 *Let $\chi = \chi(\lambda)$ be an efficiently sampleable distribution over $\mathbb{Z}_q$. Define a lossy sampler $\tilde{\mathbf{A}} \leftarrow \textsc{lossy}(1^n, 1^m, 1^\ell, q, \chi)$ by $\tilde{\mathbf{A}} = \mathbf{BC}+\mathbf{F}$, where $\mathbf{B} \leftarrow_U \mathbb{Z}_q^{m\times\ell}$, $\mathbf{C} \leftarrow_U \mathbb{Z}_q^{\ell\times n}$, $\mathbf{F} \leftarrow \chi^{m\times n}$.*

Theorem 3.2.5 (Lemma 3.2 in [4]) *Under the* $\mathrm{LWE}_{\ell,q,\chi}$ *assumption, the distribution of $\tilde{\mathbf{A}} \leftarrow \textsc{lossy}(1^n, 1^m, 1^\ell, q, \chi)$ is computationally indistinguishable from $\mathbf{A} \leftarrow_U \mathbb{Z}_q^{m\times n}$.*

3.3 Quantum Computation Preliminaries

3.3.1 Quantum Operations

We will use the X, Y and Z Pauli operators: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $Y = iXZ$. The l-qubits Pauli group consists of all elements of the form $P = P_1 \otimes P_2 \otimes \ldots \otimes P_l$ where $P_i \in \{\mathcal{I}, X, Y, Z\}$, together with the multiplicative factors -1 and $\pm i$. We will use a subset of this group, which we denote as $\mathbb{P}_l$, which includes all operators $P = P_1 \otimes P_2 \otimes \ldots \otimes P_l$ but not the multiplicative factors. We will use the fact that Pauli operators anti commute; $ZX = -XZ$. The Pauli group $\mathbb{P}_l$ is a basis to the matrices acting on l qubits. We can write any matrix U over a vector space $A \otimes B$ (where A is the space of l qubits) as $\sum_{P \in \mathbb{P}_l} P \otimes U_P$ where U_P is some (not necessarily unitary) matrix on B.

Let $\mathfrak{C}_l$ denote the l-qubit Clifford group. Recall that it is a finite subgroup of unitaries acting on l qubits generated by the Hadamard matrix $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, by $K = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$, and by controlled-NOT (CNOT) which maps $|a, b\rangle$ to $|a, a \oplus b\rangle$ (for bits a, b). The Clifford group is characterized by the property that it maps the Pauli group $\mathbb{P}_l$ to itself, up to a phase $\alpha \in \{\pm 1, \pm i\}$. That is: $\forall C \in \mathfrak{C}_l, P \in \mathbb{P}_l : \ \alpha CPC^\dagger \in \mathbb{P}_l$

The Toffoli gate T maps $|a, b, c\rangle$ to $|a, b, c \oplus ab\rangle$ (for $a, b, c \in \{0, 1\}$). We will use the fact that the set consisting of the Toffoli gate and the Hadamard gate is a universal gate set for quantum circuits ([46]).

We will use completely positive trace preserving (CPTP) maps to represent general quantum operations. A CPTP map $\mathcal{S}$ can be represented by its Kraus operators, $\{B_\tau\}_\tau$. The result of applying $\mathcal{S}$ to a state ρ is:

$$\mathcal{S}(\rho) = \sum_\tau B_\tau \rho B_\tau^\dagger \tag{3.9}$$

We say that two CPTP maps $\mathcal{S}$ and $\mathcal{S}'$ are equal if, for all density matrices ρ, $\mathcal{S}(\rho) = \mathcal{S}'(\rho)$.

3.3.2 Trace Distance

For density matrices ρ, σ, the trace distance $\|\rho - \sigma\|_{tr}$ is equal to:

$$\|\rho - \sigma\|_{tr} = \frac{1}{2} \mathrm{Tr}(\sqrt{(\rho - \sigma)^2}) \tag{3.10}$$

We will use the following fact ([51]):

$$\|\rho - \sigma\|_{tr} = \max_P \mathrm{Tr}(P(\rho - \sigma)) \tag{3.11}$$

where the maximization is carried over all projectors P. We will also use the fact that the trace distance is contractive under completely positive trace preserving maps ([51]). The following lemma relates the Hellinger distance as given in (3.1) and the trace distance of superpositions:

Lemma 3.3.1 *Let X be a finite set and $f_1, f_2 \in \mathcal{D}_x$. Let*

$$|\psi_1\rangle = \sum_{x\in X} \sqrt{f_1(x)}\,|x\rangle \qquad \text{and} \qquad |\psi_2\rangle = \sum_{x\in X} \sqrt{f_2(x)}\,|x\rangle \ .$$

Then

$$\| |\psi_1\rangle\langle\psi_1| - |\psi_2\rangle\langle\psi_2| \|_{tr} \ = \ \sqrt{1-(1-H^2(f_1,f_2))^2}\ .$$

We require the following definition, which is analogous to Definition 3.1.2:

Definition 3.3.2 ***Computational Indistinguishability of Quantum States*** *Two families of density matrices $\{\rho_{0,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\{\rho_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ (indexed by the security parameter λ) are computationally indistinguishable if for all efficiently computable CPTP maps $\mathcal{S}$ there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$:*

$$\Big| Tr((|0\rangle\langle 0| \otimes \mathcal{I})\mathcal{S}(\rho_{0,\lambda} - \rho_{1,\lambda}) \Big| \ \leq \ \mu(\lambda)\ . \tag{3.12}$$

Chapter 4

Trapdoor Claw-Free Functions in Quantum Computation

In this chapter, we introduce the primitive of trapdoor claw-free functions ([26],[25]), which will be used in both results in this thesis. This primitive was first used in the context of quantum computations in [11]. Here we describe why it is useful in the quantum setting (as shown in [11]), and roughly how it will be used in Chapter 5 and Chapter 6.

Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets (assume that $\mathcal{X} = \{0,1\}^w$). A trapdoor claw-free function $f : \mathcal{X} \to \mathcal{Y}$ is a two to one function with two more properties. First, it has a trapdoor which allows for efficient inversion: given a point y in the image of f, the trapdoor allows recovery of the two preimages x_0, x_1 of y. Second, without the trapdoor it is computationally difficult to find a pair of overlapping preimages x_0, x_1 ($f(x_0) = f(x_1)$); the pair (x_0, x_1) is referred to as a *claw* (hence the name claw-free).

4.1 Quantum Advantage

As introduced in [11], the key reason trapdoor claw-free functions are useful in the quantum setting is that a quantum algorithm can generate a superposition over a random claw:

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle) \tag{4.1}$$

and use this superposition to generate a string $d \in \mathcal{X} \setminus \{0\}$ such that $d \cdot (x_0 \oplus x_1) = 0$. This task is conjectured to be hard classically, and was used for randomness generation in [11]. We now show how this can be done.

First, the quantum machine creates a uniform superposition over the domain $\mathcal{X} = \{0,1\}^w$ and applies the function f in superposition:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x \in \mathcal{X}} |x\rangle\,|f(x)\rangle \tag{4.2}$$

The final register is measured, resulting in a point $y \in \mathcal{Y}$. The remaining state is a uniform superposition over the claw (x_0, x_1) $(f(x_0) = f(x_1) = y)$:

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle) \tag{4.3}$$

Now, the Hadamard transform is applied to the state in (4.3), resulting in:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{d \in \mathcal{X}} ((-1)^{d \cdot x_0} + (-1)^{d \cdot x_1}) \, |d\rangle \tag{4.4}$$

The amplitude of a string d is non zero only when $d \cdot (x_0 \oplus x_1) = 0$. Therefore, when the above state is measured, only such d are obtained.

Now assume a classical device (the verifier) wants to challenge a quantum device (the prover). This can be done in two ways using the procedure above. For both challenges, the verifier begins by sampling a trapdoor claw-free function f along with its trapdoor. The verifier sends the prover f and asks the prover to create a superposition over a claw (as in (4.3)) and send him y as the receipt. Note that at this point, the verifier has leverage over the prover: the verifier can compute both inverses x_0, x_1 of y (using the trapdoor of f), but it is computationally difficult for the prover to compute this information.

For the first challenge, the verifier asks the prover to return a preimage of y. If the prover did as he was told, this is easy: he simply measures (4.3). For the second challenge, the verifier asks the prover to return a string $d \neq 0$ such that $d \cdot (x_0 \oplus x_1) = 0$. Again, if the prover behaved honestly, this is easy - he can measure (4.4) to obtain such a d.

In [11], these challenges are combined: the idea is that if the prover is able to pass both challenges, he must be acting probabilistically (and therefore generating randomness). To show this, the authors rely on a strengthened version of the claw-free property (called a hardcore bit property) which states that it is computationally difficult for the prover to compute both one member of a claw (x_0, x_1) and a string $d \neq 0$ such that $d \cdot (x_0 \oplus x_1) = 0$. This implies that a deterministic prover cannot pass both challenges at once.

4.2 Applications in Homomorphic Encryption and Verification

In this thesis, we require trapdoor claw-free functions with a bit more structure: we will instead consider a pair of injective functions f_0, f_1 with the same images. The trapdoor inversion algorithm takes as input a point $y \in f_0(\cdot)$ and a bit b and returns $f_b^{-1}(y)$. The reason this structure is useful is because it allows a quantum machine to entangle a specific state with a claw; the machine can begin with an arbitrary single qubit state:

$$|\psi\rangle = \alpha_0 \, |0\rangle + \alpha_1 \, |1\rangle \tag{4.5}$$

and, for a uniformly random claw (x_0, x_1) $(f_0(x_0) = f_1(x_1))$, create

$$\alpha_0 |0\rangle |x_0\rangle + \alpha_1 |1\rangle |x_1\rangle \tag{4.6}$$

To do this, the quantum machine first creates a uniform superposition over the domain $\mathcal{X}$ and then uses $|\psi\rangle$ to determine which of the two functions (f_0, f_1) to apply:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x \in \mathcal{X}} \sum_{b \in \{0,1\}} \alpha_b |b\rangle |x\rangle |f_b(x)\rangle \tag{4.7}$$

To complete the process, the last register is measured.

In Chapter 5, we show how to build an approximate trapdoor claw-free function pair given a classical encryption of a bit s (if the encryption scheme satisfies certain properties). The resulting function pair f_0, f_1 has the special property that for all claws (x_0, x_1), the first bit of $x_0 \oplus x_1$ is equal to s. As we will show in Chapter 5, this property is extremely useful in blind quantum computation.

In Chapter 6, we use an extension of the trapdoor claw-free family built from learning with errors in [11]. The trapdoor claw-free functions are not used for functionality, as in Chapter 5, but for constraining the behavior of the prover in order to force the prover to behave as directed by the verifier. To do this, we will need to strengthen the claw-free property in two different ways. First, we require the hardcore bit property used in [11] and mentioned above, which states that it is computationally difficult to hold both one member of a claw (x_0, x_1) and a string $d \in \mathcal{X} \setminus \{0\}$ such that $d \cdot (x_0 \oplus x_1) = 0$. We also need a variant of this statement, which states that there exists a string $d \in \mathcal{X}$ which satisfies two conditions. First, $d \cdot (x_0 \oplus x_1)$ is equal to the same fixed bit c for all claws (x_0, x_1). Second, it is computationally difficult to guess the bit c with non negligible advantage.

Chapter 5

Homomorphic Encryption

In this chapter, we present a leveled fully homomorphic encryption scheme for quantum circuits with classical keys. The scheme allows a classical client to blindly delegate a quantum computation to a quantum server: an honest server is able to run the computation while a malicious server is unable to learn any information about the computation.

Our scheme follows the outline of the Pauli one time pad blind computation scheme presented in Section 2.1. The crux of this scheme was the blind application of the Toffoli gate (Section 2.1.2), which was accomplished using a quantum state, thereby requiring a quantum client. In this chapter, we remove the need for a quantum client by showing that a classical encryption, rather than a quantum state, suffices to perform the same task. To do so, we show that the server is able to convert a classical encryption of a bit s, denoted as $\mathrm{Enc}(s)$, into the quantum operation CNOT^s (i.e. the CNOT gate is applied if and only if $s = 1$). In more detail, we construct an *encrypted CNOT operation*, a quantum operation which takes as input $\mathrm{Enc}(s)$ and a 2 qubit state $|\psi\rangle$ and outputs the state $Z^z X^x \mathrm{CNOT}^s |\psi\rangle$ as well as classical encryptions of $z, x \in \{0,1\}^2$.[1]

We begin by showing that if $\mathrm{Enc}(s)$ has a specific structure, the encrypted CNOT operation can be constructed in a straightforward manner. This structure is as follows: $\mathrm{Enc}(s)$ is equal to a trapdoor claw-free function pair f_0, f_1 with the special property that each claw (x_0, x_1) hides s (the first bit of $x_0 \oplus x_1$ is equal to s). If this structure exists, the operation CNOT^s can be applied by relying on the fact that the server can efficiently create a superposition over a uniformly random claw x_0, x_1, as described in Chapter 4.

To obtain a homomorphic encryption scheme for quantum circuits, the above idea must be generalized: we must show how to perform the encrypted CNOT operation if $\mathrm{Enc}(s)$ is a ciphertext of a classical homomorphic encryption scheme which may not have the trapdoor claw-free structure described in the previous paragraph. This is less straightforward, and we do so by showing that if the classical encryption scheme satisfies certain properties, the function pair f_0, f_1 can be constructed given $\mathrm{Enc}(s)$. We call such classical encryption schemes *quantum capable*. The two main results of this chapter are that a quantum capable

[1] Each application of the encrypted CNOT operation results in $z, x \in \{0,1\}^2$ sampled uniformly at random.

scheme can be constructed from the learning with errors problem by combining two existing classical encryption schemes (Theorem 5.4.2) and that quantum capable schemes can be used for homomorphic evaluation of quantum circuits (Theorem 5.5.2). Combined, they provide the following theorem (taken from Chapter 1):

Theorem 1.0.1 (Informal) *Under the assumption that the learning with errors problem with superpolynomial noise ratio is computationally intractable for an efficient quantum machine, there exists a quantum leveled fully homomorphic encryption scheme with classical keys.*

5.1 Overview

We now present an overview of the chapter, which proceeds as follows. We first recap the blind computation scheme presented in Section 2.1, with the goal of reducing it to the encrypted CNOT operation described above. We then describe how the encrypted CNOT operation works, in the case that $\text{Enc}(s)$ is equal to a trapdoor claw-free function pair f_0, f_1 which hides the encrypted bit s. Next, we show how a classical encryption of a bit s can be used to build f_0, f_1 if the encryption scheme has certain properties and we use these properties to describe how such quantum capable homomorphic encryption schemes are defined. We conclude by describing how to combine two existing classical homomorphic encryption schemes ([23], [24]) to form a quantum capable scheme, and then showing how a quantum capable scheme can be used to build a quantum leveled fully homomorphic encryption scheme with classical keys. The chapter itself follows the outline of this overview and provides full proofs of all of our results.

5.1.1 Reduction to the Encrypted CNOT Operation

Recall the Pauli one time pad, which is the analogue of the classical one time pad. The classical one time pad encrypts a string m by XORing it with a random string r. Just as l classical bits suffice to hide an l bit string m, $2l$ classical bits suffice to hide an l bit quantum state $|\psi\rangle$. This is done by using the quantum version of the one time pad, called the *Pauli one time pad* (introduced in [5]).

An l qubit quantum state $|\psi\rangle$ is Pauli one time padded by choosing $z, x \in \{0,1\}^l$ at random and applying $Z^z X^x$ to $|\psi\rangle$, creating $Z^z X^x |\psi\rangle$. The bit strings z, x are called the *Pauli keys* and are retained by the client. Once the client sends the encrypted state to the server, the shared state held by the client and server is (the last register containing zx is held by the client):

$$\frac{1}{2^{2l}} \sum_{z,x \in \{0,1\}^l} Z^z X^x |\psi\rangle \langle\psi| (Z^z X^x)^\dagger \otimes |zx\rangle \langle zx| \tag{5.1}$$

The client's decoding process is simple: he simply uses the keys z, x to apply the Pauli operator $(Z^z X^x)^\dagger$ to the state he receives from the server. A nice property is that, in the case that $|\psi\rangle$ is a standard basis state $|m\rangle$, the quantum one time pad is the same as the classical one time pad:

$$Z^z X^x |m\rangle \langle m| X^x Z^z = |m \oplus x\rangle \langle m \oplus x| \tag{5.2}$$

It follows that the encoding and decoding of a standard basis state can be performed classically.

The key property of the Pauli one time pad used in blind computing is the fact that it can be used hide a quantum state entirely: to the server, who has no knowledge of the Pauli keys, a Pauli one time padded quantum state is equal to the maximally mixed state (the identity $\mathcal{I}$), as stated in the following lemma (which was stated informally as Lemma 2.1.1 and is proven in Section 5.2.3).

Lemma 5.1.1 (Pauli Mixing) *For a matrix ρ on two spaces A, B*

$$\frac{1}{2^{2l}} \sum_{z,x \in \{0,1\}^l} (Z^z X^x \otimes \mathcal{I}_B)\rho(Z^z X^x \otimes \mathcal{I}_B)^\dagger = \frac{1}{2^l} \mathcal{I}_A \otimes Tr_A(\rho)$$

As described in Section 2.1.3, the information theoretically secure Pauli one time pad encryption scheme can be easily transformed into a computationally secure encryption scheme ([14]). To do so, the client simply encrypts his classical Pauli keys (z, x) (using a classical homomorphic encryption scheme) and includes the encryption $\text{Enc}(z, x)$ as part of the encryption of the state $|\psi\rangle$; the encryption is now a two part encryption, containing classically encrypted keys and the Pauli one time padded quantum state $Z^z X^x |\psi\rangle$. To decode, the client requests both the Pauli key encryptions and the quantum state. He first decrypts the Pauli key encryptions to obtain the Pauli keys, and then applies the inverse of the Pauli keys to the quantum state.

5.1.1.1 Homomorphic Gate Application

In order to apply quantum gates homomorphically to the computationally secure Pauli encryption scheme, the server will need to run two separate computations: a classical homomorphic computation on the Pauli keys and a quantum computation on the one time padded state. In this section, we show how the server is able to perform the following transformations for all gates V in a universal set of gates (our universal set will be the Clifford group along with the Toffoli gate):

$$\text{Enc}(z, x) \to \text{Enc}(z', x') \tag{5.3}$$

$$Z^z X^x |\psi\rangle \to Z^{z'} X^{x'} V |\psi\rangle \tag{5.4}$$

Homomorphic Application of Pauli and Clifford Gates To achieve our goal, we simply need to show that the transformations in (5.3)/(5.4) can be computed if V is a Clifford gate or a Toffoli gate. To provide intuition, we also include the case in which V is a Pauli gate $Z^a X^b$. In this case, the server only performs the classical part of the parallel computation (in (5.3)): he homomorphically updates his Pauli keys from $\mathrm{Enc}(z, x)$ to $\mathrm{Enc}(z', x')$, for $(z' = z \oplus a, x' = x \oplus b)$. The server has now effectively applied $Z^a X^b$ to his one time padded state, since

$$Z^z X^x |\psi\rangle = Z^{z' \oplus a} X^{x' \oplus b} |\psi\rangle = Z^{z'} X^{x'} Z^a X^b |\psi\rangle \tag{5.5}$$

The equality follows up to a global phase since Pauli operators anti commute.

We continue to the case in which V is equal to a Clifford gate C. Clifford gates are applied by two parallel computations: a homomorphic Pauli key update and a quantum operation on the one time padded state. The server applies the gate C to his one time padded state, resulting in $CZ^z X^x |\psi\rangle$. We now take advantage of the fact that the Clifford group preserves the Pauli group by conjugation: for all z, x, there exist z', x' such that

$$CZ^z X^x |\psi\rangle = Z^{z'} X^{x'} C |\psi\rangle \tag{5.6}$$

To complete the Clifford application, the server homomorphically updates his Pauli keys from $\mathrm{Enc}(z, x)$ to $\mathrm{Enc}(z', x')$.

Homomorphic Application of the Toffoli Gate The Toffoli gate is more complicated. It cannot be applied by parallel quantum/classical operations by the server, as was done for Clifford gates. This is because it does not preserve Pauli operators by conjugation; applying a Toffoli directly to a 3 qubit one time padded state yields:

$$TZ^z X^x |\psi\rangle = T(Z^z X^x) T^\dagger T |\psi\rangle \tag{5.7}$$

The correction $T(Z^z X^x)T^\dagger$ is not a Pauli operator, as was the case for Clifford operators; it is instead a product of Pauli and Clifford operators, where the Clifford operator involves Hadamard gates and gates of the form $\mathrm{CNOT}^{b_{zx}}$ (b_{zx} is a bit which depends on the Pauli keys z, x). For the exact form of this operator, see Section 5.2.2. Since the correction is a Clifford gate and not a Pauli gate, it cannot be removed by a simple homomorphic Pauli key update by the server.

In order to complete the application of the Toffoli gate, the server will need to remove the operators $\mathrm{CNOT}^{b_{zx}}$ up to Pauli operators. Since the server holds the encrypted Pauli keys, we can assume the server can compute an encryption of b_{zx}. Therefore, we have reduced the question of applying a Toffoli gate on top of a one time padded state to the following question: can a BQP server use a ciphertext c encrypting a bit s to apply CNOT^s to a quantum state (up to Pauli operators)? In our setting specifically, s will be a function of the Pauli keys of the one time padded state.

5.1.2 Encrypted CNOT Operation

We now present the key idea in this chapter: we show how a BQP server can apply CNOT^s if he holds a ciphertext c encrypting a bit s. We call this procedure an *encrypted CNOT operation*. We first show how to perform this operation in an ideal scenario in which the ciphertext c is a trapdoor claw-free function pair hiding the bit s. We then generalize to the case in which c is a ciphertext from a classical homomorphic encryption scheme which satisfies certain properties, which we will describe as they are used.

In our ideal scenario, there exists finite sets $\mathcal{R}, \mathcal{Y}$ and the ciphertext c is equal to a trapdoor claw-free function pair $f_0, f_1 : \{0,1\} \times \mathcal{R} \to \mathcal{Y}$ with one additional property. As a reminder of trapdoor claw-free function pairs (introduced in Section 4.2), both f_0, f_1 are injective and their images are equal. There also exists a trapdoor which allows inversion of both functions. In this setting, we have introduced an extra bit in the domain; this bit of the preimage will be used to hide the bit s. The property we require is as follows: for all $\mu_0, \mu_1 \in \{0,1\}$ and $r_0, r_1 \in \mathcal{R}$ for which $f_0(\mu_0, r_0) = f_1(\mu_1, r_1)$, $\mu_0 \oplus \mu_1 = s$ (s is the value encrypted in the ciphertext c).

Our encrypted CNOT operation boils down to the ability to extract an encrypted bit from a classical encryption and instead store it in superposition in a quantum state. The claw-free function pair f_0, f_1 described above serves as a classical encryption for the bit s which immediately allows this task. This is because, as described in Chapter 4 (we give a reminder of how this is done in the next paragraph), it is possible for the server to compute the following superposition, for a random claw $(\mu_0, r_0), (\mu_1, r_1)$:

$$\frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} |\mu_b\rangle |r_b\rangle \tag{5.8}$$

Since $\mu_0 \oplus \mu_1 = s$, the above state can be written as:

$$(X^{\mu_0} \otimes \mathcal{I}) \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} |b \cdot s\rangle |r_b\rangle \tag{5.9}$$

Therefore, the server is able to easily convert a classical encryption of s (which is in the form a trapdoor claw-free function pair) to a quantum superposition which contains s.

It is quite straightforward to use the above process to apply the encrypted CNOT operation: the superposition over the claw in (5.8) is simply entangled with the first qubit of the quantum state on which the CNOT is to be applied. We now describe this process in detail. Assume the server would like to apply CNOT^s to a 2 qubit state $|\psi\rangle = \sum_{a,b \in \{0,1\}} \alpha_{ab} |a, b\rangle$. The server begins by entangling the first qubit of the state $|\psi\rangle$ with a random claw of f_0, f_1. This process was described in Section 4.2 and proceeds as follows. The server uses the first qubit of $|\psi\rangle$ to choose between the functions f_0, f_1 in order to create the following superposition:

$$\frac{1}{\sqrt{2|\mathcal{R}|}} \sum_{a,b,\mu \in \{0,1\}, r \in \mathcal{R}} \alpha_{ab} |a, b\rangle |\mu, r\rangle |f_a(\mu, r)\rangle \tag{5.10}$$

Now the server measures the final register to obtain $y \in \mathcal{Y}$. Let $(\mu_0, r_0), (\mu_1, r_1)$ be the two preimages of y ($f_0(\mu_0, r_0) = f_1(\mu_1, r_1) = y$). The remaining state is:

$$\sum_{a,b\in\{0,1\}} \alpha_{ab} |a, b\rangle |\mu_a\rangle |r_a\rangle \tag{5.11}$$

Recall that to apply CNOT^s, the value $a \cdot s$ must be added to the register containing b. This is where the structure in (5.9) (which relies on the fact that $\mu_0 \oplus \mu_1 = s$) comes in to play: to add $a \cdot s$, the server XORs μ_a into the second register. This is equivalent to applying the operation CNOT^s:

$$\sum_{a,b\in\{0,1\}} \alpha_{ab} |a, b \oplus \mu_a\rangle |\mu_a\rangle |r_a\rangle = \sum_{a,b\in\{0,1\}} \alpha_{ab} (\mathcal{I} \otimes X^{\mu_0}) \text{CNOT}^s_{1,2} |a, b\rangle \otimes |\mu_a, r_a\rangle \tag{5.12}$$

Finally, the server removes the interference by applying a Hadamard transform on the registers containing μ_a, r_a and measuring to obtain d. If we let (μ_a, r_a) denote the concatenation of the two values, the resulting state (up to a global phase) is

$$(Z^{d\cdot((\mu_0,r_0)\oplus(\mu_1,r_1))} \otimes X^{\mu_0}) \text{CNOT}^s_{1,2} \sum_{a,b\in\{0,1\}} \alpha_{ab} |a, b\rangle \tag{5.13}$$

In order to complete the encrypted CNOT operation, the server requires an encryption of the trapdoor of the functions f_0, f_1. The server can then homomorphically compute the bits μ_0 and $d \cdot ((\mu_0, r_0) \oplus (\mu_1, r_1))$ and use these bits to update his Pauli keys.

5.1.2.1 Trapdoor Claw-free Pair Construction

So far, we have shown how to apply the encrypted CNOT operation in the case that the ciphertext c encrypting the bit s is a trapdoor claw-free function pair f_0, f_1 which hides s. To build a homomorphic encryption scheme, we need to show how the encrypted CNOT operation can be applied if c instead comes from a classical homomorphic encryption scheme, which we call HE. We now show that if HE satisfies certain properties, the function pair f_0, f_1 hiding the bit s can be constructed (by the server) using c.

The function f_0 will be the encryption function of HE. The function f_1 is the function f_0 shifted by the homomorphic XOR of the ciphertext c encrypting the bit s: $f_0 = f_1 \oplus_H c$ ($\oplus_H$ is the homomorphic XOR operation). To ensure that f_0, f_1 are injective, we require that HE has the property of randomness recoverability: there must exist a trapdoor which allows recovery of μ_0, r_0 from a ciphertext $\text{Enc}(\mu_0; r_0)$ ($\text{Enc}(\mu_0; r_0)$ denotes the encryption of a bit μ_0 with randomness r_0). We also require that the homomorphic XOR operation is efficiently invertible using only the public key of HE.

Unfortunately, the images of the functions f_0, f_1 are not equal. We will instead require the weaker (but still sufficient) condition that there exists a distribution D over the domain

of the functions such that $f_0(D)$ and $f_1(D)$ are statistically close. We replace (5.10) with the corresponding weighted superposition:

$$\sum_{a,b,\mu\in\{0,1\},r} \alpha_{ab}\sqrt{D(\mu,r)}\,|a\rangle\,|b\rangle\,|\mu,r\rangle\,|f_a(\mu,r)\rangle \tag{5.14}$$

To do so, we require that the server can efficiently create the following superposition:

$$\sum_{\mu\in\{0,1\},r} \sqrt{D(\mu,r)}\,|\mu,r\rangle \tag{5.15}$$

Due to the negligible statistical distance between $f_0(D)$ and $f_1(D)$, when the last register of (5.14) is measured to obtain y, with high probability there exist μ_0, r_0, μ_1, r_1 such that $y = f_0(\mu_0, r_0) = f_1(\mu_1, r_1)$, which implies that the state collapses to (5.11) and that

$$\mathrm{Enc}(\mu_0; r_0) = \mathrm{Enc}(\mu_1; r_1) \oplus_H c \tag{5.16}$$

Since $\oplus_H$ is the homomorphic XOR operation, $\mu_0 \oplus \mu_1 = s$.

There is one remaining issue with the encrypted CNOT operation described above. The requirements above must hold for a classical ciphertext c which occurs at any point during the classical computation on the encrypted Pauli keys. However, in many classical homomorphic encryption schemes, the format of the ciphertext c changes throughout the computation. We know of several schemes for which the above requirements hold for a freshly encrypted ciphertext, but we do not know of any schemes which satisfy the requirements during a later stage of computation. The next section addresses this complication by sufficiently weakening the above requirements while preserving the functionality of the encrypted CNOT operation.

5.1.3 Quantum Capable Classical Homomorphic Encryption Schemes

In this section, we define quantum capable homomorphic encryption schemes, i.e. classical leveled fully homomorphic encryption schemes which can be used to evaluate quantum circuits. To justify why we must weaken the requirements listed in Section 5.1.2, we begin with a description of the ideal high level structure of a quantum capable homomorphic encryption scheme. In many classical homomorphic encryption schemes, the encryption of a bit b can be thought of as a random element of a subset S_b, perturbed by some noise term ϵ. As the computation progresses, we will require that the structure of the ciphertext remains the same; it must still be a random element of S_b, but the noise term may grow throughout the computation. We will also require that the homomorphic XOR operation is natural, in the sense that the noise of the output ciphertext is simply the addition of the two noise terms of the input ciphertexts. If these two conditions hold (invariance of the ciphertext form and the existence of a natural XOR operation), deriving a distribution $f_0(D)$ over ciphertexts which remains roughly the same after shifting by the homomorphic XOR of the ciphertext

c (as needed in Section 5.1.2) is straightforward. We simply choose D to sample the noise term from a discrete Gaussian distribution with width sufficiently larger than the magnitude of the noise term of the ciphertext c.

Unfortunately, we do not know of a classical homomorphic encryption scheme which satisfies both the conditions (ciphertext form and natural XOR operation) at once. To account for this difficulty, we define a quantum capable homomorphic encryption schemes as follows. We call a classical homomorphic encryption scheme HE quantum capable if there exists an alternative encryption scheme AltHE which satisfies the following conditions. First, given a ciphertext c under HE, it must be possible for the server to convert c to a ciphertext $\hat{c}$ under AltHE. The conversion process must maintain the decrypted value of the ciphertext. Second, AltHE must have a natural homomorphic XOR operation (which is also efficiently invertible). Third, there must exist a distribution $f_0(D)$ over encryptions under AltHE which must remain almost the same after shifting by the homomorphic XOR of $\hat{c}$ and must allow efficient construction of the superposition in (5.15). In addition, it must be possible to both decrypt and recover randomness from ciphertexts under AltHE given the appropriate secret key and trapdoor information. This definition is formalized in Section 5.3.

Finally, we describe how to connect this weaker definition to the encrypted CNOT operation given in Section 5.1.2. We begin with a quantum capable homomorphic encryption scheme HE, which is used to encrypt the Pauli keys. HE satisfies the ciphertext form requirement but may not have a natural XOR operation. Each time the server needs to apply an encrypted CNOT operation (controlled by a ciphertext c encrypting a bit s under HE), he will convert c to a ciphertext $\hat{c}$ under AltHE, which does have a natural XOR operation. Using AltHE (rather than HE) the server performs the operations described in Section 5.1.2. Upon obtaining his measurement results (denoted as y and d), the server will encrypt both $\hat{c}$ and y, d under HE. The server will then use the secret key and trapdoor information of AltHE, which are provided to him as encryptions under HE, to homomorphically recover the randomness and decrypted values from both y and $\hat{c}$. This encrypted information can be used to homomorphically compute the Pauli key updates. The entire classical homomorphic computation is done under HE.

5.1.4 Example of a Quantum Capable Classical Encryption Scheme

In Section 5.4 (Theorem 5.4.2), we show that an existing classical fully homomorphic encryption scheme is quantum capable. We use the structure of the scheme from [24], which is a leveled fully homomorphic encryption scheme built by extending the vector ciphertexts of [43] to matrices. The resulting encryption scheme does satisfy the ciphertext form requirement (as described in Section 5.1.3), but since the underlying encryption scheme ([43]) does not have the randomness recoverability property, neither does [24]. We therefore alter the scheme from [24] to use the dual encryption scheme of [43], which was introduced in [23] and allows randomness recovery, as the underlying encryption scheme. We call the resulting

scheme DualHE and the underlying scheme of [23] Dual.

We use the scheme DualHE as an instantiation of the scheme we called HE in Section 5.1.3. Although the underlying scheme Dual does have a natural XOR operation, the extension to matrices compromises the XOR operation; once the ciphertexts are matrices, addition is performed over a larger field. Luckily, it is easy to convert a ciphertext of DualHE to a ciphertext of Dual. We therefore use Dual as AltHE.

In Section 5.4, we first describe the scheme Dual from [23] and we then show how to extend it to DualHE using [24]. Next, we show that DualHE satisfies the ciphertext form requirement and that a ciphertext under DualHE can be converted to a ciphertext under Dual. In Theorem 5.4.1, we use these properties to show that DualHE is a classical leveled fully homomorphic encryption scheme. Finally, we show in Theorem 5.4.2 that DualHE is quantum capable with only a small modification of parameters (the underlying assumption for both the classical FHE and the quantum capable instantiation is the hardness of learning with errors with a superpolynomial noise ratio).

5.1.5 Extension to Quantum Leveled Fully Homomorphic Encryption

We have so far provided a quantum fully homomorphic encryption scheme with classical keys under the assumption of circular security: the server must be provided the encrypted secret key and trapdoor information in order to update his encrypted Pauli keys after each encrypted CNOT operation. Our notion of circular security here will be slightly stronger than the standard notion, due to the encryption of the trapdoor (instead of just the secret key). As an alternative to assuming circular security, we can build a quantum leveled fully homomorphic encryption scheme by employing a technique which is commonly used in classical homomorphic encryption schemes (Section 4.1 in [22]): we will encrypt the secret key and trapdoor information under a fresh public key. In other words, the i^{th} level secret key sk_i and its corresponding trapdoor information are encrypted under a fresh public key pk_{i+1} and given to the server as part of the evaluation key. The computation of the Pauli key updates corresponding to the encrypted CNOT operations of level i is performed under pk_{i+1} (i.e. the corresponding $\hat{c}, y$ and d from each encrypted CNOT operation in level i will be encrypted, by the server, under pk_{i+1} - see the last paragraph of Section 5.1.3).

Note that with the introduction of the leveled scheme, we can see the classical portion of the quantum homomorphic computation as follows. Each level of the quantum computation can be thought of as a series of Clifford gates followed by one layer of non intersecting Toffoli gates, finishing with a layer of non intersecting encrypted CNOT operations. It follows that the classical homomorphic computation of level i consists of first decrypting and recovering randomness from the cipertexts corresponding to the encrypted CNOT operations from level $i-1$, then performing the Pauli key updates corresponding to the encrypted CNOT operations from level $i-1$ and finally performing the Pauli key updates corresponding to the Clifford and Toffoli gates of level i. The ciphertexts which result from this computation

are then used as the control bits for the layer of encrypted CNOT operations of level i.

Intuitively, this leveled approach is secure since each secret key is protected by the semantic security of the encryption scheme under an independent public key. To prove security, we start with the final level of encryption. If there are L levels of the circuit, then there will be no trapdoor or secret key information provided corresponding to pk_{L+1}. It follows that all encryptions under pk_{L+1} can be replaced by encryptions of 0; now there is no encrypted information provided corresponding to sk_L. Then all encryptions under pk_L can be replaced by encryptions of 0, and we can continue in this manner until we reach pk_1, which will imply security of encryptions under the initial public key pk_1. The scheme and proof of security are presented in Section 5.5, proving that quantum capable classical encryption schemes can be used to build a quantum leveled fully homomorphic encryption scheme (see Theorem 5.5.2 for a formal statement).

5.1.6 Chapter Outline

We begin with preliminaries in Section 5.2. In Section 5.3, we define quantum capable encryption schemes by listing the requirements that a classical homomorphic encryption scheme must satisfy in order to be used to evaluate quantum circuits, as described in Section 5.1.3. We use this definition to formally prove the correctness (in Claim 5.3.3) of the encrypted CNOT operation given in Section 5.1.2. Section 5.4 covers Section 5.1.4: we provide an example of a classical homomorphic encryption scheme which is quantum capable. In Section 5.5, we formally show how to extend a quantum capable classical leveled fully homomorphic encryption scheme to a quantum leveled fully homomorphic encryption scheme (as described in Section 5.1.5).

5.2 Preliminaries

5.2.1 Homomorphic Encryption

The following definitions are modified versions of definitions from [9] and [10]. A homomorphic (public-key) encryption scheme HE = (HE.Keygen, HE.Enc, HE.Dec, HE.Eval) is a quadruple of PPT algoritms which operate as follow:

- **Key Generation.** The algorithm $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\lambda)$ takes a unary representation of the security parameter and outputs a public key encryption key pk, a public evaluation key evk and a secret decryption key sk.
- **Encryption.** The algorithm $c \leftarrow \text{HE.Enc}_{pk}(\mu)$ takes the public key pk and a single bit message $\mu \in \{0,1\}$ and outputs a ciphertext c. The notation $\text{HE.Enc}_{pk}(\mu; r)$ will be used to represent the encryption of a bit μ using randomness r.
- **Decryption.** The algorithm $\mu^* \leftarrow \text{HE.Dec}_{sk}(c)$ takes the secret key sk and a ciphertext c and outputs a message $\mu^* \in \{0,1\}$.

- **Homomorphic Evaluation** The algorithm $c_f \leftarrow \text{HE.Eval}_{evk}(f, c_1, \ldots, c_l)$ takes the evaluation key evk, a function $f : \{0,1\}^l \to \{0,1\}$ and a set of l ciphertexts $c_1, \ldots, c_l$, and outputs a ciphertext c_f. It must be the case that:

$$\text{HE.Dec}_{sk}(c_f) = f(\text{HE.Dec}_{sk}(c_1), \ldots, \text{HE.Dec}_{sk}(c_l)) \tag{5.17}$$

  with all but negligible probability in λ.

A homomorphic encryption scheme is said to be secure if it meets the following notion of semantic security:

Definition 5.2.1 (CPA Security) *A scheme* HE *is IND-CPA secure if, for any polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that*

$$\text{Adv}_{\text{CPA}}[\mathcal{A}] \stackrel{\text{def}}{=} |\Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(1)) = 1]| = \mu(\lambda) \tag{5.18}$$

where $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\lambda)$.

We now define two desirable properties of homomorphic encryption schemes:

Definition 5.2.2 (Compactness and Full Homomorphism) *A homomorphic encryption scheme HE is compact if there exists a polynomial s in λ such that the output length of HE.Eval is at most s bits long (regardless of f or the number of inputs). A compact scheme is (pure) fully homomorphic if it can evaluate any efficiently computable boolean function. A compact scheme is leveled fully homomorphic if it takes 1^L as additional input in key generation, and can only evaluate depth L Boolean circuits where L is polynomial in λ. A scheme is quantum pure or leveled fully homomorphic if we refer to quantum circuits rather than boolean circuits.*

5.2.2 Toffoli Gate Application

A Toffoli operator maps Pauli operators to Clifford operators in the following way:

$$\begin{aligned} TZ^{z_1}X^{x_1} \otimes Z^{z_2}X^{x_2} \otimes Z^{z_3}X^{x_3}T^\dagger &= CNOT^{x_2}_{1,3}CNOT^{x_1}_{2,3}\hat{Z}^{z_3}_{1,2}Z^{z_1+x_2z_3}X^{x_1} \otimes Z^{z_2+x_1z_3}X^{x_2} \otimes Z^{z_3}X^{x_1x_2+x_3} \\ &= C_{zx}P_{zx} \end{aligned} \tag{5.19}$$

where $\hat{Z}$ is the controlled phase gate:

$$\hat{Z}\,|a,b\rangle = (-1)^{ab}\,|a,b\rangle \tag{5.20}$$

$$\hat{Z}^{z_3} = (\mathcal{I} \otimes H)CNOT^{z_3}_{1,2}(\mathcal{I} \otimes H) \tag{5.21}$$

Observe that C_{zx} consists only of CNOT gates and 2 Hadamard gates. Only the CNOT gates are dependent on the Pauli keys.

5.2.3 Pauli Mixing

Here we restate Lemma 5.1.1 and prove it:
Lemma 5.1.1 *For a matrix ρ on two spaces A, B*

$$\frac{1}{2^{2l}} \sum_{z,x \in \{0,1\}^l} (Z^z X^x \otimes \mathcal{I}_B)\rho(Z^z X^x \otimes \mathcal{I}_B)^\dagger = \frac{1}{2^l}\mathcal{I}_A \otimes Tr_A(\rho)$$

*P*roof of **Lemma 5.1.1** : First, we write ρ as:

$$\sum_{ij} |i\rangle \langle j|_A \otimes \rho_{ij}$$

It follows that:

$$\mathrm{Tr}_A(\rho) = \sum_i \rho_{ii}$$

Next, observe that:

$$\sum_{z,x \in \{0,1\}^l} Z^z X^x |i\rangle \langle j| (Z^z X^x)^\dagger = \sum_{x \in \{0,1\}^l} (\sum_{z \in \{0,1\}^l} (-1)^{z \cdot (i \oplus j)}) X^x |i\rangle \langle j| (X^x)^\dagger \quad (5.22)$$

This expression is 0 if $i \neq j$. If $i = j$, we obtain $2^l \mathcal{I}_A$. Plugging in this observation to the expression in the claim, we have:

$$\begin{aligned} \frac{1}{2^{2l}} \sum_{z,x \in \{0,1\}^l} (Z^z X^x \otimes \mathcal{I}_B)\rho(Z^z X^x \otimes \mathcal{I}_B)^\dagger &= \frac{1}{2^{2l}} \sum_{ij} \sum_{z,x \in \{0,1\}^l} Z^z X^x |i\rangle \langle j|_A (Z^z X^x)^\dagger \otimes \rho_{ij} \\ &= \frac{1}{2^{2l}} \sum_{i} \sum_{z,x \in \{0,1\}^l} Z^z X^x |i\rangle \langle i|_A (Z^z X^x)^\dagger \otimes \rho_{ii} \\ &= \frac{1}{2^l}\mathcal{I}_A \otimes \mathrm{Tr}_A(\rho) \quad (5.23) \end{aligned}$$

□

5.3 Quantum Capable Classical Homomorphic Encryption Schemes

As described in Section 5.1.3, a classical leveled fully homomorphic encryption scheme is quantum capable if an encrypted CNOT operation can be applied with respect to any ciphertext which occurs during the computation. We begin by formalizing the notion of such a ciphertext. This definition is dependent on the depth parameter L (see Definition 5.2.2). Let $\mathcal{F}_L$ be the set of all functions which can be computed by circuits of depth L. Assume for convenience that all such functions have domain $\{0,1\}^l$ and range $\{0,1\}$.

Definition 5.3.1 *For a classical leveled fully homomorphic encryption scheme HE, let C_{HE} be the set of all ciphertexts which can occur during the computation:*

$$C_{HE} = \{HE.Eval_{evk}(f, HE.Enc_{pk}(\mu_1), \ldots, HE.Enc_{pk}(\mu_l)) | f \in \mathcal{F}_L, \mu_1, \ldots \mu_l \in \{0,1\}\} \quad (5.24)$$

We now define quantum capable homomorphic encryption schemes:

Definition 5.3.2 (Quantum Capable Homomorphic Encryption Schemes) *Let λ be the security parameter. Let HE be a classical leveled fully homomorphic encryption scheme. HE is quantum capable if there exists an encryption scheme AltHE such that the following conditions hold for all ciphertexts $c \in C_{HE}$.*

1. *There exists an algorithm HE.Convert which on input c produces an encryption $\hat{c}$ under AltHE, where both c and $\hat{c}$ encrypt the same value.*

2. *AltHE allows the XOR operation to be performed homomorphically. Moreover, the XOR operation is efficiently invertible using only the public key of AltHE.*

3. *There exists a distribution D which may depend on the parameters of HE and satisfies the following conditions:*

   a) *The Hellinger distance between the following two distributions is negligible in λ:*

   $$\{\text{AltHE.Enc}_{pk}(\mu; r) | (\mu, r) \xleftarrow{\$} D\} \quad (5.25)$$

   and

   $$\{\text{AltHE.Enc}_{pk}(\mu; r) \oplus_H \hat{c} | (\mu, r) \xleftarrow{\$} D\} \quad (5.26)$$

   where $\oplus_H$ represents the homomorphic XOR operation.

   b) *It is possible for a* BQP *server to create the following superposition given access to the public key pk:*

   $$\sum_{\mu \in \{0,1\}, r} \sqrt{D(\mu, r)} \, |\mu, r\rangle \quad (5.27)$$

   c) *Given $y = \text{AltHE.Enc}_{pk}(\mu_0; r_0)$ where (μ_0, r_0) is sampled from D, it must be possible to compute μ_0, r_0 given the secret key and possibly additional trapdoor information (which can be computed as part of the key generation procedure).*

For convenience, we will assume that AltHE and HE have the same public/secret key; this is the case in the example quantum capable scheme we provide in Section 5.4 and also simplifies Section 5.5. However, this assumption is not necessary.

We prove the following claim, which formalizes the encrypted CNOT operation given in Section 5.1.2:

Claim 5.3.3 *Let HE be a quantum capable homomorphic encryption scheme and let $c \in C_{HE}$ be a ciphertext encrypting a bit s. Consider a* BQP *machine with access to c and a state $|\psi\rangle$ on two qubits. The* BQP *machine can compute a ciphertext $y = AltHE.Enc_{pk}(\mu_0, r_0)$, a string d and a state within negligible trace distance of the following ideal state:*

$$(Z^{d\cdot((\mu_0,r_0)\oplus(\mu_1,r_1))} \otimes X^{\mu_0})CNOT^s_{1,2}\,|\psi\rangle \tag{5.28}$$

for ($\oplus_H$ is the homomorphic XOR operation)

$$\text{AltHE.Enc}_{pk}(\mu_0, r_0) = \text{AltHE.Enc}_{pk}(\mu_1, r_1) \oplus_H \text{HE.Convert}(c) \tag{5.29}$$

*P*roof of **Claim 5.3.3**: The server first computes $\hat{c} = \text{HE.Convert}(c)$. He then applies the encrypted CNOT operation, as described in Section 5.1.2. Recall that in Section 5.1.2, we used f_0 to denote the encryption function of AltHE and f_1 to denote the shift of f_0 by the homomorphic XOR (which we denote as $\oplus_H$) of $\hat{c}$. At the stage of (5.14), the server holds the following state:

$$\sum_{a,b,\mu\in\{0,1\},r} \alpha_{ab}\sqrt{D(\mu,r)}\,|a\rangle\,|b\rangle\,|\mu,r\rangle\,|f_a(r)\rangle \tag{5.30}$$

$$= \sum_{a,b,\mu\in\{0,1\},r} \alpha_{ab}\sqrt{D(\mu,r)}\,|a\rangle\,|b\rangle\,|\mu,r\rangle\,|\text{AltHE.Enc}_{pk}(\mu;r) \oplus_H a\cdot\hat{c}\rangle \tag{5.31}$$

Fix $a = 1$ and b and consider the resulting state:

$$\sum_{\mu_0\in\{0,1\},r_0} \sqrt{D(\mu_1,r_1)}\,|1,b\rangle\,|\mu_1,r_1\rangle\,|\text{AltHE.Enc}_{pk}(\mu_0;r_0)\rangle \tag{5.32}$$

where μ_1, r_1 are defined with respect to μ_0, r_0 and $\hat{c}$ as in (5.29). We can now apply Lemma 3.3.1 with reference to the distributions in (5.25) and (5.26), which are negligibly close. As a result, we obtain that the following state is within negligible trace distance of (5.32):

$$\sum_{\mu_0\in\{0,1\},r_0} \sqrt{D(\mu_0,r_0)}\,|1,b\rangle\,|\mu_1,r_1\rangle\,|\text{AltHE.Enc}_{pk}(\mu_0;r_0)\rangle \tag{5.33}$$

It follows immediately that the state in (5.31) is within negligible trace distance of the following state:

$$\sum_{\mu_0\in\{0,1\},r_0}\ \sum_{a,b\in\{0,1\}} \alpha_{ab}\sqrt{D(\mu_0,r_0)}\,|a,b\rangle\,|\mu_a,r_a\rangle\,|\text{AltHE.Enc}_{pk}(\mu_0;r_0)\rangle \tag{5.34}$$

Observe that, when measured, the state in (5.34) collapses exactly to the state in (5.11). The statement of Claim 5.3.3 follows.

□

5.4 Example of a Quantum Capable Classical Encryption Scheme

This section is dedicated to showing that the dual of the fully homomorphic encryption scheme in [24] is quantum capable. We begin by presenting a scheme called Dual in Section 5.4.1, which is the dual of the encryption scheme from [43]. In Section 5.4.2, we use the framework of [24] to extend Dual to a scheme called DualHE, which we prove in Theorem 5.4.1 is a classical leveled fully homomorphic encryption scheme. Finally, in Section 5.4.3 (Theorem 5.4.2), we prove that DualHE is quantum capable .

We begin by listing our initial parameters. Let λ be the security parameter. All other parameters are functions of λ. Let $q \geq 2$ be a power of 2. Let $n, m \geq 1$ be polynomially bounded functions of λ, let $N = (m+1)\log q$ and let β_{init} be a positive integer such that the following conditions hold:

$$\begin{aligned} &1.\ m = \Omega(n \log q)\ , \\ &2.\ 2\sqrt{n} \leq \beta_{init} \end{aligned} \tag{5.35}$$

5.4.1 Dual Encryption Scheme

We first describe the dual scheme of [43]. This scheme was originally given in [23], but the presentation below is taken from Section 5.2.2 in [40]. This scheme will eventually serve as the scheme AltHE in Definition 5.3.2.

Scheme 5.4.1 ***Dual Encryption Scheme [23]***

- *Dual.KeyGen: Choose* $\mathbf{e}_{sk} \in \{0,1\}^m$ *uniformly at random. Using the procedure* $\textsc{GenTrap}(1^n, 1^m, q)$ *from Theorem 3.2.3, sample a random trapdoor matrix* $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$*, together with the trapdoor information* $t_{\mathbf{A}}$*. The secret key is* $\mathbf{sk} = (-\mathbf{e}_{sk}, 1) \in \mathbb{Z}_q^{m+1}$ *and the trapdoor is* $t_{\mathbf{A}}$*. The public key is* $\mathbf{A}' \in \mathbb{Z}_q^{(m+1)\times n}$*, which is the matrix composed of* $\mathbf{A}$ *(the first* m *rows) and* $\mathbf{A}^T\mathbf{e}_{sk} \bmod q$ *(the last row).*

- *Dual.*$Enc_{pk}(\mu)$*: To encrypt a bit* $\mu \in \{0,1\}$*, choose* $\mathbf{s} \in \mathbb{Z}_q^n$ *uniformly and create* $\mathbf{e} \in \mathbb{Z}_q^{m+1}$ *by sampling each entry from* $D_{\mathbb{Z}_q, \beta_{init}}$*. Output* $\mathbf{A}'\mathbf{s} + \mathbf{e} + (0, \ldots, 0, \mu \cdot \frac{q}{2}) \in \mathbb{Z}_q^{m+1}$*.*

- *Dual.*$Dec_{sk}(\mathbf{c})$*: To decrypt, compute* $b' = \mathbf{sk}^T\mathbf{c} \in \mathbb{Z}_q$*. Output 0 if* b' *is closer to 0 than to* $\frac{q}{2} \bmod q$*, otherwise output 1.*

We make a few observations:

- For a ciphertext c with error $\mathbf{e}$ such that $\|\mathbf{e}\| < \frac{q}{4\sqrt{m+1}}$, the decryption procedure will operate correctly (since $\mathbf{sk}^T\mathbf{A}' = \mathbf{0}$).

- The trapdoor $t_{\mathbf{A}}$ can be used to recover the randomness $\mathbf{s}, \mathbf{e}$ from a ciphertext. To see this, note that the first m entries of the ciphertext can be written as $\mathbf{As} + \mathbf{e}'$, where $\mathbf{e}' \in \mathbb{Z}_q^m$. Therefore, the inversion algorithm INVERT in Theorem 3.2.3 outputs $\mathbf{s}, \mathbf{e}$ on input $\mathbf{As} + \mathbf{e}'$ and $t_{\mathbf{A}}$ as long as $\|\mathbf{e}'\| < \frac{q}{C_T\sqrt{n \log q}}$ for C_T the universal constant in Theorem 3.2.3.

- This scheme is naturally additively homomorphic; adding two ciphertexts encrypting μ_0 and μ_1 results in a ciphertext encrypting $\mu_0 \oplus \mu_1$.

5.4.2 Leveled Fully Homomorphic Encryption Scheme from Dual

We can extend the scheme Dual into a leveled fully homomorphic encryption scheme DualHE in the same way that the standard LWE scheme from [43] is extended in [24]. Namely, we map the ciphertexts to matrices and encrypt the bit μ in a matrix (the key generation procedure remains the same). We begin with the required preliminaries and then describe the scheme DualHE. Next, we prove a property of DualHE which is crucial for quantum capability: the ciphertexts retain the same form throughout the computation. Finally, we show in Theorem 5.4.1 that for a strengthened version of the parameters in (5.35), DualHE a leveled fully homomorphic encryption scheme.

To describe this scheme, we will require two operations used in [24]. The first is the linear operator $\mathbf{G} \in \mathbb{Z}_q^{(m+1)\times N}$ $(N = (m+1)\log_2 q)$, which converts a binary representation back to the original representation in $\mathbb{Z}_q^{m+1}$. More precisely, consider the N dimensional vector $\mathbf{a} = (a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{m+1,0}, \ldots, a_{m+1,l-1})$, where $l = \log_2 q$. $\mathbf{G}$ performs the following mapping:

$$\mathbf{G}(\mathbf{a}) = \Big(\sum_{j=0}^{\log_2 q - 1} 2^j \cdot a_{1,j}, \ldots, \sum_{j=0}^{\log_2 q - 1} 2^j \cdot a_{m+1,j} \Big) \tag{5.36}$$

Observe that $\mathbf{G}$ is well defined even if $\mathbf{a}$ is not a 0/1 vector. We will call the non linear inverse operation G^{-1}, which converts $\mathbf{a} \in \mathbb{Z}_q^{m+1}$ to its binary representation (a vector in $\mathbb{Z}_2^N$). G^{-1} can also be applied to a matrix by converting each column. Note that $\mathbf{G}G^{-1}$ is the identity operation. In terms of homomorphic evaluation, we will only consider the NAND gate, since we are only concerned with applying Boolean circuits. The scheme can be extended to arithmetic circuits over $\mathbb{Z}_q$, as described in further detail in [24].

The description of the scheme given below is derived from talks ([8], [49]) describing the scheme in [24]. It is equivalent to the description given in [24], but is more convenient for our purposes.

Scheme 5.4.2 ***DualHE: Classical Leveled FHE Scheme from Dual***

- *DualHE.KeyGen: This procedure is the same as Dual.KeyGen.*

- *DualHE.Enc$_{pk}(\mu)$: To encrypt a bit $\mu \in \{0,1\}$, choose $\mathbf{S} \in \mathbb{Z}_q^{n\times N}$ uniformly at random and create $\mathbf{E} \in \mathbb{Z}_q^{(m+1)\times N}$ by sampling each entry from $D_{\mathbb{Z}_q,\beta_{init}}$. Output $\mathbf{A}'\mathbf{S}+\mathbf{E}+\mu\mathbf{G} \in \mathbb{Z}_q^{(m+1)\times N}$.*

- *DualHE.Eval$(\mathbf{C}_0, \mathbf{C}_1)$: To apply the NAND gate, on input $\mathbf{C}_0, \mathbf{C}_1$ output $\mathbf{G} - \mathbf{C}_0 \cdot G^{-1}(\mathbf{C}_1)$.*

For quantum capability, we will also require the following algorithm, which converts a ciphertext under DualHE to a ciphertext under Dual:

- *DualHE.Convert$(\mathbf{C})$: Output column N of $\mathbf{C}$*

Given this algorithm, we can state decryption in terms of Dual[2]*:*

- *DualHE.Dec$_{sk}(\mathbf{C})$: Output Dual.Dec$_{sk}$(DualHE.Convert$(\mathbf{C})$)*

5.4.2.1 Ciphertext Form

We will rely on the fact that, throughout the computation of a Boolean circuit of depth L, a ciphertext encrypting a bit μ can be written in the following form:

$$\mathbf{A}'\mathbf{S} + \mathbf{E} + \mu\mathbf{G} \tag{5.37}$$

where $\|\mathbf{E}\|_\infty \leq \beta_{init}(N+1)^L$. This is clearly the structure of the ciphertext immediately after encryption and we now show that this ciphertext structure is maintained after the NAND operation. The NAND operation is performed by computing:

$$\mathbf{G} - \mathbf{C}_0 \cdot G^{-1}(\mathbf{C}_1) \tag{5.38}$$

Assume the ciphertexts we begin with are $\mathbf{C}_b = \mathbf{A}'\mathbf{S}_b + \mathbf{E}_b + \mu_b\mathbf{G}$ for $b \in \{0,1\}$. Using the fact that $\mathbf{G}G^{-1}$ is the identity operation, it is easy to see that the result of the NAND operation is:

$$\mathbf{A}'\mathbf{S}' + \mathbf{E}' + (1 - \mu_0\mu_1)\mathbf{G} \tag{5.39}$$

for

$$\mathbf{S}' = -\mathbf{S}_0 \cdot G^{-1}(\mathbf{C}_1) - \mu_0\mathbf{S}_1 \tag{5.40}$$

$$\mathbf{E}' = -\mathbf{E}_0 \cdot G^{-1}(\mathbf{C}_1) - \mu_0\mathbf{E}_1 \tag{5.41}$$

Note that if both $\|\mathbf{E}_0\|_\infty$ and $\|\mathbf{E}_1\|_\infty$ are at most β, then $\|\mathbf{E}'\|_\infty \leq \beta(N+1)$. It follows that if the scheme DualHE is used to compute a circuit of depth L, $\|\mathbf{E}\|_\infty \leq \beta_{init}(N+1)^L$ for all ciphertexts throughout the computation.

[2]This is equivalent to the decryption algorithm of [24], which is as follows. Let $\mathbf{u} = (0,\ldots,0,1) \in \mathbb{Z}_q^{m+1}$. To decrypt, compute $b' = \mathbf{s}\mathbf{k}^T\mathbf{C}G^{-1}(\frac{q}{2}\mathbf{u})$. Output 0 if b' is closer to 0 than to $\frac{q}{2}$ mod q, otherwise output 1.

5.4.2.2 Encryption Conversion

We now use the above property to prove the correctness of DualHE.Convert. Assume we begin with a ciphertext $\mathbf{C} = \mathbf{A}'\mathbf{S} + \mathbf{E} + \mu\mathbf{G}$ under DualHE. The ciphertext $\mathbf{c}$ (under Dual) will be column N of $\mathbf{C}$. To see why this is correct, first note that all individual columns of $\mathbf{A}'\mathbf{S} + \mathbf{E}$ are of the form $\mathbf{A}'\mathbf{s} + \mathbf{e}$. Second, observe that column N of $\mu\mathbf{G}$ is equal to $(0, \ldots, 0, \mu \cdot \frac{q}{2})$.

5.4.2.3 Proof of Correctness and Security of Scheme 5.4.2

The above two sections allow us to easily prove the following theorem:

Theorem 5.4.1 *Let λ be the security parameter. There exists a function η_c which is logarithmic in λ such that DualHE is IND-CPA secure and leveled fully homomorphic under the hardness assumption $LWE_{n,q,D_{\mathbb{Z}_q,\beta_{init}}}$ if the conditions in* (5.35) *as well as the following condition are satisfied:*

$$\beta_{init}(N+1)^{\eta_c} < \frac{q}{4(m+1)}. \tag{5.42}$$

Proof: We prove that DualHE is IND-CPA secure by relying on the hardness of $\text{LWE}_{n,m,q,D_{\mathbb{Z}_q,\beta_{init}}}$ (i.e. the hardness of LWE with a superpolynomial noise ratio), which implies that the ciphertext is computationally indistinguishable from a uniform string. We can use this LWE assumption as long as the public key $\mathbf{A}'$ is statistically indistinguishable from a uniformly random matrix (see Section 3.2). Since $\mathbf{A}$ is selected from a distribution which is statistically indistinguishable from the uniform distribution and $m = \Omega(n \log q)$, $\mathbf{A}'$ is statistically indistinguishable from uniform due to the leftover hash lemma in [31] (see [43] or [40] for more details).

We now show that DualHE is leveled fully homomorphic. From Section 5.4.2.1, it is clear that the evaluation of the NAND operation is correct and that DualHE is compact. Let η_c be larger than the depth of the decryption circuit of DualHE, which is logarithmic in λ. If we show that the decryption procedure operates correctly after evaluation of a circuit of depth η_c, the standard bootstrapping technique[3] of [22] can be used to turn DualHE into a leveled fully homomorphic encryption scheme. Due to Section 5.4.2.1, we can assume a ciphertext resulting from a circuit of depth η_c can be written as $\mathbf{A}'\mathbf{S} + \mathbf{E} + \mu\mathbf{G}$, where $\|\mathbf{E}\|_\infty \leq \beta_{init}(N+1)^{\eta_c}$. It is easy to check that the decryption procedure operates correctly as long as

$$\|\mathbf{E}\|_\infty < \frac{q}{4(m+1)} \tag{5.43}$$

The condition in (5.43) is implied by the condition in (5.42).

□

[3] A pure fully homomorphic encryption scheme can be obtained by assuming circular security. A leveled fully homomorphic encryption scheme can be obtained by producing a string of public and secret keys and encrypting each secret key under the next public key - see Section 4.1 of [22].

5.4.3 Quantum Capability of DualHE

We now prove the following theorem:

Theorem 5.4.2 *Let λ be the security parameter, let η_c be the logarithmic function in Theorem 5.4.1, and let η be an arbitrary logarithmic function in λ. Assume the choice of parameters satisfies the conditions in* (5.35) *as well as the following condition:*

$$\beta_{init}(N+1)^{\eta+\eta_c} < \frac{q}{4(m+1)}. \tag{5.44}$$

Under the hardness assumption of $LWE_{n,q,D_{\mathbb{Z}_q},\beta_{init}}$, the scheme DualHE is quantum capable.

Observe that the only change in parameters involved in making DualHE quantum capable is increasing the circuit depth by an additive logarithmic factor; this does not change the underlying computational assumption of the hardness of learning with errors with a superpolynomial noise ratio.

Proof: To prove Theorem 5.4.2, we begin by noting that DualHE is leveled fully homomorphic by Theorem 5.4.1. We now show that DualHE is quantum capable, by listing the requirements for quantum capability and proving that each holds. The scheme corresponding to AltHE will be Dual (Section 5.4.1). Recall the definition of C_{DualHE} from Definition 5.3.1. For all ciphertexts $c \in C_{\text{DualHE}}$:

1. *There exists an algorithm DualHE.Convert$_{pk}$ which on input c produces an encryption $\hat{c}$ under Dual, where both c and $\hat{c}$ encrypt the same value.*

   See Section 5.4.2.2.

2. *Dual allows the XOR operation to be performed homomorphically. Moreover, the XOR operation is efficiently invertible using only the public key of AltHE.*

   See Section 5.4.1.

3. *There exists a distribution D which satisfies the following conditions:*

   a) *The Hellinger distance between the following two distributions is negligible in λ:*

   $$\{\text{Dual.Enc}_{pk}(\mu; r) | (\mu, r) \xleftarrow{\$} D\} \tag{5.45}$$

   and

   $$\{\text{Dual.Enc}_{pk}(\mu; r) \oplus_H \hat{c} | (\mu, r) \xleftarrow{\$} D\} \tag{5.46}$$

   where $\oplus_H$ represents the homomorphic XOR operation.

   Let

   $$\beta_f = \beta_{init}(N+1)^{\eta_c+\eta} \tag{5.47}$$

   The distribution D will sample $\mu, \mathbf{s}$ uniformly at random and will sample $\mathbf{e}$ from the discrete Gaussian distribution $D_{\mathbb{Z}_q^{m+1},\beta_f}$. Assume that $\hat{c} = \text{DualHE.Convert}(c) =$

$\mathbf{A}'\mathbf{s}' + \mathbf{e}' + (0, \ldots, 0, s \cdot \frac{q}{2})$ where $\|\mathbf{e}'\| \leq \beta_{init}(N+1)^{\eta_c}\sqrt{m+1}$. We can assume this since we know the format of the ciphertext throughout the computation (see Section 5.4.2.1). The two distributions corresponding to (5.45) and (5.46) are as follows:

$$\{\mathbf{A}'\mathbf{s} + \mathbf{e} + (0, \ldots, 0, \mu \cdot \frac{q}{2}) | (\mu, \mathbf{s}, \mathbf{e}) \xleftarrow{\$} D\} \tag{5.48}$$

and

$$\{\mathbf{A}'(\mathbf{s} + \mathbf{s}') + \mathbf{e} + \mathbf{e}' + (0, \ldots, 0, \mu \cdot \frac{q}{2}) | (\mu, \mathbf{s}, \mathbf{e}) \xleftarrow{\$} D\} \tag{5.49}$$

The Hellinger distance between the distributions in (5.48) and (5.49) is equal to the distance between the following two distributions:

$$\{\mathbf{e} | \mathbf{e} \xleftarrow{\$} D_{\mathbb{Z}_q^{m+1}, \beta_f}\} \tag{5.50}$$

and

$$\{\mathbf{e} + \mathbf{e}' | \mathbf{e} \xleftarrow{\$} D_{\mathbb{Z}_q^{m+1}, \beta_f}\} \tag{5.51}$$

Since $\|\mathbf{e}'\| \leq \beta_{init}(N+1)^{\eta_c}\sqrt{m+1}$ and $\frac{\beta_f}{\beta_{init}(N+1)^{\eta_c}}$ is equal to the superpolynomial function $(N+1)^{\eta}$, Lemma 3.2.1 shows that the distance between the two distributions is negligible.

b) *It is possible for a* BQP *server to create the following superposition:*

$$\sum_{\mu \in \{0,1\}, r} \sqrt{D(\mu, r)} \, |\mu, r\rangle \tag{5.52}$$

In this case, $r = (\mathbf{s}, \mathbf{e})$. D samples μ and $\mathbf{s}$ according to the uniform distribution and samples $\mathbf{e}$ according to the discrete Gaussian distribution $D_{\mathbb{Z}_q^{m+1}, \beta_f}$. It is easy for a BQP server to create a superposition over a discrete Gaussian (see Lemma 3.12 in [43][4]).

c) *Given* $y = \text{AltHE.Enc}_{pk}(\mu_0; r_0)$ *where* (μ_0, r_0) *is sampled from* D*, it must be possible to compute* μ_0, r_0 *given the secret key and possibly additional trapdoor information (which can be computed as part of the key generation procedure).*

We first show that μ_0, r_0 can be recovered from y. Assume y has error $\mathbf{e} \in \mathbb{Z}_q^{m+1}$. Since $\mathbf{e}$ is sampled from $D_{\mathbb{Z}_q^{m+1}, \beta_f}$, $\|\mathbf{e}\| \leq \sqrt{m+1}\beta_f$. Therefore, it is possible to compute μ_0 as long as $\beta_f < \frac{q}{4(m+1)}$ (see Section 5.4.1). Second, it is possible to recover the randomness r_0 of y as long as the lattice trapdoor is applicable. As

[4]Taken from [11] - specifically, the state can be created using a technique by Grover and Rudolph ([30]), who show that in order to create such a state, it suffices to have the ability to efficiently compute the sum $\sum_{x=c}^{d} D_{\mathbb{Z}_q, B_P}(x)$ for any $c, d \in \{-\lfloor\sqrt{B_P}\rfloor, \ldots, \lceil\sqrt{B_P}\rceil\} \subseteq \mathbb{Z}_q$ and to within good precision. This can be done using standard techniques used in sampling from the normal distribution.

stated in Theorem 3.2.3, the lattice trapdoor is applicable if $\beta_f < \frac{q}{C_T\sqrt{n(m+1)\log q}}$. Combining these two conditions, we require that:

$$\beta_f < \min(\frac{q}{C_T\sqrt{n(m+1)\log q}}, \frac{q}{4(m+1)}) = \frac{q}{4(m+1)} \tag{5.53}$$

The equality follows since $m = \Omega(n \log q)$. Given the definition of β_f in (5.47), this condition is satisfied by (5.44).

□

5.5 Extension to Quantum Leveled Fully Homomorphic Encryption

We now present the construction of a quantum leveled fully homomorphic encryption scheme from a quantum capable classical leveled fully homomorphic encryption scheme, as described in Section 5.1.5. We first provide a full description of the scheme (Section 5.5.1) and then proceed to proving that it is a quantum leveled FHE scheme. Correctness of evaluation follows almost immediately from the correctness of the encrypted CNOT operation (see Claim 5.3.3), while CPA security follows along the lines described in Section 5.1.5.

Assume there are L levels of the quantum circuit to be computed, where each level consists of Clifford gates, followed by a layer of non intersecting Toffoli gates. Note that this circuit arrangement increases the depth of the original circuit by a factor of at most 2. Let the depth of the classical circuit corresponding to each level of this circuit be L_c (this includes decrypting and recovering randomness from ciphertexts corresponding to the encrypted CNOT operations from the previous level, performing the Pauli key updates corresponding to the encrypted CNOT operations from the previous level, and performing the Pauli key updates corresponding to the Clifford and Toffoli gates of the current level). See Section 5.1.5 for a reminder of the above description.

The scheme is quite straightforward: the server initially receives a quantum standard basis state encrypted under Pauli keys (which can be thought of as a one time padded classical string), along with the Pauli keys encrypted under a quantum capable classical homomorphic encryption scheme (which we call HE) and a string of evaluation keys (one for each level). Each evaluation key consists of the evaluation key of HE, encrypted secret key/ trapdoor information and a fresh public key for each level. Recall that the evaluation key is of this form since we need to use a fresh public/ secret key pair for each encrypted CNOT operation; this allows encryption of the secret key/ trapdoor of each level under a new, independent public key (see Section 5.1.5). The server then applies Toffoli and Hadamard gates (which compose a universal gate set) as described in Section 5.1. Finally, the decryption consists of the client first decrypting the encryptions of the Pauli keys, and then using the Pauli keys to decrypt the final measurement result sent by the server.

Scheme 5.5.1 ***Quantum Leveled Fully Homomorphic Encryption*** *Let HE be a classical leveled fully homomorphic encryption scheme which is quantum capable for depth L_c.*

- *QHE.KeyGen($1^\lambda, 1^L$):*
  1. *For $1 \leq i \leq L+1$, let $(pk_i, evk_i, sk_i, t_{sk_i})$ =HE.Keygen($1^\lambda, 1^{L_c}$), where t_{sk_i} is the trapdoor information required for randomness recovery from ciphertexts.*
  2. *The public key pk is pk_1 and the secret key sk is sk_{L+1}. The evaluation key evk consists of $(evk_1, \ldots, evk_{L+1})$ as well as (pk_{i+1},HE.Enc$_{pk_{i+1}}(sk_i)$, HE.Enc$_{pk_{i+1}}(t_{sk_i})$) for $1 \leq i \leq L$.*
- *QHE.Enc$_{pk}(m)$: For a message $m \in \{0,1\}^\lambda$, the encryption is $(Z^z X^x |m\rangle, \mathrm{HE.Enc}_{pk_1}(z,x))$[5], where $z, x \in \{0,1\}^\lambda$ are chosen at random. Note that $Z^z X^x |m\rangle$ can be represented as the classical string $x \oplus m$.*
- *QHE.Dec$_{sk}$: The input is a classical message $m \in \{0,1\}^\lambda$ and encryptions of $z, x \in \{0,1\}^\lambda$ under pk_{L+1}. The encryptions are first decrypted using sk_{L+1} to obtain z, x. The decrypted message is $Z^z X^x |m\rangle$, which can be represented as $x \oplus m$.*
- *QHE.Eval: Clifford gates and Toffoli gates are applied to an encrypted state as follows:*
  1. *To apply a Clifford gate, the Clifford is applied to the Pauli one time padded input state and the encrypted Pauli keys are homomorphically updated according to which Clifford gate was applied.*
  2. *To apply a Toffoli gate:*
     a) *The Toffoli gate is applied to the Pauli one time padded state. Assume the Toffoli is applied on top of the Pauli one time pad $Z^z X^x \in \mathbb{P}_3$.*
     b) *The Pauli key encryptions are homomorphically updated according to P_{zx}.*
     c) *Three encrypted CNOT operations are used to correct C_{zx} (see Section 5.2.2 for details on C_{zx} and P_{zx}). As part of each operation, the Pauli key encryptions are homomorphically updated (see Claim 5.5.3 for a full description of how this is done).*

5.5.1 CPA Security

In this section, we prove the following theorem:

Theorem 5.5.1 *The scheme presented in Section 5.5.1 is IND-CPA secure.*

[5]Observe that this encryption can immediately be extended to quantum states by replacing m with a λ qubit state $|\psi\rangle$. The decryption can also be extended in the same manner.

Proof: To prove CPA security as defined in Definition 5.2.1, we show that for any polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that

$$\mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] = |\Pr[\mathcal{A}(pk, evk, \mathrm{QHE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \mathrm{QHE.Enc}_{pk}(1)) = 1]| = \mu(\lambda) \tag{5.54}$$

where $(pk, evk, sk) \leftarrow \mathrm{QHE.Keygen}(1^\lambda)$.

The only difficulty in proving (5.54) is that encryptions of sk_1 and t_{sk_1} are also given to the attacker as part of the evaluation key; we need to prove that this information can be replaced with encryptions of 0. This can be done via standard techniques in proving security of leveled homomorphic encryption schemes (see Section 4.1 in [22]). We include the proof for completeness.

We proceed through L hybrids. In the final hybrid, the attacker is given only the public key pk_1 and the evaluation key $evk' = (evk_1, pk_2, evk_2, \ldots, pk_{L+1}, evk_{L+1})$ (along with many encryptions of 0). CPA security at this point follows immediately (by replacing the encryptions of z, x with 0 and then using Lemma 5.1.1). The hybrids are as follows:

Hyb_{L+1}: The evaluation key is as described in Section 5.5.1.

For $1 \leq i \leq L$, where i is decreasing:

> Hyb_i: The evaluation key is the same as in Hyb_{i+1}, except $\mathrm{HE.Enc}_{pk_{i+1}}(t_{sk_i})$ and $\mathrm{HE.Enc}_{pk_{i+1}}(sk_i)$ are replaced with encryptions of 0.

Note that in Hyb_i, the evaluation key does not contain secret key or trapdoor information corresponding to public keys $pk_i, \ldots, pk_{L+1}$. More specifically, the evaluation key in Hyb_1 is evk' (all the encryptions of secret keys and trapdoors have been replaced by encryptions of 0).

First, Hyb_{L+1} is computationally indistinguishable from Hyb_L due to the CPA security of HE under pk_{L+1} (note that encryptions of sk_{L+1} and $t_{sk_{L+1}}$ were not provided as part of the evaluation key). For all $1 \leq i \leq L-1$, Hyb_{i+1} is indistinguishable from Hyb_i due to the CPA security of HE under pk_{i+1}. This is because Hyb_{i+1} has no secret key or trapdoor information corresponding to pk_{i+1}.

It follows that there exists a negligible function μ_C such that the CPA security of QHE

$$|\Pr[\mathcal{A}(pk, evk, \mathrm{QHE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \mathrm{QHE.Enc}_{pk}(1)) = 1]| \tag{5.55}$$

can be upper bounded as follows:

$$\begin{aligned} \cdots &\leq \mu_C L + |\Pr[\mathcal{A}(pk, evk', \mathrm{QHE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk', \mathrm{QHE.Enc}_{pk}(1)) = 1]| \\ &\leq \mu_C(L+1) \end{aligned} \tag{5.56}$$

where $(pk, evk, sk) \leftarrow \mathrm{QHE.Keygen}(1^\lambda)$ and $evk' = (evk_1, pk_2, evk_2, \ldots, pk_{L+1}, evk_{L+1})$.

□

5.5.2 Quantum Leveled FHE

In this section, we prove the following theorem:

Theorem 5.5.2 *The scheme QHE presented in Section 5.5.1 is a quantum leveled fully homomorphic encryption scheme.*

Combining Theorem 5.5.2 with Theorem 5.4.2 provides the main result of this chapter (stated informally in Theorem 1.0.1). To prove Theorem 5.5.2, we need to prove that QHE can evaluate depth L quantum circuits. This is taken care of by the following claim:

Claim 5.5.3 *Assume the underlying classical encryption scheme HE of QHE is quantum capable for depth L_c. Then there exist $z', x' \in \{0,1\}^2$ such that a BQP machine with access to a ciphertext c encrypting s under pk_i, a quantum state $Z^z X^x |\psi\rangle$ on two qubits, ciphertexts encrypting z, x under pk_i, and the evaluation key of QHE can compute a state within negligible trace distance of the following ideal state*

$$CNOT^{\,s}_{1,2} Z^{z'} X^{x'} |\psi\rangle \langle\psi| (Z^{z'} X^{x'})^\dagger (CNOT^{\,s}_{1,2})^\dagger \tag{5.57}$$

as well as the encryptions of z', x' under pk_{i+1}.

Proof: Let c_{z,x,pk_i} be the concatenation of four ciphertexts, each encrypting a single bit of z, x under pk_i. The server applies the following operations:

1. As described in Section 5.1.2, the server applies the encrypted CNOT operation to the two qubit state $Z^z X^x |\psi\rangle$ using the ciphertext $\hat{c} =$HE.Convert(c). According to Claim 5.3.3, the server will obtain a ciphertext $y =$AltHE.Enc$_{pk}(\mu_0, r_0)$, a string $d \in \{0,1\}^m$ and a state within negligible trace distance of the following ideal state:

$$(Z^{d\cdot((\mu_0,r_0)\oplus(\mu_1,r_1))} \otimes X^{\mu_0}) \text{CNOT}^s_{1,2} |\psi\rangle \tag{5.58}$$

   where AltHE.Enc$_{pk}(\mu_0; r_0) =$ AltHE.Enc$_{pk}(\mu_1; r_1) \oplus_H \hat{c}$ and $\oplus_H$ is the homomorphic XOR operation.

2. The server uses pk_{i+1} to compute HE.Enc$_{pk_{i+1}}(c_{z,x,pk_i})$ and HE.Enc$_{pk_{i+1}}(\hat{c}, y, d)$.

3. The server computes the encryption of z, x under pk_{i+1} by homomorphically running the decryption circuit on inputs HE.Enc$_{pk_{i+1}}(sk_i)$ and HE.Enc$_{pk_{i+1}}(c_{z,x,pk_i})$.

4. The server homomorphically computes (μ_0, r_0) and (μ_1, r_1), using the ciphertexts encrypting $t_{sk_i}, sk_i, \hat{c}, y, d$ (all encrypted with HE under public key pk_{i+1}). The server then uses this result, along with the ciphertexts encrypting z, x, d, to homomorphically compute $z' = z + (d \cdot ((\mu_0, r_0) \oplus (\mu_1, r_1)), 0)$ and $x' = x + (0, \mu_0)$. The result of this computation is the encryption of z', x' with HE under pk_{i+1}.

□

Theorem 5.5.2 follows from Theorem 5.5.1 and Claim 5.5.3:

*P*roof of **Theorem 5.5.2**: Theorem 5.5.1 shows QHE is IND-CPA secure. From the description of the scheme (Scheme 5.5.1), it is clear that QHE is compact. Since the number of Toffoli gates is polynomial in λ, Claim 5.5.3 (along with the triangle inequality) implies that the trace distance of the server's final state from the ideal (correct) final state is at most negligible in λ.

□

Chapter 6

Verification

In this chapter, we address the question of whether quantum computations can be classically verified. We show that as long as efficient quantum computers cannot solve the learning with errors problem, such verification is possible through interaction. In more detail, we provide a method in which an efficient classical verifier (a BPP machine) can interact with an efficient quantum prover (a BQP machine), and the prover is able to convince the verifier of the correctness of the output of the desired quantum computation; the verifier rejects with high probability if the prover returns an incorrect output.

To do so, we construct a *measurement protocol* between an efficient quantum prover and a classical verifier. The goal of this protocol is to use interaction to force the prover to behave as the verifier's trusted measurement device. In our measurement protocol, an honest prover constructs an n qubit quantum state ρ of his choice, and the verifier would like each qubit to be measured in either the standard basis or the Hadamard basis. The soundness condition of the protocol guarantees that, even in the case of a dishonest prover, the measurement result $m \in \{0,1\}^n$ obtained by the verifier is indeed the result of measuring some n qubit quantum state in the desired basis. One of the features of the measurement protocol is that the choice of basis is sent to the prover in an encrypted form, and the prover ends up with no information about the measurement basis chosen by the verifier.

The goal of using the prover as the verifier's trusted measurement device is formalized in the completeness and soundness conditions of the measurement protocol, which we now describe. Denote the choice of measurement basis by an n bit string $h = (h_1, \ldots, h_n)$. For a prover $\mathbb{P}$ and a measurement basis choice $h = (h_1, \ldots, h_n)$, we define $D_{\mathbb{P},h}$ to be the resulting distribution over the measurement result $m \in \{0,1\}^n$ obtained by the verifier. For an n qubit state ρ, we define $D_{\rho,h}$ to be the distribution obtained by measuring ρ in the basis corresponding to h. Our measurement protocol is complete, in the following sense: for all efficiently computable n qubit states ρ, there exists a prover $\mathbb{P}$ such that $D_{\mathbb{P},h}$ is approximately equal to $D_{\rho,h}$ for all h. Moreover, $\mathbb{P}$ is accepted by the verifier with all but negligible probability.

Our soundness notion for the measurement protocol is as follows. We will show that if the prover $\mathbb{P}$ is accepted by the verifier with perfect probability, there exists an efficiently

computable n qubit quantum state ρ underlying the distribution over m. More precisely, for all $h \in \{0,1\}^n$, $D_{\rho,h}$ is computationally indistinguishable from $D_{\mathbb{P},h}$. This statement can easily be made robust: we will show that if a prover $\mathbb{P}$ is accepted by the verifier on basis choice h with probability $1 - p_h$, there exists a prover $\mathbb{P}'$ who is always accepted by the verifier and the statistical distance between $D_{\mathbb{P},h}$ and $D_{\mathbb{P}',h}$ is approximately $\sqrt{p_h}$ for all h.

So far, we have described the goal of our measurement protocol, which is to force the prover to behave as the verifier's trusted measurement device. To link our measurement protocol to verification, we simply need to show that a classical verifier who also has access to a trusted standard/Hadamard basis measurement device can verify the result of BQP computations.

To describe how such verification can be done, we briefly recall a method of verifying classical computations. Assume that, for a language $L \in \mathsf{BPP}$ and an instance x, the verifier wishes to check that $x \in L$. To do so, the verifier can reduce x to a 3-SAT instance, ask the prover for a satisfying variable assignment, and verify that the assignment satisfies the instance. There is an analogous setting for quantum computations, in which the language $L \in \mathsf{BQP}$, the instance x can be reduced to a local Hamiltonian instance H_x, an n bit variable assignment corresponds to an n qubit quantum state $|\psi\rangle$, and the fraction of unsatisfied clauses corresponds to the energy of the Hamiltonian H_x with respect to the state $|\psi\rangle$ ([32]). If $x \in L$, there exists a state with low energy with respect to H_x; if not, all states will have sufficiently high energy. We will rely on the fact that the energy of $|\psi\rangle$ with respect to H_x can be estimated by performing only standard/Hadamard basis measurements ([7]).

With this analogy, verification of a quantum computation can be performed by a classical verifier with access to a trusted standard/Hadamard basis measurement device as follows ([36]): the verifier first reduces the instance x to be verified to a local Hamiltonian instance H_x, then requests an n qubit state from the prover, and finally checks if the received state has low energy with respect to H_x. If so, the verifier is assured that $x \in L$.

6.1 Overview

We now present an overview of this chapter, which proceeds as follows. Our measurement protocol relies on two cryptographic primitives which give a BPP verifier some leverage over a BQP prover. We begin by describing these primitives in Section 6.1.1. Using these primitives, we can describe our measurement protocol in Section 6.1.2. In Sections 6.1.2 - 6.1.4, we show how the two cryptographic primitives can be used to guarantee soundness of the measurement protocol, in the sense that all provers must essentially be creating a state and measuring it in the basis chosen by the verifier. In Section 6.1.5, we show how to extend our measurement protocol to a verification protocol for all of BQP.

6.1.1 Cryptographic Primitives

6.1.1.1 Trapdoor Claw-Free Families

The first cryptographic primitive we will use is a function family $\mathcal{F} = \{f_{k,b} : \mathcal{X} \to \mathcal{Y}\}$ (for $b \in \{0,1\}$) called a *trapdoor claw-free* function family. For convenience, we will assume in this overview that $\mathcal{X} = \{0,1\}^w$. A trapdoor claw-free function family is a family of functions which are two to one (both $f_{k,0}(\cdot)$ and $f_{k,1}(\cdot)$ are injective and their images are equal) and for which it is computationally difficult to find a *claw*, i.e. a pair of points x_0 and x_1 which have the same image ($f_{k,0}(x_0) = f_{k,1}(x_1)$). Given y in the image of $f_{k,0}$ or $f_{k,1}$, the trapdoor t_k of the functions $f_{k,0}, f_{k,1}$ allows recovery of both preimages of y. The trapdoor claw-free family also satisfies two hardcore bit properties, which are stronger versions of the claw-free property: roughly, they state that it is computationally difficult to find a string d and the bit $d \cdot (x_0 \oplus x_1)$, where (x_0, x_1) form a claw. These two properties are specified as needed (in Claim 6.1.1 and Claim 6.1.2).

In this overview, we will assume the existence of the function family described above for simplicity. However, since we do not know how to construct such a family, in the rest of the chapter we will instead rely on an approximate version of this family. We provide the definition of the function family we will use, which we call an extended trapdoor claw-free family, in Section 6.3, Definition 6.3.4. The construction of this family (from learning with errors) is given in Section 6.8. Both the definition and construction are extensions of those given in [11].

We now describe a BQP process we call *state commitment*, which requires a function key k corresponding to functions $f_{k,0}, f_{k,1} \in \mathcal{F}$ (we assume that the functions $f_{k,0}, f_{k,1}$ can be computed given access to the function key k). The state commitment process is performed with respect to an arbitrary single qubit state $|\psi\rangle$:

$$|\psi\rangle = \sum_{b \in \{0,1\}} \alpha_b |b\rangle \tag{6.1}$$

The commitment process consists of two steps. First, the functions $f_{k,0}, f_{k,1}$ are applied in superposition, using $|\psi\rangle$ to determine whether to apply $f_{k,0}$ or $f_{k,1}$ and a uniform superposition over $x \in \mathcal{X}$ as the input to $f_{k,0}$ or $f_{k,1}$:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b \in \{0,1\}} \sum_{x \in \mathcal{X}} \alpha_b |b\rangle |x\rangle |f_{k,b}(x)\rangle \tag{6.2}$$

Second, the final register of the resulting state is measured, obtaining $y \in \mathcal{Y}$. At this point, the state is:

$$\sum_{b \in \{0,1\}} \alpha_b |b\rangle |x_{b,y}\rangle \tag{6.3}$$

where $x_{0,y}$ and $x_{1,y}$ are the two preimages of y. We will call the qubit containing b the *committed qubit*, the register containing $x_{b,y}$ the *preimage register* and the string y the

commitment string. The crucial point here is that, due to the claw-free nature of the functions $f_{k,0}, f_{k,1}$, it is computationally difficult for a BQP machine to compute both inverses $x_{0,y}$ and $x_{1,y}$ given only y. However, with access to the trapdoor t_k, both inverses can be computed from y. If we think of the state commitment process in an interactive setting, in which the verifier selects the function key and the trapdoor and the BQP prover performs the commitment process (sending the commitment y to the verifier), the BQP prover cannot compute both inverses, but the verifier can. This gives the verifier some leverage over the prover's state.

A key property of the committed state in (6.3) is that it allows a logical Hadamard measurement up to an X Pauli operator, which is performed as follows. First, a Hadamard transform is applied to both the committed qubit and preimage register of the state in (6.3):

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{d \in \mathcal{X}} X^{d \cdot (x_{0,y} \oplus x_{1,y})} H |\psi\rangle \otimes Z^{x_{0,y}} |d\rangle \tag{6.4}$$

The next step in applying the logical Hadamard measurement is to measure the second (preimage) register, obtaining $d \in \mathcal{X}$. The state at this point is:

$$X^{d \cdot (x_{0,y} \oplus x_{1,y})} H |\psi\rangle \tag{6.5}$$

To obtain the Hadamard measurement of $|\psi\rangle$, the operator $X^{d \cdot (x_{0,y} \oplus x_{1,y})}$ (which we call the decoding operator and requires the trapdoor) is first applied, followed by a standard basis measurement of $H |\psi\rangle$. Note that these two operations commute: it is equivalent to first perform a standard basis measurement of the state in (6.5) followed by applying the X decoding operator. The X decoding operator applied after measurement is simply the classical XOR operation.

We can again think of this logical Hadamard transform in the interactive setting, in which the BQP prover applies the Hadamard transform to obtain the state in (6.4) and then measures the committed qubit and preimage register, sending the measurement results $b' \in \{0,1\}$ and $d \in \{0,1\}^w$ to the verifier. The verifier decodes the measurement b' by XORing it with $d \cdot (x_{0,y} \oplus x_{1,y})$ (which can be computed using the trapdoor) to obtain the bit m, which the verifier stores as the result of the Hadamard basis measurement.

6.1.1.2 Trapdoor Injective Function Families

The second primitive is a function family $\mathcal{G} = \{g_{k,b} : \mathcal{X} \to \mathcal{Y}\}$ (for $b \in \{0,1\}$) called a *trapdoor injective* function family. A trapdoor injective function family is a family of injective functions such that the images of $g_{k,0}(\cdot)$ and $g_{k,1}(\cdot)$ are disjoint. Given $y = g_{k,b}(x_{b,y})$, the trapdoor t_k of the functions $g_{k,0}, g_{k,1}$ allows recovery of $b, x_{b,y}$. We will also require that the trapdoor injective family is computationally indistinguishable from the trapdoor claw-free family: given a function key k, it must be computationally difficult to determine whether k belongs to an injective or claw-free family. As in the case of trapdoor claw-free families, we will assume the existence of the function family described above for the purpose of this

overview, but in the rest of the chapter we will rely on an approximate version of this function family. We define the approximate version in Section 6.3 (Definition 6.3.2) and construct it from learning with errors in Section 6.8.2.

The state commitment process described in Section 6.1.1.1 can also be performed with a function key k corresponding to functions $g_{k,0}, g_{k,1} \in \mathcal{G}$. At the stage of (6.2), the following state has been created:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{b\in\{0,1\}} \sum_{x\in\mathcal{X}} \alpha_b |b\rangle |x\rangle |g_{k,b}(x)\rangle \tag{6.6}$$

However, in this case when the last register is measured to obtain $y \in \mathcal{Y}$, the superposition over b collapses. This is because the images of $g_{k,0}(\cdot)$ and $g_{k,1}(\cdot)$ are disjoint. It follows that with probability $|\alpha_b|^2$, $y \in g_{k,b}(\cdot)$. In this case, the state after measurement of y is:

$$|b\rangle |x_{b,y}\rangle \tag{6.7}$$

where $x_{b,y}$ is the unique preimage of y.

If we think of this process in the interactive setting (in which the BQP prover performs the commitment and sends the verifier y), the verifier can use the trapdoor t_k to extract $(b, x_{b,y})$ from y. Therefore, the verifier can obtain b, which is the result of standard basis measurement of the prover's state, simply by asking for the commitment y.

6.1.2 Measurement Protocol

Given the two primitives described in Section 6.1.1, we can now describe our measurement protocol (formally given in Protocol 6.4.1). Before the protocol begins, the verifier will select the basis for which he would like a measurement result (either Hadamard or standard) for each of n qubits. We will represent this basis choice by a string $h \in \{0,1\}^n$. If $h_i = 0$, it indicates that the standard basis was chosen for qubit i.

The protocol begins with the commitment round. For all i for which $h_i = 1$, the verifier samples a key k_i and a trapdoor t_{k_i} corresponding to a pair of trapdoor claw-free functions $f_{k_i,0}, f_{k_i,1} \in \mathcal{F}$. For all i for which $h_i = 0$, the verifier samples a key k_i and a trapdoor t_{k_i} corresponding to a pair of trapdoor injective functions $g_{k_i,0}, g_{k_i,1} \in \mathcal{G}$. The verifier sends the prover the function keys $k_1, \ldots, k_n$. The honest prover first constructs an n qubit state of his choice and then commits to each qubit of the state using the corresponding function key, as described in Section 6.1.1. The verifier receives the measurement results $y_1, \ldots, y_n \in \mathcal{Y}$ from the prover.

Once the verifier receives $y_1, \ldots, y_n$, he chooses at random either to run a test round or a Hadamard round. In the case of a test round, the verifier requests standard basis measurements of the committed qubit and preimage register for all n qubits. He receives $b'_i \in \{0,1\}$ and $x'_i \in \mathcal{X}$ from the prover and checks if the measurement is a preimage of y_i.

In the case of a Hadamard round, the verifier requests a Hadamard basis measurement of the committed qubit and preimage register for all i. The verifier receives $b'_i \in \{0,1\}, d_i \in \mathcal{X}$ for all i. For all i for which $h_i = 0$, the verifier ignores the measurement results b'_i and d_i

and uses the trapdoor of g_{k_i} to invert y_i. The verifier stores the first bit of the inverted value as the standard basis measurement result m_i. For all i for which $h_i = 1$, the verifier first decodes b'_i by XORing it with $d_i \cdot (x_{0,y_i} \oplus x_{1,y_i})$ (this can equivalently be thought of as applying the decoding operator $X^{d \cdot (x_{0,y_i} \oplus x_{1,y_i})}$- see the end of Section 6.1.1.1). The verifier stores the result $m_i = b'_i \oplus d_i \cdot (x_{0,y_i} \oplus x_{1,y_i})$ as the Hadamard basis measurement result.

Completeness of our measurement protocol (as defined at the start of Section 6.1) follows immediately from the description of the state commitment process given in Sections 6.1.1.1 and 6.1.1.2.

6.1.3 Measurement Protocol Soundness

We now give an overview of our soundness guarantee: we describe how to show that for $n = 1$ and a prover $\mathbb{P}$ who passes the test round perfectly, there exists a quantum state underlying the distribution over measurement results obtained by the verifier. The generalization to arbitrary n (given in Section 6.6) follows easily due to the fact that all n function keys are drawn independently. The generalization to provers $\mathbb{P}$ who do not pass perfectly (also given in Section 6.6) is straightforward as well; it is done by conditioning $\mathbb{P}$ on acceptance in a test round, thereby creating an efficient prover who passes the test round perfectly as long as $\mathbb{P}$ is accepted with non negligible probability. In this section, we begin by characterizing the behavior of a general prover. We then show that if this characterization satisfies a certain requirement, we can prove the existence of an underlying quantum state. In Section 6.1.4 (which is the crux of this chapter), we show how to enforce this requirement on general provers.

6.1.3.1 Prover Behavior

The analysis of the measurement protocol is based on understanding and characterizing the prover's Hilbert space and operations. We will rely on the following principle behind interactive proofs between a BQP prover and a BPP verifier. A round of the protocol begins with the verifier's message and ends with the prover's message. A general prover is equivalent, from the verifier's perspective, to a prover who begins each round by applying an arbitrary unitary operator to his space and then behaves exactly the same as an honest prover, culminating in a measurement (the result of which is sent to the verifier). This principle implies that an arbitrary prover measures the same registers that an honest prover does in each round, which will be particularly useful in our protocol.

Let $\mathbb{P}_0$ be an honest prover in our measurement protocol and assume the unitary operator he applies in the commitment round is $U_{C,0}$, after which he measures the commitment string register in the standard basis. As described in Section 6.1.1, $\mathbb{P}_0$ has three designated registers: the register containing the committed qubit, the preimage register, and the commitment string register. Each message of $\mathbb{P}_0$ to the verifier is the result of the measurement of one of these registers.

It follows from the principle above that a general prover $\mathbb{P}$ has the same designated registers as $\mathbb{P}_0$ and is characterized by 3 unitary operators: the unitary U_C applied in the commitment round, the unitary U_T applied in the test round, and the unitary U_H applied in the Hadamard round. We assume that both U_T and U_H do not act on the commitment string register, since it has already been measured; the measurement result could have been copied into the auxiliary space, on which U_T and U_H can act.

We now use the structure of our protocol to simplify the general prover one step further. There are only two possible messages of the verifier for the second round of our protocol: the message indicates either a test or Hadamard round. Due to this property, we can assume that the test round attack U_T is equal to the identity operator. To see this, we only need to make one observation: the attack U_T applied in the test round commutes with the measurement of the commitment string. Therefore, it could have been applied prior to reporting the commitment string.

It follows that the general prover $\mathbb{P}$ described above is identical, from the perspective of the verifier, to a prover who applies the unitary $U_0 = U_T U_{C,0} U_C$ immediately prior to measuring the commitment string register and applies $U = U_H U_T^\dagger$ prior to performing Hadamard basis measurements of the committed qubit and preimage register in the Hadamard round. We will say that such a prover is *characterized* by (U_0, U). For a formal statement and proof of the above argument, see Claim 6.4.4.

The characterization of all provers by two unitary attacks allows us to use the test round of the measurement protocol to enforce that the prover's state has a specific structure, which is derived from the cryptographic primitives in Section 6.1.1. Let $\mathbb{P}$ be a prover who passes the test round perfectly. If $h = 1$, the state of $\mathbb{P}$ at the start of either the test or the Hadamard round (i.e. immediately after reporting y) can be written as follows (the two preimages of y are $x_{0,y}, x_{1,y}$):

$$\sum_{b\in\{0,1\}} |b\rangle |x_{b,y}\rangle \left|\psi_{b,x_{b,y}}\right\rangle \tag{6.8}$$

where $\left|\psi_{b,x_{b,y}}\right\rangle$ contains all additional qubits held by the prover. This is because the verifier checks, in a test round, if he receives a valid pre-image from the prover. Since the prover simply measures the requested registers when asked by the verifier in a test round (i.e. he does not apply an attack in the test round), these registers must be in a superposition over the two preimages of the reported measurement result y.

If $h = 0$ and $\mathbb{P}$ reports y, there is only one inverse of y. If we assume this inverse is $x_{b,y}$ (i.e. $g_{k,b}(x_{b,y}) = y$), the state of $\mathbb{P}$ at the start of the test or Hadamard round can be written as follows, due to the same reasoning used in (6.8):

$$|b\rangle |x_{b,y}\rangle \left|\psi_{b,x_{b,y}}\right\rangle \tag{6.9}$$

This structure enforced by the test run is the key to proving the existence of an underlying quantum state, as we will see shortly.

6.1.3.2 Construction of Underlying Quantum State

We begin by using the characterization of general provers in Section 6.1.3.1 to define a single qubit state ρ corresponding a prover $\mathbb{P}$ who is characterized by (U_0, U). Recall that $\mathbb{P}$ has a well defined committed qubit, which he measures when the verifier asks for the measurement of a committed qubit. Let ρ' be the state of the committed qubit prior to the prover's measurement in the Hadamard round in the case that $h = 1$. We can think of ρ' as encoded by the operator $Z^{d\cdot(x_{0,y}\oplus x_{1,y})}$, which is determined by the prover's measurements d and y. This Z operator is derived from the verifier's X decoding operator applied in the measurement protocol; we have used a Z operator here since the Hadamard measurement has not yet been performed. The single qubit state ρ will be the result of applying the Z decoding operator to the committed qubit ρ'.

Define X-trivial operators to be those which commute with standard basis measurement of the committed qubit. We now show that if the prover's Hadamard round attack U is an X-trivial operator, the distribution $D_{\mathbb{P},h}$ obtained by the verifier in the measurement protocol is computationally indistinguishable from the distribution which is obtained by measuring ρ in basis specified by h.

Recall that $D_{\rho,h}$ is the distribution obtained by measuring ρ in the basis corresponding to h. By construction, $D_{\rho,1} = D_{\mathbb{P},1}$. If $h = 0$, there are two differences between the distribution $D_{\rho,h}$ and the distribution $D_{\mathbb{P},h}$. The first differences lies in the function sampling: in our measurement protocol, an injective function is sampled if $h = 0$, but in the state ρ, a claw-free function is sampled. The second difference comes from how the standard basis measurement is obtained: in $D_{\mathbb{P},h}$ the standard basis measurement is obtained from the commitment string y, but in $D_{\rho,h}$ the standard basis measurement is obtained by measuring ρ (the committed qubit) in the standard basis.

We can handle the first difference by making two key observations. First, the Z decoding operator has no effect if $h = 0$; in this case, the committed qubit will be measured in the standard basis immediately after application of Z in order to obtain $D_{\rho,h}$. Second, if the Z decoding operator is not applied, the trapdoor t_k is no longer needed to construct the distribution $D_{\rho,h}$. If $D_{\rho,h}$ is only dependent on the function key k (and not the trapdoor t_k), the function key k can be replaced with a function key which corresponds to a pair of trapdoor injective functions, rather than a pair of trapdoor claw-free functions, to obtain a computationally indistinguishable distribution. This is due to the computational indistinguishability between keys drawn from the trapdoor claw-free family $\mathcal{F}$ and the trapdoor claw-free family $\mathcal{G}$.

Let ρ_0 be the committed qubit of the prover prior to measurement in the Hadamard round in the case that $h = 0$. Due to the argument above, the distribution $D_{\rho,0}$ is computationally indistinguishable from $D_{\rho_0,0}$. To address the second difference, we now show that measuring ρ_0 in the standard basis produces the same distribution obtained from extracting the standard basis measurement from the commitment string y. First, note that measuring the committed qubit prior to application of U (i.e. at the start of the Hadamard round) results in the same measurement obtained from y; as seen in (6.9), the value of the commit-

ted qubit is equal to the value m extracted from y, since the prover passes the test round perfectly. To complete our proof, recall that U is X-trivial with respect to the committed qubit, and therefore commutes with standard basis measurement of the committed qubit.

To recap, the argument above shows that there exists a quantum state underlying the distribution $D_{\mathbb{P},h}$ as long as the prover's attack operator in the Hadamard round is an X-trivial operator. For a formal statement and complete proof of this argument, see Claim 6.4.7.

6.1.4 Replacement of a General Attack with an X-Trivial Attack

We can now proceed to the crux of this chapter: assuming that $n = 1$ and $\mathbb{P}$ passes the test round perfectly, we show that there exists a prover $\mathbb{P}'$ such that $D_{\mathbb{P},h}$ is computationally indistinguishable from $D_{\mathbb{P}',h}$ for both h and $\mathbb{P}'$ attacks with an X-trivial operator in the Hadamard round. By the argument in Section 6.1.3.2 and the triangle inequality, this implies that there exists a state ρ for which $D_{\mathbb{P},h}$ and $D_{\rho,h}$ are computationally indistinguishable, thereby proving our soundness guarantee.

Assume $\mathbb{P}$ is characterized by (U_0, U). Then $\mathbb{P}'$ is characterized by $(U_0, \{U_x\}_{x\in\{0,1\}})$, where $\{U_x\}_{x\in\{0,1\}}$ is an X-trivial CPTP map:

$$U = \sum_{x,z\in\{0,1\}} X^x Z^z \otimes U_{xz} \tag{6.10}$$

$$U_x = \sum_{z\in\{0,1\}} Z^z \otimes U_{xz} \tag{6.11}$$

Observe that if $h = 0$, $D_{\mathbb{P},h} = D_{\mathbb{P}',h}$; this is simply because the standard basis measurement is obtained from the commitment y, which is measured prior to the Hadamard round attack U. This argument requires a bit more detail for $n > 1$ and is given formally in Claim 6.6.4. We proceed to describing how to replace the attack U in (6.10) with the CPTP map $\{U_x\}_{x\in\{0,1\}}$ in (6.11) in the case that the verifier chooses the Hadamard basis ($h = 1$). We will rely heavily on the structure of the prover's state, as written in (6.8).

The replacement of U with $\{U_x\}_{x\in\{0,1\}}$ will be done by using the Z Pauli twirl (Corollary 6.2.2). The Z Pauli twirl is a technique which allows the replacement of U with the CPTP map $\{U_x\}_{x\in\{0,1\}}$ by conjugating U by a random Z Pauli operator. More formally, Corollary 6.2.2 states that the following two CPTP maps are equivalent when followed by Hadamard basis measurement:

$$\{\frac{1}{\sqrt{2}}(Z^r \otimes \mathcal{I})U(Z^r \otimes \mathcal{I})\}_{r\in\{0,1\}} \tag{6.12}$$

$$\{U_x\}_{x\in\{0,1\}} \tag{6.13}$$

To apply the Z Pauli twirl in this setting, it suffices to show that replacing the prover's attack U with the unitary attack $(Z \otimes \mathcal{I})U(Z \otimes \mathcal{I})$ results in a computationally indistinguishable distribution.

To prove this statement, we will rely on the fact that there is already computational randomness, due to the trapdoor claw-free function, which is hiding both the Z operator applied prior to U and the Z operator applied after. The computational randomness hiding the posterior Z operator comes from the verifier's decoding operator $X^{d\cdot(x_{0,y}\oplus x_{1,y})}$ applied at the end of the measurement protocol (see Section 6.1.2); if this decoding operator is shifted prior to the Hadamard transform on the committed qubit, it acts as a Z operator immediately after the attack U. The computational randomness hiding the anterior Z operator results from the format of the prover's state. Recall that, since the prover is perfect, we can assume the prover begins the Hadamard round with a state of the form in (6.8):

$$|\phi_y\rangle = \sum_{b\in\{0,1\}} |b\rangle\,|x_{b,y}\rangle\,\left|\psi_{b,x_{b,y}}\right\rangle \tag{6.14}$$

Applying a Hadamard transform to the preimage register of the state in (6.14) results in a Z operator acting on the committed qubit:

$$(\mathcal{I}\otimes H^{\otimes w}\otimes\mathcal{I})\sum_{b\in\{0,1\}} |b,x_{b,y}\rangle\left|\psi_{b,x_{b,y}}\right\rangle = \frac{1}{\sqrt{|\mathcal{X}|}}\sum_{d\in\mathcal{X},b\in\{0,1\}} Z^{d\cdot(x_{0,y}\oplus x_{1,y})}\,|b\rangle\otimes Z^{x_{0,y}}\,|d\rangle\otimes\left|\psi_{b,x_{b,y}}\right\rangle \tag{6.15}$$

In order to use these two sources of computational randomness to hide the difference between U and $(Z\otimes\mathcal{I})U(Z\otimes\mathcal{I})$, it must be the case that the bit $d\cdot(x_{0,y}\oplus x_{1,y})$ is computationally indistinguishable from a uniformly random bit. Formalizing this requirement is a bit tricky, since d is sampled from the state created by the prover. In the next section, we show how to prove computational indistinguishability between the distributions resulting from U and $(Z\otimes\mathcal{I})U(Z\otimes\mathcal{I})$. As part of this process, we formalize the computational randomness requirement regarding $d\cdot(x_{0,y}\oplus x_{1,y})$ as two different hardcore bit conditions for the function pair $f_{k,0}, f_{k,1}$.

6.1.4.1 Computational Indistinguishability of Phase Flip

Let $\mathbb{P}$ be the prover characterized by (U_0, U) and let $\hat{\mathbb{P}}$ be the prover characterized by $(U_0, (Z\otimes\mathcal{I})U(Z\otimes\mathcal{I}))$. In this section, we will show that the distributions resulting from the two provers ($D_{\mathbb{P},h}$ and $D_{\hat{\mathbb{P}},h}$) are computationally indistinguishable for all h. For convenience, we will instead refer to these two distributions as mixed states; let σ_0 be the mixed state corresponding to $D_{\mathbb{P},h}$ and let σ_1 be the mixed state corresponding to $D_{\hat{\mathbb{P}},h}$, i.e.

$$\sigma_0 = \sum_{m\in\{0,1\}} D_{\mathbb{P},h}(m)\,|m\rangle\langle m| \tag{6.16}$$

To prove the computational indistinguishability of σ_0 and σ_1, each state is split into two terms (for $r\in\{0,1\}$):

$$\sigma_r = \sigma_r^D + \sigma_r^C \tag{6.17}$$

By a straightforward application of the triangle inequality, we obtain that if σ_0 is computationally distinguishable from σ_1, either σ_0^D and σ_1^D are computationally distinguishable or σ_0^C and σ_1^C are. Note that even if the terms are not quantum states, the notion of computational indistinguishability (Definition 3.3.2) is still well defined: to show that two terms, for example σ_0^C and σ_1^C, are computationally indistinguishable, we need to show (informally) that there does not exist an efficiently computable CPTP map $\mathcal{S}$ such that the following expression is non negligible

$$|\mathrm{Tr}((|0\rangle\langle 0| \otimes \mathcal{I})\mathcal{S}(\sigma_0^C - \sigma_1^C)| \tag{6.18}$$

In more detail, the density matrices σ_0 and σ_1 are created by beginning with the state $|\phi_y\rangle$ in (6.14) and applying the operations of both the prover and verifier in the Hadamard round, followed by tracing out all but the first qubit. Therefore, to split σ_0 and σ_1 into two parts, we can equivalently split the density matrix of $|\phi_y\rangle$ into the following two parts, corresponding to the diagonal and cross terms:

$$\sum_{b\in\{0,1\}} |b\rangle\langle b| \otimes |x_{b,y}\rangle\langle x_{b,y}| \otimes \left|\psi_{b,x_{b,y}}\right\rangle\left\langle\psi_{b,x_{b,y}}\right| \tag{6.19}$$

$$\sum_{b\in\{0,1\}} |b\rangle\langle b\oplus 1| \otimes |x_{b,y}\rangle\langle x_{b\oplus 1,y}| \otimes \left|\psi_{b,x_{b,y}}\right\rangle\left\langle\psi_{b,x_{b\oplus 1,y}}\right| \tag{6.20}$$

Let σ_0^D and σ_1^D be the result of applying the operations of both the prover and the verifier in the Hadamard round to (6.19), followed by tracing out all but the first qubit. Recall the difference between σ_0^D and σ_1^D: in the latter, the prover's attack U is conjugated by $(Z \otimes \mathcal{I})$. Define σ_0^C and σ_1^C similarly, but replace (6.19) with (6.20). In the following two sections, we show that both pairs of terms are computationally indistinguishable.

Diagonal Terms In this section, we will show that if there exists a BQP attacker $\mathcal{A}'$ who can distinguish between the terms σ_0^D and σ_1^D, then there exists a BQP attacker $\mathcal{A}$ who can violate the following informal hardcore bit property of the function family $\mathcal{F}$ (the formal statement is part of Definition 6.3.1):

Claim 6.1.1 *Assume $f_{k,0}$ and $f_{k,1}$ are sampled from a trapdoor claw-free family $\mathcal{F}$. Then there does not exist an attacker who, on input k, can produce $b \in \{0,1\}$, $x_b \in \mathcal{X}$, $d \in \{0,1\}^w \setminus \{0^w\}$ and $c \in \{0,1\}$ such that $c = d \cdot (x_0 \oplus x_1)$ where $f_{k,0}(x_0) = f_{k,1}(x_1)$.*

We first describe the state σ_0^D, which is created by beginning with the state in (6.19), in more detail. Note that the state in (6.19) can be efficiently created by following the prover's commitment process and then measuring the committed qubit and preimage register. To create σ_0^D, the attack U is applied to the state in (6.19), followed by Hadamard measurement of the committed qubit and preimage register and application of the verifier's X decoding operator. Finally, all qubits but the first are traced out. σ_1^D is almost the same as σ_0^D,

except the attack U is replaced with the attack $(Z \otimes \mathcal{I})U(Z \otimes \mathcal{I})$. Note that the initial phase operator has no effect, since it acts on the diagonal state in (6.19). The final phase flip, once it is shifted past the Hadamard transform, is equivalent to flipping the decoding bit of the verifier; it follows that $\sigma_1^D = X\sigma_0^D X$.

We now construct the BQP attacker $\mathcal{A}$ who will use $\mathcal{A}'$ to violate Claim 6.1.1. Let σ_D be the state σ_0^D except for the verifier's decoding. Observe from the description in the previous paragraph that this state can be efficiently created, and as part of creating the state, the measurements $b, x_{b,y}$ and d are obtained. The attacker $\mathcal{A}$ creates the state σ_D. For the string d obtained by $\mathcal{A}$, the decoding bit $d \cdot (x_{0,y} \oplus x_{1,y})$ determines which of the two states σ_0^D and σ_1^D $\mathcal{A}$ has created; if $d \cdot (x_{0,y} \oplus x_{1,y}) = r$, $\mathcal{A}$ has created σ_r^D. Now $\mathcal{A}$ can run $\mathcal{A}'$ on the resulting mixed state in order to learn $d \cdot (x_{0,y} \oplus x_{1,y})$. As a result, $\mathcal{A}$ holds the following information: $b, x_{b,y}, d$, and $d \cdot (x_0 \oplus x_1)$, therefore violating Claim 6.1.1.

Cross Terms In this section, we will show that the cross terms σ_0^C and σ_1^C are computationally indistinguishable. Since the cross terms are not quantum states, we first show below that if there exists a CPTP map $\mathcal{S}$ which distinguishes between σ_0^C and σ_1^C, i.e. if the following expression is non negligible:

$$|\mathrm{Tr}((|0\rangle\langle 0| \otimes \mathcal{I})\mathcal{S}(\sigma_0^C - \sigma_1^C)| \tag{6.21}$$

then there exists an efficiently computable CPTP map $\mathcal{S}'$ such that the CPTP map $\mathcal{S}\mathcal{S}'$ distinguishes between the quantum states $\hat{\sigma}_0$ and $\hat{\sigma}_1$, defined as follows. The density matrix $\hat{\sigma}_r$ corresponds to the following pure state (recall $|\phi_y\rangle$ from (6.14)):

$$(Z^r \otimes \mathcal{I})|\phi_y\rangle = (Z^r \otimes \mathcal{I})(\sum_{b\in\{0,1\}} |b\rangle |x_{b,y}\rangle |\psi_{b,x_{b,y}}\rangle) \tag{6.22}$$

To do this, it suffices to show that $\sigma_0^C - \sigma_1^C = \mathcal{S}'(\hat{\sigma}_0 - \hat{\sigma_1})$. This equality is straightforward for two reasons. First, $\frac{1}{2}(\hat{\sigma}_0 - \hat{\sigma_1})$ is equal to the cross term in (6.20). Second, both σ_0^C and σ_1^C also begin with (6.20), but followed by a CPTP map which is inefficient due to the verifier's decoding. To prove the existence of $\mathcal{S}'$, we show that taking the difference between σ_0^C and σ_1^C effectively removes the verifier's decoding, creating an efficient CPTP map $\mathcal{S}'$.

Finally, we will show that an attacker who can distinguish between $\hat{\sigma}_0$ and $\hat{\sigma}_1$ can violate the following informal hardcore bit property of the function family $\mathcal{F}$ (the formal statement is part of Definition 6.3.4):

Claim 6.1.2 *Assume $f_{k,0}$ and $f_{k,1}$ are sampled from a trapdoor claw-free family $\mathcal{F}$. Then there exists $d \in \{0,1\}^w$ which satisfies two conditions. First, there exists a bit c_k such that $d \cdot (x_0 \oplus x_1) = c_k$ for all claws (x_0, x_1) ($f_{k,0}(x_0) = f_{k,1}(x_1)$). Second, there does not exist an attacker who, on input k, can determine the bit c_k.*

We begin by describing the cross term σ_0^C (which is not a quantum state) in more detail. σ_0^C is created by beginning with the expression in (6.20), copied here for reference:

$$\sum_{b\in\{0,1\}} |b\rangle\langle b \oplus 1| \otimes |x_{b,y}\rangle\langle x_{b\oplus 1,y}| \otimes |\psi_{b,x_{b,y}}\rangle\langle\psi_{b,x_{b\oplus 1,y}}| \tag{6.23}$$

then applying the attack U, followed by Hadamard measurement of the committed qubit and preimage register and application of the verifier's X decoding operator. Finally, all qubits but the first are traced out. σ_1^C is almost the same, except the attack U is replaced with the attack $(Z \otimes \mathcal{I})U(Z \otimes \mathcal{I})$. As in Section 6.1.4.1, the phase flip acting after U is equivalent to flipping the decoding operator of the verifier (i.e. applying an X operator to the matrix σ_0^C). The initial phase flip, which acts on the first qubit of (6.23), results in a phase of -1. Combining these two observations yields the following equality:

$$\sigma_1^C = -X\sigma_0^C X \tag{6.24}$$

Taking the difference between σ_0^C and σ_1^C results in a matrix which has a uniform X operator applied:

$$\sigma_0^C - \sigma_1^C = \sum_{r\in\{0,1\}} X^r \sigma_0^C X^r \tag{6.25}$$

Observe that the CPTP map applied to (6.23) to create σ_0^C is efficiently computable except for the verifier's X decoding operator. In (6.25), there is a uniform X operator acting on σ_0^C, effectively replacing the verifier's decoding operator. Let $\mathcal{S}'$ be the resulting efficiently computable CPTP map. It follows immediately that $\sigma_0^C - \sigma_1^C = \mathcal{S}'(\hat{\sigma}_0 - \hat{\sigma}_1)$.

We now proceed to showing that an attacker $\mathcal{A}'$ who can distinguish between $\hat{\sigma}_0$ and $\hat{\sigma}_1$ can be used to violate Claim 6.1.2. Since the state $\hat{\sigma}_r$ is the state $|\phi_y\rangle$ from (6.14) with the operator Z^r applied to the committed qubit, an attacker who can distinguish between $\hat{\sigma}_0$ and $\hat{\sigma}_1$ can distinguish whether or not a Z operator is applied to the committed qubit of $|\phi_y\rangle$. The following equality (which holds up to a global phase) shows that a Z operator on the preimage register is equivalent to a Z operator on the committed qubit:

$$(\mathcal{I} \otimes Z^d \otimes \mathcal{I})(\sum_{b\in\{0,1\}} |b\rangle |x_{b,y}\rangle \left|\psi_{b,x_{b,y}}\right\rangle) = (Z^{d\cdot(x_{0,y}\oplus x_{1,y})} \otimes \mathcal{I})(\sum_{b\in\{0,1\}} |b\rangle |x_{b,y}\rangle \left|\psi_{b,x_{b,y}}\right\rangle) \tag{6.26}$$

This equality, along with the attacker $\mathcal{A}'$, can be used to construct a BQP attacker $\mathcal{A}$ who can determine $d \cdot (x_{0,y} \oplus x_{1,y})$ for an arbitrary fixed string d. $\mathcal{A}$ first constructs $|\phi_y\rangle$ (this is simply the prover's state after reporting the commitment string y, so it can be constructed efficiently). Next, $\mathcal{A}$ applies Z^d to the preimage register of $|\phi_y\rangle$. Due to the equality in (6.26), this is equivalent to instead applying $Z^{d\cdot(x_{0,y}\oplus x_{1,y})}$ to the committed qubit. By running the attacker $\mathcal{A}'$, $\mathcal{A}$ can determine $d \cdot (x_{0,y} \oplus x_{1,y})$, therefore violating Claim 6.1.2.

6.1.5 Extension of Measurement Protocol to a Verification Protocol for BQP

Our goal is to verify that an instance $x \in L$ for a language $L \in$ BQP. Recall that each instance can be converted into a local Hamiltonian H with the following property: if $x \in L$, H has ground state energy at most a and if $x \notin L$, H has ground state energy at least b, where the gap $b - a$ is inverse polynomial. Therefore, to verify that an instance $x \in L$,

a verifier with a quantum computer can simply ask the prover for the ground state and estimate the energy of the received state with respect to the Hamiltonian H. The soundness of such a protocol rests on the fact that if an instance $x \notin L$, all possible states sent by the prover will have energy $\geq b$.

To use such a verification procedure in our setting, we need to rely on one more fact: the Hamiltonian H can be written as a sum over terms which are each a product of X and Z operators [7]. Therefore, when the verifier is estimating the energy of a state sent by the prover, he only needs to perform Hadamard or standard basis measurements on each individual qubit. In [36], the authors formalize the resulting protocol and use it to build a protocol in which a verifier with access to a single qubit register can verify the result of a BQP computation. Their protocol achieves a completeness/ soundness gap which is negligibly close to 1 by performing polynomially many repetitions of the energy estimation process described above.

In [36], the prover sends single qubits to the verifier, who performs either Hadamard or standard basis measurements. To obtain a verification protocol for BQP, we simply replace this step of their protocol with our measurement protocol. Completeness and soundness follow, since our measurement protocol allows the verifier to collect standard and Hadamard basis measurements of a given state, and our soundness claim guarantees that the distribution over measurement results obtained by the verifier comes from the measurement of an underlying quantum state. The extension of our measurement protocol to a verification protocol is described in Section 6.7.

6.1.6 Chapter Outline

As noted in Section 6.1.1.1, the trapdoor function families we use in the rest of the chapter are not the ideal families used so far in the overview. We instead use approximations of these function families, which are defined in Section 6.3. The protocol as described used several properties of the ideal families. We take care to define our approximate families to make sure that they satisfy these required properties, at the expense of additional notation. At the start of Section 6.3, there is a summary of the differences between the approximate functions and the ideal functions (taken from [11]).

We begin with the definition of our extended trapdoor claw-free family in Section 6.3. Section 6.4 covers Sections 6.1.1 to 6.1.3 of the overview. In Section 6.5, we present the argument outlined in Section 6.1.4, in which we replace a general attack with an attack which commutes with standard basis measurement. In Section 6.6, we prove the soundness of the measurement protocol. Finally, the extension of the measurement protocol to a QPIP_0 (described in Section 6.1.5) is given in Section 6.7, providing the main result of this chapter (the following informal statement is taken from Chapter 1 and is stated formally as Theorem 6.7.6 in Section 6.7):

Theorem 1.0.2 (Informal) *Assuming the existence of an extended trapdoor claw-free family as defined in Definition 6.3.4,* $\mathsf{BQP} = \mathsf{QPIP}_0$.

In Section 6.8, we provide a learning with errors based construction of an extended trapdoor family, providing a proof of the following theorem (the following informal statement is taken from Chapter 1 and is stated formally as Theorem 6.8.1 in Section 6.8):

Theorem 1.0.3 (Informal) *Under the assumption that the learning with errors problem with superpolynomial noise ratio is computationally intractable for an efficient quantum machine, there exists an extended trapdoor claw-free family.*

6.2 Preliminaries

6.2.1 Pauli Twirl

We call the conjugation of a unitary operator (or a CPTP map) by a random Pauli a Pauli twirl ([19]). The twirled version of a unitary U is the CPTP map $\{(X^x Z^z)^\dagger U (X^x Z^z)\}_{x,z}$. If the Pauli is a random X (or Z) Pauli operator, we call the conjugation an X (or Z) Pauli twirl. A Z Pauli twirl has the following effect:

Lemma 6.2.1 ***Z Pauli Twirl*** *For a CPTP map with Kraus operators $\{B_\tau\}_\tau$, the following two CPTP maps are equal:*

$$\left\{ \frac{1}{\sqrt{2}} (Z^r \otimes \mathcal{I}) B_\tau (Z^r \otimes \mathcal{I}) \right\}_{r \in \{0,1\}, \tau} = \{ (X^x \otimes \mathcal{I}) B'_{x,\tau} \}_{x \in \{0,1\}, \tau} \tag{6.27}$$

where $B_\tau = \sum_{x,z \in \{0,1\}} X^x Z^z \otimes B_{xz\tau}$ and the CPTP map $\{B'_{x,\tau}\}_{x \in \{0,1\},\tau}$ is equal to $B'_{x,\tau} = \sum_{z \in \{0,1\}} Z^z \otimes B_{xz\tau}$.

Proof: To prove the lemma, we show that applying either of the two CPTP maps in equation (6.27) on an arbitrary density matrix ρ results in the same state. We begin with the CPTP map on the left of (6.27):

$$\frac{1}{2} \sum_{r \in \{0,1\}, \tau} (Z^r \otimes \mathcal{I}) B_\tau (Z^r \otimes \mathcal{I}) \rho (Z^r \otimes \mathcal{I}) B_\tau^\dagger (Z^r \otimes \mathcal{I})^\dagger \tag{6.28}$$

Using the fact that $B_\tau = \sum_{x,z \in \{0,1\}} X^x Z^z \otimes B_{xz\tau}$, we can rewrite the expression from (6.28) as:

$$\frac{1}{2} \sum_{\substack{r \in \{0,1\}, \tau \\ x,z,x',z' \in \{0,1\}}} (Z^r X^x Z^z Z^r \otimes B_{xz\tau}) \rho (Z^r X^{x'} Z^{z'} Z^r \otimes B_{x'z'\tau})^\dagger \tag{6.29}$$

Next, we use the anti commutation properties of Pauli operators (by commuting Z^r with both X^x and $X^{x'}$) to obtain the following state:

$$\frac{1}{2} \sum_{\substack{r\in\{0,1\},\tau \\ x,z,x',z'\in\{0,1\}}} (-1)^{r\cdot(x\oplus x')} (X^x Z^z \otimes B_{xz\tau}) \rho (X^{x'} Z^{z'} \otimes B_{x'z'\tau})^\dagger \tag{6.30}$$

At this point, we can sum over r to obtain $x = x'$, resulting in the following expression:

$$\sum_{\substack{\tau \\ x,z,z'\in\{0,1\}}} (X^x Z^z \otimes B_{xz\tau}) \rho (X^x Z^{z'} \otimes B_{x'z\tau})^\dagger \tag{6.31}$$

$$= \sum_{x\in\{0,1\},\tau} (X^x \otimes \mathcal{I}) B'_{x,\tau} \rho ((X^x \otimes \mathcal{I}) B'_{x,\tau})^\dagger \tag{6.32}$$

□

The following corollary follows from Lemma 6.2.1 and captures the effect of a Z Pauli twirl followed by Hadamard basis measurement:

Corollary 6.2.2 ***Z Pauli Twirl with Measurement*** *For a CPTP map with Kraus operators $\{B_\tau\}_\tau$, the following two CPTP maps are equal:*

$$\left\{ \frac{1}{\sqrt{2}} (|b\rangle \langle b| H Z^r \otimes \mathcal{I}) B_\tau (Z^r \otimes \mathcal{I}) \right\}_{b,r\in\{0,1\},\tau} = \{ (|b\rangle \langle b| H \otimes \mathcal{I}) B'_{x,\tau} \}_{b,x\in\{0,1\},\tau} \tag{6.33}$$

where $B_\tau = \sum_{x,z\in\{0,1\}} X^x Z^z \otimes B_{xz\tau}$ and the CPTP map $\{B'_{x,\tau}\}_{x\in\{0,1\},\tau}$ is equal to $B'_{x,\tau} = \sum_{z\in\{0,1\}} Z^z \otimes B_{xz\tau}$.

Proof: We begin by applying Lemma 6.2.1 to obtain the following equality:

$$\left\{ \frac{1}{\sqrt{2}} (|b\rangle \langle b| H Z^r \otimes \mathcal{I}) B_\tau (Z^r \otimes \mathcal{I}) \right\}_{b,r\in\{0,1\},\tau} = \{ (|b\rangle \langle b| H X \otimes \mathcal{I}) B'_{x,\tau} \}_{b,x\in\{0,1\},\tau} \tag{6.34}$$

To prove the corollary, we show that

$$\{ (|b\rangle \langle b| H X \otimes \mathcal{I}) B'_{x,\tau} \}_{b,x\in\{0,1\},\tau} = \{ (|b\rangle \langle b| H \otimes \mathcal{I}) B'_{x,\tau} \}_{x\in\{0,1\},\tau} \tag{6.35}$$

This is straightforward and only requires one intermediate step:

$$\{ (|b\rangle \langle b| H X \otimes \mathcal{I}) B'_{x,\tau} \}_{b,x\in\{0,1\},\tau} = \{ (|b\rangle \langle b| Z H \otimes \mathcal{I}) B'_{x,\tau} \}_{x\in\{0,1\},\tau} \tag{6.36}$$

The second CPTP map in (6.36) is equal to the second CPTP map in (6.35), since a Z operator applied prior to standard basis measurement has no effect, and can be replaced with the identity operator. □

6.2.2 QPIP Definition

A QPIP is defined as follows (this definition is taken from [1]/ [3]):

Definition 6.2.3 *A language $\mathcal{L}$ is said to have a Quantum Prover Interactive Proof (QPIP_τ) with completeness c and soundness s (where $c - s$ is at least a constant) if there exists a pair of algorithms $(\mathbb{P}, \mathbb{V})$, where $\mathbb{P}$ is the prover and $\mathbb{V}$ is the verifier, with the following properties:*

- *The prover $\mathbb{P}$ is a BQP machine, which also has access to a quantum channel which can transmit τ qubits.*
- *The verifier $\mathbb{V}$ is a hybrid quantum-classical machine. Its classical part is a BPP machine. The quantum part is a register of τ qubits, on which the verifier can perform arbitrary quantum operations and which has access to a quantum channel which can transmit τ qubits. At any given time, the verifier is not allowed to possess more than τ qubits. The interaction between the quantum and classical parts of the verifier is the usual one: the classical part controls which operations are to be performed on the quantum register, and outcomes of measurements of the quantum register can be used as input to the classical machine.*
- *There is also a classical communication channel between the prover and the verifier, which can transmit polynomially many bits at any step.*
- *At any given step, either the verifier or the prover perform computations on their registers and send bits and qubits through the relevant channels to the other party.*

We require:

- ***Completeness**: if $x \in \mathcal{L}$, then after interacting with $\mathbb{P}$, $\mathbb{V}$ accepts with probability $\geq c$.*
- ***Soundness**: if $x \notin \mathcal{L}$, then the verifier rejects with probability $\geq 1 - s$ regardless of the prover $\mathbb{P}'$ (who has the same description as $\mathbb{P}$) with whom he is interacting.*

Abusing notation, we denote the class of languages for which such a proof exists also by QPIP_τ.

6.3 Function Definitions

6.3.1 Noisy Trapdoor Claw-Free Functions

This section is taken directly from [11]. Let λ be a security parameter, and $\mathcal{X}$ and $\mathcal{Y}$ be finite sets (depending on λ). For our purposes an ideal family of functions $\mathcal{F}$ would have the following properties. For each public key k, there are two functions $\{f_{k,b} : \mathcal{X} \to \mathcal{Y}\}_{b\in\{0,1\}}$

that are both injective and have the same range (equivalently, $(b, x) \mapsto f_{k,b}(x)$ is 2-to-1), and are invertible given a suitable trapdoor t_k (i.e. t_k can be used to compute x given b and $y = f_{k,b}(x)$). Furthermore, the pair of functions should be claw-free: it must be hard for an attacker to find two pre-images $x_0, x_1 \in \mathcal{X}$ such that $f_{k,0}(x_0) = f_{k,1}(x_1)$. Finally, the functions should satisfy an adaptive hardcore bit property, which is a stronger form of the claw-free property: assuming for convenience that $\mathcal{X} = \{0,1\}^w$, we would like that it is computationally infeasible to simultaneously generate an $(b, x_b) \in \{0,1\} \times \mathcal{X}$ and a $d \in \{0,1\}^w \backslash \{0^w\}$ such that with non-negligible advantage over $\frac{1}{2}$ the equation $d \cdot (x_0 \oplus x_1) = 0$, where x_{1-b} is defined as the unique element such that $f_{k,1-b}(x_{1-b}) = f_{k,b}(x_b)$, holds.

Unfortunately, we do not know how to construct a function family that exactly satisfies all these requirements under standard cryptographic assumptions. Instead, we construct a family that satisfies slightly relaxed requirements, that we will show still suffices for our purposes, based on the hardness of the learning with errors (LWE) problem. The requirements are relaxed as follows. First, the range of the functions is no longer a set $\mathcal{Y}$; instead, it is $\mathcal{D}_\mathcal{Y}$, the set of probability densities over $\mathcal{Y}$. That is, each function returns a density, rather than a point. The trapdoor injective pair property is then described in terms of the support of the output densities: these supports should either be identical, for a colliding pair, or be disjoint, in all other cases.

The consideration of functions that return densities gives rise to an additional requirement of efficiency: there should exist a quantum polynomial-time procedure that efficiently prepares a superposition over the range of the function, i.e. for any key k and $b \in \{0,1\}$, the procedure can prepare the state

$$\frac{1}{\sqrt{\mathcal{X}}} \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \sqrt{f_{k,b}(x)(y)} \, |x\rangle \, |y\rangle \ . \tag{6.37}$$

In our instantiation based on LWE, it is not possible to prepare (6.37) perfectly, but it is possible to create a superposition with coefficients $\sqrt{f'_{k,b}(x)}$ such that the resulting state is within negligible trace distance of (6.37). The density $f'_{k,b}(x)$ is required to satisfy two properties used in our protocol. First, it must be easy to check, without the trapdoor, if an element $y \in \mathcal{Y}$ lies in the support of $f'_{k,b}(x)$. Second, the inversion algorithm should operate correctly on all y in the support of $f'_{k,b}(x)$.

We slightly modify the adaptive hardcore bit requirement as well. Since the set $\mathcal{X}$ may not be a subset of binary strings, we first assume the existence of an injective, efficiently invertible map $J : \mathcal{X} \to \{0,1\}^w$. Next, we only require the adaptive hardcore bit property to hold for a subset of all nonzero strings, instead of the set $\{0,1\}^w \backslash \{0^w\}$. Finally, membership in the appropriate set should be efficiently checkable, given access to the trapdoor.

A formal definition follows.

Definition 6.3.1 (NTCF Family) *Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $\mathcal{K}_\mathcal{F}$ be a finite set of keys. A family of functions*

$$\mathcal{F} = \left\{ f_{k,b} : \mathcal{X} \to \mathcal{D}_\mathcal{Y} \right\}_{k \in \mathcal{K}_\mathcal{F}, b \in \{0,1\}}$$

is called a ***noisy trapdoor claw-free (NTCF) family*** *if the following conditions hold:*

1. ***Efficient Function Generation.*** *There exists an efficient probabilistic algorithm $GEN_{\mathcal{F}}$ which generates a key $k \in \mathcal{K}_{\mathcal{F}}$ together with a trapdoor t_k:*

$$(k, t_k) \leftarrow GEN_{\mathcal{F}}(1^{\lambda}) \ .$$

2. ***Trapdoor Injective Pair.*** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ the following conditions hold.*

   a) *Trapdoor: For all $b \in \{0,1\}$ and $x \neq x' \in \mathcal{X}$, $\textsc{Supp}(f_{k,b}(x)) \cap \textsc{Supp}(f_{k,b}(x')) = \emptyset$. Moreover, there exists an efficient deterministic algorithm $INV_{\mathcal{F}}$ such that for all $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \textsc{Supp}(f_{k,b}(x))$, $INV_{\mathcal{F}}(t_k, b, y) = x$.*

   b) *Injective pair: There exists a perfect matching $\mathcal{R}_k \subseteq \mathcal{X} \times \mathcal{X}$ such that $f_{k,0}(x_0) = f_{k,1}(x_1)$ if and only if $(x_0, x_1) \in \mathcal{R}_k$.*

3. ***Efficient Range Superposition.*** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ and $b \in \{0,1\}$ there exists a function $f'_{k,b} : \mathcal{X} \mapsto \mathcal{D}_{\mathcal{Y}}$ such that*

   a) *For all $(x_0, x_1) \in \mathcal{R}_k$ and $y \in \textsc{Supp}(f'_{k,b}(x_b))$, $INV_{\mathcal{F}}(t_k, b, y) = x_b$ and $INV_{\mathcal{F}}(t_k, b \oplus 1, y) = x_{b \oplus 1}$.*

   b) *There exists an efficient deterministic procedure $CHK_{\mathcal{F}}$ that, on input k, $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, returns 1 if $y \in \textsc{Supp}(f'_{k,b}(x))$ and 0 otherwise. Note that $CHK_{\mathcal{F}}$ is not provided the trapdoor t_k.*

   c) *For every k and $b \in \{0,1\}$,*

$$\mathrm{E}_{x \leftarrow_U \mathcal{X}} \big[\, H^2(f_{k,b}(x), \, f'_{k,b}(x)) \, \big] \ \leq \ \mu(\lambda) \ ,$$

   for some negligible function $\mu(\cdot)$. Here H^2 is the Hellinger distance; see (3.1). Moreover, there exists an efficient procedure $SAMP_{\mathcal{F}}$ that on input k and $b \in \{0,1\}$ prepares the state

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \sqrt{(f'_{k,b}(x))(y)} \, |x\rangle \, |y\rangle \ . \tag{6.38}$$

4. ***Adaptive Hardcore Bit.*** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ the following conditions hold, for some integer w that is a polynomially bounded function of λ.*

   a) *For all $b \in \{0,1\}$ and $x \in \mathcal{X}$, there exists a set $G_{k,b,x} \subseteq \{0,1\}^w$ such that $\Pr_{d \leftarrow_U \{0,1\}^w}[d \notin G_{k,b,x}]$ is negligible, and moreover there exists an efficient algorithm that checks for membership in $G_{k,b,x}$ given k, b, x and the trapdoor t_k.*

b) There is an efficiently computable injection $J : \mathcal{X} \to \{0,1\}^w$, such that J can be inverted efficiently on its range, and such that the following holds. If

$$\begin{aligned} H_k &= \left\{(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1))) \mid b \in \{0,1\},\ (x_0, x_1) \in \mathcal{R}_k,\ d \in G_{k,0,x_0} \cap G_{k,1,x_1}\right\}, \text{[1]} \\ \overline{H}_k &= \left\{(b, x_b, d, c) \mid (b, x, d, c \oplus 1) \in H_k\right\}, \end{aligned}$$

then for any quantum polynomial-time procedure $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that

$$\left| \Pr_{(k,t_k) \leftarrow GEN_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k) \leftarrow GEN_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in \overline{H}_k] \right| \leq \mu(\lambda) . \quad (6.39)$$

6.3.2 Extended Trapdoor Claw-Free Functions

In this section, we define the extended trapdoor claw-free family we will use in this chapter, which is a NTCF family (Definition 6.3.1) with two additional properties. In order to define an extended trapdoor claw-free family, we must first define a trapdoor injective family. A trapdoor injective family differs from a NTCF family in two ways: the function pairs have disjoint images (rather than perfectly overlapping images) and there is no adaptive hardcore bit condition.

Definition 6.3.2 (Trapdoor Injective Function Family) *Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $\mathcal{K}_\mathcal{G}$ be a finite set of keys. A family of functions*

$$\mathcal{G} = \left\{g_{k,b} : \mathcal{X} \to \mathcal{D}_\mathcal{Y}\right\}_{b \in \{0,1\}, k \in \mathcal{K}_\mathcal{G}}$$

is called a ***trapdoor injective family*** *if the following conditions hold:*

1. ***Efficient Function Generation.*** *There exists an efficient probabilistic algorithm $GEN_\mathcal{G}$ which generates a key $k \in \mathcal{K}_\mathcal{G}$ together with a trapdoor t_k:*

$$(k, t_k) \leftarrow GEN_\mathcal{G}(1^\lambda) .$$

2. ***Disjoint Trapdoor Injective Pair.*** *For all keys $k \in \mathcal{K}_\mathcal{G}$, for all $b, b' \in \{0,1\}$ and $x, x' \in \mathcal{X}$, if $(b, x) \neq (b', x')$, $\text{Supp}(g_{k,b}(x)) \cap \text{Supp}(g_{k,b'}(x')) = \emptyset$. Moreover, there exists an efficient deterministic algorithm $INV_\mathcal{F}$ such that for all $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \text{Supp}(g_{k,b}(x))$, $INV_\mathcal{G}(t_k, y) = (b, x)$.*

3. ***Efficient Range Superposition.*** *For all keys $k \in \mathcal{K}_\mathcal{G}$ and $b \in \{0,1\}$*

   a) There exists an efficient deterministic procedure $CHK_\mathcal{G}$ that, on input k, $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, outputs 1 if $y \in \text{Supp}(g_{k,b}(x))$ and 0 otherwise. Note that $CHK_\mathcal{G}$ is not provided the trapdoor t_k.

[1] Note that although both x_0 and x_1 are referred to to define the set H_k, only one of them, x_b, is explicitly specified in any 4-tuple that lies in H_k.

b) There exists an efficient procedure $SAMP_{\mathcal{G}}$ that on input k and $b \in \{0,1\}$ returns the state

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \sqrt{(g_{k,b}(x))(y)}\, |x\rangle\, |y\rangle \ . \tag{6.40}$$

Definition 6.3.3 (Injective Invariance) *A noisy trapdoor claw-free family $\mathcal{F}$ is **injective invariant** if there exists a trapdoor injective family $\mathcal{G}$ such that:*

1. *The algorithms $CHK_{\mathcal{F}}$ and $SAMP_{\mathcal{F}}$ are the same as the algorithms $CHK_{\mathcal{G}}$ and $SAMP_{\mathcal{G}}$.*
2. *For all quantum polynomial-time procedures $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that*

$$\Big| \Pr_{(k,t_k) \leftarrow GEN_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) = 0] - \Pr_{(k,t_k) \leftarrow GEN_{\mathcal{G}}(1^\lambda)} [\mathcal{A}(k) = 0] \Big| \leq \mu(\lambda) \tag{6.41}$$

Definition 6.3.4 (Extended Trapdoor Claw-Free Family) *A noisy trapdoor claw-free family $\mathcal{F}$ is an **extended trapdoor claw-free family** if:*

1. *It is injective invariant.*
2. *For all $k \in \mathcal{K}_{\mathcal{F}}$ and $d \in \{0,1\}^w$, let:*

$$H'_{k,d} \ = \ \{d \cdot (J(x_0) \oplus J(x_1)) | (x_0, x_1) \in \mathcal{R}_k\} \tag{6.42}$$

For all quantum polynomial-time procedures $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ and a string $d \in \{0,1\}^w$ such that

$$\Big| \Pr_{(k,t_k) \leftarrow GEN_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in H'_{k,d}] - \frac{1}{2} \Big| \leq \mu(\lambda) \tag{6.43}$$

6.4 Measurement Protocol

We begin by introducing the state commitment process (described in Section 6.1.1) followed by the measurement protocol (given in Section 6.1.2). The presentation below is slightly more involved than the overview, since we are not using the perfect trapdoor claw-free/ injective families used in the overview; we are instead using the families given in Definitions 6.3.1, 6.3.2 and 6.3.4. After presenting the measurement protocol, we provide notation which will be used throughout the rest of the chapter. Next, we prove completeness of our measurement protocol and characterize the behavior of general provers (as described in Section 6.1.3.1). This section ends with the construction of the quantum state underlying the measurement distribution obtained by the verifier (given in Section 6.1.3.2).

6.4.1 How to Commit Using a Noisy Trapdoor Claw-Free Family

Here we describe the process of state commitment with a NTCF function (as defined in Definition 6.3.1). The state commitment process requires a function key $k \in \mathcal{K}_{\mathcal{F}}$ (which corresponds to functions $f_{k,0}, f_{k,1} \in \mathcal{F}$) and is performed with respect to the first qubit of an arbitrary state $|\psi\rangle$:

$$|\psi\rangle = \sum_{b\in\{0,1\}} \alpha_b |b\rangle |\psi_b\rangle \tag{6.44}$$

The first step of the commitment process is to apply the $\text{SAMP}_{\mathcal{F}}$ procedure in superposition, with k and the first qubit containing b as input:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{\substack{b\in\{0,1\}\\ x\in\mathcal{X},y\in\mathcal{Y}}} \alpha_b \sqrt{f'_{k,b}(x)(y)}\, |b\rangle |x\rangle |\psi_b\rangle |y\rangle \tag{6.45}$$

By condition 3(c) of Definition 6.3.1 and Lemma 3.3.1 (6.45) is within negligible trace distance of the following state:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{\substack{b\in\{0,1\}\\ x\in\mathcal{X},y\in\mathcal{Y}}} \alpha_b \sqrt{f_{k,b}(x)(y)}\, |b\rangle |x\rangle |\psi_b\rangle |y\rangle \tag{6.46}$$

The second step of the commitment process is to measure the last register, obtaining the *commitment string* $y \in \mathcal{Y}$. Let $x_{b,y} = \text{INV}_{\mathcal{F}}(t_k, b, y)$ (t_k is the trapdoor corresponding to the key k). The remaining state at this point is within negligible trace distance of the following state

$$\sum_{b\in\{0,1\}} \alpha_b |b\rangle |x_{b,y}\rangle |\psi_b\rangle \tag{6.47}$$

The fact that the superposition collapses in this manner is due to both the trapdoor and injective pair conditions in Definition 6.3.1. The trapdoor condition implies that for each b, there can be at most one remaining element $x \in \mathcal{X}$ in the superposition after measuring y. The injective pair condition states that for all $x_{b,y} \in \mathcal{X}$, there exists exactly one $x_{b\oplus 1,y} \in \mathcal{X}$ such that $(x_{0,y}, x_{1,y}) \in \mathcal{R}_k$ (i.e. $f_{k,b}(x_b) = f_{k,b}(x_{b\oplus 1})$). Therefore, if $y \in \text{Supp}(f_{k,b}(x_b))$, it follows that $y \in \text{Supp}(f_{k,b\oplus 1}(x_{b\oplus 1}))$. We will call the first qubit of (6.47) the *committed qubit* and the second register (containing $x_{b,y}$) the *preimage register*.

6.4.1.1 Hadamard Measurement of a Committed State

The first step in measuring a committed state in the Hadamard basis is to apply the unitary U_J, which uses the injective map J defined in condition 4(b) of Definition 6.3.1:

$$U_J\Big(\sum_{b\in\{0,1\}} \alpha_b |b\rangle |x_{b,y}\rangle |\psi_b\rangle |0\rangle^{e_J}\Big) = \sum_{b\in\{0,1\}} \alpha_b |b\rangle |J(x_{b,y})\rangle |\psi_b\rangle |0\rangle^{e'_J} \tag{6.48}$$

where the number of auxiliary qubits (e_J and e'_J) is determined by the map J. The map U_J is unitary since J is both efficiently computable and efficiently invertible. The second step is to apply the Hadamard transform $H^{\otimes w+1}$ to the first two registers of the state in (6.48). The resulting state is:

$$\frac{1}{\sqrt{2^w}} \sum_{\substack{d\in\{0,1\}^w \\ b\in\{0,1\}}} \alpha_b X^{d\cdot(J(x_{0,y})\oplus J(x_{1,y}))} H\left|b\right\rangle \otimes (-1)^{d\cdot J(x_{0,y})}\left|d\right\rangle \otimes \left|\psi_b\right\rangle \otimes \left|0\right\rangle^{e'_J} \tag{6.49}$$

The third step is measurement of the preimage register, obtaining a string $d \in \{0,1\}^w$ and resulting in the following state (recall the state $|\psi\rangle$ from (6.44)):

$$(X^{d\cdot(J(x_{0,y})\oplus J(x_{1,y}))} H \otimes \mathcal{I})\left|\psi\right\rangle \left|0\right\rangle^{e'_J} \tag{6.50}$$

The final step is measuring the committed qubit to obtain a bit b'. The Hadamard measurement result of the first qubit of $|\psi\rangle$ is $b' \oplus d \cdot (J(x_{0,y}) \oplus J(x_{1,y})) \in \{0,1\}$ ($x_{0,y}$ and $x_{1,y}$ can be recovered from y using the trapdoor t_k and the function $\mathrm{INV}_{\mathcal{F}}$).

6.4.1.2 How to Commit Using a Trapdoor Injective Family

The commitment process described in Section 6.4.1 can also be performed using a key $k \in \mathcal{K}_{\mathcal{G}}$ corresponding to trapdoor injective functions $g_{k,0}, g_{k,1} \in \mathcal{G}$ (see Definition 6.3.2). Prior to measuring y (at the stage of (6.45)), the state is:

$$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{\substack{b\in\{0,1\} \\ x\in\mathcal{X}, y\in\mathcal{Y}}} \alpha_b \sqrt{g_{k,b}(x)(y)}\left|b\right\rangle \left|x\right\rangle \left|\psi_b\right\rangle \left|y\right\rangle \tag{6.51}$$

Now the last register is measured to obtain $y \in \mathcal{Y}$. Since the sets $\textsc{Supp}(g_{k,b}(x))$ and $\textsc{Supp}(g_{k,b'}(x'))$ are disjoint for all $(b,x) \neq (b',x')$ (see the trapdoor condition of Definition 6.3.2), $y \in \bigcup_{x\in\mathcal{X}} \textsc{Supp}(g_{k,b}(x))$ with probability $|\alpha_b|^2$. Let $(b, x_{b,y}) = \mathrm{INV}_{\mathcal{G}}(t_k, y)$. The remaining state after measurement is:

$$\left|b\right\rangle \left|x_{b,y}\right\rangle \left|\psi_b\right\rangle \tag{6.52}$$

Therefore, measuring y acts as a standard basis measurement of the first qubit of the state $|\psi\rangle$ in (6.44). The standard basis measurement b can be obtained (with access to only the trapdoor t_k of $g_{k,0}, g_{k,1}$ and y) by running the function $\mathrm{INV}_{\mathcal{G}}$.

6.4.2 Measurement Protocol

We now use the commitment process in Section 6.4.1 to construct our measurement protocol for n qubits, where n is polynomial in the security parameter λ. We require an extended trapdoor claw-free family $\mathcal{F}$ (Definition 6.3.4) as well as its corresponding trapdoor injective

family $\mathcal{G}$ (Definition 6.3.2). The measurement protocol depends on a string $h \in \{0,1\}^n$, called the basis choice, which represents the basis for which the verifier would like measurement results of the n qubits; $h_i = 0$ indicates the standard basis and $h_i = 1$ indicates the Hadamard basis. We now provide the measurement protocol:

Protocol 6.4.1 ***Measurement Protocol (for*** $h \in \{0,1\}^n$***)***

1. *The verifier performs the following operations for* $1 \leq i \leq n$*:*

   a) *If the standard basis is chosen for qubit* i *(i.e.* $h_i = 0$*), the verifier runs* $GEN_{\mathcal{G}}(1^\lambda)$ *to produce a function key* $k_i \in \mathcal{K}_{\mathcal{G}}$ *and its corresponding trapdoor* t_{k_i}*.*

   b) *If the Hadamard basis is chosen for qubit* i *(*$h_i = 1$*), the verifier runs* $GEN_{\mathcal{F}}(1^\lambda)$ *to produce a function key* $k_i \in \mathcal{K}_{\mathcal{F}}$ *and its corresponding trapdoor* t_{k_i}*.*

   Let $k' = (k_1, \ldots, k_n)$*. The verifier sends the function choice* k' *to the prover.*

2. *The verifier receives* $y' = (y_1, \ldots, y_n) \in \mathcal{Y}^n$ *from the prover.*

3. *The verifier chooses at random to run a test round or a Hadamard round (each is chosen with probability* $\frac{1}{2}$*).*

4. *For a test round:*

   a) *The verifier asks the prover for standard basis measurements of committed qubit* i *and preimage register* i*, for* $1 \leq i \leq n$*.*

   b) *For* $1 \leq i \leq n$*, the verifier receives a bit* b'_i *and a string* $x'_i \in \mathcal{X}$*. For all* i *such that* $h_i = 0$*, the verifier rejects if* $CHK_{\mathcal{G}}(k_i, b'_i, x'_i, y_i) = 0$*. For all* i *such that* $h_i = 1$*, the verifier rejects if* $CHK_{\mathcal{F}}(k_i, b'_i, x'_i, y_i) = 0$*.*

5. *For a Hadamard round:*

   a) *The verifier asks the prover for Hadamard measurements of committed qubit* i *and preimage register* i *for* $1 \leq i \leq n$*.*

   b) *For* $1 \leq i \leq n$*, the verifier receives a bit* b'_i *and a string* $d_i \in \{0,1\}^w$*.*

   c) *For qubits* i *for which* $h_i = 0$*, the results* (b'_i, d_i) *are ignored. The verifier computes*

   $$(m_i, x_{m_i, y_i}) \quad = \quad INV_{\mathcal{G}}(t_{k_i}, y_i) \tag{6.53}$$

   If the inverse does not exist, the verifier stores a random bit as the measurement result and rejects. Otherwise the verifier stores m_i *as the standard basis measurement result.*

d) For qubits i for which $h_i = 1$, the verifier computes

$$x_{0,y_i} = INV_{\mathcal{F}}(t_{k_i}, 0, y_i) \quad (6.54)$$
$$x_{1,y_i} = INV_{\mathcal{F}}(t_{k_i}, 1, y_i) \quad (6.55)$$

If either of the inverses does not exist, the verifier stores a random bit as the measurement result and rejects. The verifier uses t_{k_i} to check if $d_i \in G_{k_i,0,x_{0,y_i}} \cap G_{k_i,1,x_{1,y_i}}$. If not, the verifier stores a random bit as the measurement result and rejects. Otherwise, the verifier stores $m_i = b'_i \oplus d_i \cdot (J(x_{0,y_i}) \oplus J(x_{1,y_i}))$ as the Hadamard basis measurement result.

6.4.2.1 Honest Prover

We now provide an honest prover's behavior in Protocol 6.4.1, assuming the prover would like to report measurement results of an n qubit state ρ:

Protocol 6.4.2 ***Honest Prover in Measurement Protocol (for an efficiently computable n qubit state ρ)***

1. *The prover creates the state ρ. Upon receipt of k' from the verifier, the prover commits to qubit i of ρ using k_i as described in Section 6.4.1. The prover reports the measurement results $y' = (y_1, \ldots, y_n) \in \mathcal{Y}^n$ obtained from each commitment process to the verifier.*

2. *For a test round:*

   a) *The prover measures each of the n committed qubits and preimage registers in the standard basis, sending the verifier the resulting bit b'_i and string $x'_i \in \mathcal{X}$ for $1 \leq i \leq n$.*

3. *For a Hadamard round:*

   a) *The prover first applies the unitary U_J to all n preimage registers. The prover then measures each of the n committed qubits and preimage registers in the Hadamard basis, sending the verifier the resulting bit b'_i and string $d_i \in \{0,1\}^w$ for $1 \leq i \leq n$.*

6.4.3 Notation

We now introduce some notation (and provide reminders of previously used notation) which will be useful throughout the rest of the chapter.

1. The string $h \in \{0,1\}^n$ is called the basis choice; $h_i = 0$ indicates the standard basis and $h_i = 1$ indicates the Hadamard basis.

2. We will call k' (produced in step 1 of Protocol 6.4.1) the *function choice* of the verifier. Let $D_{\mathbb{V},h}$ be the distribution which it is sampled from (this is the distribution produced by $\mathrm{GEN}_{\mathcal{F}}$ and $\mathrm{GEN}_{\mathcal{G}}$).

3. A *perfect* prover is a prover who is always accepted by the verifier on the test round.

4. For a density matrix ρ on n qubits and a string $h \in \{0,1\}^n$, let $D_{\rho,h}$ be the distribution over $\{0,1\}^n$ which results from measuring all qubits of ρ in the basis specified by h.

5. For every prover $\mathbb{P}$ and basis choice $h \in \{0,1\}^n$, let $D_{\mathbb{P},h}$ be the distribution over measurement results $m \in \{0,1\}^n$ obtained by the verifier when interacting with $\mathbb{P}$ on basis choice h in a Hadamard round. Let $D^C_{\mathbb{P},h}$ be the same distribution, but conditioned on acceptance by the verifier (in a Hadamard round). Let $\sigma_{\mathbb{P},h}$ be the density matrix corresponding to the distribution $D_{\mathbb{P},h}$:

$$\sigma_{\mathbb{P},h} \stackrel{\text{def}}{=} \sum_{m\in\{0,1\}^n} D_{\mathbb{P},h}(m) \left|m\right\rangle \left\langle m\right| \tag{6.56}$$

   We will frequently use the fact that for provers $\mathbb{P}$ and $\mathbb{P}'$, $\sigma_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}',h}$ are computationally indistinguishable if and only if $D_{\mathbb{P},h}$ and $D_{\mathbb{P}',h}$ are computationally indistinguishable, by definition of computational indistinguishability of distributions (Definition 3.1.2) and of density matrices (Definition 3.3.2). Also note that by definition of trace distance in (3.10) and total variation distance in (3.2):

$$\left\|\sigma_{\mathbb{P},h} - \sigma_{\mathbb{P}',h}\right\|_{tr} = \left\|D_{\mathbb{P},h} - D_{\mathbb{P}',h}\right\|_{TV} \tag{6.57}$$

6. As introduced in Section 6.4.1, a *committed qubit* is the qubit which is used to determine whether to apply $f_{k,0}$ or $f_{k,1}$ (or $g_{k,0}$ or $g_{k,1}$) and the *preimage register* is the register which contains the inverse after the measurement; i.e. in the following state from (6.47):

$$\sum_b \alpha_b \left|b\right\rangle \left|x_{b,y}\right\rangle \left|\psi_b\right\rangle \tag{6.58}$$

   the first qubit is the committed qubit and the second register (containing $x_{b,y}$) is the preimage register. The *commitment string* is the string $y \in \mathcal{Y}$.

6.4.4 Completeness of Measurement Protocol

Claim 6.4.3 ***Completeness of Measurement Protocol (Protocol 6.4.1)*** *For all n qubit states ρ and for all basis choices $h \in \{0,1\}^n$, the prover $\mathbb{P}$ described in Protocol 6.4.2 is a perfect prover ($\mathbb{P}$ is accepted by the verifier in a test round for basis choice h with perfect probability). There exists a negligible function μ such that in the Hadamard round for basis choice h, the verifier accepts $\mathbb{P}$ with probability $\geq 1-\mu$ and $\left\|D^C_{\mathbb{P},h} - D_{\rho,h}\right\|_{TV} \leq \mu$.*

Proof of **Claim 6.4.3**: First, assume that the prover could produce the ideal states in the commitment procedure, as written in (6.46) for the Hadamard basis and (6.51) for the standard basis. Call such a prover $\mathbb{P}'$. The distribution over measurement results obtained by the verifier when interacting with $\mathbb{P}'$ (prior to conditioning on acceptance) is equal to the distribution over measurement results obtained by measuring ρ in the basis specified by h, i.e.:

$$\|D_{\mathbb{P}',h} - D_{\rho,h}\|_{TV} = 0 \tag{6.59}$$

We now return to analyzing the prover $\mathbb{P}$ given in Protocol 6.4.2. First note that $\mathbb{P}$ is a perfect prover: when measured, the superpositions created by $\mathbb{P}$ during the commitment process (in (6.45) and (6.51)) pass the CHK procedure perfectly, by definition (see Definition 6.3.1 and Definition 6.3.2). Moving on to the Hadamard round, $\mathbb{P}$ is rejected by the verifier only if there exists an i such that the measurement result d_i is not in the set $G_{k_i,0,x_{0,y_i}} \cap G_{k_i,1,x_{1,y_i}}$ (and $h_i = 1$). The adaptive hardcore bit clause of Definition 6.3.1 (item 4(a)) implies that since d_i is sampled uniformly (it is the result of a Hadamard transform), there exists a negligible function μ_H such that the probability that the verifier rejects $\mathbb{P}$ in the Hadamard round is at most μ_H:

$$\left\|D_{\mathbb{P},h} - D^C_{\mathbb{P},h}\right\|_{TV} \leq \mu_H \tag{6.60}$$

Next, observe that the prover $\mathbb{P}$ in Protocol 6.4.2 can produce the state in (6.51) (which is used by the prover $\mathbb{P}'$), but can only create a state within negligible trace distance of the state in (6.46). It follows that there exists a negligible function μ' such that:

$$\|D_{\mathbb{P},h} - D_{\mathbb{P}',h}\|_{TV} \leq \mu' \tag{6.61}$$

Using the triangle inequality, we obtain:

$$\left\|D^C_{\mathbb{P},h} - D_{\rho,h}\right\|_{TV} \leq \left\|D^C_{\mathbb{P},h} - D_{\mathbb{P},h}\right\|_{TV} + \|D_{\mathbb{P},h} - D_{\mathbb{P}',h}\|_{TV} + \|D_{\mathbb{P}',h} - D_{\rho,h}\|_{TV} \tag{6.62}$$

We complete the calculation by plugging in (6.59), (6.60) and (6.61):

$$\left\|D^C_{\mathbb{P},h} - D_{\rho,h}\right\|_{TV} \leq \mu_H + \mu' \tag{6.63}$$

To complete the proof of the claim, set $\mu = \mu_H + \mu'$. □

6.4.5 Prover Behavior

We now give a claim which characterizes the behavior of a general prover in Protocol 6.4.1 (the overview of this claim and its proof are given in Section 6.1.3.1). The only difference between the following claim and the version given in the overview is the inclusion of the operator U_J (as defined in (6.48)):

Claim 6.4.4 ***Prover Behavior*** *For all* BQP *provers* $\mathbb{P}$ *in Protocol 6.4.1, there exist two efficiently computable unitary operators* U_0, U *and a prover* $\mathbb{P}'$ *(described below) such that for all basis choices* $h \in \{0,1\}^n$, $\mathbb{P}$ *and* $\mathbb{P}'$ *are accepted by the verifier with the same probability*

in a test round and the distribution over measurement results $D_{\mathbb{P},h}$ produced by the prover and verifier as a result of Protocol 6.4.1 is equal to the distribution $D_{\mathbb{P}',h}$ corresponding to the prover $\mathbb{P}'$. We say that $\mathbb{P}$ is characterized by (U_0, U).

1. *$\mathbb{P}'$ designates his first n qubits as committed qubits, the next n registers as preimage registers and the final n registers as commitment string registers. All other registers contain auxiliary space.*

2. *Upon receipt of the function choice k' from the verifier, the prover $\mathbb{P}'$ applies U_0 to his initial state:*

$$|0\rangle^{\otimes e} \otimes |k'\rangle \tag{6.64}$$

where e is determined by U_0, U and U_0 uses the last register (containing k') as a control register; i.e. there exists a unitary $U_{0,k'}$ such that

$$U_0(|0\rangle^{\otimes e} \otimes |k'\rangle) = U_{0,k'}(|0\rangle^{\otimes e}) \otimes |k'\rangle \tag{6.65}$$

3. *$\mathbb{P}'$ measures the commitment string registers to obtain $y' = (y_1, \ldots, y_n) \in \mathcal{Y}^n$, which is sent to the verifier.*

4. *For a Hadamard round:*

   a) *$\mathbb{P}'$ appends $e_J \cdot n$ auxiliary 0 qubits to his state and applies the unitary U_J to all n preimage registers, followed by application of the unitary U to his entire state.*

   b) *$\mathbb{P}'$ measures the n committed qubits and preimage registers in the Hadamard basis. $\mathbb{P}'$ sends the verifier the resulting bit b'_i and string $d_i \in \{0,1\}^w$ for $1 \leq i \leq n$.*

5. *For a test round, $\mathbb{P}'$ measures each of the n committed qubits and preimage registers in the standard basis, sending the verifier the resulting bit b'_i and string $x'_i \in \mathcal{X}$ for $1 \leq i \leq n$.*

Notation 6.4.5 *We will also frequently say that a prover $\mathbb{P}$ is characterized by two CPTP maps $(\mathcal{S}_0, \mathcal{S})$. This means that for all basis choices $h \in \{0,1\}^n$, $\mathbb{P}$ and $\mathbb{P}'$ are accepted with the same probability in a test round and $D_{\mathbb{P},h} = D_{\mathbb{P}',h}$, and the prover $\mathbb{P}'$ follows steps 1 - 5 in Claim 6.4.4, but uses the CPTP maps $\mathcal{S}_0, \mathcal{S}$ rather than the unitary operators U_0, U.*

*P*roof of **Claim 6.4.4**: We will follow the principle given in Section 6.1.3.1: a general prover is equivalent from the verifier's perspective to a prover $\mathbb{P}$ who begins each round by applying an arbitrary unitary attack and then behaves honestly. The first implication of the principle is that $\mathbb{P}$ measures the same registers as an honest prover; therefore, like the honest prover, $\mathbb{P}$ designates the first n qubits as committed qubits, the next n registers as preimage registers, and the final n registers as commitment string registers. All other registers of $\mathbb{P}$ contain the auxiliary space.

The second implication is that there exist unitary operators U', U_T and U_C such that $\mathbb{P}$ acts as follows. $\mathbb{P}$ begins with the initial state $|0\rangle^{\otimes e} \otimes |k'\rangle$ and then applies a unitary operator U' to his state, followed by standard basis measurement of the commitment string registers to obtain y'. If the verifier chooses a test round, the prover applies another unitary U_T followed by standard basis measurements of the committed qubit and preimage registers to obtain the requested measurement results. If the verifier chooses a Hadamard round, the prover first appends $e_J \cdot n$ auxiliary 0 qubits to his state. Next, the prover applies a unitary U_C to his state. He then applies the unitary U_J (see Section 6.4.1.1) to all n preimage registers. Finally, the prover measures all n committed qubits and preimage registers in the Hadamard basis to obtain the requested measurement results. We can assume both U_T and U_C do not act on the register containing y'. This is because y' could have been copied into the prover's auxiliary space prior to measurement, and U_T and U_C can instead act on this space. It follows that both U_T and U_C commute with the measurement of y'.

To obtain the attacks U_0 and U which characterize $\mathbb{P}$, we make two changes. First, we use the fact that U_T commutes with measurement of y' to shift it prior to the measurement. Due to this change, we also need to append $U_T^\dagger$ to the start of the Hadamard round attack. Our second change is to shift the unitary U_J so that it is prior to the Hadamard round attack; this can be done by conjugating the attack by U_J. It follows that if we let $U_0 = U_T U'$, $U = U_J^{\otimes n} U_C U_T^\dagger (U_J^{\otimes n})^\dagger$ and consider the prover $\mathbb{P}'$ described in the statement of Claim 6.4.4 (with respect to U_0 and U), $\mathbb{P}$ and $\mathbb{P}'$ are accepted with the same probability in a test round and $D_{\mathbb{P},h} = D_{\mathbb{P}',h}$ for all basis choices h.

□

6.4.6 Construction of Underlying Quantum State

We require the following definition:

Definition 6.4.6 ***Trivial Prover*** *A perfect prover $\mathbb{P}$ in Protocol 6.4.1 characterized by $(U_0, \mathcal{S})$ (where U_0 is a unitary, $\mathcal{S}$ is a CPTP map and both are efficiently computable) is called trivial if $\mathcal{S}$ commutes with standard basis measurement on the first n qubits.*

In this section, we prove the following claim (the overview of this claim is given in Section 6.1.3.2):

Claim 6.4.7 *For all trivial provers $\mathbb{P}$, there exists an n qubit state ρ (which can be created using a BQP circuit) such that for all $h \in \{0,1\}^n$, the distribution over measurement results $D_{\mathbb{P},h}$ produced in Protocol 6.4.1 with respect to $\mathbb{P}$ for basis choice h is computationally indistinguishable from the distribution $D_{\rho,h}$ which results from measuring ρ in the basis determined by h.*

Proof: For a unitary U_0 and CPTP map $\mathcal{S}$, let the prover $\mathbb{P}$ be characterized by $(U_0, \mathcal{S})$. The state ρ is constructed as follows:

Protocol 6.4.8 ***Construction of ρ corresponding to $\mathbb{P}$***

1. *For $1 \leq i \leq n$: sample $(k_i, t_{k_i}) \leftarrow GEN_{\mathcal{F}}(1^\lambda)$.*
2. *Follow steps 1-4(a) in Claim 6.4.4 (with respect to $U_0, \mathcal{S}$).*
3. *Measure all preimage registers in the Hadamard basis to obtain $d_1, \ldots, d_n \in \{0,1\}^w$.*
4. *For $1 \leq i \leq n$, use the trapdoor t_{k_i} to apply $Z^{d_i \cdot (x_{0,y_i} \oplus x_{1,y_i})}$ to the i^{th} committed qubit.*
5. *Trace out all qubits except n committed qubits.*

We now argue that, for all $h \in \{0,1\}^n$, $D_{\rho,h}$ is computationally indistinguishable from $D_{\mathbb{P},h}$. We will proceed through two families of hybrid states which are dependent on the basis choice h. In the first family $\{\rho_h^{(1)}\}_{h \in \{0,1\}^n}$, we simply remove the Z decoding operator (step 4 of Protocol 6.4.8) if $h_i = 0$. This also eliminates the need for the trapdoor t_{k_i} if $h_i = 0$:

Protocol 6.4.9 ***Construction of $\rho_h^{(1)}$ corresponding to $\mathbb{P}$***

1. *For $1 \leq i \leq n$: sample $(k_i, t_{k_i}) \leftarrow GEN_{\mathcal{F}}(1^\lambda)$. If $h_i = 0$, discard the trapdoor t_{k_i}.*
2. *Apply steps 2-3 of Protocol 6.4.8.*
3. *For $1 \leq i \leq n$, if $h_i = 1$, use the trapdoor t_{k_i} to apply $Z^{d_i \cdot (x_{0,y_i} \oplus x_{1,y_i})}$ to the i^{th} committed qubit.*
4. *Trace out all qubits except the n committed qubits.*

The distributions $D_{\rho,h}$ and $D_{\rho_h^{(1)},h}$ differ only on i for which $h_i = 0$. To address this difference, note that if $h_i = 0$, the Z operator applied in step 4 of Protocol 6.4.8 has no effect on $D_{\rho,h}$: to obtain $D_{\rho,h}$ the i^{th} committed qubit is measured in the standard basis immediately after application of the Z operator. Therefore, $D_{\rho,h} = D_{\rho_h^{(1)},h}$ for all h.

Our next hybrid is:

Protocol 6.4.10 ***Construction of $\rho_h^{(2)}$ corresponding to $\mathbb{P}$***

1. *For $1 \leq i \leq n$: if $h_i = 1$, sample $(k_i, t_{k_i}) \leftarrow GEN_{\mathcal{F}}(1^\lambda)$. If $h_i = 0$, sample $(k_i, t_{k_i}) \leftarrow GEN_{\mathcal{G}}(1^\lambda)$ and discard the trapdoor t_{k_i}.*
2. *Apply steps 2-4 of Protocol 6.4.9.*

The computational indistinguishability of $D_{\rho_h^{(1)},h}$ and $D_{\rho_h^{(2)},h}$ follows due to the injective invariance (Definition 6.3.3) of $\mathcal{F}$ with respect to $\mathcal{G}$: as long as the trapdoor t_{k_i} is unknown, a key k_i sampled from $\mathcal{K}_{\mathcal{F}}$ is computationally indistinguishable from a key k_i sampled from $\mathcal{K}_{\mathcal{G}}$. We can apply this argument for all i such that $h_i = 0$ since the trapdoor t_{k_i} was discarded for all such i in Protocols 6.4.9 and 6.4.10.

We have so far shown that $D_{\rho,h}$ is computationally indistinguishable from $D_{\rho_h^{(2)},h}$ for all $h \in \{0,1\}^n$. To complete our proof, we now show that $D_{\rho_h^{(2)},h} = D_{\mathbb{P},h}$. The two distributions

differ as follows: if $h_i = 0$, the distribution of the i^{th} bit of $D_{\mathbb{P},h}$ is obtained from the commitment string y_i (see step 5(c) of Protocol 6.4.1), but the distribution of the i^{th} bit of $D_{\rho_h^{(2)},h}$ is obtained from measuring the i^{th} committed qubit of $\rho_h^{(2)}$ in the standard basis.

To see that these two distributions are equal, we begin by observing that since the prover $\mathbb{P}$ is perfect, if $h_i = 0$, measuring the i^{th} committed qubit prior to the attack $\mathcal{S}$ (i.e. at the start of the Hadamard round) results in the same outcome as extracting the measurement outcome from y_i. To complete our proof, recall that since the prover is trivial, the attack $\mathcal{S}$ commutes with standard basis measurement. □

6.5 Replacement of a General Attack with an X-Trivial Attack for Hadamard Basis

In this section, we analyze Protocol 6.4.1 with a perfect prover (a prover who passes the test round of Protocol 6.4.1 with perfect probability). We will rely on notation introduced in Section 6.4.3. This section is dedicated to proving the following claim:

Claim 6.5.1 ***General to X-Trivial Attack for Hadamard Basis*** *Let $1 \leq j \leq n$. Let $\mathcal{S} = \{B_\tau\}_\tau$ and $\mathcal{S}_j = \{B'_{j,x,\tau}\}_{x\in\{0,1\},\tau}$ be CPTP maps written in terms of their Kraus operators:*

$$B_\tau = \sum_{x,z\in\{0,1\}} X^x Z^z \otimes B_{jxz\tau} \tag{6.66}$$

$$B'_{j,x,\tau} = \sum_{z\in\{0,1\}} Z^z \otimes B_{jxz\tau} \tag{6.67}$$

where B_τ and $B'_{j,x,\tau}$ have been rearranged so that $X^x Z^z$ and Z^z act on the j^{th} qubit. For a unitary operator U_0, let $\mathbb{P}$ be a perfect prover characterized by $(U_0, \mathcal{S})$ (see Claim 6.4.4 and notation 6.4.5). Let $\mathbb{P}_j$ be a perfect prover characterized by $(U_0, \mathcal{S}_j)$. If $h_j = 1$, $D_{\mathbb{P},h}$ and $D_{\mathbb{P}_j,h}$ are computationally indistinguishable.

The overview of the proof of Claim 6.5.1 is given in Section 6.1.4. Claim 6.5.1 is slightly more general than the statement in the overview: n does not have to be equal to 1, and we are proving that we can replace the attack $\mathcal{S}$ with an attack which acts trivially on any one of the committed qubits j for which $h_j = 1$. We begin by writing out the state $\sigma_{\mathbb{P},h}$ which corresponds to the distribution $D_{\mathbb{P},h}$ (as defined in (6.56)). This requires some care, since we need to go through the steps of Protocol 6.4.1 in order to construct $\sigma_{\mathbb{P},h}$. Once we write down the state $\sigma_{\mathbb{P},h}$, we can proceed to proving computational indistinguishability between $\sigma_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}_j,h}$. As written below (6.56), proving computational indistinguishability between $\sigma_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}_1,h}$ is equivalent to proving indistinguishability between $D_{\mathbb{P},h}$ and $D_{\mathbb{P}_1,h}$.

*P*roof of **Claim 6.5.1**: We will assume for convenience that $j = 1$; the proof for all other values of j is equivalent. To analyze the state $\sigma_{\mathbb{P},h}$, we will assume that $\mathbb{P}$ follows steps 1-4

in Claim 6.4.4. We can do this since $\mathbb{P}$ is characterized by $U_0, \mathcal{S}$; therefore, the state $\sigma_{\mathbb{P},h}$ can be obtained by following the steps in Claim 6.4.4.

We first provide some notation we will require. Let $k = k_1 \in \mathcal{K}_{\mathcal{F}}$ be the first function key received by the prover $\mathbb{P}$ in Protocol 6.4.1. Throughout this proof, we will only be focusing on the first committed qubit (since $j = 1$). Therefore, for notational convenience, we will drop the subscript of 1 for values pertaining to the first committed qubit (i.e. the basis choice, function key, commitment string, etc.). Let $h_{>1} = (h_2, \ldots, h_n)$, let $k_{>1} = (k_2, \ldots, k_n)$ and define $t_{k>1}$ similarly. We will also require the following mixed state, which contains the distribution over all function keys and trapdoors except the first (as sampled by the verifier).

$$\sum_{k_{>1}} D_{\mathbb{V},h_{>1}} |k_{>1}\rangle \langle k_{>1}| \otimes |t_{k>1}\rangle \langle t_{k>1}| \tag{6.68}$$

This mixed state is required to create $\sigma_{\mathbb{P},h}$: the function keys are part of the prover's input and the trapdoors are used for the verifier's decoding. For convenience, let $|\phi_{k>1}\rangle$ be a purification of the above state; when analyzing the state $\sigma_{\mathbb{P},h}$, we can consider a purification since we will eventually be tracing out all but the committed qubits. For $b \in \{0,1\}$, let $T_{k,b} = \bigcup_{x \in \mathcal{X}} \textsc{Supp}(f_{k,b}(x))$ ($\textsc{Supp}(f_{k,b}(x))$ is the support of the probability density function $f_{k,b}(x)$ - see Definition 6.3.1 for a reminder). Let $T_k = T_{k,0} \cup T_{k,1}$.

We begin by writing the state of $\mathbb{P}$ after application of U_0. Recall from Claim 6.4.4 that when the verifier requests test round measurement results from $\mathbb{P}$, $\mathbb{P}$ simply measures the requested registers in the standard basis and sends the results to the verifier. Since $\mathbb{P}$ is a perfect prover, it follows that the state of $\mathbb{P}$ after applying U_0 must yield measurement results which pass the test round perfectly. The state of $\mathbb{P}$ after applying U_0 can therefore be written as:

$$U_0 |0\rangle^{\otimes e} |k\rangle |\phi_{k>1}\rangle = \sum_{\substack{b \in \{0,1\} \\ y \in T_{k,b}}} |b, x_{b,y}\rangle |\psi_{b,y,k}\rangle |y\rangle \tag{6.69}$$

where $x_{b,y}$ is the output of $\text{INV}_{\mathcal{F}}(t_k, b, y)$. We have suppressed the dependence of $x_{b,y}$ on k for convenience. The state in (6.69) can be written in this format since if the prover returns $y \in \mathcal{Y}$ in the commitment stage (step 2 of Protocol 6.4.1), in the test round he must return $b \in \{0,1\}$ and $x \in \mathcal{X}$ such that $\text{CHK}_{\mathcal{F}}(t_k, b, x, y) = 1$. Conditions 3(a) and 3(b) of Definition 6.3.1 imply that only $x_{b,y} = \text{INV}_{\mathcal{F}}(t_k, b, y)$ satisfies this condition. The auxiliary space represented by $|\psi_{b,y,k}\rangle$ includes the remaining $n-1$ committed qubits, preimage registers and commitment strings as well as the state $|\phi_{k>1}\rangle$.

For convenience, we will instead write the state in (6.69) as follows:

$$U_0 |0\rangle^{\otimes e} |k\rangle = \sum_{\substack{b \in \{0,1\} \\ y \in T_k}} |b, x_{b,y}\rangle |\psi_{b,y,k}\rangle |y\rangle \tag{6.70}$$

The only change we have made is we have replaced the summation over $y \in T_{k,b}$ with the summation over $y \in T_k = T_{k,0} \cup T_{k,1}$. Note that the existence of two inverses of y ($x_{0,y}$

and $x_{1,y}$) is guaranteed since $y \in T_k$ - see Definition 6.3.1. For b, y for which $y \notin T_{k,b}$, let $|\psi_{b,y,k}\rangle = 0$.

After the prover measures y and sends it to the verifier, the state shared between the prover and verifier is:

$$\sum_{y \in T_k} (\sum_{b \in \{0,1\}} |b, x_{b,y}\rangle |\psi_{b,y,k}\rangle)(\sum_{b \in \{0,1\}} |b, x_{b,y}\rangle |\psi_{b,y,k}\rangle)^\dagger \otimes |y\rangle \langle y| \tag{6.71}$$

and the last register (containing y) is held by the verifier. Next, the prover applies the injective map U_J (see Section 6.4.1.1) to all n preimage registers along with auxiliary 0 qubits, which we assume have already been included in the extra space $|\psi_{b,y,k}\rangle$. At this point, the state shared between the prover and verifier is:

$$\rho_k = \sum_{y \in T_k} \rho_{y,k} \tag{6.72}$$

where

$$\rho_{y,k} = (\sum_{b \in \{0,1\}} |b, J(x_{b,y})\rangle \left|\psi'_{b,y,k}\right\rangle)(\sum_{b \in \{0,1\}} |b, J(x_{b,y})\rangle \left|\psi'_{b,y,k}\right\rangle)^\dagger \otimes |y\rangle \langle y| \tag{6.73}$$

The auxiliary space $|\psi_{b,y,k}\rangle$ has changed to $\left|\psi'_{b,y,k}\right\rangle$ to account for the fact that the commitment strings corresponding to indices $i > 1$ were measured and the unitary U_J was applied to the corresponding preimage registers in the auxiliary space (we are considering a purification of the auxiliary space for convenience).

The prover then applies his CPTP map $\mathcal{S} = \{B_\tau\}_\tau$ followed by Hadamard basis measurement of the first committed qubit and preimage register of the state in (6.72). The state shared between the prover and verifier at this point is:

$$\sum_{\substack{b' \in \{0,1\}, \tau \\ d \in \{0,1\}^w}} (|b'\rangle \langle b'| \otimes |d\rangle \langle d| \otimes \mathcal{I})(H^{\otimes l+1} \otimes \mathcal{I}) B_\tau \rho_k B_\tau^\dagger (H^{\otimes l+1} \otimes \mathcal{I})^\dagger (|b'\rangle \langle b'| \otimes |d\rangle \langle d| \otimes \mathcal{I})^\dagger \tag{6.74}$$

Next, if the measurement result $d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$, the verifier decodes the first qubit by applying the operator $X^{d \cdot (J(x_{0,y}) \oplus J(x_{1,y}))}$ (see step 5(d) of Protocol 6.4.1). If not, the verifier stores a random bit as his measurement result; we can equivalently assume the verifier decodes the first qubit by applying a random X operator. Note that the verifier's decoding (the application of the X operator) commutes with the prover's measurement of the first qubit. Therefore, the entire state, including the verifier's decoding, can be written as:

$$\sigma_{0,k} = \sum_{\substack{b', c \in \{0,1\}, \tau \\ d \in \{0,1\}^w, y \in R_{c,d,k}}} \delta_{d,y} (|b'\rangle \langle b'| X^c \otimes |d\rangle \langle d| \otimes \mathcal{I})(H^{\otimes l+1} \otimes \mathcal{I}) B_\tau \rho_{y,k} B_\tau^\dagger (H^{\otimes l+1} \otimes \mathcal{I})^\dagger (|b'\rangle \langle b'| X^c \otimes |d\rangle \langle d| \otimes \mathcal{I})^\dagger \tag{6.75}$$

where $\delta_{d,y} = \frac{1}{2}$ if $d \notin G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$ and 1 if $d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$ and

$$R_{c,d,k} = \{y \in T_k | (d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}} \wedge d \cdot (J(x_{0,y}) \oplus J(x_{1,y})) = c) \vee (d \notin G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}})\} \tag{6.76}$$

For ease of notation, we will instead write the state in (6.75) as:

$$\sigma_{0,k} = \sum_{\substack{b',c\in\{0,1\},\tau \\ d\in\{0,1\}^w, y\in R_{c,d,k}}} \delta_{d,y} O_{b',c,d,\tau} \rho_{y,k} O^\dagger_{b',c,d,\tau} \tag{6.77}$$

where

$$O_{b',c,d,\tau} \quad = \quad (|b'\rangle\langle b'| X^c \otimes |d\rangle\langle d| \otimes \mathcal{I})(H^{\otimes l+1} \otimes \mathcal{I}) B_\tau \tag{6.78}$$

Let $\mathcal{S}_{>1}$ be the CPTP map which contains all operations done on the remaining $n-1$ committed qubits and preimage registers after application of the attack $\mathcal{S}$: $\mathcal{S}_{>1}$ consists of the Hadamard measurement of the remaining $n-1$ committed qubits and preimage registers as well as the verifier decoding of those committed qubits. $\mathcal{S}_{>1}$ is independent of the function key k and trapdoor t_k; it is only dependent on the remaining $n-1$ function keys and trapdoors, which are drawn independently and included in the auxiliary space of $\sigma_{0,k}$. Given this, the state $\sigma_{\mathbb{P},h}$ is obtained by applying $\mathcal{S}_{>1}$, and then tracing out all but the first n qubits (the committed qubits):

$$\sigma_{\mathbb{P},h} = \mathrm{Tr}_{>n}(\mathcal{S}_{>1}(\sum_{k\in\mathcal{K}_\mathcal{F}} D_{\mathbb{V},h}(k)\sigma_{0,k})) \tag{6.79}$$

where $D_{\mathbb{V},h}$ is the distribution over the set of function keys $\mathcal{K}_\mathcal{F}$ (since $h=1$) produced by $\mathrm{GEN}_\mathcal{F}$.

To prove the claim, we need to show that $\sigma_{\mathbb{P},h}$ is computationally indistinguishable from $\sigma_{\mathbb{P}_1,h}$. Let

$$\sigma_{\mathbb{P},h,E} = \sum_{k\in\mathcal{K}_\mathcal{F}} D_{\mathbb{V},h}(k)\sigma_{0,k} \tag{6.80}$$

We will instead prove the stronger statement that $\sigma_{\mathbb{P},h,E}$ is computationally indistinguishable from $\sigma_{\mathbb{P}_1,h,E}$ (to see why this is stronger, observe from (6.79) that $\sigma_{\mathbb{P},h}$ can be obtained from $\sigma_{\mathbb{P},h,E}$ by applying the efficiently computable superoperator $\mathcal{S}_{>1}$ and tracing out all but the first n qubits). Recall from the statement of the claim that $\mathbb{P}_1$ is characterized by $(U_0, \mathcal{S}_1)$. Since $\mathcal{S}_1$ is followed by Hadamard basis measurement, Corollary 6.2.2 implies that $\mathbb{P}_1$ is also characterized by $(U_0, \{\frac{1}{\sqrt{2}}(Z^r\otimes\mathcal{I})\mathcal{S}_1(Z^r\otimes\mathcal{I})\}_{r\in\{0,1\}})$. If we let $\hat{\mathbb{P}}_1$ be the prover characterized by $(U_0, (Z\otimes\mathcal{I})\mathcal{S}(Z\otimes\mathcal{I}))$, it follows by linearity that

$$\frac{1}{2}(\sigma_{\mathbb{P},h,E} + \sigma_{\hat{\mathbb{P}}_1,h,E}) = \sigma_{\mathbb{P}_1,h,E} \tag{6.81}$$

Therefore, to complete the proof of Claim 6.5.1, we can instead show that $\sigma_{\mathbb{P},h,E}$ is computationally indistinguishable from $\sigma_{\hat{\mathbb{P}}_1,h,E}$, which implies that $\sigma_{\mathbb{P},h,E}$ is computationally indistinguishable from $\sigma_{\mathbb{P}_1,h,E}$.

Computational Indistinguishability For convenience, let $\sigma_{\mathbb{P},h,E} = \sigma_0$ and $\sigma_{\hat{\mathbb{P}}_1,h,E} = \sigma_1$.

We now prove that σ_0 and σ_1 are computationally indistinguishable; our proof follows the outline given in Section 6.1.4.1. As given in (6.80):

$$\sigma_r = \sum_{k \in \mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\sigma_{r,k} \tag{6.82}$$

and using (6.77):

$$\sigma_{r,k} = \sum_{\substack{b',c \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in R_{c,d,k}}} \delta_{d,y} O_{b',c\oplus r,d,\tau}(Z^r \otimes \mathcal{I})\rho_{y,k}(Z^r \otimes \mathcal{I})O^{\dagger}_{b',c\oplus r,d,\tau} \tag{6.83}$$

Note that, in the case that $r = 1$, the operator $(Z \otimes \mathcal{I})$ acting after B_τ was absorbed into $O_{b',c,d,\tau}$ to create $O_{b',c\oplus r,d,\tau}$. Recall from (6.73) that:

$$\rho_{y,k} = (\sum_{b \in \{0,1\}} |b, J(x_{b,y})\rangle \left|\psi'_{b,y,k}\right\rangle)(\sum_{b \in \{0,1\}} |b, J(x_{b,y})\rangle \left|\psi'_{b,y,k}\right\rangle)^{\dagger} \otimes |y\rangle\langle y| \tag{6.84}$$

We can break down the state $\rho_{y,k}$ into two components:

$$\rho_{y,k} = \rho^D_{y,k} + \rho^C_{y,k} \tag{6.85}$$

The components are as follows:

$$\rho^D_{y,k} = \sum_{b \in \{0,1\}} |b\rangle\langle b| \otimes |J(x_{b,y})\rangle\langle J(x_{b,y})| \otimes \left|\psi'_{b,y,k}\right\rangle\left\langle\psi'_{b,y,k}\right| \otimes |y\rangle\langle y| \tag{6.86}$$

$$\rho^C_{y,k} = \sum_{b \in \{0,1\}} |b\rangle\langle b \oplus 1| \otimes |J(x_{b,y})\rangle\langle J(x_{b\oplus 1,y})| \otimes \left|\psi'_{b,y,k}\right\rangle\left\langle\psi'_{b\oplus 1,y}\right| \otimes |y\rangle\langle y| \tag{6.87}$$

Since Z operators acting on the first qubit have no effect on (6.86) and add a phase of -1 to (6.87), we can rewrite (6.83) as:

$$\sigma_{r,k} = \sum_{\substack{b',c \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in R_{c,d,k}}} \delta_{d,y} O_{b',c\oplus r,d,\tau}(Z^r \otimes \mathcal{I})(\rho^D_{y,k} + \rho^C_{y,k})(Z^r \otimes \mathcal{I})O^{\dagger}_{b',c\oplus r,d,\tau} \tag{6.88}$$

$$= \sum_{\substack{b',c \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in R_{c,d,k}}} \delta_{d,y} O_{b',c\oplus r,d,\tau}(\rho^D_{y,k} + (-1)^r\rho^C_{y,k})O^{\dagger}_{b',c\oplus r,d,\tau} \tag{6.89}$$

To show that σ_0 and σ_1 are computationally indistinguishable, we reduce the problem to showing that the components corresponding to the diagonal and cross terms of the committed state $\rho_{y,k}$ are computationally indistinguishable:

Claim 6.5.2 *If σ_0 is computationally distinguishable from σ_1, then one of the following must hold:*

1. *Let*

$$\sigma_r^D \stackrel{\text{def}}{=} \sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\sigma_{r,k}^D \tag{6.90}$$

$$\sigma_{r,k}^D \stackrel{\text{def}}{=} \sum_{\substack{b',c\in\{0,1\},\tau \\ d\in\{0,1\}^w, y\in R_{c,d,k}}} \delta_{d,y} O_{b',c\oplus r,d,\tau}(\rho_{y,k}^D) O_{b',c\oplus r,d,\tau}^\dagger \tag{6.91}$$

The density matrices σ_0^D and σ_1^D are computationally distinguishable.

2. *Let*

$$\hat{\sigma}_r \stackrel{\text{def}}{=} \sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\hat{\sigma}_{r,k} \tag{6.92}$$

$$\hat{\sigma}_{r,k} \stackrel{\text{def}}{=} (Z^r\otimes\mathcal{I})(\sum_{y\in T_k}\rho_{y,k})(Z^r\otimes\mathcal{I}) \tag{6.93}$$

The density matrices $\hat{\sigma}_0$ and $\hat{\sigma}_1$ are computationally distinguishable.

The first pair of density matrices is equal to the terms of σ_0 and σ_1 resulting from the diagonal term of the committed state and the second pair represents the cross terms. The proof of Claim 6.5.2 is a simple application of the triangle inequality and is given in Section 6.5.3. We complete the proof of Claim 6.5.1 with the following two claims:

Claim 6.5.3 *If σ_0^D is computationally distinguishable from σ_1^D, then there exists a BQP attacker $\mathcal{A}$ who can violate the hardcore bit clause of Definition 6.3.1.*

Claim 6.5.4 *If $\hat{\sigma}_0$ is computationally distinguishable from $\hat{\sigma}_1$, then there exists a BQP attacker $\mathcal{A}$ who can violate the hardcore bit clause of Definition 6.3.4.*

We prove Claims 6.5.3 and 6.5.4 in the next two sections.

□

6.5.1 Indistinguishability of Diagonal Terms (Proof of Claim 6.5.3)

The overview of the following proof is given in Section 6.1.4.1.

*P*roof of **Claim 6.5.3**: Recall that we would like to show that the density matrices σ_0^D and σ_1^D are computationally indistinguishable. We will proceed by contradiction. Assume σ_0^D and σ_1^D are distinguishable using an efficiently computable CPTP map $\mathcal{S}$. It follows by the definition of computational indistinguishability (Definition 3.3.2) and by the expression for σ_r^D in (6.90) that the following expression is non negligible:

$$|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\sum_{r\in\{0,1\}}(-1)^r\sigma_{r,k}^D))| \tag{6.94}$$

We will use the CPTP map $\mathcal{S}$ to construct an attacker $\mathcal{A}$ who violates the hardcore bit clause of Definition 6.3.1.

Let

$$R^D_{c,d,k} = \{y \in T_k | d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}} \wedge d \cdot (J(x_{0,y}) \oplus J(x_{1,y})) = c\} \tag{6.95}$$

We will require the unnormalized state $\tilde{\sigma}^D_{r,k}$, which is the state $\sigma^D_{r,k}$ from (6.91) conditioned on obtaining measurements y, d such that $d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$:

$$\tilde{\sigma}^D_{r,k} = \sum_{\substack{b',c \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in R^D_{c,d,k}}} O_{b',c\oplus r,d,\tau}(\rho^D_{y,k}) O^\dagger_{b',c\oplus r,d,\tau} \tag{6.96}$$

Observe that for all $k \in \mathcal{K}_\mathcal{F}$

$$\sum_{r\in\{0,1\}} (-1)^r \sigma^D_{r,k} = \sum_{r\in\{0,1\}} (-1)^r \tilde{\sigma}^D_{r,k} \tag{6.97}$$

This is because $\sigma^D_{0,k}$ and $\sigma^D_{1,k}$ are identical when conditioned on $d \notin G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$ (both have a uniform X decoding operator applied). It follows that the expression in (6.94) is equal to:

$$|\sum_{k\in\mathcal{K}_\mathcal{F}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|0\rangle\langle 0| \otimes \mathcal{I})\mathcal{S}(\sum_{r\in\{0,1\}} (-1)^r \tilde{\sigma}^D_{r,k}))| \tag{6.98}$$

The attacker $\mathcal{A}$ (on input $k \in \mathcal{K}_\mathcal{F}$) first constructs the following state from (6.72):

$$\sum_{y\in T_k} \rho_{y,k} \tag{6.99}$$

He does this exactly as the BQP prover $\mathbb{P}$ would have: by applying the prover's initial operator U_0 to the input state and measuring the last register to obtain y. Then $\mathcal{A}$ measures both the committed qubit and preimage register, obtaining $(b, x_{b,y})$. The resulting state is a summation over the state in (6.86):

$$\sum_{y\in T_k} \rho^D_{y,k} \tag{6.100}$$

$\mathcal{A}$ now continues as $\mathbb{P}$ would have: he applies the CPTP map $\{B_\tau\}_\tau$, followed by Hadamard measurement of the committed qubit and preimage register, obtaining results $b' \in \{0,1\}$ and $d \in \{0,1\}^w$. $\mathcal{A}$ then chooses a bit c' at random, stores the bit c' in an auxiliary register, and applies $X^{c'}$ to the committed qubit (this operation commutes with measurement of the committed qubit). The unnormalized state created by $\mathcal{A}$ (conditioned on d, y such that $d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$) is equal to:

$$\frac{1}{2} \sum_{\substack{b',c' \in \{0,1\},\tau,y\in T_k \\ d\in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}}} O_{b',c',d,\tau}(\rho^D_{y,k}) O^\dagger_{b',c',d,\tau} \otimes |c'\rangle\langle c'| \tag{6.101}$$

We will partition the above state into components using the following projection (the set $R^D_{c,d,k}$ is defined in (6.95)):

$$P^D_{c,k} = \mathcal{I} \otimes \sum_{d \in \{0,1\}^w, y \in R^D_{c,d,k}} |d\rangle\langle d| \otimes \mathcal{I} \otimes |y\rangle\langle y| \tag{6.102}$$

The state of $\mathcal{A}$ in (6.101) can now be written in terms of the state $\tilde{\sigma}^D_{r,k}$, as defined in (6.96):

$$= \frac{1}{2} \sum_{c,c' \in \{0,1\}} P^D_{c,k} \tilde{\sigma}^D_{c \oplus c',k} P^D_{c,k} \otimes |c'\rangle\langle c'| \tag{6.103}$$

Finally, $\mathcal{A}$ applies the efficiently computable CPTP map $\mathcal{S}$ (which is used to distinguish between σ^D_0 and σ^D_1) to the state in (6.103) and measures the first qubit. If the result of the measurement is $r \in \{0,1\}$, $\mathcal{A}$ outputs $b, x_{b,y}, d, c' \oplus r$.

In order to violate the hardcore bit clause of Definition 6.3.1, $\mathcal{A}$ must output $(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1)))$ with non negligible advantage (over outputting $(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1)) \oplus 1)$). More formally, we need to show that the following advantage of $\mathcal{A}$ (taken from Definition 6.3.1) is non negligible:

$$\left| \Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in \overline{H}_k] \right| \leq \mu \tag{6.104}$$

where

$$\begin{aligned} H_k &= \{(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1)))| b \in \{0,1\}, (x_0, x_1) \in \mathcal{R}_k, d \in G_{k,0,x_0} \cap G_{k,1,x_1}\} \\ \overline{H}_k &= \{(b, x_b, d, c) | (b, x_b, d, c \oplus 1) \in H_k\} \end{aligned}$$

$\mathcal{A}$ outputs a string in H_k if, on components $P^D_{c,k} \tilde{\sigma}^D_{0,k} P^D_{c,k}$ and $P^D_{c,k} \tilde{\sigma}^D_{1,k} P^D_{c,k}$, the final bit of $\mathcal{A}$'s output is c. This occurs as long as the distinguishing operator $\mathcal{S}$ outputs r on components $P^D_{0,k} \tilde{\sigma}^D_{r,k} P^D_{0,k}$ and $P^D_{1,k} \tilde{\sigma}^D_{r,k} P^D_{1,k}$. It follows that the probability that $\mathcal{A}$ outputs a string in H_k is equal to:

$$\begin{aligned} \Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in H_k] &= \frac{1}{2} \sum_{\substack{k \in \mathcal{K}_{\mathcal{F}} \\ r \in \{0,1\}}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|r\rangle\langle r| \otimes \mathcal{I}) \mathcal{S}(\sum_{c \in \{0,1\}} P^D_{c,k} \tilde{\sigma}^D_{r,k} P^D_{c,k})) \\ &= \frac{1}{2} \sum_{\substack{k \in \mathcal{K}_{\mathcal{F}} \\ r \in \{0,1\}}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|r\rangle\langle r| \otimes \mathcal{I}) \mathcal{S}(\tilde{\sigma}^D_{r,k})) \end{aligned} \tag{6.105}$$

By similar reasoning,

$$\Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in \overline{H}_k] = \frac{1}{2} \sum_{\substack{k \in \mathcal{K}_{\mathcal{F}} \\ r \in \{0,1\}}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|r \oplus 1\rangle\langle r \oplus 1| \otimes \mathcal{I}) \mathcal{S}(\tilde{\sigma}^D_{r,k})) \tag{6.106}$$

By combining (6.105) and (6.106) and then using the equality in (6.97), we obtain that the advantage of $\mathcal{A}$ in (6.104) is equal to:

$$\Big|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathrm{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\tilde{\sigma}^D_{0,k}-\tilde{\sigma}^D_{1,k}))\Big| = \Big|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathrm{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\sigma^D_{0,k}-\sigma^D_{1,k}))\Big| \tag{6.107}$$

The expression in (6.107) is non negligible, due to our initial assumption that the CPTP map $\mathcal{S}$ can distinguish between σ^D_0 and σ^D_1 (see (6.94)).

□

6.5.2 Indistinguishability of Cross Terms (Proof of Claim 6.5.4)

The overview of the following proof is given in Section 6.1.4.1.

Proof of **Claim 6.5.4**: Recall that we would like to show that the density matrices $\hat{\sigma}_0$ and $\hat{\sigma}_1$ are computationally indistinguishable. We will proceed by contradiction. Assume the two matrices $\hat{\sigma}_0$ and $\hat{\sigma}_1$ are computationally distinguishable using the efficiently computable CPTP map $\mathcal{S}$. It follows by the definition of computational indistinguishability (Definition 3.3.2) and the expression for $\hat{\sigma}_r$ in (6.92) that the following expression is non negligible:

$$|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathrm{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\sum_{r\in\{0,1\}}(-1)^r\hat{\sigma}_{r,k}))| \tag{6.108}$$

We will use the CPTP map $\mathcal{S}$ to construct a BQP attacker $\mathcal{A}$ who will violate the hardcore bit clause of Definition 6.3.4.

Fix a string $d\in\{0,1\}^w$. The attacker $\mathcal{A}$ (on input $k\in\mathcal{K}_{\mathcal{F}}$) first constructs the following state from (6.72):

$$\sum_{y\in T_k}\rho_{y,k} \tag{6.109}$$

He does this exactly as the BQP prover $\mathbb{P}$ would have: by applying the prover's initial operator U_0 the to input state and measuring the last register to obtain y. Then $\mathcal{A}$ applies Z^d to the preimage register. The resulting state is:

$$(\mathcal{I}\otimes Z^d\otimes\mathcal{I})(\sum_{y\in T_k}\rho_{y,k})(\mathcal{I}\otimes Z^d\otimes\mathcal{I}) = \sum_{y\in T_k}(Z^{d\cdot(J(x_{0,y})\oplus J(x_{1,y}))}\otimes\mathcal{I})\rho_{y,k}(Z^{d\cdot(J(x_{0,y})\oplus J(x_{1,y}))}\otimes\mathcal{I}) \tag{6.110}$$

The equality is due to the format of the state $\rho_{y,k}$, as written in (6.73):

$$\rho_{y,k} = (\sum_{b\in\{0,1\}}|b,J(x_{b,y})\rangle\left|\psi'_{b,y,k}\right\rangle)(\sum_{b\in\{0,1\}}|b,J(x_{b,y})\rangle\left|\psi'_{b,y,k}\right\rangle)^\dagger\otimes|y\rangle\langle y| \tag{6.111}$$

If we let

$$R^C_{c,d,k} = \{y\in T_k | d\cdot(J(x_{0,y})\oplus J(x_{1,y}))=c\} \tag{6.112}$$

the expression in (6.110) is equal to:

$$\sum_{c\in\{0,1\},y\in R^C_{c,d,k}} (Z^c\otimes\mathcal{I})\rho_{y,k}(Z^c\otimes\mathcal{I}) \tag{6.113}$$

Finally, $\mathcal{A}$ chooses a random bit c', applies $Z^{c'}$ to the committed qubit and stores c' in an auxiliary register. Continuing from (6.113), the state of $\mathcal{A}$ at this point is equal to:

$$\frac{1}{2}\sum_{c,c'\in\{0,1\},y\in R^C_{c,d,k}} (Z^{c\oplus c'}\otimes\mathcal{I})\rho_{y,k}(Z^{c\oplus c'}\otimes\mathcal{I})\otimes|c'\rangle\langle c'| \tag{6.114}$$

We partition the state in (6.114) into components using the following projection:

$$P^C_{c,k} \;=\; \mathcal{I}\otimes\sum_{y\in R^C_{c,d,k}}|y\rangle\langle y| \tag{6.115}$$

The state in (6.114) can now be written in terms of $\hat{\sigma}_{r,k}$, as defined in (6.93)

$$\hat{\sigma}_{r,k} = (Z^r\otimes\mathcal{I})(\sum_{y\in T_k}\rho_{y,k})(Z^r\otimes\mathcal{I}) \tag{6.116}$$

as follows:

$$= \frac{1}{2}\sum_{c,c'\in\{0,1\}} P^C_{c,k}(Z^{c\oplus c'}\otimes\mathcal{I})(\sum_{y\in T_k}\rho_{y,k})(Z^{c\oplus c'}\otimes\mathcal{I})P^C_{c,k}\otimes|c'\rangle\langle c'| \tag{6.117}$$

$$= \frac{1}{2}\sum_{c,c'\in\{0,1\}} P^C_{c,k}(\hat{\sigma}_{c\oplus c',k})P^C_{c,k}\otimes|c'\rangle\langle c'| \tag{6.118}$$

Finally, $\mathcal{A}$ applies the CPTP map $\mathcal{S}$ (which is used to distinguish between $\hat{\sigma}_0$ and $\hat{\sigma}_1$) to the state in (6.118) and measures the first qubit. If the result of the measurement is $r\in\{0,1\}$, $\mathcal{A}$ outputs $c'\oplus r$.

In order to violate the hardcore bit clause of Definition 6.3.4, $\mathcal{A}$ must guess the value of $d\cdot(J(x_{0,y})\oplus J(x_{1,y}))$ with non negligible advantage. More formally, we need to show that the following advantage of $\mathcal{A}$ (taken from Definition 6.3.4) is non negligible:

$$\Big|\Pr_{(k,t_k)\leftarrow\mathrm{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k)\in H'_{k,d}]-\frac{1}{2}\Big| \tag{6.119}$$

where

$$H'_{k,d} \;=\; \{d\cdot(J(x_0)\oplus J(x_1))|(x_0,x_1)\in\mathcal{R}_k\} \tag{6.120}$$

The bit output by $\mathcal{A}$ is in $H'_{k,d}$ if, on components $P^C_{c,k}\hat{\sigma}_{0,k}P^C_{c,k}$ and $P^C_{c,k}\hat{\sigma}_{1,k}P^C_{c,k}$, $\mathcal{A}$ outputs c. This occurs as long as the distinguishing operator $\mathcal{S}$ outputs r on components $P^C_{0,k}\hat{\sigma}_{r,k}P^C_{0,k}$ and $P^C_{1,k}\hat{\sigma}_{r,k}P^C_{1,k}$. It follows that the probability that $\mathcal{A}$ outputs a value in $H'_{k,d}$ is equal to:

$$\begin{aligned}
\Pr_{(k,t_k)\leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k,d)\in H'_{k,d}] &= \frac{1}{2}\sum_{\substack{k\in\mathcal{K}_{\mathcal{F}}\\ r\in\{0,1\}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|r\rangle\langle r|\otimes\mathcal{I})\mathcal{S}(\sum_{c\in\{0,1\}} P^C_{c,k}\hat{\sigma}_{r,k}P^C_{c,k})) \\
&= \frac{1}{2}\sum_{\substack{k\in\mathcal{K}_{\mathcal{F}}\\ r\in\{0,1\}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|r\rangle\langle r|\otimes\mathcal{I})\mathcal{S}(\hat{\sigma}_{r,k})) \\
&= \frac{1}{2}\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\hat{\sigma}_{0,k}-\hat{\sigma}_{1,k}))+\frac{1}{2}
\end{aligned}$$

We can use the above expression to write the advantage of $\mathcal{A}$ from (6.119) as:

$$\frac{1}{2}\Big|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\hat{\sigma}_{0,k}-\hat{\sigma}_{1,k}))\Big| \tag{6.121}$$

The expression in (6.121) is non negligible, due to our initial assumption that the CPTP map $\mathcal{S}$ can distinguish between $\hat{\sigma}_0$ and $\hat{\sigma}_1$ (see (6.108)).

□

6.5.3 Reduction to Diagonal/Cross Terms (Proof of Claim 6.5.2)

Proof: If σ_0 is computationally distinguishable from σ_1, by Definition 3.3.2 and the expression for σ_r in (6.82) there exists an efficiently computable CPTP map $\mathcal{S}$ for which the following expression is non negligible:

$$\Big|\sum_{k\in\mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k)\cdot \mathrm{Tr}((|0\rangle\langle 0|\otimes\mathcal{I})\mathcal{S}(\sum_{r\in\{0,1\}}(-1)^r\sigma_{r,k}))\Big| \tag{6.122}$$

We use the expression for $\sigma_{r,k}$ from (6.89) and the matrix $\sigma^D_{r,k}$ from (6.91) to define the matrix $\sigma^C_{r,k}$:

$$\begin{aligned}
\sigma_{r,k} &= \sum_{\substack{b',c\in\{0,1\},\tau\\ d\in\{0,1\}^w, y\in R_{c,d,k}}} \delta_{d,y} O_{b',c\oplus r,d,\tau}(\rho^D_{y,k}+(-1)^r\rho^C_{y,k})O^\dagger_{b',c\oplus r,d,\tau} &(6.123)\\
&= \sigma^D_{r,k}+\sigma^C_{r,k} &(6.124)
\end{aligned}$$

By combining the following equality

$$\sum_{r\in\{0,1\}}(-1)^r\sigma_{r,k} = \sum_{r\in\{0,1\}}(-1)^r\sigma^D_{r,k}+\sum_{r\in\{0,1\}}(-1)^r\sigma^C_{r,k} \tag{6.125}$$

with the triangle inequality, it follows that if the quantity in (6.122) is non negligible, one of the following two quantities must be non negligible:

$$\Big| \sum_{k \in \mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|0\rangle \langle 0| \otimes \mathcal{I}) \mathcal{S}(\sum_{r \in \{0,1\}} (-1)^r \sigma^D_{r,k})) \Big| \tag{6.126}$$

$$\Big| \sum_{k \in \mathcal{K}_{\mathcal{F}}} D_{\mathbb{V},h}(k) \cdot \mathrm{Tr}((|0\rangle \langle 0| \otimes \mathcal{I}) \mathcal{S}(\sum_{r \in \{0,1\}} (-1)^r \sigma^C_{r,k})) \Big| \tag{6.127}$$

If the quantity in (6.126) is non negligible, σ^D_0 is computationally distinguishable from σ^D_1. To complete the proof of Claim 6.5.2, we will show that if the quantity in 6.127 is non negligible, $\hat{\sigma}_0$ is computationally distinguishable from $\hat{\sigma}_1$ ($\hat{\sigma}_r$ is defined in (6.93) and copied below in (6.134)). To do this, we show below that for the efficiently computable CPTP map $\mathcal{S}' = \{\frac{1}{\sqrt{2}} O_{b',c,d,\tau}\}_{b',c,d}$ ($O_{b',c,d,\tau}$ is introduced in (6.78)):

$$\sum_{r \in \{0,1\}} (-1)^r \sigma^C_{r,k} = \mathcal{S}'(\sum_{r \in \{0,1\}} (-1)^r \hat{\sigma}_{r,k}) \tag{6.128}$$

Therefore, if the quantity in (6.127) is non negligible, $\hat{\sigma}_0$ is computationally distinguishable from $\hat{\sigma}_1$ by using the CPTP map $\mathcal{S}\mathcal{S}'$.

We now prove (6.128), beginning with the expression for $\sigma^C_{r,k}$ in (6.124). First, we observe that

$$\sigma^C_{1,k} = -(X \otimes \mathcal{I}) \sigma^C_{0,k} (X \otimes \mathcal{I}) \tag{6.129}$$

Therefore:

$$\begin{aligned} \sigma^C_{0,k} - \sigma^C_{1,k} &= \sum_{r \in \{0,1\}} (X^r \otimes \mathcal{I}) \sigma^C_{0,k} (X^r \otimes \mathcal{I}) && (6.130) \\ &= \sum_{\substack{b',c,r \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in R_{c,d,k}}} \delta_{d,y} O_{b',r,d,\tau}(\rho^C_{y,k}) O^\dagger_{b',r,d,\tau} && (6.131) \\ &= \sum_{\substack{b',c \in \{0,1\},\tau \\ d \in \{0,1\}^w, y \in T_k}} O_{b',c,d,\tau}(\rho^C_{y,k}) O^\dagger_{b',c,d,\tau} && (6.132) \end{aligned}$$

The second inequality follows since $\delta_{d,y} = \frac{1}{2}$ if $d \notin G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$ and 1 if $d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}}$ and (from (6.76))

$$R_{c,d,k} = \{y \in T_k | (d \in G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}} \wedge d \cdot (J(x_{0,y}) \oplus J(x_{1,y})) = c) \vee (d \notin G_{k,0,x_{0,y}} \cap G_{k,1,x_{1,y}})\} \tag{6.133}$$

Recall the state $\hat{\sigma}_{r,k}$ from (6.93):

$$\hat{\sigma}_{r,k} = (Z^r \otimes \mathcal{I})(\sum_{y \in T_k} \rho_{y,k})(Z^r \otimes \mathcal{I}) \tag{6.134}$$

We use the equality $\rho_{y,k} = \rho^D_{y,k} + \rho^C_{y,k}$ from (6.85):

$$\hat{\sigma}_{0,k} - \hat{\sigma}_{1,k} = \sum_{y \in T_k} (\rho^D_{y,k} + \rho^C_{y,k}) - (Z \otimes \mathcal{I}) \sum_{y \in T_k} (\rho^D_{y,k} + \rho^C_{y,k})(Z \otimes \mathcal{I}) \tag{6.135}$$

$$= 2 \sum_{y \in T_k} \rho^C_{y,k} \tag{6.136}$$

Plugging the equality in (6.136) into (6.132) yields (6.128).

□

6.6 Measurement Protocol Soundness

In this section, we prove soundness of the measurement protocol, as stated in the following claim:

Claim 6.6.1 ***Soundness of Protocol 6.4.1*** *For a prover $\mathbb{P}$ in Protocol 6.4.1, let $1-p_{h,H}$ be the probability that the verifier accepts $\mathbb{P}$ on basis choice h in the Hadamard round and $1-p_{h,T}$ be the probability that the verifier accepts $\mathbb{P}$ in the test round. There exists a state ρ, a prover $\mathbb{P}'$ and a negligible function μ such that for all h, $\|D_{\mathbb{P},h} - D_{\mathbb{P}',h}\|_{TV} \leq p_{h,H} + \sqrt{p_{h,T}} + \mu$ and $D_{\mathbb{P}',h}$ is computationally indistinguishable from the distribution $D_{\rho,h}$ which results from measuring ρ in the basis determined by h.*

Throughout this section, we will use notation introduced in Section 6.4.3. To prove Claim 6.6.1, we will proceed as follows. We begin by transitioning from a general prover to a *perfect* prover, i.e. a prover who is always accepted in the test run by the verifier. We show in the following claim (which we prove in Section 6.6.1) that for all provers $\mathbb{P}$, there exists a perfect prover $\mathbb{P}'$ such that the statistical distance between the distributions $D_{\mathbb{P},h}$ and $D_{\mathbb{P}',h}$ is a function of the probability of acceptance of the general prover.

Claim 6.6.2 *For a prover $\mathbb{P}$ in Protocol 6.4.1, let $1-p_{h,H}$ be the probability that the verifier accepts $\mathbb{P}$ on basis choice h in the Hadamard round and $1-p_{h,T}$ be the probability that the verifier accepts $\mathbb{P}$ on basis choice h in the test round. There exists a perfect prover $\mathbb{P}'$ and a negligible function μ such that for all h, $\left\|D^C_{\mathbb{P},h} - D_{\mathbb{P}',h}\right\|_{TV} \leq p_{h,H} + \sqrt{p_{h,T}} + \mu$.*

$\mathbb{P}'$ is created by conditioning $\mathbb{P}$ on acceptance in the test round. This argument is straightforward, but does have one delicate aspect: we need to ensure that the perfect prover is still efficient, even though we have conditioned on acceptance in the test run. This is taken care of by recalling that in the test round, the verifier computes the procedure $\text{CHK}_{\mathcal{F}}$ or $\text{CHK}_{\mathcal{G}}$ (see Protocol 6.4.1). By definition (Definitions 6.3.1 and 6.3.2), both of these procedures require only the function key and not the trapdoor, which implies that the procedures can be computed efficiently by the prover.

Next, we transition from a perfect prover to a trivial prover (as defined in Definition 6.4.6, a trivial prover's Hadamard round attack commutes with standard basis measurement on the n committed qubits):

Claim 6.6.3 *For all perfect provers $\mathbb{P}$, there exists a trivial prover $\hat{\mathbb{P}}$ such that for all h, $D_{\mathbb{P},h}$ is computationally indistinguishable from $D_{\hat{\mathbb{P}},h}$.*

We prove Claim 6.6.3 in Section 6.6.2 as follows. First, in Claim 6.6.4, we prove a statement analogous to Claim 6.5.1 for the standard basis: we show that for an index j such that $h_j = 0$, we can replace the prover's Hadamard round attack with an attack which acts X-trivially on the j^{th} committed qubit to obtain the same distribution over measurements. The proof of Claim 6.6.4 is quite straightforward, since the prover's Hadamard round attack has no effect on the standard basis measurement obtained by the verifier; this measurement is obtained from the commitment y_j. To prove Claim 6.6.3, we sequentially apply Claim 6.5.1 and Claim 6.6.4 to each of the n committed qubits, ending with an attack which commutes with standard basis measurement on all of the committed qubits.

Assuming Claim 6.6.2 and Claim 6.6.3, the proof of Claim 6.6.1 is straightforward:
*P*roof of **Claim 6.6.1**: We begin with a prover $\mathbb{P}$ and apply Claim 6.6.2 to transition to a perfect prover $\mathbb{P}'$ such that for all h, $\left\|D^C_{\mathbb{P},h} - D_{\mathbb{P}',h}\right\|_{TV} \leq p_{h,H} + \sqrt{p_{h,T}} + \mu$ for a negligible function μ. Combining Claim 6.6.3 and Claim 6.4.7 tells us that there exists a state ρ such that for all h, $D_{\mathbb{P}',h}$ is computationally indistinguishable from $D_{\rho,h}$. In more detail, Claim 6.6.3 shows that there exists a trivial prover $\hat{\mathbb{P}}$ such that for all h, $D_{\mathbb{P}',h}$ is computationally indistinguishable from $D_{\hat{\mathbb{P}},h}$. Next, Claim 6.4.7 shows that there exists a state ρ such that for all h, $D_{\hat{\mathbb{P}},h}$ is computationally indistinguishable from $D_{\rho,h}$.
□

6.6.1 General to Perfect Prover (Proof of Claim 6.6.2)

Proof: We begin by observing that by definition of total variation distance (see (3.2)):

$$\left\|D^C_{\mathbb{P},h} - D_{\mathbb{P},h}\right\|_{TV} = p_{h,H} \tag{6.137}$$

It follows by (6.57) that $\left\|\sigma^C_{\mathbb{P},h} - \sigma_{\mathbb{P},h}\right\|_{tr} = p_{h,H}$. In the rest of this proof, we will show that there exists a perfect prover $\mathbb{P}'$ and a negligible function μ such that for all h for which $1 - p_{h,T}$ is non negligible, the trace distance between $\sigma_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}',h}$ is $\leq \sqrt{p_{h,T}} + \mu$. For all h for which $1 - p_{h,T}$ is negligible, the trace distance bound of $\sqrt{p_{h,T}} + \mu$ is trivial and therefore satisfied. It follows by the triangle inequality that for all h, the trace distance between $\sigma^C_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}',h}$ is $\leq p_{h,H} + \sqrt{p_{h,T}} + \mu$ (to complete the proof of the claim, recall from (6.57) that $\left\|\sigma^C_{\mathbb{P},h} - \sigma_{\mathbb{P}',h}\right\|_{tr} = \left\|D^C_{\mathbb{P},h} - D_{\mathbb{P}',h}\right\|_{TV}$).

Distance from perfect prover (between $\sigma_{\mathbb{P},h}$ and $\sigma_{\mathbb{P}',h}$) By Claim 6.4.4, there exist unitaries U_0, U such that the prover $\mathbb{P}$ is characterized by (U_0, U) and the state $\sigma_{\mathbb{P},h}$ can be

created by following steps 1-4 in the statement of the claim. The state after application of U_0 (step 1 of Claim 6.4.4) is:

$$\sigma_{\mathbb{P},h,0} = \sum_{k'} D_{\mathbb{V},h}(k') \cdot U_0(|0\rangle\langle 0|^{\otimes e} \otimes |k'\rangle\langle k'|)U_0^\dagger \tag{6.138}$$

The state above is a mixed state over the verifier's choice of the function key k', which is sampled according to the distribution $D_{\mathbb{V},h}$ (see Section 6.4.3 for a notation reminder). To create $\sigma_{\mathbb{P},h}$ (i.e. the state resulting from the Hadamard round defined in (6.56)), the prover will measure the second to last register (obtaining $y' \in \mathcal{Y}^n$), apply his attack U, measure all committed qubits and preimage registers in the Hadamard basis, send the results to the verifier and discard all other qubits. The verifier will decode the appropriate registers and discard all other measurement results (as described in Protocol 6.4.1). Note that for all provers $\mathbb{P}, \mathbb{P}'$:

$$\|\sigma_{\mathbb{P},h} - \sigma_{\mathbb{P}',h}\|_{tr} \leq \|\sigma_{\mathbb{P},h,0} - \sigma_{\mathbb{P}',h,0}\|_{tr} \tag{6.139}$$

This is because the operators described above which are applied to $\sigma_{\mathbb{P},h,0}$ to create $\sigma_{\mathbb{P},h}$ represent a CPTP map.

We now construct a perfect prover $\mathbb{P}'$ and provide an upper bound for $\|\sigma_{\mathbb{P},h,0} - \sigma_{\mathbb{P}',h,0}\|_{tr}$. We begin by partitioning the state $\sigma_{\mathbb{P},h,0}$ from (6.138) according to acceptance or rejection by the verifier in the test round:

$$\sigma_{\mathbb{P},h,0} = \sum_{k'} D_{\mathbb{V},h}(k') \cdot (|\psi_{ACC,k'}\rangle + |\psi_{REJ,k'}\rangle)(\langle\psi_{ACC,k'}| + \langle\psi_{REJ,k'}|) \otimes |k'\rangle\langle k'| \tag{6.140}$$

where

$$U_0 |0\rangle^{\otimes e} |k'\rangle = U_{0,k'}(|0\rangle^{\otimes e}) \otimes |k'\rangle = (|\psi_{ACC,k'}\rangle + |\psi_{REJ,k'}\rangle) \otimes |k'\rangle \tag{6.141}$$

The first equality in (6.141) is given in (6.65) in Claim 6.4.4, $|\psi_{ACC,k'}\rangle$ (resp. $|\psi_{REJ,k'}\rangle$) is the part of the state $U_{0,k'}(|0\rangle^{\otimes e})$ which will be accepted (resp. rejected) by the verifier in the test round for function choice k', and $\langle\psi_{ACC,k'}|\psi_{REJ,k'}\rangle = 0$. Consider the following state:

$$\sigma_{perfect} = \frac{1}{1 - p_{h,T}} \sum_{k'} D_{\mathbb{V},h}(k') \cdot (|\psi_{ACC,k'}\rangle)(\langle\psi_{ACC,k'}|) \otimes \langle k'|\langle k'| \tag{6.142}$$

The trace distance between $\sigma_{perfect}$ and $\sigma_{\mathbb{P},h,0}$ is at most $\sqrt{p_{h,T}}$. We show below that there exists a CPTP map $\mathcal{S}_0$, a perfect prover $\mathbb{P}'$ characterized by $(\mathcal{S}_0, U)$ and a negligible function μ such that for all h for which $1 - p_{h,T}$ is non negligible, $\|\sigma_{\mathbb{P}',h,0} - \sigma_{perfect}\|_{tr} \leq \mu$. It follows by the triangle inequality that:

$$\begin{aligned} \|\sigma_{\mathbb{P},h,0} - \sigma_{\mathbb{P}',h,0}\|_{tr} &\leq \|\sigma_{\mathbb{P},h,0} - \sigma_{perfect}\|_{tr} + \|\sigma_{perfect} - \sigma_{\mathbb{P}',h,0}\|_{tr} && (6.143) \\ &\leq \sqrt{p_{h,T}} + \mu && (6.144) \end{aligned}$$

It then follows by (6.139) that $\|\sigma_{\mathbb{P},h} - \sigma_{\mathbb{P}',h}\|_{tr} \leq \sqrt{p_{h,T}} + \mu$, which completes the proof of Claim 6.6.2.

We now describe the CPTP map S_0. To do so, we will require the unitary V, which consists of first applying U_0, then applying the verifier's test round check (the algorithm CHK_F) in superposition, and storing the result in an extra register. The result of applying V is:

$$V\Big(\sum_{k'} D_{V;h}(k') \; |0\rangle\langle 0|^{e} \otimes |k'\rangle\langle k'|\Big)V^{\dagger} \tag{6.145}$$

$$= \sum_{k'} D_{V;h}(k') \; (|1\rangle|\mathrm{ACC};k'\rangle + |0\rangle|\mathrm{REJ};k'\rangle)(\langle 1|\langle \mathrm{ACC};k'| + \langle 0|\langle \mathrm{REJ};k'|)^{\dagger} \otimes |k'\rangle\langle k'| \tag{6.146}$$

where the first register contains the result of the algorithm CHK_F. If the first register is measured to obtain a bit b, the probability of obtaining $b = 1$ is $1 - p_{h;T}$. Therefore, for all h for which $1 - p_{h;T}$ is non negligible, there exists a polynomial n_h' such that if V is applied n_h' times to the input state in (6.145) the probability of never obtaining $b = 1$ over n_h' iterations is negligible.

Let $n' = \max_h n_h'$, where the maximum is taken over all h such that $1 - p_{h;T}$ is non negligible. The CPTP map S_0 will apply the unitary V followed by measurement of the first qubit to the input $|0\rangle^{e}|k'\rangle$ at most n' times or until measuring $b = 1$. In the case that the measurement result $b = 1$ is never obtained, the CPTP map S_0 applies a unitary operator which would be applied by an honest prover (described in Protocol 6.4.2). Since the prover in Protocol 6.4.2 is perfect (see Claim 6.4.3), it follows that the prover P' characterized by $(S_0; U)$ is perfect.

Due to the choice of n', it follows that there exists a negligible function ε such that for all h for which $1 - p_{h;T}$ is non negligible

$$\rho_{P';h;0} = S_0\Big(\sum_{k'} D_{V;h}(k') \; (|0\rangle\langle 0|^{e} \otimes |k'\rangle\langle k'|)\Big) = (1 - \varepsilon)\,\rho_{\mathrm{perfect}} + \varepsilon\,\rho_{\mathrm{fail}} \tag{6.147}$$

where ρ_{fail} is the state created in the case that all n' applications of V do not yield 1 as the measurement result. It follows by the convexity of trace distance that for all h for which $1 - p_{h;T}$ is non negligible the trace distance between $\rho_{P';h;0}$ in (6.147) and ρ_{perfect} is ε.

6.6.2 Perfect to Trivial Prover (Proof of Claim 6.6.3)

We begin by proving a claim which is analogous to Claim 6.5.1, but for the standard basis (i.e. if $h_j = 0$):

Claim 6.6.4 Let $1 \le j \le n$. Let $S = \{B_x\}$ and $S_j = \{B^0_{j;x;}\}_{x \in \{0,1\};}$ be CPTP maps written in terms of their Kraus operators:

$$B_x = \sum_{z \in \{0,1\}} X^x Z^z \, B_{jxz} \tag{6.148}$$

$$B'_{j;x;} = \sum_{z \in \{0,1\}} Z^z \quad B_{jxz} \tag{6.149}$$

where B and $B'_{j;x;}$ have been rearranged so that $X^x Z^z$ and Z^z act on the j^{th} qubit. For a unitary operator U_0, let P be a perfect prover characterized by $(U_0; S)$ (see Claim 6.4.4 and notation 6.4.5). Let P_j be a perfect prover characterized by $(U_0; S_j)$. If $h_j = 0$, $D_{P;h} = D_{P_j;h}$.

Proof of Claim 6.6.4 : For convenience, we will assume that $h_1 = 0$ and we will prove the claim for $j = 1$. The proof is identical for all other values of j. The key observation is that we can change the CPTP maps S to act in an arbitrary way on the first qubit (which is also the first committed qubit), as long as its action on the remaining qubits is unchanged. This is because the measurement of the first committed qubit will be ignored by the verifier; the verifier will obtain the standard basis measurement for the first qubit from the commitment y_1. In other words, the first committed qubit is traced out after application of S and standard basis measurement (it is not part of the distribution $D_{P;h}$).

To prove the claim, we will show that the distribution over measurement results remains the same if $S = \{B\}$ is replaced with $S_1 = \{B'_{1;x;}\}_{x \in \{0,1\};}$. Our first step is to replace S with $S^{(1)} = \{\frac{1}{\sqrt{2}}(Z^r \otimes I)B\}_{r \in \{0,1\};}$. As described above, this change has no impact since the measurement of the first committed qubit is ignored. Therefore, if we let $P^{(1)}$ be a perfect prover characterized by $(U_0; S^{(1)})$, $D_{P;h} = D_{P^{(1)};h}$. Next, we replace $S^{(1)}$ with $S^{(2)} = \{\frac{1}{\sqrt{2}}(Z^r \otimes I)B(Z^r \otimes I)\}_{r \in \{0,1\};}$. Observe that the added Z operator (acting prior to B) has no effect on the state of a perfect prover: it acts on the first committed qubit, which must be in a standard basis state since $h_1 = 0$ (the function key k_1 corresponds to a pair of injective functions $g_{k;0}; g_{k;1}$ for which there is only one valid preimage for each commitment string y - see equation 6.9 for more details). It follows that if we let $P^{(2)}$ be the prover characterized by $(U_0; S^{(2)})$, $D_{P^{(1)};h} = D_{P^{(2)};h}$. Finally, since $S^{(2)}$ is followed by Hadamard basis measurement, we can apply Corollary 6.2.2 to replace $S^{(2)}$ with S_1, showing that $D_{P^{(2)};h} = D_{P_1;h}$ (therefore $D_{P;h} = D_{P_1;h}$).

Using Claim 6.5.1 and Claim 6.6.4, we now prove Claim 6.6.3, which is copied here for reference:

Claim 6.6.3 For all perfect provers P, there exists a trivial prover $\hat{P}$ such that for all h, $D_{P;h}$ is computationally indistinguishable from $D_{\hat{P};h}$.

Proof of Claim 6.6.3 : Fix a basis choice $h \in \{0,1\}^n$. Assume for convenience that for indices $1 \leq i \leq t$, $h_i = 1$ and for indices $t+1 \leq i \leq n$, $h_i = 0$. We apply Claim 6.5.1 t times, beginning with the prover P (who is characterized by $(U_0; U)$). Let $\hat{P}_0 = P$ and let $\hat{P}_j$ be

the prover characterized by $(U_0; \{U_x\}_{x \in \{0,1\}^j})$:

$$U = \sum_{x,z \in \{0,1\}^j} X^x Z^z \otimes U^j_{xz} \tag{6.150}$$

$$U_x = \sum_{z \in \{0,1\}^j} Z^z \otimes U^j_{xz} \tag{6.151}$$

where $X^x Z^z$ acts on qubits $1, \ldots, j$ (which are also committed qubits $1, \ldots, j$). The i^{th} application of Claim 6.5.1 shows that $D_{\hat{P}_{i-1};h}$ is computationally indistinguishable from $D_{\hat{P}_i;h}$. It follows by the triangle inequality that $D_{P;h}$ is computationally indistinguishable from $D_{\hat{P}_t;h}$. We complete the proof of Claim 6.6.3 by applying Claim 6.6.4 t times, which tells us that $D_{\hat{P}_t;h} = D_{\hat{P};h}$, where $\hat{P} = \hat{P}_n$ is a trivial prover characterized by $(U_0; \{U_x\}_{x \in \{0,1\}^n})$.

6.7 Extension of Measurement Protocol to a Verification Protocol for BQP

To extend Protocol 6.4.1 to a $QPIP_0$, we will use the $QPIP_1$ protocol in [36]. We begin by presenting this protocol.

6.7.1 Morimae-Fitzsimons Protocol

Most of this description is taken directly from [36]. Let L be a language in BQP. Since BQP is contained in QMA, for all $x \in L$, there exists a local Hamiltonian H such that

1. if $x \in L$, then the ground energy of H is $\leq a$
2. if $x \notin L$, then the ground energy of H is $\geq b$

where $b - a \geq \frac{1}{\text{poly}(|x|)}$. It is known that H can be a 2-local Hamiltonian with only X and Z operators ([7]).

Let us write the 2-local Hamiltonian as $H = \sum_S d_S S$, where d_S is a real number and S is a tensor product of Pauli operators, where only two operators are Z or X and others are I. We define the rescaled Hamiltonian:

$$H' = \sum_S \alpha_S P_S; \tag{6.152}$$

where $\alpha_S = \frac{|d_S|}{\sum_S |d_S|} \geq 0$ and $P_S = \frac{I + \text{sign}(d_S) S}{2}$.

We now present the protocol:

Protocol 6.7.1 [36] This protocol is used to verify that an instance $x \in L$ for a language $L \in BQP$. Let H be the Hamiltonian which corresponds to x, and define H' as in 6.152.

1. The verifier randomly chooses S with probability α_S.
2. The prover sends the verifier a state σ, sending one qubit at a time.
3. The verifier measures S by performing single qubit measurements of only two qubits of σ in the X or Z basis, discarding all other qubits of σ without measuring them.
4. The verifier computes the product of the measurement results. If the verifier obtains the result $\text{sign}(d_S)$ the verifier accepts.

An honest prover would simply send the ground state of H^0.

Theorem 6.7.2 [36] *Protocol 6.7.1 is a $QPIP_1$ for all languages in BQP with completeness c and soundness s, where $c - s$ is inverse polynomial in $|x|$.*

Proof of Theorem 6.7.2: It was shown (in [37]) that the acceptance probability of each round of Protocol 6.7.1 is

$$p_{acc} = 1 - \frac{1}{\sum_S 2|d_S|}\left(\text{Tr}(H\sigma) + \sum_S |d_S|\right) \tag{6.153}$$

which is

$$p_{acc} \geq \frac{1}{2} + \frac{a}{\sum_S 2|d_S|} \tag{6.154}$$

when $x \in L$, and

$$p_{acc} \leq \frac{1}{2} + \frac{b}{\sum_S 2|d_S|} \tag{6.155}$$

when $x \notin L$. Their difference is $\frac{1}{\text{poly}(|x|)}$.

We require the following version of Protocol 6.7.1 with an amplified completeness/soundness gap:

Protocol 6.7.3 [36] *This protocol is used to verify that an instance $x \in L$ for a language $L \in BQP$. Let H be the Hamiltonian which corresponds to x, and define H^0 as in 6.152. Let k' be a polynomial in $|x|$.*

1. The verifier randomly chooses $S_1, \ldots, S_{k'}$ independently, each with probability α_{S_i}.
2. The prover sends the verifier a state σ', sending one qubit at a time.
3. The verifier measures each S_i (for $1 \leq i \leq k'$) by performing single qubit measurements of only two qubits of σ' in the X or Z basis, discarding all other qubits of σ' without measuring them.
4. The verifier computes the product of the measurement results for each S_i. If the verifier obtains the result $\text{sign}(d_{S_i})$ more than half of the time the verifier accepts.

In Protocol 6.7.3, an honest prover would simply send k^0 copies of the ground state of H^0.

Theorem 6.7.4 [36] Protocol 6.7.1 is a QPIP$_1$ for all languages in BQP with completeness 1 and soundness $$, where $$ is negligible in the size of the instance.

Proof: This theorem follows from Theorem 6.7.2; since s is $\frac{1}{\text{poly}(|x|)}$ the verifier can distinguish the case where $x \in L$ from the case where $x \notin L$ with probability of error bounded to be exponentially small with only polynomially many repetitions.

6.7.2 Extending the Measurement Protocol

We now present the extension of the measurement protocol (Protocol 6.4.1) to a QPIP$_0$. To do this, we use the structure of Protocol 6.7.3, but we replace the prover sending qubits to the verifier to perform the measurement (i.e. steps 2 and 3 of Protocol 6.7.3) with Protocol 6.4.1. Assume that in Protocol 6.7.3, n qubits are sent from the prover to the verifier. The QPIP protocol is as follows:

Protocol 6.7.5 Measurement Protocol QPIP This protocol is used to verify that an instance $x \in L$ for a language $L \in$ BQP.

1. The verifier performs step 1 of Protocol 6.7.3, which partially defines a basis choice $h \in \{0,1\}^n$. For all undefined h_i, the verifier sets $h_i = 0$.
2. The prover and verifier run Protocol 6.4.1 on basis choice h. The verifier accepts or rejects as specified in Protocol 6.4.1.
3. In the case of a Hadamard round of Protocol 6.4.1, the verifier performs step 4 of Protocol 6.7.3 using the measurement results obtained from Protocol 6.4.1.

We now prove the following theorem, which is the main result of this chapter (and was stated earlier as Theorem 1.0.2):

Theorem 6.7.6 Protocol 6.7.5 is a QPIP$_0$ for all languages in BQP with completeness negligibly close to 1 and soundness negligibly close to $\frac{3}{4}$.

We require the completeness and soundness guarantees of Protocol 6.4.1, copied below for reference. Both claims use notation from Section 6.4.3.

Claim 6.4.3 Completeness of Protocol 6.4.1 For all n qubit states $$ and for all basis choices $h \in \{0,1\}^n$, the prover P described in Protocol 6.4.2 is a perfect prover (P is accepted by the verifier in a test run for basis choice h with perfect probability). There exists a negligible function $$ such that in the Hadamard round for basis choice h, the verifier accepts P with probability 1 and $\| D^{C}_{P,h} - D_{,h} \|_{TV}$.

Claim 6.6.1 Soundness of Protocol 6.4.1 For a prover P in Protocol 6.4.1, let $1 - p_{h;H}$ be the probability that the verifier accepts P on basis choice h in the Hadamard round and $1 - p_{h;T}$ be the probability that the verifier accepts P in the test round. There exists a state, a prover P' and a negligible function μ such that for all h, $D^C_{P;h} \quad D_{P';h} \quad_{TV} \quad p_{h;H} + \sqrt{p_{h;T}} + \mu$ and $D_{P';h}$ is computationally indistinguishable from the distribution $D_{;h}$ which results from measuring $\quad$ in the basis determined by h.

Proof of Theorem 6.7.6 : We will require some notation. For all provers P, let $E^H_{P;h}$ be the event that the verifier accepts in a Hadamard round of Protocol 6.4.1 with basis choice h while interacting with P, let $E^T_{P;h}$ be the same event in a test round, and let $E_{P;h}$ be the event that the verifier accepts in step 3 of Protocol 6.7.5. Let v_h be the probability that the verifier chooses basis choice h in step 1 of Protocol 6.7.5. As a reminder, in step 3 of Protocol 6.7.5 (which is step 4 of Protocol 6.7.3), the verifier determines whether or not to accept by computing the product of relevant measurement results; the verifier's decision is a function of the basis choice h, the measurement result and the BQP instance, but we leave off the dependence on the BQP instance for convenience. For a distribution D over n bit strings and a basis choice h, let $p_h(D)$ be the probability that the verifier rejects an n bit string drawn from D for basis choice h in step 3 of Protocol 6.7.5.

Completeness Recall that in Protocol 6.7.3, an honest prover sends polynomially many copies of the ground state for the Hamiltonian corresponding to an instance $x \in L$ (where $L \in BQP$). Let the entire state sent by the prover be $\quad$, and assume it contains n qubits. To compute the completeness parameter of Protocol 6.7.5, we will consider the prover P for the state $\quad$, as described in Protocol 6.4.2, and upper bound the probability that the verifier rejects in Protocol 6.7.5. To do this, we need to upper bound the probability that the verifier rejects in step 2 of Protocol 6.7.5 (i.e. in Protocol 6.4.1) or the verifier rejects in step 3 of Protocol 6.7.5:

$$1 - c = \frac{1}{2} \sum_{h \in \{0,1\}^n} v_h (\Pr[E^T_{P;h}] + \Pr[\overline{E^H_{P;h}} \cup \overline{E_{P;h}}]) \tag{6.156}$$

$$\frac{1}{2} \sum_{h \in \{0,1\}^n} v_h (\Pr[E^H_{P;h}] + \Pr[\overline{E_{P;h}}]) \tag{6.157}$$

$$\frac{1}{2} + \frac{1}{2} \sum_{h \in \{0,1\}^n} v_h \Pr[E_{P;h}] \tag{6.158}$$

The last two expressions follow due to Claim 6.4.3: we know that for all basis choices $h \in \{0,1\}^n$, the prover P described in Protocol 6.4.2 is accepted by the verifier in a test round with perfect probability and is accepted in a Hadamard round with probability 1 for a negligible function $\quad$. $\Pr[E_{P;h}]$ is the probability that the verifier rejects the distribution $D^C_{P;h}$ for basis choice h in step 3 of Protocol 6.7.5:

$$\Pr[E_{P;h}] = p_h(D^C_{P;h}) \tag{6.159}$$

Recall from Section 6.4.3 that for a density matrix σ on n qubits and basis choice h, we let $D_{\sigma,h}$ be the distribution obtained by measuring σ in the basis corresponding to h. It follows by Lemma 3.1.1 and Claim 6.4.3 that

$$p_h(D^C_{P,h}) \quad p_h(D_{\sigma,h}) \quad D^C_{P,h} \quad D_{\sigma,h} \quad _{TV} \tag{6.160}$$

Due to the completeness parameter of Protocol 6.7.3 (see Section 6.7.1), there exists a negligible function $_C$ such that:

$$\sum_h v_h p_h(D_{\sigma,h}) \quad _C \tag{6.161}$$

We now use (6.159), (6.160) and (6.161) to wrap up the calculation of the completeness parameter c (continuing from (6.158)):

$$1 \quad c \quad \frac{1}{2} + \frac{1}{2} \sum_{h \in \{0,1\}^n} v_h p_h(D^C_{P,h}) \tag{6.162}$$

$$+ \frac{1}{2} \quad _C \tag{6.163}$$

Therefore, the completeness parameter c is negligibly close to 1.

Soundness To compute the soundness parameter, we will fix an arbitrary prover P^* and upper bound the probability that the verifier accepts in Protocol 6.7.5 for an instance $x \notin L$. To do so, we need to upper bound the probability that the verifier accepts in step 2 of Protocol 6.7.5 (i.e. in Protocol 6.4.1) and the verifier accepts in step 3 of Protocol 6.7.5. The intuition here is that as long as there exists a state σ such that for all h, $D^C_{P,h}$ is close (computationally) to $D_{\sigma,h}$, the soundness parameter should be close to the soundness parameter of Protocol 6.7.3, which is negligible. We will rely on Claim 6.6.1 (we also use the same notation used in Claim 6.6.1):

$$s = \sum_{h \in \{0,1\}^n} v_h \left(\frac{1}{2}\Pr[E^T_{P,h}] + \frac{1}{2}\Pr[E^H_{P,h} \setminus E_{P,h}]\right) \tag{6.164}$$

$$= \sum_{h \in \{0,1\}^n} v_h \left(\frac{1}{2}(1 \quad p_{h,T}) + \frac{1}{2}\Pr[E^H_{P,h}]\Pr[E_{P,h} | E^H_{P,h}]\right) \tag{6.165}$$

$$= \sum_{h \in \{0,1\}^n} v_h \left(\frac{1}{2}(1 \quad p_{h,T}) + \frac{1}{2}(1 \quad p_{h,H})(1 \quad p_h(D^C_{P,h}))\right) \tag{6.166}$$

where the last equality follows because $\Pr[E_{P,h} | E^H_{P,h}]$ is the probability that the verifier accepts a string drawn from the distribution $D^C_{P,h}$ for basis choice h in step 3 of Protocol 6.7.5.

Claim 6.6.1 guarantees the existence of a state, a prover P^0 and a negligible function such that for all h, $D^C_{P,h}$ $D_{P^0,h}$ $_{TV}$ $p_{h,H} + \sqrt{p_{h,T}} +$ and $D_{P^0,h}$ is computationally indistinguishable from $D_{,h}$. By Lemma 3.1.1 and Claim 6.6.1:

$$p_h(D_{P^0,h}) \quad p_h(D^C_{P,h}) \quad D^C_{P,h} \quad D_{P^0,h} \; _{TV} \quad p_{h,H} + \sqrt{p_{h,T}} + \tag{6.167}$$

We now return to the calculation of the soundness parameter of Protocol 6.7.5 in (6.166):

$$s \quad \sum_{h \in \{0,1\}^n} v_h\left(\frac{1}{2}(1 \quad p_{h,T}) + \frac{1}{2}(1 \quad p_{h,H})(1 \quad p_h(D_{P^0,h}) + p_{h,H} + \sqrt{p_{h,T}} +)\right) \tag{6.168}$$

$$\frac{1}{2} + \frac{1}{2}\sum_{h \in \{0,1\}^n} v_h(1 \quad p_{h,T} + (1 \quad p_{h,H})(p_{h,H} + \sqrt{p_{h,T}})) + \frac{1}{2}\sum_{h \in \{0,1\}^n} v_h(1 \quad p_h(D_{P^0,h}))$$

$$\frac{1}{2} + \frac{3}{4} + \frac{1}{2}\sum_{h \in \{0,1\}^n} v_h(1 \quad p_h(D_{P^0,h})) \tag{6.169}$$

Next, Claim 6.6.1 guarantees that for all h, $D_{P^0,h}$ and $D_{,h}$ are computationally indistinguishable. It follows that for all h:

$$p_h(D_{,h}) \quad p_h(D_{P^0,h}) \quad _h \tag{6.170}$$

To see this implication, assume there did exist an $h \in \{0,1\}^n$ such that the difference in (6.170) was non negligible. Then $D_{P^0,h}$ and $D_{,h}$ could be distinguished by computing whether or not the verifier would reject for basis choice h in step 3 of Protocol 6.7.5, which is step 4 of Protocol 6.7.3. This is because the computational indistinguishability of $D_{P^0,h}$ and $D_{,h}$ holds even if h is known; the indistinguishability is due to the hardcore bit and injective invariance properties of the extended trapdoor claw-free family (Definition 6.3.4).

Due to the soundness parameter of Protocol 6.7.3, we also know that there exists a negligible function $_S$ such that:

$$\sum_{h \in \{0,1\}^n} v_h(1 \quad p_h(D_{,h})) \quad _S \tag{6.171}$$

We return to calculating the soundness parameter of the QPIP (continuing from (6.169)):

$$s \quad \frac{1}{2} + \frac{3}{4} + \frac{1}{2}\sum_{h \in \{0,1\}^n} v_h(\; _h + 1 \quad p_h(D_{,h})) \tag{6.172}$$

$$\frac{1}{2} + \frac{3}{4} + \frac{1}{2}\max_{h \in \{0,1\}^n} \; _h + \frac{1}{2} \; _S \tag{6.173}$$

Therefore, the soundness parameter of Protocol 6.7.5 is negligibly close to $\frac{3}{4}$

6.8 Extended Trapdoor Claw-Free Family from LWE

6.8.1 Parameters

We will use the same parameters used in [11]. Let n be the security parameter. All other parameters are functions of n. Let $q \geq 2$ be a prime integer. Let $\ell(n), m(n), w(n) \geq 1$ be polynomially bounded functions of n and B_L, B_V, B_P be positive integers such that the following conditions hold:

1. $n = \Omega(\ell \log q)$ and $m = \Omega(n \log q)$,
2. $w = n\lceil \log q \rceil$,
3. $B_P = \frac{q}{2C_T \sqrt{mn \log q}}$, for C_T the universal constant in Theorem 3.2.3,
4. $2\sqrt{n} \cdot B_L < B_V < B_P$,
5. The ratios $\frac{B_P}{B_V}$ and $\frac{B_V}{B_L}$ are both super-polynomial in n.

(6.174)

Given a choice of parameters satisfying all conditions above, we describe the function family $\mathcal{F}_{LWE}$ (taken from [11]). Let $X = \mathbb{Z}_q^n$ and $Y = \mathbb{Z}_q^m$. The key space is $\mathcal{K}_{\mathcal{F}_{LWE}} = \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. For $b \in \{0,1\}$, $x \in X$ and key $k = (A, As + e)$, the density $f_{k,b}(x)$ is defined as

$$\forall y \in Y, \quad (f_{k,b}(x))(y) = D_{\mathbb{Z}_q^m, B_P}(y - Ax - b \cdot As), \tag{6.175}$$

where the definition of $D_{\mathbb{Z}_q^m, B_P}$ is given in (3.5). Note that $f_{k,b}$ is well-defined given k, as for our choice of parameters k uniquely specifies s. The following theorem was proven in [11]:

Theorem 22 [11] For any choice of parameters satisfying the conditions (6.174), the function family $\mathcal{F}_{LWE}$ is a noisy trapdoor claw-free family under the hardness assumption $LWE_{\ell, q, D_{\mathbb{Z}_q, B_L}}$.

We will prove the following theorem:

Theorem 6.8.1 For any choice of parameters satisfying the conditions (6.174), the function family $\mathcal{F}_{LWE}$ is an extended trapdoor claw-free family under the hardness assumption $LWE_{\ell, q, D_{\mathbb{Z}_q, B_L}}$.

We begin by providing a trapdoor injective family and then use this family to show that $\mathcal{F}_{LWE}$ is an extended trapdoor claw-free function family.

6.8.2 Trapdoor Injective Family from LWE

We now describe the trapdoor injective family $\mathcal{G}_{LWE}$. Let $X = \mathbb{Z}_q^n$ and $Y = \mathbb{Z}_q^m$. The key space is $\mathcal{K}_{\mathcal{G}_{LWE}} = \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. For $b \in \{0,1\}$, $x \in X$ and key $k = (A, u)$, the density $g_{k,b}(x)$

is defined as

$$\forall y \in Y: \quad (g_{k;b}(x))(y) = D_{Z_q^m;B_P}(y - Ax - bu); \tag{6.176}$$

where the definition of $D_{Z_q^m;B_P}$ is given in (3.5). The three properties required for a trapdoor injective family, as specified in Definition 6.3.2, are verified in the following subsections, providing a proof of the following theorem.

Theorem 6.8.2 For any choice of parameters satisfying the conditions (6.174), the function family G_{LWE} is a trapdoor injective family under the hardness assumption $LWE_{\ell;q;D_{Z_q;B_L}}$.

6.8.2.1 Efficient Function Generation

$GEN_{G_{LWE}}$ is defined as follows. First, the procedure samples a random $A \in Z_q^{m \times n}$, together with trapdoor information t_A. This is done using the procedure $GenTrap(1^n, 1^m, q)$ from Theorem 3.2.3. The trapdoor allows the evaluation of an inversion algorithm $Invert$ that, on input A, t_A and $b = As + e$ returns s and e as long as $\|e\| \leq \frac{1}{C_T}\sqrt{\frac{q}{n \log q}}$. Moreover, the distribution on matrices A returned by $GenTrap$ is negligibly close to the uniform distribution on $Z_q^{m \times n}$.

Next, the sampling procedure selects $u \in Z_q^m$ uniformly at random. By using the trapdoor t_A, the sampling procedures checks if there exists e such that $\|e\| \leq \frac{1}{C_T}\sqrt{\frac{q}{n \log q}}$ and $As + e = u$. If so, the sampling algorithm discards u and samples u again. Since u is discarded with negligible probability, the distribution over u is negligibly close to the uniform distribution. $GEN_{G_{LWE}}$ returns $k = (A, u)$ and $t_k = t_A$.

6.8.2.2 Trapdoor Injective Functions

It follows from (6.176) and the definition of the distribution $D_{Z_q^m;B_P}$ in (3.5) that for any key $k = (A, u) \in K_{G_{LWE}}$ and for all $x \in X$,

$$Supp(g_{k;0}(x)) = \{y = Ax + e_0 \mid \|e_0\| \leq B_P\sqrt{m}\}; \tag{6.177}$$

$$Supp(g_{k;1}(x)) = \{y = Ax + e_0 + u \mid \|e_0\| \leq B_P\sqrt{m}\}. \tag{6.178}$$

Since there do not exists s, e such that $\|e\| \leq \frac{1}{C_T}\sqrt{\frac{q}{n \log q}}$ and $u = As + e$, the intersection $Supp(g_{k;0}(x)) \cap Supp(g_{k;1}(x))$ is empty as long as

$$B_P \leq \frac{1}{2C_T}\sqrt{\frac{q}{mn \log q}}. \tag{6.179}$$

The procedure $INV_{G_{LWE}}$ takes as input the trapdoor t_A and $y \in Y$. It first runs the algorithm $Invert$ on input y. If $Invert$ outputs s_0, e_0 such that $y = As_0 + e_0$ and $\|e_0\| \leq B_P\sqrt{m}$, the procedure $INV_{F_{LWE}}$ outputs $(0, s_0)$. Otherwise, it runs the algorithm $Invert$ on input $y - u$ to obtain s_0, e_0 and outputs $(1, s_0)$ if $y - u = As_0 + e_0$. Using Theorem 3.2.3, this procedure returns the unique correct outcome provided $y = bu = As_0 + e_0$ for some e_0 such that $\|e_0\| \leq \frac{1}{C_T}\sqrt{\frac{q}{n \log q}}$. Due to (6.179), this condition is satisfied for all $y \in Supp(f_{k;b}(x))$.

6.8.2.3 Efficient Range Superposition

The procedures $\text{CHK}_{G_{LWE}}$ and $\text{SAMP}_{G_{LWE}}$ are the same as the procedures $\text{CHK}_{F_{LWE}}$ and $\text{SAMP}_{F_{LWE}}$.

6.8.3 Injective Invariance

We now show that F_{LWE} as given in (6.175) is injective invariant with respect to G_{LWE} (see Definition 6.3.3). To show this, we need to show that for all QPT attackers $\mathcal{A}$, the distributions produced by $\text{GEN}_{F_{LWE}}$ and $\text{GEN}_{G_{LWE}}$ are computationally indistinguishable. This is equivalent to proving the hardness of LWE (as defined in Definition 3.2.2) with a binary secret, as seen in the following lemma:

Lemma 6.8.3 Assume a choice of parameters satisfying the conditions (6.174). Assume the hardness assumption $\text{LWE}_{\ell,q,D_{\mathbb{Z},q,B_L}}$ holds. Then the distributions

$$D_0 = ((A, As + e) \leftarrow \text{GEN}_{F_{LWE}}(1^\lambda)) \tag{6.180}$$

and

$$D_1 = ((A, u) \leftarrow \text{GEN}_{G_{LWE}}(1^\lambda)) \tag{6.181}$$

are computationally indistinguishable.

The hardness of LWE with a binary secret is well studied, and the above lemma is implied by several results, starting with [27] and improved in [12]. To be precise, it is also immediately implied by Theorem B.5 of [4].

6.8.4 Extended Trapdoor Claw-Free Family

We have already shown that F_{LWE} is injective invariant. To show that F_{LWE} is an extended trapdoor claw-free family, we now prove the second condition (the hardcore bit condition) of Definition 6.3.4. First recall (from [11]) that $X = \mathbb{Z}_q^n$, $w = n\lceil \log q \rceil$ and $J : X \to \{0,1\}^w$ is the map such that $J(x)$ returns the binary representation of $x \in X$. The key point, which we prove in Lemma 6.8.5, is that the inner product appearing in the definition of $H^0_{k,d}$ (in condition 2 of Definition 6.3.4) is equal to the inner product $\hat{d} \cdot s$ (for $\hat{d} \in \{0,1\}^n$) if $d = J(\hat{d})$. We first show in Lemma 6.8.4 that producing an inner product $\hat{d} \cdot s$ is computationally difficult given $A, As + e$. Next, in Lemma 6.8.5, we use Lemma 6.8.4 to prove condition 2 of Definition 6.3.4.

Lemma 6.8.4 Assume a choice of parameters satisfying the conditions (6.174). Assume the hardness assumption $\text{LWE}_{\ell,q,D_{\mathbb{Z},q,B_L}}$ holds. For all $\hat{d} \in \{0,1\}^n \setminus \{0^n\}$, the distributions

$$D_0 = (A, As + e) \leftarrow \text{GEN}_{F_{LWE}}(1^\lambda),\ \hat{d} \cdot s \mod 2 \tag{6.182}$$

and

$$D_1 = (A; As + e) \quad GEN_{F_{LWE}}(1^n); r \quad ; \tag{6.183}$$

where $r \in_U \{0,1\}$, are computationally indistinguishable.

Proof: This proof is a simpler version of the proof of Lemma 27 in [11]. We first transition from D_0 to the following computationally indistinguishable distribution:

$$D^{(1)} = (\tilde{A}C + F; \tilde{B}Cs + e; \hat{d} \cdot s \mod 2) \tag{6.184}$$

where $\tilde{A} = BC + F$ lossy $(1^n; 1^m; 1^{\ell}; q; D_{Z_q;\tilde{B}_L})$ is sampled from a lossy sampler. The transition from D_0 to $D^{(1)}$ is the same as the transition from (47) to (50) in the proof of Lemma 27 in [11]: the first step is to replace A with a lossy matrix $\tilde{A}$ to obtain a computationally indistinguishable distribution and the second step is to remove the terms F from the lossy LWE sample $As + e$. Next, we apply Lemma 23 ([11]) to $D^{(1)}$ to replace $\hat{d} \cdot s \mod 2$ with a uniformly random bit r, resulting in the following statistically indistinguishable distribution:

$$D^{(2)} = (\tilde{A}C + F; \tilde{B}Cs + e; r) \tag{6.185}$$

Computational indistinguishability between $D^{(2)}$ and D_1 follows similarly to between $D^{(1)}$ and D_0.

We now show that the second condition of Definition 6.3.4 holds:

Lemma 6.8.5 Assume a choice of parameters satisfying the conditions (6.174). Assume the hardness assumption $LWE_{\ell;q;D_{Z_q;\tilde{B}_L}}$ holds. Let $s \in \{0,1\}^n$ and for $d \in \{0,1\}^w$ let [2]

$$H^0_{s;d} = \{d \cdot (J(x) \oplus J(x \oplus (1^n)^b s)) \mid x \in X\}; \tag{6.186}$$

Then for all $\hat{d} \in \{0,1\}^n$ and for any quantum polynomial-time procedure

$$A : Z_q^{m \times n} \times Z_q^m \to \{0,1\}$$

there exists a negligible function $(\cdot)$ such that

$$\Pr_{(A;As+e) \leftarrow GEN_{F_{LWE}}(1^n)}[A(A; As + e) \in H^0_{s;J(\hat{d})}] \leq \frac{1}{2} + (n). \tag{6.187}$$

Proof: This proof is very similar to the proof of Lemma 28 in [11]. The proof is by contradiction. Assume that there exists $\hat{d} \in \{0,1\}^n$ and a quantum polynomial-time procedure A such that the left-hand side of (6.187) is at least some non-negligible function (n). We derive a contradiction by showing that for $\hat{d}$, the two distributions D_0 and D_1 in Lemma 6.8.4 are computationally distinguishable, giving a contradiction.

Let $(A; As + e) \leftarrow GEN_{F_{LWE}}(1^n)$. To link A to the distributions in Lemma 6.8.4 we relate the inner product condition in (6.186) to the inner product $\hat{d} \cdot s$ appearing in (6.182). This is based on the following claim.

[2] We write the set as $H^0_{s;d}$ instead of $H^0_{k;d}$ to emphasize the dependence on s.

Claim 6.8.6 For all $b \in \{0,1\}, x \in X, \hat{d} \in \{0,1\}^n$ and $s \in \{0,1\}^n$ the following equality holds:

$$J(\hat{d}) \oplus (J(x) \oplus J(x \oplus (-1)^b s)) = \hat{d} \cdot s. \tag{6.188}$$

Proof: We do the proof in case $n = 1$ and $w = \lceil \log q \rceil$, as the case of general n follows by linearity. In this case s is a single bit. If $s = 0$ then both sides of (6.188) evaluate to zero, so the equality holds trivially. If $s = 1$, then the least significant bit of $J(x) \oplus J(x \oplus (-1)^b s)$ is s and the least significant bit of $J(\hat{d}) \cdot \hat{d}$. Since the remaining $w - 1$ bits of $J(\hat{d})$ are 0, the claim follows.

To conclude we construct a distinguisher A' between the distributions D_0 and D_1 in Lemma 6.8.4. Consider two possible distinguishers A'_u for $u \in \{0,1\}$. Given a sample $((A, As + e), t)$, A'_u computes $c = A(A, As + e)$ and returns 0 if $c = t + u$, and 1 otherwise. The sum of the advantages of A'_0 and A'_1 is:

$$\sum_{u \in \{0,1\}} \Pr_{((A,As+e),\hat{d}\cdot s) \leftarrow D_0} [A'_u((A, As + e), \hat{d} \cdot s) = 0] - \Pr_{((A,As+e),r) \leftarrow D_1} [A'_u((A, As + e), r) = 0]$$

$$= \sum_{u \in \{0,1\}} \Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) = \hat{d} \cdot s + u] - \Pr_{((A,As+e),r) \leftarrow D_1} [A(A, As + e) = r + u]$$

$$= \sum_{u \in \{0,1\}} \Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) = \hat{d} \cdot s + u] - \frac{1}{2} \tag{6.189}$$

$$\Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) = \hat{d} \cdot s] - \Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) = \hat{d} \cdot s + 1]$$

$$2 \cdot \Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) = \hat{d} \cdot s] - \frac{1}{2} \tag{6.190}$$

By Claim 6.8.6, we can replace $\hat{d} \cdot s$ with $J(\hat{d}) \oplus (J(x) \oplus J(x \oplus (-1)^b s))$ to obtain:

$$= 2 \cdot \Pr_{(A,As+e) \leftarrow \mathrm{GEN}_{F_{LWE}}(1^\lambda)} [A(A, As + e) \in H^0_{s, J(\hat{d})}] - \frac{1}{2} \tag{6.191}$$

$$2 \cdot (\quad) \tag{6.192}$$

Therefore, at least one of A'_0 or A'_1 must successfully distinguish between D_0 and D_1 with advantage at least , a contradiction with the statement of Lemma 6.8.4.

Bibliography

[1] Dorit Aharonov, Michael Ben-Or, and Elad Eban. \Interactive Proofs For Quantum Computations". In: Arxiv preprint arXiv:0810.5375 (2008).

[2] Dorit Aharonov and Umesh Vazirani. \Is Quantum Mechanics Falsi able? A computational perspective on the foundations of Quantum Mechanics". In: Arxiv preprint arXiv:1206.3686 (2012).

[3] Dorit Aharonov et al. \Interactive Proofs for Quantum Computations". In: Arxiv preprint 1704.04487 (2017).

[4] Joel Alwen et al. Learning with Rounding, Revisited: New Reduction, Properties and Applications. Cryptology ePrint Archive, Report 2013/098. https://eprint.iacr.org/2013/098 . 2013.

[5] Andris Ambainis et al. \Private Quantum Channels". In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 547{. isbn: 0-7695-0850-2. url : http://dl.acm.org/citation.cfm?id=795666.796592 .

[6] Wojciech Banaszczyk. \New bounds in some transference theorems in the geometry of numbers". In: Mathematische Annalen 296.1 (1993), pp. 625{635.

[7] Jacob D. Biamonte and Peter J. Love. \Realizable Hamiltonians for universal adiabatic quantum computers". In: Phys. Rev. A 78 (1 July 2008), p. 012352. doi : 10.1103/PhysRevA.78.012352. url : https://link.aps.org/doi/10.1103/PhysRevA.78.012352

[8] Zvika Brakerski. Fully Homomorphic Encryption. Simons Institute. https://www.youtube.com/watch?v=O8IvJAIvGJo. 2015.

[9] Zvika Brakerski and Vinod Vaikuntanathan. E cient Fully Homomorphic Encryption from (Standard) LWE. Siam Journal on Computing, Vol 43, No 2, pp. 831-871. http://epubs.siam.org/doi/pdf/10.1137/120868669 . 2014.

[10] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. Cryptology ePrint Archive, Report 2013/541. http://eprint.iacr.org/2013/541 . 2013.

[11] Zvika Brakerski et al. \Certi able Randomness from a Single Quantum Device". In: Arxiv preprint 1804.00640 (2018).

[12] Zvika Brakerski et al. \Classical hardness of learning with errors". In: Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013 2013, pp. 575{584.

[13] Anne Broadbent, Joseph F. Fitzsimons, and Elham Kashe . \Universal blind quantum computation". In: Arxiv preprint arXiv:0807.4154 (2008).

[14] Anne Broadbent and Stacey Je eryQuantum homomorphic encryption for circuits of low T-gate complexity Cryptology ePrint Archive, Report 2015/551.http://eprint.iacr.org/2015/551 . 2015.

[15] Andrew M. Childs. \Secure Assisted Quantum Computation". In: Quantum Info. Comput. 5.6 (Sept. 2005), pp. 456{466.

[16] Kai-Min Chung, Yael Kalai, and Salil Vadhan.Improved Delegation of Computation using Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2010/241. https://eprint.iacr.org/2010/241 . 2010.

[17] John F. Clauser et al. \Proposed Experiment to Test Local Hidden-Variable Theories". In: Phys. Rev. Lett. 23 (15 1969), pp. 880{884.doi : 10.1103/PhysRevLett.23.880 . url : https://link.aps.org/doi/10.1103/PhysRevLett.23.880 .

[18] Alexandru Cojocaru et al.Delegated Pseudo-Secret Random Qubit Generation2018.

[19] Christoph Dankert et al. \Exact and approximate unitary 2-designs and their application to delity estimation". In: Phys. Rev. A 80 (1 July 2009), p. 012304.doi : 10.1103/PhysRevA.80.012304. url : https://link.aps.org/doi/10.1103/PhysRevA.80.012304

[20] Yfke Dulek, Christian Scha ner, and Florian Speelman. \Quantum Homomorphic Encryption for Polynomial-Sized Circuits". In: Advances in Cryptology { CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3{32. isbn: 978-3-662-53015-3.doi : 10.1007/978-3-662-53015-3_1 . url : http://dx.doi.org/10.1007/978-3-662-53015-3_1 .

[21] Joseph F. Fitzsimons and Elham Kashe . \Unconditionally veri able blind quantum computation". In: Phys. Rev. A96 (1 July 2017), p. 012303.doi : 10.1103/PhysRevA.96.012303. url : https://link.aps.org/doi/10.1103/PhysRevA.96.012303 .

[22] Craig Gentry. \A fully homomorphic encryption scheme".crypto.stanford.edu/craig . PhD thesis. Stanford University, 2009.

[23] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan.Trapdoors for Hard Lattices and New Cryptographic Constructions.Cryptology ePrint Archive, Report 2007/432. http://eprint.iacr.org/2007/432 . 2007.

[24] Craig Gentry, Amit Sahai, and Brent Waters.Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based.Cryptology ePrint Archive, Report 2013/340.http://eprint.iacr.org/2013/340 . 2013.

[25] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks". In: *SIAM J. Comput.* 17.2 (Apr. 1988), pp. 281–308. DOI: 10.1137/0217017. URL: http://dx.doi.org/10.1137/0217017.

[26] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Paradoxical Solution to The Signature Problem". In: (1985), pp. 467–467.

[27] Shafi Goldwasser et al. "Robustness of the Learning with Errors Assumption". In: *ICS* (2010), pp. 230–240.

[28] Daniel Gottesman. As referenced in [http://www.scottaaronson.com/blog/?p=284; accessed 13-Apr-2017]. 2004.

[29] Daniel Gottesman and Isaac L. Chuang. "Quantum Teleportation is a Universal Computational Primitive". In: *Arxiv preprint arXiv:quant-ph/9908010* (1999).

[30] Lov Grover and Terry Rudolph. *Creating superpositions that correspond to efficiently integrable probability distributions*. 2002.

[31] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "Pseudo-random Generation from One-way Functions". In: *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*. STOC '89. New York, NY, USA: ACM, 1989, pp. 12–24.

[32] A.Y. Kitaev, A. Shen, and M.N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, 2002.

[33] Ching-Yi Lai and Kai-Min Chung. *On statistically-secure quantum homomorphic encryption*. 2017.

[34] Atul Mantri et al. "Flow Ambiguity: A Path Towards Classically Driven Blind Quantum Computation". In: *Phys. Rev. X* 7 (3 July 2017), p. 031004. DOI: 10.1103/PhysRevX.7.031004. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.031004.

[35] Daniele Micciancio and Chris Peikert. *Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller*. Cryptology ePrint Archive, Report 2011/501. http://eprint.iacr.org/2011/501. 2011.

[36] Tomoyuki Morimae and Joseph F. Fitzsimons. "Post hoc verification with a single prover". In: *Arxiv preprint arXiv:1603.06046* (2016).

[37] Tomoyuki Morimae, Daniel Nagaj, and Norbert Schuch. *Quantum proofs can be verified using only single qubit measurements*. 2015.

[38] Michael Newman and Yaoyun Shi. *Limitations on Transversal Computation through Quantum Homomorphic Encryption*. 2017.

[39] Yingkai Ouyang, Si-Hui Tan, and Joseph F. Fitzsimons. *Quantum homomorphic encryption from quantum codes*. 2015.

[40] Chris Peikert. *A Decade of Lattice Cryptography*. Cryptology ePrint Archive, Report 2015/939. `http://eprint.iacr.org/2015/939`. 2015.

[41] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem: extended abstract". In: *STOC*. 2009, pp. 333–342.

[42] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. "Pseudorandomness of ring-LWE for any ring and modulus". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM, 2017, pp. 461–473. ISBN: 978-1-4503-4528-6. DOI: `10.1145/3055399.3055489`. URL: `http://doi.acm.org/10.1145/3055399.3055489`.

[43] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*. STOC '05. New York, NY, USA: ACM, 2005, pp. 84–93.

[44] B. Reichardt, F. Unger, and U. Vazirani. "A classical leash for a quantum system". In: *Arxiv preprint arXiv:1209.0448* (2012).

[45] A. Shamir. "IP= PSPACE". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 869–877.

[46] Yaoyun Shi. "Both Toffoli and controlled-NOT Need Little Help to Do Universal Quantum Computing". In: *Quantum Info. Comput.* 3.1 (Jan. 2003), pp. 84–92. ISSN: 1533-7146. URL: `http://dl.acm.org/citation.cfm?id=2011508.2011515`.

[47] Si-Hui Tan et al. *A quantum approach to homomorphic encryption. Scientific reports*. Scientific Reports, 6. 2016.

[48] Umesh Vazirani. *CS 294 Lecture 1 Fall 2016*. [`https://people.eecs.berkeley.edu/~vazirani/f16quantum/lec1.pdf`; accessed 5-May-2018]. 2016.

[49] Daniel Wichs. *Homomorphic Commitments and Signatures*. Simons Institute. `https://www.youtube.com/watch?v=1cQP1QYVjAI`. 2015.

[50] Wikipedia. *Quantum teleportation*. [`https://en.wikipedia.org/wiki/Quantum_teleportation`; accessed 5-May-2018]. 2018.

[51] Wikipedia. *Trace Distance — Wikipedia, The Free Encyclopedia*. [`https://en.wikipedia.org/wiki/Trace_distance`; accessed 7-Jan-2018]. 2018.