Quantum Shell Games: How to Classically Delegate the Preparation of Authenticated Quantum States

MSc Thesis *(Afstudeerscriptie)*

written by

John Hunter McKnight
(born April 12, 1991 in Columbus, Mississippi, United States of America)

under the supervision of **dr. Christian Schaffner** and **dr. Christian Majenz**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
September 12, 2019	prof. dr. Ronald de Wolf (chair)
	prof. dr. Serge Fehr
	dr. Christian Majenz
	dr. Maris Ozols
	dr. Christian Schaffner

Institute for Logic, Language and Computation

Abstract

We propose novel protocols for verifiable, classically instructed remote state preparation. Our "Shell Game" protocols require constantly many rounds of communication to prepare an arbitrary number of qubits, and the prepared qubits can optionally be authenticated using a trap code. In the Shell Game, a classical client and quantum server use a new cryptographic resource called a "magic box" to perform secret CNOT and SWAP operations that encode, permute, and encrypt a quantum state held by the server. The keys to this encoding/encryption are private to the client. The client tests the server's honesty by asking the server to measure some of the prepared state and evaluating the consistency of the outcomes against the honest state. The remaining, unmeasured state is the quantum output of the protocol.

We give a practical construction of magic boxes using Mahadev's secret-CNOT gadget [Mah18a] in the quantum honest-but-curious setting. We prove the Shell Game protocols secure against adversaries in this limited setting. We include a case study on the security of magic boxes against adversaries as a first step toward proving security of the Shell Game against active adversaries.

Acknowledgments

I am tremendously grateful to my official supervisors, Christian Schaffner and Christian Majenz, as well as my unofficial supervisors, Alex Grilo and Yfke Dulek. This thesis would not have been possible without their mentorship, patience, encouragement, and willingness to video conference. I only learned as much about quantum cryptography as I did by having all the hard stuff explained four different ways. For lack of any one of them, this thesis could not have been completed.

I would also like to thank the other members of my committee, Ronald de Wolf, Serge Fehr, and Maris Ozols, for taking the time to read and evaluate this thesis. In particular, I must thank Ronald de Wolf, who, along with Christian Schaffner, introduced me to the world of quantum information with their LoLaCo lectures. Who knows what the topic of this thesis might have been without their influence.

Last but not least, I thank Early and Wanda McKnight, Mark Smith, Nederlandse Spoorwegen, the public libraries of Coffee and Knox Counties, and several extended-stay hotels in two countries for providing me with space to think and write over the last few months.

Contents

Chapter 1

Introduction

Consider the following scenario. Alice needs to prepare a particular quantum state, but she does not have direct access to a quantum device. What she does have is a (non-quantum) telephone and Bob, a friend with his own private quantum computer. Alice calls Bob to describe the quantum state she needs, and Bob prepares it for her. Alice and Bob have just performed a task called (classically instructed) remote (quantum) state preparation. In protocols for remote state preparation, we typically refer to Alice as the classical "client" and to Bob as the quantum "server".

Now imagine that, although Bob still offers to prepare the state on Alice's behalf, Alice and Bob are not friends, and Alice cannot fully trust Bob. Alice wants to conceal as much information as possible about the prepared state from Bob, and she wants some assurance that Bob has actually followed her instructions honestly. What Alice needs now is a *secure*, *verifiable* protocol for remote state preparation.

At face value, it seems unlikely that any protocol could enable a classical client to instruct a much more computationally powerful quantum server to prepare a quantum state without the server learning everything about the prepared state. Indeed, the existence of remote state preparation protocols that are both information theoretically secure and leak almost no information about the prepared state have been shown to be implausible [ACGK17].[1] However, astonishingly, computationally blind protocols for remote preparation of single random qubits are known [DK16][CCKW18]. More recently, two protocols for secure, authenticated remote state preparation have been independently proposed [CCKW19][GV19]. These protocols are based on noisy/lossy injective trapdoor claw-free function families[2] (or similar 2-regular collision-resistant

[1] Specifically, information theoretically secure blind delegated quantum computation with a strictly classical client implies inclusions of computational complexity classes that are considered implausible by many computer scientists; information theoretically secure remote state preparation with a classical client implies information theoretically secure blind delegated quantum computation with a classical client.

[2] Informally, a trapdoor claw-free function pair (f_0, f_1) is a pair of injective functions with

function families),[3] which can be constructed based on the computational assumption that the learning with errors problem with superpolynomial noise is hard [Reg05][PW08]. In this thesis, we present a third protocol also based on trapdoor claw-free functions. Our protocol is distinguished from others in that the number of rounds of communication required between the client and server is independent of the number of qubits prepared.

1.1 Background: Delegated Quantum Computation and the Role of Remote State Preparation

Regardless of how optimistic their predictions for the near-term development of quantum computers might be, few experts expect average consumers to have quantum personal computers on their desks any time soon. Instead, most users will probably interact with quantum computers by delegation: A few entities will invest in developing and maintaining large quantum servers, and clients will buy server time on the occasions when they need to run quantum algorithms. This model is justified for two reasons. First, even current small-scale quantum computers are expensive and physically quite large; there is no reason to suspect near-term quantum computers will be small and affordable enough to find one in every home. Secondly, although quantum algorithms for certain interesting tasks, such as factoring large primes and simulating quantum systems, offer an exponential speedup over the best known classical algorithms, the quantum advantage is known to be much more modest for most tasks. Additionally, consumers are already quite accustomed to delegation in the realm of classical computing, buying server time on classical supercomputers or "clouds" of graphics processing units to train deep neural models on large data sets. Since 2016, IBM Q has allowed users to test limited quantum algorithms on three of their small (5- to 16-qubits, as of 2019) quantum processors, so one might say that the era of delegated quantum computation has already begun.

Given that most of us will not be performing our own quantum computations but will instead delegate them to (possibly untrusted or insecure) outside entities, cryptographic protocols are necessary to protect our delegated computations from eavesdropping and tampering. Protocols for verifiable, information theoretically blind[4] delegated quantum computation (BQC) have been known for over a decade, although existing protocols require clients to have at least some quantum capability [Chi05][BFK09] [GMMR13][MPDF13][HM15]

equal domains and equal images. These functions are easy to compute but difficult to invert without some extra information called a "trapdoor". Additionally, it is difficult to find a pair of elements (x_0, x_1) (called a "claw") such that $f_0(x_0) = f_1(x_1)$.

[3] Informally, a 2-regular collision-resistant function f maps exactly two elements of its domain to each element of its image, and finding a pair of elements with the same image under f is difficult. As with trapdoor claw-free functions, computing the function is easy, but inverting it is hard without a trapdoor.

[4] "Blind" here means private but for leaking an upper bound on the size of the computation.

[ABOEM17] [FK17]. (The minimal quantum capability required of clients is the ability to prepare and transmit single qubits or the ability to receive and measure single qubits.) A verifiable and semantically secure scheme for the related task of quantum fully homomorphic encryption (FHE) is also known; this protocol also requires some "quantumness" on the part of the client whenever the initial and final states of the delegated computation are not both classical [ADSS17]. A great many other protocols exist for delegated quantum computation, although none are simultaneously blind, verifiable, suitable for evaluating arbitrary quantum circuits (i.e., universal), and strictly classical on the client's side [AS06][MDK10][DKL12][MF12][BJ15][Lia15][Mah18a][Bra18][Mah18b].

Until recently, the question of how much the necessary quantumness of the client in a secure, verifiable protocol for delegated quantum computation can be theoretically reduced has been an open question. In particular, cryptographers have hoped to prove secure verifiable delegation possible for strictly classical clients. This is a question of practical importance for the future of quantum computing. If at least some quantumness is necessary for secure, verifiable delegation, then quantum cloud computing depends not only on the continuing development of quantum computers, but also on the development of large-scale and reliable quantum internet infrastructure.

One way to reduce the quantumness required for delegated quantum computation is to examine the quantum requirements placed on clients by existing protocols and to find ways to offload these requirements to the server. In particular, Fitzsimons and Kashefi give a BQC protocol that only requires the client to be able to prepare a small, fixed number of single-qubit quantum states and transmit them to the server [FK17]. Therefore, BQC with a fully classical client reduces to classically instructed remote quantum state preparation—in particular, the ability of a classical client to delegate the preparation of the qubits required for Fitzsimons' and Kashefi's BQC protocol to the quantum server. The existence of the computationally blind and verifiable protocols for remote state preparation presented below thus guarantees that universal and verifiable BQC is indeed possible with strictly classical clients, contingent on the computational assumptions of the remote state preparation protocols.

1.2 Current Approaches to Remote State Preparation

1.2.1 QFactory

The QFactory family of protocols was introduced in [CCKW19].[5] The basic QFactory protocol produces a single *random* qubit from the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ (known in the quantum cryptography literature as the BB84 states). The blindness guarantee given for the basic protocol is that, under computational assump-

[5] The authors previously published a similar but distinct "delegated pseudo-secret random qubit generator" also called QFactory in [CCKW18], although this protocol was not verifiable, and its security was not proven against fully malicious adversaries.

tions (in particular, the hardness of learning with errors), a quantum server has negligble advantage in guessing the basis of the output qubit. The basic protocol can be extended to produce a single random qubit from the set $|+_\theta\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$ for $\theta \in \{0, \frac{\pi}{4}, \ldots, \frac{7\pi}{4}\}$ (precisely those needed for the Fitzsimons and Kashefi protocol); this is accomplished by running the basic protocol twice (once in a slightly rotated basis) and "merging" the two output qubits using a particular gadget. This 8-state QFactory also satisfies basis blindness.

Either version of QFactory can be made verifiable using a protocol the authors call "blind self-testing", named by analogy to self-testing protocols for entanglement. Essentially, the client remotely prepares a large number of random qubits using QFactory and then asks the server to measure a random subset of those qubits, measuring each qubit in a basis chosen uniformly at random (from set of possible preparation bases). The client aborts if the reported measurement outcomes are too unlikely with respect to the measurement statistics of the honest state. Dishonestly producing a passing transcript of measurement outcomes is presumed to be difficult due to the server's basis blindness for each qubit. An honest server passes blind self-testing with all but negligible probability. The authors prove that, in a restricted adversarial context, an adversary that passes blind self-testing with high probability must hold a state close (up to an efficient isometry) to the honest state, possibly including some side information that does not depend on the actual bases of the prepared qubits. The authors conjecture, but do not prove, that a similar guarantee holds for blind self-testing against general adversaries. The communication complexity of the QFactory protocol with verification scales with the total number of qubits produced before testing, although it may be possible to reduce this complexity if the total number of qubits to prepare is known in advance.

For interested readers, a simplified overview of the basic QFactory protocol follows. This protocol is based on a family $\mathcal{G} = \{g_k : \{0,1\}^{n-1} \to \{0,1\}^m\}_k$ of injective trapdoor functions with the homomorphic property that for each k and all $z, z' \in \{0,1\}^{n-1}$

$$g_k(z \oplus z') = g_k(z) \oplus g_k(z')$$

together with a predicate $h : \{0,1\}^{n-1} \to \{0,1\}$, hardcore with respect to any g_k,[6] with the homomorphic property

$$h(z \oplus z') = h(z) \oplus h(z').$$

From $\mathcal{G}$, Cojocaru et al. construct a 2-regular, collision resistant trapdoor function family $\mathcal{F} = \{f_k : \{0,1\}^n \to \{0,1\}^m\}_k$ by choosing a random $z^\star \in \{0,1\}^{n-1}/\{0^{n-1}\}$ and taking

$$f_k(z\|c) = \begin{cases} g_k(z), & \text{if } c = 0, \\ g_k(z) \oplus y^\star, & \text{if } c = 1 \end{cases}$$

[6]Informally, this means that for a random z, $h(z)$ is hard to compute on average given $g_k(z)$ but easy given z itself.

for each k, $z \in \{0,1\}^{n-1}$, and $c \in \{0,1\}$, with $y^\star := g_k(z^\star)$. Note that, by the homomorphic property of g_k,

$$f_k(z\|1) = g_k(z) \oplus g_k(z^\star) = g_k(z \oplus z^\star),$$

so every collision in f_k is of the form $(z\|0, (z \oplus z^\star)\|1)$.

The client randomly samples a key and trapdoor (k, t_k) as well as $z^\star$ at the beginning of the protocol. The client sends $(k, y^\star)$ to the server in order to classically instruct the preparation of the state

$$\frac{1}{\sqrt{2^n}} \sum_{z\|c \in \{0,1\}^n} |h(z)\rangle_A |z\|c\rangle_B |f_k(z\|c)\rangle_C .$$

The server measures the C register in the computational basis, observing an outcome y before discarding the measured qubits. This updates the state to

$$\frac{1}{\sqrt{2}} \big(|h(z_0)\rangle_A |z_0\|0\rangle_B + |h(z_1)\rangle_A |z_1\|1\rangle_B \big) \tag{1.1}$$

for $f_k(z_0, 0) = f_k(z_1, 1) = y$. The server then measures the B register in the Hadamard basis,[7] observing an outcome d. This updates the state to

$$H^{B_1} X^{B_2} |0\rangle ,$$

for bits B_1 and B_2 which can be computed from $z_0\|0$, $z_1\|1$, and d. In particular,

$$B_1 = h(z_0) \oplus h(z_1) = h(z_0 \oplus z_1) = h(z_0 \oplus (z_0 \oplus z^\star)) = h(z^\star).$$

The server forwards the measurement outcomes y and d to the client, who can then compute B_2 using their trapdoor information.[8] The one-way property of g_k and f_k prevents the server from efficiently inverting y and $y^\star$ to compute B_1 and B_2. The collision-resistance of f_k prevents the server from choosing their own collision and counterfeiting the state from 1.1. The hardcore property of h with respect to g_k ensures that $y^\star$ does not give the server advantage in guessing B_1.

1.2.2 Buffered Remote State Preparation

The buffered remote state preparation protocol (BRSP) was introduced by Gheorgiu and Vidick in [GV19]. The protocol has its theoretical roots in an earlier application of trapdoor claw-free functions as a "cryptographic test of quantumness" for untrusted devices [BCM+18], as well as earlier work using trapdoor claw-free functions for quantum FHE by Mahadev [Mah18a][Mah18b] and

[7] In the "rotated" QFactory employed as a subroutine of the 8-states protocol, these measurements are performed in the $|\pm\frac{\pi}{2}\rangle$ basis instead.

[8] A closed formula for B_2 is complicated and unintuitive, but essentially if $h(z^\star) = 0$, then $B_2 = h(z_0)$, and if $h(z^\star) = 1$, then $B_2 = d \cdot (z_0\|0 \oplus z_1\|1)$. Observe that the client actually knows the basis of the qubit prepared by the honest protocol as soon as $z^\star$ is sampled. When the basis is the computational basis, the server knows precisely which basis state is prepared as well.

optimality results for $2 \mapsto 1$ quantum random-access codes [ALMO08]. The protocol consists of repeatedly instructing a server to prepare a particular kind of quantum state and testing the server's compliance by performing a random test (chosen from a small, fixed set of tests). If the server passes a high proportion of the tests, the client instructs the server to distill a single random qubit from either the set $\{|b\rangle\}_{b\in\{0,1\}}$ or the set $\{|+_\theta\rangle\}_{\theta\in\{0,\frac{\pi}{4},\ldots,\frac{7\pi}{4}\}}$ from the quantum state instead of testing it.[9] Importantly, the server does not know how many test rounds will occur, and the distillation process looks the same as a test from the server's perspective until the final step. The tests are constructed such that a server that can pass each test with high probability must have held a state close to the honest one, without side information that confers advantage in guessing b or θ (under computational assumptions). The protocol can be made ϵ-indistinguishable (up to isometry) from an ideal functionality for remote qubit preparation[10] using $O(1/\epsilon^3)$ rounds of communication. BRSP is able to make strong blindness and verification claims about its output; however, this security comes at the cost of high communication complexity per qubit prepared.

Again, a simplified overview of the underlying theory of the BRSP protocol follows for interested readers. The test of quantumness from [BCM+18] relies on a family $\mathcal{F} = \{f_{k,b} : \{0,1\}^n \to \{0,1\}^m\}_k$ of injective trapdoor functions such that each pair $(f_{k,0}, f_{k,1})$ is claw-free and $f_{k,0}$ and $f_{k,1}$ have identical images. The client classically instructs the server to prepare the state

$$\frac{1}{\sqrt{2}}(|0\rangle_A |x_0\rangle_B + |1\rangle_A |x_1\rangle_B) \tag{1.2}$$

such that (x_0, x_1) is a random claw of a random function pair (i.e., $f_{k,0}(x_0) = f_{k,1}(x_1) = y$ for some random k and y). By computational assumptions, this claw can be computed efficiently by the client (using the trapdoor) but not by the server. The server is then asked to perform one of two tests, which consist of measuring the state in some basis and sending the outcomes to the client. In the "preimage test", the server measures the entire state in the computational basis and returns a pair $(b, x_b) \in \{0,1\} \times \{0,1\}^n$ to the server; the server passes only if $f_b(x_b) = y$. In the "equation test", the server measures the entire state in the Hadamard basis and returns a pair $(b, d) \in \{0,1\} \times \{0,1\}^n$; the server passes only if $b = d \cdot (x_0 \oplus x_1)$. The authors of [BCM+18] demonstrate that a server can pass these tests with high probability only by preparing a state close to the state 1.2. Each round of BRSP also begins by preparing a state with this form, and variations of both of aforementioned tests are included in the test rounds of BRSP.

When $\{0,1\}^n$ is reinterpreted as $\mathbb{Z}_8^{n/3}$, the server can measure the B register of the state 1.2 in the $\mathbb{Z}_8$ Fourier basis with outcome $d \in \mathbb{Z}_8^{n/3}$ to update the

[9] The client can choose the set, but the qubit prepared from that set is random.

[10] The ideal functionality is a black box that assigns a random qubit (from the set of possible qubits) to the server and a classical description of that qubit to the client whenever the server is honest and aborts otherwise.

state to

$$\frac{1}{\sqrt{2}} e^{\frac{2i\pi}{8} d \cdot x_0} \left(|0\rangle_A + e^{\frac{2i\pi}{8} d \cdot (x_0 + x_1)} |1\rangle_A \right), \tag{1.3}$$

where addition and inner product are taken modulo 8. This state is equivalent to a state from $\{|+_\theta\rangle\}_{\theta \in \{0, \frac{\pi}{4}, \ldots, \frac{7\pi}{4}\}}$ (with $\theta = \frac{2i\pi}{8} d \cdot (x_0 + x_1)$) up to a global phase. During the final round of BRSP, this state is the output of the protocol.[11] Gheorgiu and Vidick also observe that when $\theta \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$ this state can also be interpreted as an optimal $2 \mapsto 1$ quantum random-access code for some pair of bits (b_1, b_2). Thus BRSP includes an additional kind of test distilling the state 1.3 from 1.2, treating it as a quantum random-access code (when valid), and requesting one of the encoded bits uniformly at random. The server passes this test with near-optimal probability only by producing a state close to 1.3 (up to isometry).

1.3 Our Contributions

We present a new protocol for quantum remote state preparation which we call the Shell Game protocol. This protocol relies on a new primitive called a Magic SWAP Box which acts as a trusted third party between the client and server. The magic box takes as input two qubits from the server and a secret control bit s from the client. The qubits are swapped if and only if $s = 1$. The qubits are returned to the server encrypted with a quantum one-time pad, and the client receives the encryption keys. The protocol proceeds by instructing the server to prepare a large quantum state consisting of many $|0\rangle$ and $|+\rangle$ qubits and using the magic box many times to privately permute and encrypt this state. The client then attempts to verify the preparation by asking the server to measure some of the prepared qubits in random bases, aborting if the outcomes are physically inconsistent with the honest state. The remaining state after testing consists of a number of qubits (chosen by the client at the start of the protocol as a security parameter) in random BB84 states. A variation of the protocol can be used to encode the output qubits in the trap code authentication scheme from [BGS13]. The number of rounds of communication required by either variation of the protocol is constant and independent of the number of qubits produced; one round is used during the "commitment phase" to permute the initial state, and one round is used in the "test phase" to test the committed state. We give a construction of computationally secure magic boxes in the quantum honest-but-curious setting from trapdoor claw-free function families. We give a full analysis of the security of the Shell Game in the quantum honest-but-curious setting, demonstrating that such adversaries cannot learn the permutation or quantum one-time pad applied to the prepared state. Ideally, we would like to

[11] To prepare a $|0\rangle$ or $|1\rangle$ instead, the client uses a family of injective trapdoor function pairs $\mathcal{G}$ instead of $\mathcal{F}$ to instruct the preparation of the output state. Function pairs from $\mathcal{G}$ have disjoint (rather than identical) images. The server should not be able to distinguish pairs sampled from $\mathcal{F}$ from pairs sampled from $\mathcal{G}$.

prove security against stronger adversaries. To this end, we include a small case study of the security of the magic box resource against active adversaries.

1.4 Organization of This Thesis

The thesis is organized as follows. Chapter 2 gives brief overview of notation and basic concepts employed throughout this work. Chapter 3 describes magic boxes in detail and describes how to construct computationally secure magic boxes in the quantum honest-but-curious setting using a secret-CNOT gadget (itself based on trapdoor claw-free function families) first described by Mahadev. Chapter 4 describes the Shell Game protocol in detail and analyzes its security against quantum honest-but-curious adversaries. Chapter 5 begins addressing the security of the Shell Game against fully malicious servers, including ways secret-CNOT gadgets may fail to implement magic boxes against active adversaries. Chapter 6 discusses possible improvements to the Shell Game and directions for future research.

Chapter 2

Notation and Preliminaries

2.1 Notation

2.1.1 Quantum States

We use standard notation for quantum information theory and assume familiarity with the subject equivalent to a first course in quantum computation.[1] We use bra-ket notation for pure states. An arbitrary pure, normalized quantum state is represented by $|\psi\rangle$, and its adjoint (conjugate transpose) is represented by $|\psi\rangle^\dagger := \langle\psi|$. We denote the single-qubit states

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and

$$|1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and refer to the orthonormal basis for $\mathbb{C}^2$ defined by these states as the computational basis or Z basis. We also define

$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

and

$$|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

[1] The first two chapters of Ronald de Wolf's publicly available lecture notes should provide sufficient background [dW19].

and refer to the basis defined by them as the Hadamard or X basis. We denote the two-qubit entangled states known as the Bell states by

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \text{ and}$$
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

and we refer to the basis on $\mathbb{C}^4$ defined by them as the Bell basis. We also sometimes refer to $|\Phi^+\rangle$ as an EPR pair.

When a quantum state exists on particular registers of a quantum server's workspace, we may add the name of the registers to the quantum state as a subscript, as in $|0\rangle_A$ or $|\psi\rangle_{AB}$. When we speak of mixed quantum states, we usually denote them by ρ. We use $\mathrm{Tr}[\rho]$ to denote the linear algebraic trace of ρ. We use $\mathrm{Tr}_B[\rho_{AB}]$ refer to the partial trace of ρ_{AB} "tracing out" the B subsystem. We sometimes abuse this notation slightly, writing $\mathrm{Tr}_B[|\psi\rangle_{AB}]$ to refer to the remaining state after discarding the B register of the pure state $|\psi\rangle_{AB}$.

When we discuss the distance between two quantum states ρ and σ, we use the trace distance, defined as

$$T(\rho, \sigma) = \frac{1}{2}||\rho - \sigma||_1,$$

i.e., the half the 1-norm (or Schatten norm) of the matrix $\rho - \sigma$. This is a measure of the distinguishability of ρ from σ by measurement. If $T(\rho, \sigma) = 0$, then the two states are physically identical and cannot be distinguished by any measurement. If $T(\rho, \sigma) = 1$, then the two states are orthogonal and can be distinguished perfectly by some measurement. We will often be interested in showing that two states are "close" in the sense of having small trace distance.

2.1.2 Unitary Matrices and Quantum Logic Gates

We denote the unitary Pauli matrices (or quantum logic gates implementing them) by

$$X = |+\rangle\langle+| - |-\rangle\langle-| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$
$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \text{ and}$$
$$Y = -iZX = iXZ = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

When a pure quantum state can be written as $Z^{\vec{z}}X^{\vec{x}}|\psi\rangle$, where ψ is composed of n qubits, $Z^{\vec{z}} = Z^{z_1} \otimes \ldots \otimes Z^{z_n}$, and $X^{\vec{x}} = X^{x_1} \otimes \ldots \otimes X^{x_n}$, we refer to $Z^{\vec{z}}X^{\vec{x}}$ as the "Pauli Padding" of $|\psi\rangle$ and $(\vec{x}, \vec{z})$ as the "Pauli keys". When the exponents $\vec{x}$ and $\vec{z}$ are chosen uniformly at random, the Pauli padding is called a quantum one-time pad (QOTP) by analogy with a classical one-time pad. As a classical one-time pad provides information theoretically secure encryption, a QOTP provides information theoretically secure encryption of a quantum state.

Other important single-qubit matrices/gates we refer to include the identity matrix

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and the Hadamard gate for switching between the computational and Hadamard bases

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

Important two-qubit gates include the CNOT_{AB} gate, which implements an X on the qubit in the B register if and only if the qubit in the A register is the state $|1\rangle$, and the SWAP_{AB} gate, which swaps the qubits in the A and B registers. Our Shell Game protocol implements a secret permutation of a state held by the quantum server by SWAPing two qubits at a time. We also note that

$$\text{SWAP}_{AB} = \text{CNOT}_{AB}\text{CNOT}_{BA}\text{CNOT}_{AB}.$$

To see that this so, consider an arbitrary two-qubit state

$$|\psi\rangle_{AB} = \sum_{(a,b)\in\{0,1\}^2} \alpha_{ab} |a\rangle_A |b\rangle_B .$$

Then, using the fact that $\text{CNOT}_{AB}|a\rangle_A|b\rangle_B = |a\rangle_A|a \oplus b\rangle_B$,

$$\begin{aligned}
\text{CNOT}_{AB}\text{CNOT}_{BA}\text{CNOT}_{AB}|\psi\rangle_{AB} &= \text{CNOT}_{AB}\text{CNOT}_{BA}\sum \alpha_{ab}|a\rangle_A|a\oplus b\rangle_B \\
&= \text{CNOT}_{AB}\sum \alpha_{ab}|a\oplus(a\oplus b)\rangle_A|a\oplus b\rangle_B \\
&= \text{CNOT}_{AB}\sum \alpha_{ab}|b\rangle_A|a\oplus b\rangle_B \\
&= \sum \alpha_{ab}|b\rangle_A|b\oplus(a\oplus b)\rangle_B \\
&= \sum \alpha_{ab}|b\rangle_A|a\rangle_B \\
&= \text{SWAP}_{AB}|\psi\rangle_{AB}.
\end{aligned}$$

2.2 Trapdoor Claw-Free Functions

A family of trapdoor claw-free function (TCF) pairs is a function family $\mathcal{F} = \{f_{k,b} : \mathcal{X} \to \mathcal{Y}\}$ such that each function is indexed by a key k and a bit $b \in \{0, 1\}$. All functions in $\mathcal{F}$ are injective, and pairs $f_{k,0}, f_{k,1}$ have equal images. Each

function $f_{k,b}$ in the family is "one-way" in the sense that, given $x \in \mathcal{X}$, $f_{k,b}$ can be computed efficiently, but inverting random $y \in \mathcal{Y}$ is inefficient without extra information. The trapdoor property guarantees that there exists such a piece of extra information t_k (called a "trapdoor") for each pair $f_{k,0}, f_{k,1}$ that allows efficient inversion when it is known. For any TCF pair, a pair $(x_0, x_1) \in \mathcal{X} \times \mathcal{X}$ such that $f_{k,0}(x_0) = f_{k,1}(x_1)$ is called a claw; the claw-free property guarantees that it is difficult to find such claws. In this work, $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^m$ for some positive integers n and m. We will also typically drop the k subscript whenever possible and speak only of TCF pairs f_0 and f_1.

TCFs have become popular in quantum cryptography for two reasons. The first is that, when the trapdoor information is known to a classical client but hidden from a quantum server, the client can efficiently compute a claw of preimages (x_0, x_1) given an image y, but the server cannot. Clients can leverage this asymmetry of information in order to interact with much more computationally powerful servers on more equal footing during cryptographic protocols involving TCFs. The second reason is that the claw structure of TCFs can be exploited in the quantum setting to develop novel cryptographic protocols. In particular, a classical client can direct a quantum server to produce a superposition over a random claw of a TCF pair:

$$\sum_{a \in \{0,1\}} |a\rangle |x_a\rangle .$$

Subsection 3.2.1 explains the details of preparing this superposition, as well as how to use it to build a secret-CNOT gadget (due to Mahadev [Mah18a]) for use in our Shell Game protocol.

2.3 Steane Quantum Error Correcting Code

The Steane code is a $[[7, 1, 3]]$ quantum error correcting code, encoding a single qubit into a seven-qubit quantum code word, also called a "logical qubit". This quantum error correcting code is able to correct arbitrary errors (or attacks by adversaries) that affect only a single qubit of the quantum code word. A circuit for implementing the Steane encoding is given in Figure 2.1

The theoretical basis for the Steane code and the details of how it is used to correct errors are not directly relevant to our Shell Game. (The interested reader can find those details in Steane's original paper [Ste96].) What is important is that when a server holds a surplus of $|0\rangle$ and $|+\rangle$ qubits to use as ancilla, the Steane encoding can be implemented using only CNOT gates. This means that the same secret-CNOT operations a client uses to secretly permute a server's state can also be used to secretly encode part of the server's state.

Note that, in the Steane encoding circuit, operations of the form CNOT_{AB} and CNOT_{AC} (i.e., two CNOTs controlled on the same qubit but targeting different qubits) commute, as do operations of the form CNOT_{AC} and CNOT_{BC} (i.e., two CNOTs controlled by different qubits but targeting the same qubit).

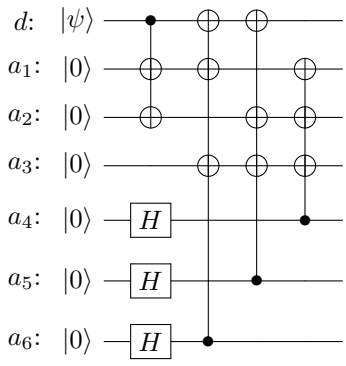


Figure 2.1: A circuit for encoding a single data qubit $|\psi\rangle$ (labeled d) into a single logical qubit $|\tilde{\psi}\rangle$ using six $|0\rangle$ ancilla qubits (labeled a_1 through a_6). Filled dots represent controls for CNOT gates; $\oplus$s represent targets of those gates.

To see this, consider an arbitrary state

$$|\psi\rangle_{ABC} = \sum_{(a,b,c)\in\{0,1\}^3} \alpha_{abc} |a\rangle_A |b\rangle_B |c\rangle_C .$$

Then

$$\begin{aligned} \mathrm{CNOT}_{AB}\mathrm{CNOT}_{AC} |\psi\rangle_{ABC} &= \sum \alpha_{abc} |a\rangle_A |a\oplus b\rangle_B |a\oplus c\rangle_C \\ &= \mathrm{CNOT}_{AC}\mathrm{CNOT}_{AB} |\psi\rangle_{ABC} \end{aligned}$$

and

$$\begin{aligned} \mathrm{CNOT}_{AC}\mathrm{CNOT}_{BC} |\psi\rangle_{ABC} &= \sum \alpha_{abc} |a\rangle_A |b\rangle_B |a\oplus b\oplus c\rangle_C \\ &= \mathrm{CNOT}_{BC}\mathrm{CNOT}_{AC} |\psi\rangle_{ABC} \end{aligned}$$

On the other hand, operations of the form CNOT_{AB} and CNOT_{BC} do *not* commute in general, since in general

$$\begin{aligned} \mathrm{CNOT}_{AB}\mathrm{CNOT}_{BC} |\psi\rangle_{ABC} &= \sum \alpha_{abc} |a\rangle_A |a\oplus b\rangle_B |b\oplus c\rangle_C \\ &\neq \sum \alpha_{abc} |a\rangle_A |a\oplus b\rangle_B |(a\oplus b)\oplus c\rangle_C \\ &= \mathrm{CNOT}_{BC}\mathrm{CNOT}_{AB} |\psi\rangle_{ABC} \end{aligned}$$

Thus, when we implement the Steane encoding during our Shell Game protocol, it is important that all CNOTs controlled on the data qubit occur before all CNOTs that target the data qubit, but otherwise the order of the CNOTs in the encoding circuit is not important.

2.4 Trap Code Authentication

Trap encoding is an authentication scheme based on error-correcting codes introduced in [BGS13]. Given a $[[n, 1, d]]$ quantum error correcting code, one can

implement a family $\mathcal{E}$ of $(3n)!$ distinct $[[3n, 1, d]]$ trap codes. The trap encoding of a single data qubit $|\psi\rangle$ proceeds as follows:

1. $|\psi\rangle$ is encoded as an n-qubit logical qubit $|\tilde{\psi}\rangle$ using an error correcting code.

2. $|\tilde{\psi}\rangle$ is adjoined with n $|0\rangle$ and n $|+\rangle$ ancillary qubits. These qubits are referred to as "trap" qubits.

3. A permutation π chosen uniformly at random from S_{3n} is applied to the entire $3n$ qubit state. Each different permutation implements a different trap encoding in the family $\mathcal{E}$.

The entire trap encoded state should then be encrypted with a quantum one-time pad. Authentication is performed by measuring and verifying the trap qubits.

Such a family $\mathcal{E}$ of trap codes is said to be $(2/3)^{d/2}$-secure against Pauli attacks. This means that the probability (over a uniform choice over the permutation π) that an arbitrary Pauli attack has a nontrivial effect on the data $|\psi\rangle$ yet is undetectable is at most $(2/3)^{d/2}$. Any Pauli attack that cannot be detected by the underlying $[[n, 1, d]]$ error correcting code must have Pauli weight at least d and so must apply an X to at least $d/2$ of the encoded qubits or a Z to at least $d/2$ of the encoded qubits. Each qubit an adversary attacks is a trap qubit with probability $2/3$; $|0\rangle$ traps detect X attacts, and $|+\rangle$ traps detect Z-attcks. Furthermore, any attack is equivalent to a mixture of Pauli attacks when the trap encoded state is also quantum one-time padded. For more details about trap code authentication schemes, see [BGS13] and [DS18].

Our Shell Game protocol can be used to remotely prepare trap code authenticated qubits using the Steane encoding. More broadly, trap authentication schemes were one of the primary inspirations for the Shell Game, which works by applying a secret permutation to a large quantum state of $|0\rangle$ and $|+\rangle$ qubits and attempts to catch cheating by measuring most of the state.

2.5 Adversaries

Since some versions of our Shell Game protocol rely on Mahadev's secret-CNOT gadget, which itself relies on the security of TCFs, the security of these versions of the Shell Game relies on the same computational assumptions required to make TCFs secure. Thus we cannot prove information theoretic security against computationally unbounded adversaries. Instead, we assume adversaries are bounded-error quantum polynomial time (BQP) servers.

Although we are ultimately interested in proving security against active, fully malicious adversaries (and we make some attempt to address them in chapter 5), we mostly focus on 0-specious adversaries, which we will refer to as quantum honest-but-curious (QHBC) adversaries. This captures the quantum generalization of classical honest-but-curious adversaries, who follow protocols honestly but attempt to learn more than they should.

Specious adversaries are a restricted class of adversaries that may deviate from a protocol, but any step, they must be able to reconstruct a state "close" to the honest state. 0-specious adversaries must be able to reconstruct the honest state exactly. This means that our QHBC adversaries are restricted to follow protocols honestly, except that they may (1) remember any classical information they learn, (2) arbitrarily purify the honest state, and (3) perform measurements which they know will be nondestructive. For a fully formal description of speciousness, see [DNS10].

Chapter 3

Magic Boxes

In this chapter, we define several kinds of cryptographic magic boxes that will be the building blocks for our Shell Game protocol. We also describe how gadgets similar to magic boxes can be constructed using a secret CNOT gadget developed by Mahadev and discuss the security of such gadgets against QHBC adversaries.

3.1 Ideal Magic Boxes

An ideal magic box is essentially a trusted third party between the classical client and quantum server. The box takes some input from each party, performs some operation on the input, and returns output to each party. Neither party's input or output is explicitly revealed to the other party. Essentially, a magic box allows a client to direct a server to perform some operation on its quantum state while withholding some information about the operation from the server. A few examples of relevant magic boxes are given below.

3.1.1 Magic QOTP Box

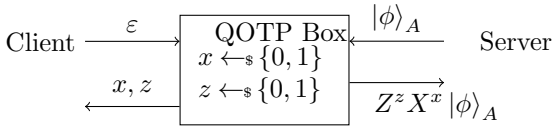


Figure 3.1: Magic QOTP Box functionality. The client's input ε is the empty string.

The simplest magic box we consider is a Magic Quantum One-Time Pad (QOTP) Box. This box takes as input a single qubit from the server and nothing (represented by the empty string ε in Figure 3.1) from the client. The

box applies a QOTP to the input before returning the padded state to the server and the QOTP Pauli keys to the client. This box essentially allows a classical client to quantum one-time pad a qubit held by the server without having to share the keys with the server. Although the Magic QOTP Box is not used directly in our shell game protocols and we do not attempt to construct a gadget for it, this box plays an essential role in our security analysis of larger boxes in chapter 5.

3.1.2 Magic CNOT Box

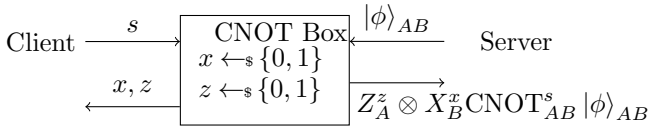


Figure 3.2: Magic CNOT Box functionality, with s a secret bit chosen by the client.

The next magic box we consider is a Magic CNOT Box, illustrated in Figure 3.2. This box takes a secret bit s from the client and a two-qubit quantum state $|\phi\rangle_{AB}$ from the server. The box returns the state $Z_A^z \otimes X_B^x \mathrm{CNOT}_{AB}^s |\phi\rangle_{AB}$ to the quantum server (with x and z generated uniformly at random) and returns x and z to the client. This box allows a classical client to direct a quantum server to perform a "secret" CNOT operation: the client knows whether a CNOT was actually performed on the server's qubits, but the server itself does not (i.e., the server does not learn s). This magic box is an ideal version of the secret CNOT gadget developed by Mahadev, which we describe in subsection 3.2.1 and analyze in section 3.3.

Note that the $Z^z \otimes X^x$ Pauli padding on the output is necessary to keep the client's secret bit s private from adversarial QHBC servers. To see why, consider what happens when the server's quantum input is, for example, $|1\rangle |0\rangle$. If this padding were not present, the server could learn s by measuring the second qubit in the computational basis. However, with the X padding, the outcome of that measurement is $x \oplus s$ (rather than simply s). Since x is uniformly random, the outcome of the measurement is also uniformly random and therefore totally uncorrelated. (Similarly, without the Z padding, the server could learn s on input $|+\rangle |-\rangle$ by measuring the first output qubit in the Hadamard basis.) There is no nondestructive measurement the QHBC server can perform on any input that will reveal s. Indeed, even a computationally unbounded, fully malicious server would have no advantage in guessing s. The following proposition formalizes this idea.

Proposition 1. *Regardless of the state the server inputs to the Magic CNOT Box, the server's output when $s = 0$ is indistinguishable from the server's output when $s = 1$.*

Proof. Let the server's initial state be

$$|\phi\rangle_{ABC} = \sum_{(a,b)\in\{0,1\}^2, c\in\{0,1\}^{|C|}} \alpha_{abc} |a\rangle_A |b\rangle_B |c\rangle_C ,$$

where the A and B subsystems are the two qubits entered into the magic box and the C subsystem is an arbitrary number of qubits as side information. Also let

$$\begin{aligned}\rho_{ABC} &= |\phi\rangle\langle\phi|_{ABC} \\ &= \sum_{(a,b,a',b')\in\{0,1\}^4, (c,c')\in\{0,1\}^{2|C|}} \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle\langle a'|_A \otimes |b\rangle\langle b'|_B \otimes |c\rangle\langle c'|_C .\end{aligned}$$

Recall that the exponents of the Pauli padding applied by the box are chosen uniformly at random and that these exponents are not explicitly output to the server. For s=0, the output of the box from the server's informational perspective is

$$\begin{aligned}\rho_{s=0} =& \frac{1}{4}\Big(\rho_{ABC} + (I_A\otimes X_B)\rho_{ABC}(I_A\otimes X_B) \\ &+ (Z_A\otimes I_B)\rho_{ABC}(Z_A\otimes I_B) + (Z_A\otimes X_B)\rho_{ABC}(Z_A\otimes X_B)\Big) \\ =& \frac{1}{4}\Big(\sum \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle\langle a'|_A \otimes |b\rangle\langle b'|_B \otimes |c\rangle\langle c'|_C \\ &+ \sum \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle\langle a'|_A \otimes |1-b\rangle\langle 1-b'|_B \otimes |c\rangle\langle c'|_C \\ &+ \sum (-1)^{a\oplus a'}\alpha_{abc}\alpha^*_{a'b'c'} |a\rangle\langle a'|_A \otimes |b\rangle\langle b'|_B \otimes |c\rangle\langle c'|_C \\ &+ \sum (-1)^{a\oplus a'}\alpha_{abc}\alpha^*_{a'b'c'} |a\rangle\langle a'|_A \otimes |1-b\rangle\langle 1-b'|_B \otimes |c\rangle\langle c'|_C \Big)\end{aligned}$$

Observe that when we add the first and third sum and the second and fourth sum, terms with $a \neq a'$ cancel out.

$$\begin{aligned}=& \frac{1}{2}\Big(\sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle\langle a|_A \otimes |b\rangle\langle b'|_B \otimes |c\rangle\langle c'|_C \\ &+ \sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle\langle a|_A \otimes |1-b\rangle\langle 1-b'|_B \otimes |c\rangle\langle c'|_C \Big)\end{aligned}$$

Finally, we combine like terms in the two remaining sums.

$$= \sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle\langle a|_A \otimes \Big(\frac{1}{2}|b\rangle\langle b'| + \frac{1}{2}|1-b\rangle\langle 1-b'|\Big)_B \otimes |c\rangle\langle c'|_C .$$

In the case that $s = 1$, the server's box output is

$$\begin{aligned}
\rho_{s=1} =& \frac{1}{4}\Big(\mathrm{CNOT}_{AB}\rho_{ABC}\mathrm{CNOT}_{AB} \\
&+ (I_A \otimes X_B)\mathrm{CNOT}_{AB}\rho_{ABC}\mathrm{CNOT}_{AB}(I_A \otimes X_B) \\
&+ (Z_A \otimes I_B)\mathrm{CNOT}_{AB}\rho_{ABC}\mathrm{CNOT}_{AB}(Z_A \otimes I_B) \\
&+ (Z_A \otimes X_B)\mathrm{CNOT}_{AB}\rho_{ABC}\mathrm{CNOT}_{AB}(Z_A \otimes X_B)\Big) \\
=& \frac{1}{4}\Big(\sum \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle \langle a'|_A \otimes |b \oplus a\rangle \langle b' \oplus a'|_B \otimes |c\rangle \langle c'|_C \\
&+ \sum \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle \langle a'|_A \otimes |b \oplus (1-a)\rangle \langle b' \oplus (1-a')|_B \otimes |c\rangle \langle c'|_C \\
&+ \sum (-1)^{a \oplus a'} \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle \langle a'|_A \otimes |b \oplus a\rangle \langle b' \oplus b'|_B \otimes |c\rangle \langle c'|_C \\
&+ \sum (-1)^{a \oplus a'} \alpha_{abc}\alpha^*_{a'b'c'} |a\rangle \langle a'|_A \otimes |b \oplus (1-a)\rangle \langle b' \oplus (1-a')|_B \otimes |c\rangle \langle c'|_C \Big)
\end{aligned}$$

Again, we can add the first and third sums and second and fourth sums, using the $(-1)^{a \oplus a'}$ phase to cancel terms with $a \neq a'$.

$$\begin{aligned}
=& \frac{1}{2}\Big(\sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle \langle a|_A \otimes |b \oplus a\rangle \langle b' \oplus a|_B \otimes |c\rangle \langle c'|_C \\
&+ \sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle \langle a|_A \otimes |b \oplus (1-a)\rangle \langle b' \oplus (1-a)|_B \otimes |c\rangle \langle c'|_C \Big)
\end{aligned}$$

We can also add like terms in this case.

$$= \sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle \langle a|_A \otimes \Big(\frac{1}{2} |b \oplus a\rangle \langle b' \oplus a| + \frac{1}{2} |b \oplus (1-a)\rangle \langle b' \oplus (1-a)|\Big)_B \otimes |c\rangle \langle c'|_C$$

Now observe that regardless of the value of a, the B subsystem of each term can be expressed as $\frac{1}{2} |b\rangle \langle b'| + \frac{1}{2} |1-b\rangle \langle 1-b'|$.

$$= \sum \alpha_{abc}\alpha^*_{ab'c'} |a\rangle \langle a|_A \otimes \Big(\frac{1}{2} |b\rangle \langle b'| + \frac{1}{2} |1-b\rangle \langle 1-b'|\Big)_B \otimes |c\rangle \langle c'|_C$$

This is precisely the state we computed for $\rho_{s=0}$.

$$= \rho_{s=0}.$$

Thus the two cases are indistinguishable from the perspective of the server. □

Note also that the client's *output* (i.e., the Pauli keys) is *not* always entirely private from a QHBC server. For example, if the server's quantum input is $|+\rangle |+\rangle$, a QHBC server can learn z simply by measuring the first qubit of the box output in the Hadamard basis, although it will still have no advantage in guessing either x or s in that case.

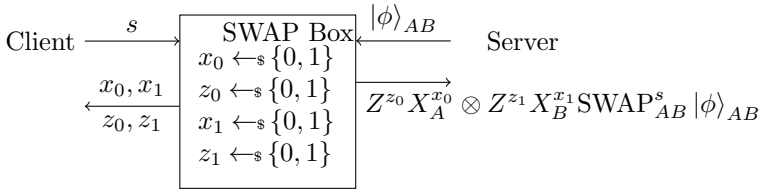


Figure 3.3: Magic SWAP Box functionality.

3.1.3 Magic SWAP Boxes

The Magic SWAP Box, illustrated in Figure 3.3, is similar to a Magic CNOT box in that it takes a secret bit s from the client and a two-qubit quantum state $|\phi\rangle_{AB}$ from the server. The box generates a QOTP uniformly at random, returns the one-time padded state

$$Z^{z_0} X_A^{x_0} \otimes Z^{z_1} X_B^{x_1} \mathrm{SWAP}_{AB}^{s} |\phi\rangle_{AB}$$

to the server, and returns the QOTP Pauli keys to the client. Our shell game protocol uses boxes like these to secretly permute and encrypt a quantum state held by the server.

Observe that, even if a Magic SWAP Box is unavailable, the client and server could achieve a similar effect by "chaining" three Magic CNOT Boxes, using the fact that

$$\mathrm{CNOT}_{AB} \mathrm{CNOT}_{BA} \mathrm{CNOT}_{AB} = \mathrm{SWAP}_{AB}.$$

First, the client and server input s and $|\phi\rangle$ into a Magic CNOT_{AB} Box. Then, the server feeds the ouput of that box into a Magic CNOT_{BA} Box, into which the client inputs the same s. Finally, the server feeds the output of the second box to a Magic CNOT_{AB} Box, into which the client inputs the same s. The final quantum state held by the server is a quantum one-time padded $\mathrm{SWAP}_{AB}^{s} |\phi\rangle_{AB}$. The client can compute the QOTP Pauli keys from the output received from the three Magic CNOT Boxes. (See section 3.3 for details.)

The weakness of performing a secret-SWAP in this way, rather than with a Magic SWAP Box, is that an adversarial server has more opportunity to deviate from an honest protocol (e.g., by inputting some quantum state besides the output of the previous box into the next box). The strength is that a client can also be more flexible; that is, the client need not choose the same s for every Magic CNOT Box. Depending on the choice of s_1, s_2, s_3 input to the three Magic CNOT Boxes, the client could alter the secret operation delegated by the protocol to be $\mathrm{SWAP}_{AB}, \mathrm{CNOT}_{AB}, \mathrm{CNOT}_{BA}$, or $I_A \otimes I_B$. This is the inspiration for the modified Magic SWAP Box in Figure 3.4, which is used in one version of our shell game protocol to secretly permute, encrypt, and *encode* a quantum state held by the server with an error correcting code. The ability to simultaneously secretly encode the server's state while secretly permuting it allows the client to remotely prepare a trap-encoded quantum state using our Shell Game protocol.

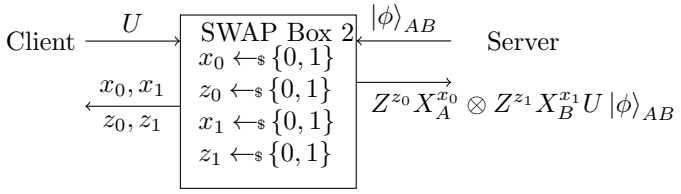


Figure 3.4: Modified Magic SWAP Box functionality. U is a classical description of a unitary chosen from the set $\{\text{SWAP}_{AB}, \text{CNOT}_{AB}, \text{CNOT}_{BA}, I_A \otimes I_B\}$.

3.2 Mahadev's Secret CNOT Gadget

In this section, we describe the construction of a secret-CNOT gadget first published by Mahadev in [Mah18a]. We relate this gadget to the magic boxes described in the previous section. In the following section, we analyze the security of the gadget against bounded-error quantum polynomial time (BQP) QHBC adversaries.[1]

3.2.1 Gadget Description

Mahadev's gadget assumes the existence of quantum-safe TCF pairs[2]. In the first step, the client generates an encryption of the secret bit s as a trapdoor claw-free function pair $\mathsf{Enc}(s) = (f_0, f_1)$, together with a trapdoor for inverting each function. The function pairs have the form $f_b : \{0,1\}^n \to \{0,1\}^m$ for $b \in \{0,1\}$ and must satisfy the additional restriction that if $f_0(x_0) = f_1(x_1)$, then $x_0 = \mu_0 \| r_0$ and $x_1 = \mu_1 \| r_1$ such that $\mu_0 \oplus \mu_1 = s$. (In other words, the XOR of the initial bits of any claw is always s.) The client then sends some classical description of $\mathsf{Enc}(s)$ to the server, which we will refer to as a public key k. This public key allows the server to implement the following $(1+n+m)$-qubit controlled unitary $\hat{U}_{\mathsf{Enc}(s)}$:

$$\hat{U}_{\mathsf{Enc}(s)} |a\rangle_A |x\rangle_X |y\rangle_Y = |a\rangle_A |x\rangle_X |y \oplus f_a(x)\rangle_Y ,$$

for $a \in \{0,1\}$, $x \in \{0,1\}^n$, and $y \in \{0,1\}^m$. The client retains the trapdoor information privately so that she can efficiently invert the functions and find claws, but the polytime-bounded quantum server cannot.

Suppose the server's initial state on which the secret-CNOT is to be performed is $|\phi\rangle_{AB} = \sum_{a,b \in \{0,1\}} \alpha_{ab} |a\rangle_A |b\rangle_B$. The server first prepares $n+m$ ancillary registers in the all-zero state.

$$|\phi\rangle_{AB} |\vec{0}\rangle_X |\vec{0}\rangle_Y$$

[1] BQP is the complexity class of decision problems efficiently solvable by quantum computers as defined in [BV97]. Refer to section 2.5 for the definition of QHBC.

[2] Of course, the existence of true families of TCFs, as with all one-way function families, is conjectural. However, Mahadev also gives a construction of a family that behaves similarly to a family of TCF pairs using the Learning with Errors (LWE) problem. This family is quantum-safe assuming the hardness of LWE. For details, see [Mah18a]

The server then applies a Hadamard transform $H_X^{\otimes n}$ to the X register to produce a uniform superposition over all strings in $\{0,1\}^n$, the domain of $\mathsf{Enc}(s)$.

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |\phi\rangle_{AB} |x\rangle_X |\vec{0}\rangle_Y$$

Next, the server applies $\hat{U}_{\mathsf{Enc}(s)}$, controlled on the A and X registers and targeting the Y register.

$$\frac{1}{\sqrt{2^n}} \sum_{a,b,x} \alpha_{ab} |a\rangle_A |b\rangle_B |x\rangle_X |f_a(x)\rangle_Y$$

Note that at this point, the Y register is entangled with the A and X registers. The server then measures the Y register in the computational basis and reports the outcome y to the client. The server's remaining state collapses to

$$\sum_{a,b} \alpha_{ab} |a\rangle_A |b\rangle_B |x_a\rangle_X ,$$

with $f_0(x_0) = f_1(x_1) = y$ a claw of $\mathsf{Enc}(s)$. Under the assumption that $\mathsf{Enc}(s)$ is quantum-safe and without the trapdoor, the server cannot compute this claw efficiently.[3]

To leverage our additional restriction on the structure of claws of $\mathsf{Enc}(s)$, we can also relabel the server's X register as $|x\rangle_X = |\mu, r\rangle_X$.

$$\sum_{a,b} \alpha_{ab} |a\rangle_A |b\rangle_B |\mu_a, r_a\rangle_X$$

Since $\mu_0 \oplus \mu_1 = s$, we have $\mu_a = (a \cdot s) \oplus \mu_0$. The server now performs a CNOT controlled by the first bit of the X register (containing μ) and targeting the B register. Since $\mathrm{CNOT} |b_1\rangle |b_2\rangle = |b_1\rangle |b_2 \oplus b_1\rangle$, the server's state becomes

$$\sum_{a,b} \alpha_{ab} |a\rangle_A |b \oplus ((a \cdot s) \oplus \mu_0)\rangle_B |\mu_a, r_a\rangle_X .$$

Since $X^\mu |b\rangle = |b \oplus \mu\rangle$ and $\mathrm{CNOT}^s |a\rangle |b\rangle = |a\rangle |b \oplus (a \cdot s)\rangle$, we can rewrite this state as

$$(I_A \otimes X^{\mu_0} \otimes I_X)\mathrm{CNOT}^s_{AB} \sum_{a,b} \alpha_{ab} |a\rangle_A |b\rangle_B |x_a\rangle_X .$$

Next, the server performs another Hadamard transform $H_X^{\otimes n}$ on the X register, changing the state to

$$(I_A \otimes X^{\mu_0} \otimes I_X)\mathrm{CNOT}^s_{AB} \sum_{a,b} \alpha_{ab} |a\rangle_A |b\rangle_B \big(\frac{1}{\sqrt{2^n}} \sum_{d \in \{0,1\}^n} (-1)^{d \cdot x_a} |d\rangle_X \big).$$

[3] Claw-freeness also guarantees that a dishonest server cannot efficiently counterfeit this state by choosing the claw themselves.

Finally, the server measures the X register in the computational basis and reports the outcome d to the client. The server's remaining quantum state collapses to

$$(I_A \otimes X^{\mu_0})\mathrm{CNOT}^s_{AB} \sum_{a,b} (-1)^{d \cdot x_a} \alpha_{ab} |a\rangle |b\rangle ,$$

or equivalently, up to an irrelevant global phase,

$$(Z_A^{(d \cdot x_0) \oplus (d \cdot x_1)} \otimes X_B^{\mu_0})\mathrm{CNOT}^s_{AB} |\phi\rangle .$$

(Note that a Hadamard transform followed by a computational basis measurement is equivalent to a Hadamard basis measurement.)

Both the Pauli exponents $(d \cdot x_0) \oplus (d \cdot x_1)$ (equivalently, $d \cdot (x_0 \oplus x_1)$, and hereafter assigned the symbol $\bar{d}$) and μ_0 can be efficiently computed by the client after learning y and d, since the client retains the trapdoor information that allows inversion of the TCF pair.

3.2.2 Secret-CNOT Gadget vs. Magic CNOT Box

Certain similarities between the Secret-CNOT gadget and the Magic CNOT Box are obvious. They both require a secret bit s from the client and a bipartite quantum state $|\psi\rangle$ from the server, and they both result in the server holding a state of the form $Z^z \otimes X^x \mathrm{CNOT}^s_{AB} |\phi\rangle$ such that the client knows the values of the exponents x and z. But dissimilarities are equally obvious. The gadget has no trusted third party; the server is in charge of manipulating its own quantum state and reporting outcomes to the client, and so there is more opportunity for an adversarial server to cheat in a protocol built from secret-CNOT gadgets than in one built from Magic CNOT Boxes. Furthermore, the exponents of the Pauli padding of the gadget are not truly generated uniformly at random, and the server is prevented from computing them only by the hardness of inverting one-way functions and finding claws.

However, in the QHBC setting we focus on, having extra opportunities for dishonest behavior is not much of a problem, since adversaries are so restricted. Furthermore, since we consider servers which are computationally bounded, our servers are unable to distinguish the exponents of the Pauli padding generated by the secret-CNOT gadget from truly random bits except with negligible probability (as we shall prove in section 3.3). Thus, in our particular setting, secret-CNOT gadgets are "almost as good" as true magic boxes. Furthermore, we can use secret-CNOT gadgets to build secret-SWAP gadgets that are almost as good as true Magic SWAP boxes.

3.2.3 Secret-SWAP from Secret-CNOT

In the same way that we can approximate a Magic Swap Box using three Magic CNOT Boxes, we can construct a secret-SWAP Gadget using three secret-CNOT gadgets. After applying three secret-CNOT gadgets (reversing the control and

target for the second secret-CNOT) on an initial bipartite state $|\psi\rangle$, the state becomes

$$(Z^{\bar{d}^{(3)}} \otimes X^{\mu_0^{(3)}})\mathrm{CNOT}_{AB}^{s^{(3)}}(X^{\mu_0^{(2)}} \otimes Z^{\bar{d}^{(2)}})\mathrm{CNOT}_{BA}^{s^{(2)}}(Z^{\bar{d}^{(1)}} \otimes X^{\mu_0^{(1)}})\mathrm{CNOT}_{AB}^{s^{(1)}} |\psi\rangle ,$$

where each superscript (i) is a reminder of which of the three secret-CNOTs a particular variable is associated with. Note that the client is free to choose a different s for each secret-CNOT, and even if the same s is reused every time, then a different encryption (that is, a different TCF pair) should be used for each secret-CNOT. Furthermore, the exponents μ_0 and $\bar{d}$ generated by each gadget will not in general be the same, even if the same secret and encryption were reused, because the measurement outcomes y and d that generate them are uniformly distributed each time.

We can move the secret CNOTs to the inside using the following rules:

$$\mathrm{CNOT}^s(X \otimes I) = (X \otimes X^s)\mathrm{CNOT}^s$$

$$\mathrm{CNOT}^s(I \otimes X) = (I \otimes X)\mathrm{CNOT}^s$$

$$\mathrm{CNOT}^s(Z \otimes I) = (Z \otimes I)\mathrm{CNOT}^s$$

$$\mathrm{CNOT}^s(I \otimes Z) = (Z^s \otimes Z)\mathrm{CNOT}^s$$

We can also re-arrange the accumulated Pauli operators into the form $Z^z X^x$ using anticommutativity,

$$X^x Z^z = (-1)^{x \cdot z} Z^z X^x,$$

though we can effectively disregard the $(-1)^{x \cdot z}$ phase as it is global.

Thus the final state (up to global phase) without Pauli padding is

$$\mathrm{CNOT}_{AB}^{s^{(3)}}\mathrm{CNOT}_{BA}^{s^{(2)}}\mathrm{CNOT}_{AB}^{s^{(1)}} |\psi\rangle ,$$

which is equal to $\mathrm{SWAP}_{AB} |\psi\rangle$, $\mathrm{CNOT}_{AB} |\psi\rangle$, $\mathrm{CNOT}_{BA} |\psi\rangle$, or $(I \otimes I) |\psi\rangle$ depending on how the client chose the secret bits for the three secret-CNOT gadgets. (Compare to the Modified Magic SWAP Box in Figure 3.4.) If the client restricts its choices such that $s^{(1)} = s^{(2)} = s^{(3)} = s$, then the unpadded state is $\mathrm{SWAP}_{AB}^{s} |\psi\rangle$. (Compare with the unmodified Magic SWAP Box in Figure 3.3.)

In either case, the Pauli padding on top of this state is

$$Z^{\bar{d}^{(3)} \oplus (\bar{d}^{(1)} \oplus ((\bar{d}^{(2)} \oplus (\bar{d}^{(1)} \cdot s^{(2)})) \cdot s^{(3)}))} X^{\mu_0^{(2)} \oplus (\mu_0^{(1)} \cdot s^{(2)})}$$
$$\otimes Z^{\bar{d}^{(2)} \oplus (\bar{d}^{(1)} \cdot s^{(2)})} X^{\mu_0^{(3)} \oplus (\mu_0^{(1)} \oplus (\mu_0^{(2)} \oplus (\mu_0^{(1)} \cdot s^{(2)})) \cdot s^3)}.$$

We can further rewrite the exponents in a more intuitive form. Note that for bits a and b we have $(a \oplus (b \cdot s^{(i)})) \cdot s^{(j)} = (a \cdot s^{(j)}) \oplus (b \cdot s^{(i)} s^{(j)})$ and $b \oplus (b \cdot s^{(i)} s^{(j)}) = b \cdot (1 - s^{(i)} s^{(j)})$. Thus the Pauli padding is

$$Z^{\bar{d}^{(3)} \oplus (\bar{d}^{(2)} \cdot s^{(2)}) \oplus (\bar{d}^{(1)} \cdot (1 - s^{(2)} s^{(3)}))} X^{\mu_0^{(2)} \oplus (\mu_0^{(1)} \cdot s^{(2)})}$$
$$\otimes Z^{\bar{d}^{(2)} \oplus (\bar{d}^{(1)} \cdot s^{(2)})} X^{\mu_0^{(3)} \oplus (\mu_0^{(2)} \cdot s^{(3)}) \oplus (\mu_0^{(1)} \cdot (1 - s^{(2)} s^{(3)}))}.$$

If the client is restricted to choosing $s^{(1)} = s^{(2)} = s^{(3)} = s$, then the padding becomes

$$Z^{\bar{d}^{(3)} \oplus (\bar{d}^{(2)} \cdot s) \oplus (\bar{d}^{(1)} \cdot (1-s))} X^{\mu_0^{(2)} \oplus (\mu_0^{(1)} \cdot s)} \otimes Z^{\bar{d}^{(2)} \oplus (\bar{d}^{(1)} \cdot s)} X^{\mu_0^{(3)} \oplus (\mu_0^{(2)} \cdot s) \oplus (\mu_0^{(1)} \cdot (1-s))}.$$

If these exponents appear random to the server (which, by computational assumptions, cannot determine s), then the state is effectively quantum one-time padded on top of the secret-SWAP. This is a weaker (computationally secure) version of the true (information theoretically secure) QOTP applied by the Magic SWAP Box.

3.3 Gadget Security

We must now confirm that exponents of the Pauli padding really do appear random to a computationally bounded QHBC server. We begin with a few informal observations and conclude with proofs of specific claims. First, note that when the server behaves honestly, the outcomes y and d that generate the exponents result from measuring an equal superposition over all possible outcomes. Thus, these measurement outcomes are truly random if anything in the universe is.

Next, note that, by construction, μ_0 is a uniformly random bit when the server honestly measures a random y. A computationally bounded server cannot learn μ_0 by inverting y. A QHBC server cannot learn μ_0 by measurement except in the degenerate case that server knows the control qubit was in the state $|0\rangle$, as in this case the X register never contains a superposition over claws, but merely the separable state $|x_0\rangle$, which can be nondestructively measured. Of course, performing a secret-CNOT is pointless in this case, and we can always avoid this situation in our Shell Game protocol. Furthermore, if a server were able to learn an x_0 generated by a truly random outcome y, then assuming our families of TCF pairs used to encrypt s have the adaptive hardcore bit property, the server would have negligible advantage in computing $\bar{d}$.

It is perhaps a bit more difficult to form strong intuitions about why it should be difficult to guess $\bar{d}$. We can say something about $\bar{d}$ given a few specific outcomes d: For example, when $d = \vec{0}$, then necessarily $\bar{d} = 0$; when $d = 1\|\vec{0}$, the server knows $\bar{d} = s$, even if the server cannot be sure of the value of s. However, the chances of observing these particular outcomes is negligible in n, the length of strings in the domain of our TCFs. We shall see in Theorem 4 that $\bar{d}$ is a hardcore bit over a uniform choice of d by a version of the Goldreich-Levin (GL) argument.

3.3.1 Security of the Secret CNOT Against QHBC Adversaries

In our Shell Game protocol, the server is initially asked to prepare a multipartite quantum state composed of many $|0\rangle$s and $|+\rangle$s. Because, as we have seen, degenerate secret-CNOT operations may leak information by allowing a curious

server to make measurements it knows will be nondestructive, we will never use a secret-CNOT gadget controlled by a qubit the server knows is in the $|0\rangle$ state or targeting a qubit the server knows is in the $|+\rangle$ state. If our permutations and encodings are done secretly, the only time the honest-but-curious server will know the state of the control and target qubit with certainty is at the beginning of the protocol, when control qubits are in the $|+\rangle$ state and target qubits are in the $|0\rangle$ state. We focus on this scenario.

Theorem 2. *A polytime-bounded quantum honest-but-curious (QHBC) server has negligible advantage in guessing $\bar{d}$, μ_0, or their parity by running the secret CNOT protocol on the initial state $|+\rangle|0\rangle$.*

Proof. The QHBC server is allowed to remember any classical data it can learn "for free". This information can be communicated to the server by the client, the outcomes of measurements made by the server in the course of the honest execution of the protocol, or the outcomes of extraneous measurements the server knows will not be destructive. In the case of the secret CNOT protocol with initial state $|+\rangle|0\rangle$, the classical information available to the server is:

- k (some public key used to point to the pair of TCFs used to encrypt s),
- y (the outcome of the measurement of the codomain register Y, effectively drawn from a uniformly random distribution over the codomain of f_0 and f_1), and
- d (the outcome of the Hadamard measurement on the domain register X, effectively chosen uniformly at random from the set of binary strings the length of the register).

We do not know much about k in the abstract, but we will assume it is chosen in some sensible way and that our family of TCF pairs is large enough that keys are unlikely to be repeated.

It is sufficient to show that the server has no advantage in guessing (i.e., cannot "approximate") the bits $\bar{d}$, μ_0, or their parity. To show that a QHBC server cannot approximate these bits, consider ways the server could efficiently approximate s (against our assumption that the encryption of s is secure) that would be possible if a QHBC server could approximate those bits.

Guessing $\bar{d}$: If a QHBC server could approximate $\bar{d}$, it would be possible for a dishonest server to hold the state $Z^{\bar{d}}X^{\mu_0}\mathrm{CNOT}^s|+\rangle|0\rangle$ while knowing a guess $\bar{d}'$ for $\bar{d}$ which is correct with probability nonnegligibly greater than $1/2$. The server could apply the unitary $Z^{\bar{d}'}\otimes I$ to the state and measure both qubits in the Bell basis. If the outcome of the measurement is Φ^- or Ψ^-, guess $s=0$. Otherwise, guess $s=1$.

Analysis: Suppose $s=0$. This attack will guess s correctly with probability $1/2$. (Note that it does not matter whether the guess $\bar{d}'$ is correct in this case; either way, the measurement outcome is uniformly random.) If $s=1$, then if the guess $\bar{d}'=\bar{d}$ (which happens with advantage), the server will definitely guess s correctly, as the state is either $|\Phi^+\rangle$ or $|\Psi^+\rangle$ (depending on μ_0). If $\bar{d}'\neq\bar{d}$,

the server will definitely guess s incorrectly, since the state is either $|\Phi^-\rangle$ or $|\Psi^-\rangle$. Thus the server has an inverse polynomial advantage in detecting $s = 1$, and so it also has an inverse polynomial advantage in guessing s over a uniform distribution on s.

Guessing μ_0: If a QHBC server could approximate μ_0, it would be possible for a dishonest server to hold the state $Z^{\bar{d}}X^{\mu_0}\mathrm{CNOT}^s |+\rangle |0\rangle$ while knowing a guess μ_0' for μ_0 which is correct with probability nonnegligibly greater than $1/2$. The server could apply the unitary $I \otimes X^{\mu_0'}$ to the state and measure both qubits in the Bell basis. If the outcome of the measurement is Ψ^+ or Ψ^-, guess $s = 0$. Otherwise, guess $s = 1$.

Analysis: Suppose $s = 0$. This attack will guess s correctly with probability $1/2$. (Again, it does not matter whether the guess μ_0' is correct in this case because the outcome is uniformly random.) If $s = 1$, then if the guess $\mu_0' = \mu_0$ (which happens with advantage), the server will definitely guess s correctly, as the state is either $|\Phi^+\rangle$ or $|\Phi^-\rangle$ (depending on $\bar{d}$). If $\mu_0' \neq= \mu_0$, the server will definitely guess s incorrectly, since the state is either $|\Psi^+\rangle$ or $|\Psi^-\rangle$. Thus the server has an inverse polynomial advantage in detecting $s = 1$, and so it also has an inverse polynomial advantage in guessing s over a uniform distribution on s.

Guessing parity: If a QHBC server could approximate $\bar{d} \oplus \mu_0$, the server could hold the same state as before while knowing a guess p for $\bar{d} \oplus \mu_0$. If $p = 1$, the server applies the unitary $X \otimes I$. In either case, the server measures in the Bell basis. If the outcome is Φ^- or Ψ^+, the server guesses $s = 0$. Otherwise, the server guesses $s = 1$.

Analysis: Again, the server's measurement outcome is uniformly random if $s = 0$ and the server's winning probability in that case is only $1/2$. But if $s = 1$, the server's guess for s is correct if and only if its guess $p = \bar{d} \oplus \mu_0$, so again the server has inverse polynomial advantage in this case and over a uniform distribution on s.

Observe that all three attacks described here are specific instances of a more general attack that is possible whenever the server's Bayesian prior on the joint distribution of $(\bar{d}, \mu_0)$ is non negligibly far from uniform. □

Additionally, note that we can make an argument that the server cannot learn $\bar{d}$ by adapting classical arguments about hardcore bits. This argument holds even without assuming the server is merely QHBC or that its input state is $|+\rangle |0\rangle$ (though computational assumptions are still necessary). The following two propositions sketch the argument.

Proposition 3. *The quantum server has negligible advantage in guessing $\tilde{x} = x_0 \oplus x_1$ given $y = f_0(x_0) = f_1(x_1)$ for trapdoor claw-free function pairs f_0, f_1.*

Proof. Suppose the server could guess the XOR with non-negligible advantage using an algorithm A. Then this gives an algorithm for computing claws with non-negligible advantage: Given x_0, efficiently compute $f_0(x_0)$. Use A to efficiently compute $x_0 \oplus x_1$ such that $f_0(x_0) = f_1(x_1)$ with non-negligible advantage. Compute $(x_0 \oplus x_1) \oplus x_0$ to obtain x_1. But since f_0, f_1 is a claw-free pair, an

efficient algorithm for computing claws with non-negligible advantage must not exist, a contradiction. □

Proposition 4. *Each $\bar{d}$ term is hard to compute on average over a uniform distribution over measurement outcomes d.*

Proof. Follows from an argument similar to the Goldreich-Levin (GL) hardcore predicate proof [GL89]; if the server could compute with nonnegligible advantage (over a uniform distribution of measurement outcomes d) the XOR of a random nonempty subset (induced by d) of $x_0 \oplus x_1$ given y and d, the server could efficiently approximate $x_0 \oplus x_1$, contradicting Proposition 3.

The original GL proof does not quite apply, but a version given in Appendix B of [CCKW18] does. In the original GL proof, the ability to compute $\langle x, r\rangle$ given $f(x), r$ (efficiently and with advantage) leads to a "sampling" algorithm which returns a polynomial number of guesses $\{x'\}$ for x such that at least one is correct with probability negligibly far from 1, contradicting the hardness of inverting $f(x)$. In our case, the algorithm instead returns a polynomial number of guesses $\{\tilde{x}'\}$ for $\tilde{x} = x_0 \oplus x_1$, contradicting the hardness of computing claws by way of Proposition 3. □

We will find these additional guarantees on the hardness of guessing $\bar{d}$ useful in the following section, in which we must address correlations arising from chaining multiple secret-CNOT gadgets into a secret-SWAP gadget.

3.3.2 Security of the Secret SWAP Against QHBC Adversaries

We have seen that the exponents of the Pauli padding generated by Mahadev's secret-CNOT gadget appear close to uniformly random to a computationally bounded QHBC server in the sense that such a server has no advantage in guessing those exponents on average. We have also seen that the exponents of the Pauli padding after performing a secret-SWAP are XORs of the exponents generated by each of its constituent secrect-CNOTs. Thus when there is no correlation among the exponents generated by each secret-CNOT gadget, as is the case when the choice of secret bit for each gadget is unrestricted, the padding generated by the secret-SWAP also appears random to a computationally bounded server. Thus we may think of this padding as a "computational" QOTP.

When the server is restricted to always choose the same secret bit s for each secret-CNOT gadget used in the same secret-SWAP, we must be a little more careful. In this case, there *is* a correlation among the secret-CNOT exponents that allows more information to leak from their XOR than from each exponent individually in certain situations. In particular, in the (negligibly probable) case that $d^{(1)} = d^{(2)} = d^{(3)} = 1\|\vec{0}$, the exponent of the Pauli Z applied to the A register becomes

$$s \oplus (s \cdot s) \oplus (s \cdot (1 - s)) = s \oplus s = 0.$$

However, this extra information leak occurs only with negligible probability. Note that there is potential for extra information leak only when all three $d^{(i)}$ measurement outcomes have 1 as their initial bit, which occurs with probability 1/8. In this case, the exponent of the Pauli Z on the A subsystem is the XOR of two bits, each of which is generated by taking the XOR of a random subset of bits of $r_0 \oplus r_1$ for some claw $(\mu_0 \| r_0, \mu_1 \| r_1)$.

There is no apparent correlation between any two randomly chosen claws $(\mu_0^{(i)} \| r_0^{(i)}, \mu_1^{(i)} \| r_1^{(i)})$ and $(\mu_0^{(j)} \| r_0^{(j)}, \mu_1^{(j)} \| r_1^{(j)})$ for any two randomly chosen TCF pairs in our family with the same secret s besides that

$$\mu_0^{(i)} \oplus \mu_1^{(i)} = \mu_0^{(j)} \oplus \mu_1^{(j)} = s.$$

By a GL argument, the XOR of a random subset of bits of $r_0^{(i)} \oplus r_1^{(i)}$ (or $r_0^{(j)} \oplus r_1^{(j)}$) is a hardcore bit such that a computationally bounded server's advantage in guessing is negligible in $n-1$. Since there is no correlation between $(r_0^{(i)}, r_1^{(i)})$ and $(r_0^{(j)}, r_1^{(j)})$, the XOR of these hardcore bits is also hard to compute on average. Thus we may treat the Pauli padding as a "computational" QOTP in this case as well.

Chapter 4

Shell Game

We now have all the cryptographic machinery necessary to describe our Shell Game protocol for remote state preparation in detail and analyze it in the QHBC context. Our protocol is named for a sleight-of-hand trick using one pea, two players, and three walnut shells. Let the players be our familiar friends Alice and Bob. The game begins with Alice hiding the pea under one of the walnut shells in full view of Bob. Alice then rapidly shuffles the shells before allowing Bob to guess which shell the pea is under. Bob wins if he guesses correctly, and Alice wins if he fails. Alice has the best chance of winning (and the game is most interesting) when she has mastered the art of shuffling the shells in a sneaky or misleading way.

At a high level, our Shell Game protocol reflects its namesake, only using qubits and magic boxes instead of peas and walnut shells. We give two protocols, one permutation-only protocol for remotely preparing ℓ' $|0\rangle$ and $(\ell - \ell')$ $|+\rangle$ qubits (for some integer ℓ chosen by the client and random $\ell' \leq \ell$) and one permute-and-encode protocol for remotely preparing ℓ trap encoded qubits. The ideal functionalities these protocols are intended to implement are given in Figure 4.1 and Figure 4.2, respectively.

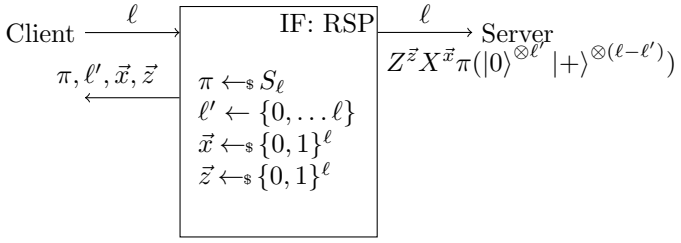


Figure 4.1: Ideal Functionality for Remote State Preparation (IF: RSP). This functionality prepares ℓ' $|0\rangle$ and $(\ell - \ell')$ $|+\rangle$ qubits, permuted and encrypted with a QOTP, on the server's workspace. ℓ' is sampled from a distribution with probability mass function $p(\ell') = \binom{\ell^2}{\ell'}\binom{\ell^2}{\ell-\ell'}/\binom{2\ell^2}{\ell}$.

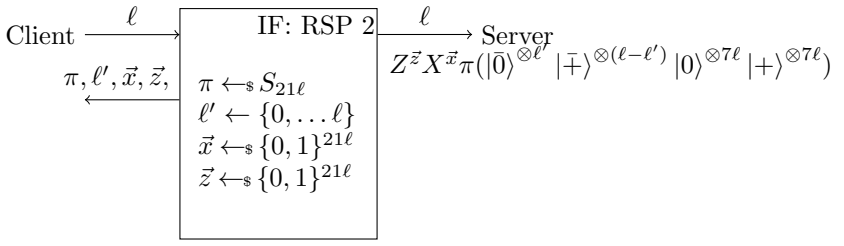


Figure 4.2: Ideal Functionality for Remote Authenticated State Preparation (IF: RSP_A). This functionality prepares ℓ' trap encoded $|0\rangle$ and $(\ell - \ell')$ $|+\rangle$ qubits encrypted with a QOTP on the server's workspace. Here $|\bar{\psi}\rangle$ represents a Steane logical qubit (encoded across 7 physical qubits) with data $|\psi\rangle$. ℓ' is sampled from a distribution with probability mass function $p(\ell') = \binom{21\ell^2}{\ell'}\binom{21\ell^2}{\ell-\ell'}/\binom{42\ell^2}{\ell}$.

Each protocol consists of a commitment phase, in which a client and server interact to prepare a particular quantum state. The server prepares an initial state of many $|+\rangle$ and $|0\rangle$ qubits (in particular, $|+\rangle^{\otimes k} |0\rangle^{\otimes k}$, for a value of k that we shall specify later), and the client and server shuffle the initial state using magic boxes. We call this the commitment phase because at its conclusion, the server is committed to holding a particular state (i.e., a particular permutation and encryption of the initial state) by information sent back to the client by magic boxes, from which the client computes the exponents of the Pauli padding an honest server's state should have. If we were only interested in QHBC servers, our protocol could stop here: At this point, the client can be sure the honest server is holding an encrypted quantum state, and the honest server does not know the keys to decrypt the state. Since we are ultimately interested in being able to authenticate states prepared by potentially more adversarial servers, we include a second phase during which the client tests the server.

During this test phase, the client asks the server to measure the entire prepared state except for ℓ of the qubits. (In permute-and-encode protocols, these are actually ℓ trap-encoded logical qubits, encoded as 21ℓ qubits.) Each qubit is measured in either the computational Z basis or the Hadamard X basis uniformly at random. After measuring all the qubits, the server reports the list of measurement outcomes to the client. The client assesses the consistency of the outcomes in the following sense: If a qubit committed in the previous phase to a particular basis state is measured in its corresponding basis, and the measurement outcome is not the expected one, then the outcome is inconsistent. For example, if the server committed the first qubit to be in the state $|1\rangle$, then it would be inconsistent for the outcome of a computational basis test measurement of the first qubit to be 0. (Note that no outcome is inconsistent if the committed basis and measurement basis are mismatched.) If any reported outcomes are inconsistent, the client rejects and aborts the protocol. Otherwise, the client accepts that it has remotely prepared a particular encrypted state of ℓ (logical) qubits. The number of rounds of communication required by this

protocol is constant and independent of the number of qubits produced. These qubits can be fed forward into some new computation or protocol that requires the secret preparation of qubits, such as a blind quantum computation (BQC) protocol.

The following chapter proceeds by describing in detail how the client can use magic boxes to secretly permute and encode a quantum state prepared by an honest server. We then present the Shell Game protocols in full and analyze their security in the QHBC setting, analogous to the case in the pea-and-walnut shell game in which Bob plays along dutifully but tries his best to learn the trick of the game. We assume clients and servers have access to a shared magic box resource, which may be implemented by secret-SWAP gadgets in this setting with computational assumptions. In chapter 5, we will begin addressing security against more active adversaries, although our analysis is by no means fully comprehensive.

4.1 Efficiently Generating Permutations

In order for our shell game to be viable, it must be possible for the client to direct the server to implement an arbitrary permutation of the initial state efficiently. The fact that an arbitrary permutation on n objects can be generated using polynomially many transpositions is implied by the fact that the Bubble Sort algorithm can sort any arbitrarily permuted list of n objects using $O(n^2)$ adjacent transpositions. In particular, we can use the list of $\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$ transpositions

$$L_{\text{perm}}(n) := [(1\ 2), \ldots, (1\ n), (2\ 3), \ldots, (2\ n), \ldots, ((n-1)\ n)]$$

to generate any permutation using secret-SWAPs. That is, first we either swap the first two elements or do not, then we either swap the first and third element or do not, and so on. This list allows us to code any permutation of n objects as a string of length $O(n^2)$.

Proposition 5. *Any permutation of n elements can be generated using at most n transpositions of the form $(i\ j)$ for $1 \leq i < j \leq n$, such that each integer in $[n]$ appears as the first index in a transposition at most once, and if for a pair of transpositions $(i\ j)$ and $(i'\ j')$ we have $i < i'$, then the transposition $(i\ j)$ occurs before the transposition $(i'\ j')$.*

Proof. By induction on the number of elements to permute.

Basis: For a single element, there is only one permutation, generated by performing no transpositions.

Induction: Suppose any permutation of $n-1$ elements can be generated as described in the proposition. For an n-element list, begin by either performing at most one transposition of the form $(1\ j)$ for $1 < j \leq n$. There are n unique ways to do this. Then perform an arbitrary permutation on the last $n-1$ elements (indices 2 to n), which by inductive hypothesis can be done using at

most $n-1$ transpositions of the form $(i\ j)$ for $2 \leq i < j \leq n$ such that each integer in $[2, n]$ appears as the first index in a transposition at most once, and where the transpositions have the right kind of order. There are $(n-1)!$ unique ways to do this. The overall process is of the form described in the proposition, and it generates $n \cdot (n-1)! = n!$ unique permutations, which exhausts all possibilities. □

We can start the commitment phase of the shell game by separating the state into a "block" of k $|+\rangle$ qubits followed by a block of k $|0\rangle$ qubits (i.e., $|+\rangle^{\oplus k} |0\rangle^{\oplus k}$). This is the initial state prepared by an honest server. For $1 \leq i \leq k$, secret SWAP the ith $|+\rangle$ with the ith $|0\rangle$. This corresponds to a list of possible "inter-block" transpositions of the form

$$L_{\text{block}}(k) := [(1\ \ (k+1)), (2\ \ (k+2)), \ldots, (k\ \ 2k)].$$

At this point, there is no qubit for which the server (or an adversary) knows the basis with certainty. Then apply a random permutation for $2k$ qubits by performing a secret SWAP for each transposition in the list $L_{\text{perm}}(2k)$ described above. In fact, we will actually run through $L_{\text{perm}}(2k)$ *four* times during the permute-and-encode variation of our Shell Game protocol instead of just once, since this will help with implementing a quantum error correcting code. (See subsection 4.1.2 for details.)

This ordered list of possible transpositions—one round of inter-block transpositions $L_{\text{block}}(k)$ followed by one or four rounds through $L_{\text{perm}}(2k)$—is public. It should be common knowledge among all parties and need not be communicated explicitly during a Shell Game protocol. Since the "wiring" of the magic boxes or gadgets (i.e., the number and ordering of secret-SWAPs in the protocol) is fixed by the size of the initial state, the client can begin the protocol by sending (1) a security parameter that determines the initial size to the client and (2) an "instruction" string to the magic box resource representing the permutation to be performed. We describe in precise detail how the client should generate these instruction strings in the following subsections before introducing the full Shell Game protocols in section 4.2.

4.1.1 Instruction Strings for Permutations Only

Consider a protocol in which the client attempts to permute but not encode the server's state. In these protocols, we will use our original Magic SWAP Boxes. We also use the public list $L_{\text{block}}(k) + L_{\text{perm}}(2k)$ (i.e., $L_{\text{block}}(k)$ concatenated with $L_{\text{perm}}(2k)$) for a value of k determined by the client's security parameter. The ith bit of the instruction string is the secret bit s used as the clients input to the Magic SWAP Box for the ith transposition in the public list. (Recall that the box SWAPs the qubits input by the server if and only if $s = 1$.)

The client begins by choosing a security parameter ℓ. This parameter determines the size of the honest server's quantum state, with the block size $k = \ell^2$. The first $|L_{\text{block}}(k)|$ bits of the instruction string, corresponding to the initial inter-block transpositions, are chosen uniformly at random. The inter-block

transpositions induce some permutation σ of the initial state. The client then chooses a permutation τ uniformly at random from S_{2k}. The client computes a decomposition of τ into transpositions of the form in Proposition 5. For the next (and last) $|L_{\text{perm}}(2k)|$ bits of the instruction string, the bit is set to 1 if and only if the corresponding transposition in the public list occurs in the decomposition of τ. [1] The permutations σ and τ induce a permutation $\pi = \tau \circ \sigma$. In our notation, we identify an instruction string with the permutation π it induces in the honest protocol. We denote the process described above for generating an instruction string given a security parameter $\texttt{Gen}_{\text{perm}}(\ell)$.

4.1.2 Instruction Strings for Permutations with Encoding

Consider a protocol in which the client attempts to simultaneously permute and encode the server's state using a trap code based on the Steane quantum error-correction code (see section 2.3 and section 2.4). In these protocols, we use modified Magic SWAP Boxes. We use the public list $L_{\text{block}}(k) + 4L_{\text{perm}}(2k)$ (i.e., $L_{\text{block}}(k)$ concatenated with four copies of $L_{\text{perm}}(2k)$.) Running through $L_{\text{perm}}(2k)$ four times helps us implement the trap encoding on a pre-selected subset of the server's qubits while permuting the entire state. For these protocols, the instruction string is not a string of secret bits, but rather a string of $|L_{\text{block}}(k)| + 4|L_{\text{perm}}(2k)|$ classical descriptions of unitaries in the set

$$\{I_i \otimes I_j, \text{SWAP}_{ij}, \text{CNOT}_{ij}, \text{CNOT}_{ji}\}.$$

(Each description could be, for example, a two-bit index into the set of possible unitaries.)

As in the permutation-only protocol, the client begins by choosing a security parameter ℓ. In the permute-and-encode protocol, an honest server will prepare a state with block size $k = 21\ell^2$. The client also randomly chooses 10ℓ distinct indicies from $[k]$ (corresponding to $|+\rangle$ qubits in the state prepared by an honest server), 10ℓ distinct indices from $[k+1, 2k]$ (corresponding to $|0\rangle$ qubits), and ℓ more distinct inidices from $[2k]$. The client labels the 21ℓ selected qubits as follows:

- Each of the ℓ indices chosen from $[2k]$ (labeled $d_1, \ldots, d_\ell$) will be data qubits encoded using a Steane encoding.
- Three $|0\rangle$ qubits (labeled $a_{q,1}, \ldots, a_{q,3}$, for $q \in [\ell]$) and three $|+\rangle$ qubits (labeled $a_{q,4}, \ldots a_{q,6}$) will be used as ancilla for Steane encoding each logical qubit.
- Seven $|0\rangle$ qubits (labeled $t_{q,1}, \ldots, t_{q,7}$) and seven $|+\rangle$ qubits (labeled $t_{q,8}$, $\ldots$, $t_{q,14}$) will be used as trap qubits for the trap encoding of each logical

[1] Instead of choosing the permutation τ uniformly at random, decomposing it into transpositions, and preparing instructions accordingly, the client could simply prepare an instruction with s chosen uniformly at random for each possible transposition in the public list. However, the permutation generated by such a list would not be uniformly random.

qubit.[2]

Call this assignment of labels $\mathcal{A}$. (The reader may find it helpful to refer back to the labels of Figure 2.1.)

Next, analogously to the permutation-only protocol, the first $|L_{\text{block}}(k)|$ unitaries are chosen uniformly at random from $\{I_i \otimes I_j, \text{SWAP}_{ij}\}$, thus inducing an inter-block permutation σ on $42\ell^2$ qubits. For the next $4|L_{\text{perm}}(2k)|$ unitaries, corresponding to four passes through the $L_{\text{perm}}(2k)$ list, the client chooses as follows:

First pass: The client chooses a permutation τ_1 uniformly at random from S_{2k}. The client computes a decomposition of τ_1 into transpositions of the form in Proposition 5. For the next $|L_{\text{perm}}(2k)|$ unitaries, the client chooses $SWAP_{ij}$ if the corresponding transposition in the public list occurs in the decomposition of τ and chooses $I_i \otimes I_j$ otherwise.

Second pass: For each $(i\ j)$ in $L_{\text{perm}}(2k)$, let $i' := \sigma^{-1}(\tau_1^{-1}(i))$ and $j' := \sigma^{-1}(\tau_1^{-1}(j))$. If i' and j' are both unlabeled, the client chooses the corresponding unitary in the instruction string to $SWAP_{ij}$ or $I_i \otimes I_j$ uniformly at random. This induces a permutation τ_2 on top of $\tau_1 \circ \sigma$. If i' and j' have the labels (in any order) d_q and $a_{q,1}$ or d_q and $a_{q,2}$, the client prepares an instruction according to $\text{CNOT}_{q_p a_{p,2}}$. This implements the first part of the Steane encoding, entangling each data qubit with two of its $|0\rangle$ ancilla. (Refer to Figure 2.1.) If i' and j' have any other combination of labels, the client chooses $I_i \otimes I_j$, leaving them in place.

Third pass: For each $(i\ j)$ in $L_{\text{perm}}(2k)$, let $i' := \sigma^{-1}(\tau_1^{-1}(\tau_2^{-1}(i)))$ and $j' := \sigma^{-1}(\tau_1^{-1}(\tau_2^{-1}(j)))$. If i' and j' are both unlabeled, the client chooses the corresponding unitary in the instruction string to be $SWAP_{ij}$ or $I_i \otimes I_j$ uniformly at random. This induces a permutation τ_3 on top of $\tau_2 \circ \tau_1 \circ \sigma$. If i' and j' have the labels (in any order)

- $a_{q,6}$ and (a_{q_3}, $a_{q,1}$, or d_q), or
- $a_{q,5}$ and ($a_{q,3}$, $a_{q,2}$, or d_q), or
- $a_{q,4}$ and ($a_{q,3}$, $a_{q,2}$, or $a_{q,1}$),

the client chooses the unitary $\text{CNOT}_{q_p a_{p,2}}$. This completes the Steane encoding of the pre-selected qubits; we shall refer to the encoding induced by the assignment of labels $\mathcal{A}$ as $E_{\mathcal{A}}$. If i' and j' have any other combination of labels, the client chooses the corresponding unitary in the instruction string to be $I_i \otimes I_j$, leaving them in place.

Fourth pass: The client chooses another permutation τ_4 uniformly at random from S_{2k} and chooses the $|L_{\text{perm}}(2k)|$ unitaries implementing it as in the first pass.

[2]The precise labels applied to traps are not actually important so long as 7ℓ $|0\rangle$ qubits and 7ℓ $|+\rangle$ qubits are chosen as traps. However, labels are necessary to keep track of which ancilla belong to which data qubit and what role each ancilla plays in the encoding circuit for its data.

This process chooses one unitary corresponding to each possible transposition in the public list $L_{\text{block}}(k) + 4L_{\text{perm}}(2k)$. It also induces an encoding $E_{\mathcal{A}}$ and a permutation

$$\pi = \tau_4 \circ \tau_3 \circ \tau_2 \circ \tau_1 \circ \sigma.$$

As in the permutation-only variation, we identify an instruction string with the permutation and encoding $\pi \circ E_{\mathcal{A}}$ it induces. We will denote the process for generating such an instruction string given a security parameter by $\texttt{Gen}_{\text{encode}}(\ell)$.

4.2 Protocol Descriptions

We are now ready to describe our protocols in full. The full details of the permute-only version of the protocol are given in Figure 4.3, and the permute-and-encode version are detailed in Figure 4.4.

Each protocol begins with a commitment phase, in which the client chooses a parameter ℓ determining the number of qubits to prepare and sends it to the server. The client generates a list of magic box instructions in the way described in the previous section.[3] The server prepares the quantum state $|+\rangle^{\otimes k} |0\rangle^{\otimes k}$, where k is determined by ℓ and the variation of the Shell Game being performed. The server uses the public list associated with the Shell Game variation to choose which qubits to input to the Magic SWAP Box in which order. This induces a secret permutation, encryption, and possibly encoding on the initial state, which the client can compute using their instructions and classical magic box output.[4]

After the commitment phase comes the test phase. In the permutation-only protocol, the client chooses a list M of $2\ell^2 - \ell$ qubits uniformly at random to be tested. In the permute-and-encode protocol, the client chooses $\bar{\mathcal{A}}$, the list of all qubits not assigned a label by $\mathcal{A}$ and therefore not used in the trap encoding induced by $\mathcal{A}$.[5] In either case, the list of qubits to be measured is sent to the server, together with a list of measurement bases for each qubit to be tested, with each basis chosen randomly from $\{X, Z\}$. The server performs the measurements and returns the list of outcomes to the client. The client aborts if the measurement outcomes are physically inconsistent with the honest committed state.

[3] We have assumed a magic box resource is available to the server and client throughout this chapter. It is worth noting that, if the boxes are implemented with secret gadgets, the client should send the instructions for all the secret gadgets along with ℓ in their first message to the server.

[4] If magic boxes are implemented with secret gadgets, the server should send a single message containing all gadget measurement outcomes to the client after following the instructions.

[5] Note that $\mathcal{A}$ picked out indices in the initial honest state, but $\bar{\mathcal{A}}$ should pick its indices from the honest *committed* state instead.

Shell Game Protocol 1: Permutation-Only Version

Client		**Server**
..... Commitment Phase		
1 : Choose ℓ		
2 : $\pi \leftarrow_{\$} \mathtt{Gen}_{\mathrm{perm}}(\ell)$		
3 : Input π to magic box		
4 :	$\ell \longrightarrow$	
5 :		Prepare $\lvert+\rangle^{\otimes \ell^2} \lvert 0\rangle^{\otimes \ell^2}$
6 :		**for** $(i\ j) \in [L_{\mathrm{block}}(\ell^2) + L_{\mathrm{perm}}(2\ell^2)]$ **do**
		Input qubits i, j to box
		Honest state: $Z^{\vec{z}} X^{\vec{x}} \pi(\lvert+\rangle^{\otimes \ell^2} \lvert 0\rangle^{\otimes \ell^2})$
7 : Get all box output		
8 : Compute $\vec{x}$ and $\vec{z}$ from box output		
..... Test Phase		
9 : $M \leftarrow_{\$} \mathcal{F}_M(\ell)$, random list of $2\ell^2 - \ell$ registers to measure		
10 : $\Theta \leftarrow_{\$} \{X, Z\}^{\lvert M\rvert}$		
11 :	$(M, \Theta) \longrightarrow$	
12 :		**for** $(m, \theta) \in \mathtt{zip}(M, \Theta)$ **do**
		Measure qubit m in basis θ
		for outcome o
		Call list of outcomes O
		Honest state: $\mathrm{Tr}_M[Z^{\vec{z}} X^{\vec{x}} \pi(\lvert+\rangle^{\otimes \ell^2} \lvert 0\rangle^{\otimes \ell^2})]$
13 :	$\longleftarrow O$	
14 : Assess consistency of O with respect to $Z^{\vec{z}} X^{\vec{x}} \pi(\lvert+\rangle^{\otimes \ell^2} \lvert 0\rangle^{\otimes \ell^2})$		
15 : **Accept** if O is consistent; Otherwise, **reject**		

Figure 4.3: Permutation-only Shell Game protocol to remotely prepare ℓ qubits. M is a list of $2\ell^2 - \ell$ distinct registers to measure. $\mathcal{F}_M(\ell)$ is the family of all such lists. The function `zip` takes two ordered lists $[x_i]_{i \in I}$ and $[y_i]_{i \in I}$ and returns an ordered list of pairs $[(x_i, y_i)]_{i \in I}$.

Shell Game Protocol 2: Permutate-and-Encode Version

Client		**Server**
..... Commitment Phase		
1 : Choose ℓ		
2 : $\pi \circ E_{\mathcal{A}} \leftarrow \mathtt{Gen}_{\text{encode}}(\ell)$		
3 : Input $\pi \circ E_{\mathcal{A}}$ to magic box		
4 :	$\ell \longrightarrow$	
5 :		Prepare $\lvert+\rangle^{\otimes 21\ell^2} \lvert 0\rangle^{\otimes 21\ell^2}$
6 :		**for** $(i\ j) \in [L_{\text{block}}(\ell^2) + 4L_{\text{perm}}(2\ell^2)]$ **do**
		Input qubits i, j to box
		Honest state: $Z^{\vec{z}} X^{\vec{x}} \pi(E_{\mathcal{A}}(\lvert+\rangle^{\otimes 21\ell^2} \lvert 0\rangle^{\otimes 21\ell^2}))$
7 : Get all box output		
8 : Compute $\vec{x}$ and $\vec{z}$ from box output		
..... Test Phase		
9 : Let $\bar{\mathcal{A}}$ be the list of $42\ell^2 - 21\ell$ indices of honest state unlabeled by $\mathcal{A}$		
10 : $\Theta \leftarrow_{\$} \{X, Z\}^{\lvert\bar{A}\rvert}$		
11 :	$(\bar{\mathcal{A}}, \Theta) \longrightarrow$	
12 :		**for** $(a, \theta) \in \mathtt{zip}(\bar{\mathcal{A}}, \Theta)$ **do**
		Measure qubit a in basis θ
		for outcome o
		Call list of outcomes O
		Honest state:
		$\mathrm{Tr}_{\bar{\mathcal{A}}}[Z^{\vec{z}} X^{\vec{x}} \pi(E_{\mathcal{A}}(\lvert+\rangle^{\otimes 21\ell^2} \lvert 0\rangle^{\otimes 21\ell^2}))]$
13 :	$\longleftarrow O$	
14 : Assess consistency of O W.R.T. $Z^{\vec{z}} X^{\vec{x}} \pi(E_{\mathcal{A}}(\lvert+\rangle^{\otimes 21\ell^2} \lvert 0\rangle^{\otimes 21\ell^2}))$		
15 : **Accept** if O is consistent; Otherwise, **reject**		

Figure 4.4: Shell Game protocol to prepare ℓ trap-encoded qubits. $\bar{\mathcal{A}}$ is a list of registers of the honest committed state containing qubits not used in the encoding $\mathcal{A}$.

4.3 Protocol Security: Quantum Honest-But-Curious Adversaries

We are now ready to formalize the correctness and security of the Shell Game protocols in the QHBC setting.

Theorem 6. *In the QHBC setting, Shell Game Protocol 1 (Figure 4.3) implements Ideal Functionality for Remote State Preparation(Figure 4.1), and Shell Game Protocol 2 (Figure 4.4) implements Ideal Functionality for Remote Authenticated State Preparation (Figure 4.2).*

Proof. In the QHBC setting, the theorem follows if the Shell Game protocols are correct, i.e. if (1) a server produces the state

$$\mathrm{Tr}_M[Z^{\vec{z}}X^{\vec{x}}\pi(|+\rangle^{\ell^2}|0\rangle^{\ell^2})]$$

by following a permute-only protocol honestly and produces the state

$$\mathrm{Tr}_{\bar{\mathcal{A}}}[Z^{\vec{z}}X^{\vec{x}}\pi(E_{\mathcal{A}}(|+\rangle^{\ell^2}|0\rangle^{\ell^2}))]$$

by following a permute-and-encode protocol, and (2) the client can compute $\vec{x}$ and $\vec{z}$ from the magic box output. Then we need only show that

$$\mathrm{Tr}_M[Z^{\vec{z}}X^{\vec{x}}\pi(|+\rangle^{\ell^2}|0\rangle^{\ell^2})] = Z^{\vec{z'}}X^{\vec{x'}}\pi'(|0\rangle^{\otimes\ell'}|+\rangle^{\otimes(\ell-\ell')})$$

for some $z' \leftarrow_{\$} \{0,1\}^{\ell}$, $x' \leftarrow_{\$} \{0,1\}^{\ell}$, $\pi' \leftarrow_{\$} S_{\ell}$, and $\ell' \leftarrow_{\$} \{0,\ldots,\ell\}$, and similarly

$$\mathrm{Tr}_{\bar{\mathcal{A}}}[Z^{\vec{z}}X^{\vec{x}}\pi(E_{\mathcal{A}}(|+\rangle^{\ell^2}|0\rangle^{\ell^2}))] = Z^{\vec{z'}}X^{\vec{x'}}\pi'(|\bar{0}\rangle^{\otimes\ell'}|\bar{+}\rangle^{\otimes(\ell-\ell')}|0\rangle^{\otimes 7\ell}|+\rangle^{\otimes 7\ell})$$

for some $z' \leftarrow_{\$} \{0,1\}^{21\ell}$, $x' \leftarrow_{\$} \{0,1\}^{21\ell}$, $\pi' \leftarrow_{\$} S_{21\ell}$, and $\ell' \leftarrow_{\$} \{0,\ldots,\ell\}$.

That the server really does produce the correct states follows from the construction of instruction strings in section 4.1. That the client can compute the Pauli exponents follows from the fact that the client chooses the unitary implemented by each use of the magic box and learns the QOTP applied by each use of the magic box.

We see that by construction both protocols leave the right kind of remaining state after test measurements: ℓ qubits in Protocol 1 and ℓ Steane-encoded qubits with 7ℓ trap $|0\rangle$ qubits and 7ℓ trap $|+\rangle$ qubits in Protocol 2. Note that in both protocols, each bit of $\vec{z}$ and $\vec{x}$ are generated uniformly random by the magic box. $\vec{z'}$ and $\vec{x'}$ are derived from $\vec{z}$ and $\vec{x}$ discarding the bits of $\vec{z}$ and $\vec{x}$ associated with qubits measured in the test phase. Since M and $\mathcal{A}$ are chosen uniformly at random, the subsets of measured qubits are also uniformly random and in particular have no dependence on $\vec{z}$ or $\vec{x}$. Thus each bit of $\vec{z'}$ and $\vec{x'}$ is also uniformly random in both protocols. Furthermore, since π, M, and $\mathcal{A}$ are chosen uniformly at random, π' is uniformly random, and ℓ' is sampled from the kind of distribution described in the ideal functionalities.

□

Theorem 7. *In the QHBC setting, the server has no advantage in guessing the Pauli keys $(\vec{x}, \vec{z})$ or the permutation π applied during either Shell Game protocol at the end of the commitment phase.*

Proof. By construction, each use of the magic box applies a QOTP with uniformly random exponents, and each qubit is entered into the box and is padded at least once. Thus $(\vec{x}, \vec{z})$ is uniformly random. The permutation π is also constructed as a composition of multiple permutations, at least one of which is chosen uniformly at random from S_{2k}. Thus π is also uniformly random. Thus the server has advantage in guessing the keys and permutation only if information about them is leaked by the protocol.

QHBC servers must prepare the correct initial state, must enter the correct pair of qubits into the magic box at each step, and may not perform any extraneous measurements unless they are certain (from the server's informational perspective) to be nondestructive. The inter-block secret transpositions performed at the beginning of each protocol ensure that the server never knows (and in fact has no advantage in guessing) the basis of either qubit output by the magic box at any step in the protocol. Thus the server never has an opportunity to perform a nondestructive measurement on box output in order to learn some of the exponents of the Pauli padding. Furthermore, since the box output is quantum one-time padded, the outcome of any measurement (nondestructive or otherwise) is uncorrelated with the unitary applied by the box. Thus no information about the keys or permutation leaks during the commitment phase. □

At the start of the test phase, the client reveals to the server which qubits will be measured and hence also which qubits will be untested (i.e., which qubits will be the quantum output of the protocol). This revelation may leak information about π to the server. For example, after learning $\bar{\mathcal{A}}$, the server knows that π maps at least 10ℓ of the first $21\ell^2$ qubits and 10ℓ of the last $21\ell^2$ qubits of the initial state outside of $\bar{\mathcal{A}}$. (However, since $\mathcal{A}$ was chosen at random, this is all the honest server learns from $\bar{\mathcal{A}}$.) The set of possible permutations is still large,[6] but it is no longer S_{2k}. Thus, at least for protocol 2, we cannot hope for the server to have no advantage in guessing $\pi \in S_{2k}$ after the test phase has begun. Instead, we show that the remaining state still looks very "mixed up" to the server in the sense that the server has no advantage in guessing π'.

Theorem 8. *In the QHBC setting, the server has no advantage in guessing the Pauli keys $(\vec{x}, \vec{z})$ or π' (the permutation of the honest output) at the end of the test phase of either protocol.*

Proof. By the previous theorem, nothing about the keys or permutation leak during the commitment phase. The measurements performed during the test phase are also not correlated with the QOTP keys, since the measurement basis is chosen uniformly at random and independently of the measured qubit.

[6] The cardinality is precisely $\binom{21\ell^2}{10\ell}^2 \cdot \binom{42\ell^2-20\ell}{\ell} \cdot 21\ell!$, although many permutations will map to equivalent quantum output.

In protocol 1, since M is chosen uniformly at random from all possible lists of $2\ell^2 - \ell$ distinct registers, it is in particular independent of π. Thus nothing about π is revealed when the server learns M, and so nothing is revealed about π' or ℓ' either.

In protocol 2, information about π *always* leaks when $\bar{\mathcal{A}}$ is revealed to the server in protocol 2. However, since π is uniformly random and $\mathcal{A}$ is chosen at random, nothing is revealed about π' without measuring part of the untested state.

Since the server is QHBC, the server can only measure an untested qubit when it knows the basis. The server only knows the basis of an untested qubit either when it knows all the untested qubits are in the computational basis or when it knows all the untested qubits are in the Hadamard basis. This is never possible in protocol 2 (for which the untested state must contain trap qubits in both bases), and it is also not possible in protocol 1 when M is chosen uniformly at random. Thus the server cannot measure untested qubits in either protocol.

Since the QHBC server has no advantage in guessing $(\vec{x}, \vec{z})$ or π at the beginning of the test phase, learns nothing about the Pauli keys from performing test measurements, and cannot measure untested qubits to distinguish between candidate permutations in S_ℓ or $S_{21\ell}$, the server has no advantage in guessing $(\vec{x}, \vec{z})$ or π' at the end of the test phase. □

Here we make a comment about choosing M and $\mathcal{A}$. Clients would probably prefer to choose M and $\mathcal{A}$ freely, since M and $\mathcal{A}$ determine the combination of (possibly encoded) $|0\rangle$ and $|+\rangle$ qubits prepared by the Shell Game protocols. Choosing M and $\mathcal{A}$ uniformly at random is not strictly necessary for the protocols to be secure in the QHBC setting, but we have enforced random selection anyway to prevent the client from choosing in particularly insecure ways. For example, in Protocol 1, if the client always chooses M in such a way that ℓ encrypted $|0\rangle$ qubits are prepared, then the protocol is totally insecure. After learning M, the server knows all untested qubits are in the computational basis, and so a QHBC server can nondestructively measure all untested qubits in the computational basis. In protocol 2, if $\mathcal{A}$ is chosen in some deterministic and public way, then an actively malicious server can freely manipulate the data qubits at the start of the protocol without being caught during the test phase. Furthermore, choosing M and $\mathcal{A}$ uniformly at random is not as big a drawback as it may initially appear. If the client needs a particular combination of $|0\rangle$ and $|+\rangle$ qubits, the client should just prepare more qubits than strictly necessary in order to obtain a probabilistic guarantee that a subset of them will be in the desired combination of bases.

Chapter 5

Active Security

In this chapter, we begin exploring the security of the shell game and its constituent parts (magic boxes, TCF-based secret gadgets, and trap code authentication schemes) against more actively malicious adversaries. Two observations are worth making immediately. The first is that in permute-only protocols, the client is only able to detect dishonest behavior by evaluating the consistency of test measurement outcomes. This means that an adversarial server could behave honestly until it learns (M, Θ) from the client. At that point, it knows precisely which qubits will be tested and which will remain unmeasured. The adversarial server can then perform an arbitrary attack that affects only unmeasured qubits. As long as the server performs and reports the test measurements honestly, the client will accept without detecting the server's deviation from the protocol. (In permute-and-encode protocols, the trap encoding on reserved qubits provides additional opportunities for authentication after the protocol has ended.)

We note, however, that a similar problem exists for competing remote state preparation protocols as well (and indeed even for our ideal remote state preparation functionalities): At the end of any protocol, when the server knows it will not be tested further, the server can always deviate arbitrarily with no immediate consequence. Since the qubits prepared by these protocols are typically intended to be fed forward into other protocols, it is possible that deviation during remote state preparation will be detected during later protocols if the prepared state differs too much from the honest state. In any case, we will focus our active security analysis on the relationship between the probability that the adversary passes the test phase and the "closeness" between the actual state held by the server at the end of the commitment phase and the honest state.

The second observation is that there exist attacks that pass the test phase with probability $1-\mathsf{poly}(\ell)$ such that the trace distance between the actual state held by the server at the end of the commitment phase and the honest state is 1. For example, the server can prepare the honest initial state and choose one qubit to flip (from $|0\rangle$ to $|1\rangle$ or from $|+\rangle$ to $|-\rangle$). The server then behaves honestly for the remainder of the protocol. At the end of the commitment phase, the

flipped qubit causes the trace distance between the actual and honest states to be 1. With probability $\frac{1}{2\ell}$, the flipped qubit is not selected for testing and so not detected. Ideally, we would like to guarantee that our protocol can detect attacks that cause the actual state held in the commitment phase to be so far from the honest state to be detected with all but negligible probability, but this is unfortunately not possible.

We begin our formal analysis of active security by returning to our magic box primitives from chapter 3. We explore a miniature version of the Shell Game test phase as a self-testing game played between a classical verifier (in the role of the client) and a quantum prover (in the role of the server) using magic boxes. We argue that the prover is unlikely to win these games unless the quantum state input to the box is "close" to the honest state. Next, we explore the ways in which secret-SWAP gadgets fail to implement Magic SWAP Boxes as a black box when the server is actively malicious. Finally, we address ways to strengthen the trap code authentication used in Shell Game protocol 2.

5.1 Magic QOTP Box Game

Recall the Magic QOTP Box first introduced in subsection 3.1.1. We reproduce Figure 3.1 depicting the Magic QOTP Box below, using slightly different notation to emphasize that the qubit entered into the box by the prover need not be in any pure state. We use the QOTP Box to play the following self-testing game between a verifier V and a prover P.

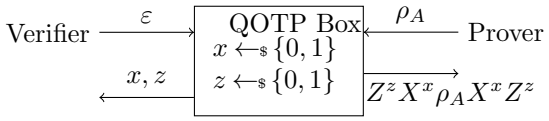


Figure 5.1: Magic QOTP Box functionality. The verifier's input ε is the empty string.

Magic QOTP Box Game

1 : V chooses whether she will ask P to guess x or z and notifies P of her choice.

2 : P prepares some quantum state ρ_{AB} of his choice.

3 : P inputs the single qubit $\rho_A = \mathrm{Tr}_B[\rho_{AB}]$ into the Magic QOTP Box.

4 : V and P receive their respective box output.

5 : P performs measurement of his choice.

6 : P guesses the exponent chosen by V and announces his guess to V.

7 : P wins if his guess is correct; else, V wins.

Theorem 9. *If a prover P wins the magic QOTP box guess-x game with probability $1-\epsilon$ using a state*

$$|\phi\rangle_{AB} = \sum_i \sqrt{\lambda_i}\, |e_i\rangle_A\, |f_i\rangle_B\,,$$

then there exists a diagonal density matrix

$$\sigma_A = p\,|0\rangle\langle 0| + (1-p)\,|1\rangle\langle 1|$$

with $0 \leq p \leq 1$ such that

$$T(\rho_A, \sigma_A) \leq \sqrt{\epsilon(1-\epsilon)},$$

where ρ_A is the reduced density matrix of $|\phi\rangle$, and $T(\cdot,\cdot)$ is the trace distance (see section 2.1).

Similarly, if P wins the guess-z game with probability $1-\epsilon$ using $|\phi\rangle$, there exists a density matrix

$$\sigma_A = p\,|+\rangle\langle +| + (1-p)\,|-\rangle\langle -|$$

with $0 \leq p \leq 1$ such that

$$T(\rho_A, \sigma_A) \leq \sqrt{\epsilon(1-\epsilon)}.$$

Proof. Begin by considering the guess-x game. Define the following density matrices, corresponding to the output of the magic box conditioned on the value of x:

$$\rho_{x=0} := \frac{1}{2}\sum_{i,j,z} \sqrt{\lambda_i\lambda_j} Z^z\, |e_i\rangle\langle e_j|_A\, Z^z \otimes |f_i\rangle\langle f_j|_B$$

and

$$\rho_{x=1} := \frac{1}{2}\sum_{i,j,z} \sqrt{\lambda_i\lambda_j} Z^z X\, |e_i\rangle\langle e_j|_A\, X Z^z \otimes |f_i\rangle\langle f_j|_B$$

From the prover's informational perspective, the box outputs $\rho_{x=0}$ or $\rho_{x=1}$ with probability 1/2 each, and the prover's ability to win the game depends on his ability to distinguish between these two states. The probability of distinguishing them is bounded above by $\frac{1}{2}(1+T(\rho_{x=0},\rho_{x=1}))$. Recalling that $T(\rho_{x=0},\rho_{x=1}) = \frac{1}{2}||\rho_{x=0}-\rho_{x=1}||_1$, we have

$$1-\epsilon \leq \frac{1}{2}(1+T(\rho_{x=0},\rho_{x=1})) = \frac{1}{2}(1+\frac{1}{2}||\rho_{x=0}-\rho_{x=1}||_1) = \frac{1}{2}+\frac{1}{4}\sum_{i=1}^{4}|\xi_i|$$

where $\{\xi_i\}_{i\in[4]}$ is the set of eigenvalues of $\rho_{x=0}-\rho_{x=1}$. The above inequality implies (by algebraic manipulation)

$$1-2\epsilon \leq \frac{1}{2}\sum_{i=1}^{4}|\xi_i|,$$

which we shall find more convenient to work with.

Fix the Schmidt decomposition of the prover's quantum state $|\phi\rangle_{AB}$ as $|e_0\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$ and $|e_1\rangle = e^{i\phi} \begin{pmatrix} b^* \\ -a^* \end{pmatrix}$ (for arbitrary complex amplitudes a and b such that $|a|^2 + |b|^2 = 1$ and $0 \leq \phi < 2\pi$) with $|f_0\rangle = |0\rangle$ and $|f_1\rangle = |1\rangle$.[1] Observe that the $\{|e_i\rangle\}$ basis is completely general. Since trace distance is invariant under unitary transformations, there is no harm in fixing the $\{|f_i\rangle\}$ basis as the computational basis.

With this choice of bases, the four eigenvalues of $\rho_{x=0} - \rho_{x=1}$ are of the form[2]

$$(\pm(|a|^2 - |b|^2)(\lambda_0 - \lambda_1) \pm \sqrt{1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2})/2.$$

Note that

$$(|a|^2 - |b|^2)(\lambda_0 - \lambda_1) \leq \sqrt{1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2},$$

so we can identify two eigenvalues that are necessarily nonnegative and two that are necessarily nonpositive:

$$(\pm(|a|^2 - |b|^2)(\lambda_0 - \lambda_1) + \sqrt{1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2})/2 \geq 0$$

and

$$(\pm(|a|^2 - |b|^2)(\lambda_0 - \lambda_1) - \sqrt{1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2})/2 \leq 0$$

Then

$$1 - 2\epsilon \leq \frac{1}{2} \sum_{i=1}^{4} |\xi_i| = \sqrt{1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2}.$$

Squaring both (nonnegative) sides of the inequality gives

$$1 - 4\epsilon + 4\epsilon^2 \leq 1 - 4|a|^2|b|^2(\lambda_0 - \lambda_1)^2,$$

which implies by basic algebra

$$\epsilon(1 - \epsilon) \geq |a|^2|b|^2(\lambda_0 - \lambda_1)^2.$$

Now consider the reduced density matrix of the subsystem the prover inputs to the magic box

$$\rho_A = \lambda_0 |e_0\rangle \langle e_0| + \lambda_1 |e_1\rangle \langle e_1| = \begin{pmatrix} \lambda_0|a|^2 + \lambda_1|b|^2 & (\lambda_0 - \lambda_1)ab^* \\ (\lambda_0 - \lambda_1)a^*b & \lambda_0|b|^2 + \lambda_1|a|^2 \end{pmatrix}.$$

Now define

$$\sigma_A := (\lambda_0|a|^2 + \lambda_1|b|^2) |0\rangle \langle 0| + (\lambda_0|b|^2 + \lambda_1|a|^2) |1\rangle \langle 1|.$$

[1] More precisely, if the B subsystem consists of n qubits, choose $|f_0\rangle = |0^n\rangle$ and $|f_1\rangle = |10^{n-1}\rangle$. The nonzero eigenvalues are the same as in the 1-qubit case.

[2] See the Mathematica notebook `approx-qotp-game.nb` on the GitHub repository https://github.com/huntermcknight/mol-thesis.

Taking the 2-norm of $\rho_A - \sigma_A$, we find

$$||\rho_A - \sigma_A||_2 = \sqrt{2|a|^2|b|^2(\lambda_0 - \lambda_1)^2} \leq \sqrt{2\epsilon(1-\epsilon)}.$$

By Cauchy-Schwartz,

$$||\rho_A - \sigma_A||_1 \leq \sqrt{2}||\rho_A - \sigma_A||_2 \leq 2\sqrt{\epsilon(1-\epsilon)},$$

and so

$$T(\rho_A, \sigma_A) = \frac{1}{2}||\rho_A - \sigma_A||_1 \leq \sqrt{\epsilon(1-\epsilon)}.$$

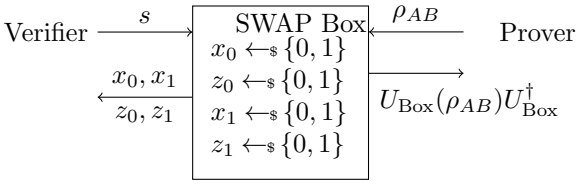
The argument for the guess-z game is analogous. □

Setting $\epsilon = 0$ in the above theorem leads to the following corollary.

Corollary 10. *A prover P wins the Magic QOTP Box guess-x game with probability 1 only if the qubit input to the box is a mixture of computational basis states; likewise, P wins the guess-z game with probability 1 only if the qubit input to the box is a mixture of Hadamard basis states.*

5.2 Magic Swap Box Game

Now recall our Magic SWAP Box, reprinted for convenience in Figure 5.2. The prover and the verifier play the following game with the box.



$$U_{\text{Box}} = (Z^{z_0}X_A^{x_0} \otimes Z^{z_1}X_B^{x_1})\text{SWAP}_{AB}^s$$

Figure 5.2: Magic SWAP Box functionality.

Magic SWAP Box Game

1 : V inputs $s \in \{0,1\}$ to Magic SWAP Box.
2 : V asks P to prepare and input $|0\rangle_A |+\rangle_B$ to the Box.
3 : P prepares a state ρ_{ABC} of his choice and inputs it.
4 : V samples observables $\theta_1, \theta_2 \leftarrow_\$ \{X, Z\}^2$
5 : V asks P to measure θ_0 of A and θ_1 of B.
6 : P performs measurements of his choice and reports outcomes to V.
7 : P wins if outcomes are consistent with honest measurements of $U_{\text{Box}}|0\rangle|+\rangle$.

Theorem 11. *If a prover P wins the magic SWAP box game with probability 1, then the A subsystem of the quantum input must be a mixture of computational basis states (i.e., its reduced density matrix is $\rho_A = p\,|0\rangle\langle 0| + (1-p)\,|1\rangle\langle 1|$ for some $0 \leq p \leq 1$), and the B subsystem must be a mixture of Hadamard basis states (i.e., $\rho_B = q\,|+\rangle\langle +| + (1-q)\,|-\rangle\langle -|$ for some $0 \leq q \leq 1$).*

Proof. If P wins the game with certainty, then in particular P wins with certainty whenever $s = 0$, $\theta_0 = Z$, and $\theta_1 = X$. Winning in this case requires guessing x_0 and z_1. Thus any strategy (i.e., choice of input and choice of measurement) that succeeds in this case would also succeed in playing a guess-x magic QOTP box game on the A subsystem in parallel with a guess-z magic QOTP box game on the B subsystem. The result then follows from Corollary 9. □

If we want to use magic boxes to build protocols for remote state preparation, then we need to be able to say something about the quantum state that remains if we only measure some of the box output. Consider a variation on the above magic SWAP box game. In this variation, everything proceeds as before until both players receive their output from the box. Then, the verifier samples an observable $\theta \leftarrow_{\$} \{X, Z\}$ and asks the prover to measure the A subsystem in the θ basis. The prover performs some (not necessarily honest) measurement and reports the outcome to the verifier. If the outcome is physically consistent with performing the honest θ measurement on the A subsystem of

$$(Z^{z_0}X_A^{x_0} \otimes Z^{z_1}X_B^{x_1})\mathrm{SWAP}_{AB}^{s}\,|0\rangle_A\,|+\rangle_B\,,$$

then the verifier accepts and the prover wins the game. The following theorem gives a guarantee on the state of the B subsystem at the moment it is output by the box.

Theorem 12. *If a prover P wins the variant magic SWAP box game with probability 1, then the B subsystem output by the box is a mixture of Hadamard basis states if $s = 0$ or else a mixture of computational basis states if $s = 1$.*

Proof. The crucial fact is that the prover does not know s, so the prover's perfectly winning strategy must be independent of s. If the prover wins with probability 1, then in particular he wins when $s = 0$ and $\theta = Z$, and so by the argument in the proof theorem 11, the A subsystem must be in a mixture of Z-axis basis states when it is input to the box. Similarly, the prover must always win when $s = 1$ and $\theta = X$, which requires the prover to guess z_0. A strategy that wins in this case will also win the guess-z magic QOTP box game played on the B subsystem, and so by theorem 9, the B system must be in a mixture of X-axis basis states when it is input into the box. Thus, if $s = 0$, the subsystems are not swapped by the box, and the output B subsystem remains in a mixture of X-axis basis states. When $s = 1$, the box swaps the A and B subsystems, and the B subsystem output is in a mixture of Z-axis basis states. □

5.3 Secret Gadgets with Active Adversaries

We have already mentioned that a secret-SWAP gadget fails to implement a Magic SWAP Box against active adversaries in the sense that a gadget has more points of failure. For a magic box, the prover chooses only the input to the box and nothing more; for a secret gadget, there are many steps at which an adversarial prover could deviate from the honest implementation. In this section, we discuss a few of these possible deviations and their security implications.

5.3.1 Fabricated Gadget Measurement Outcomes

A malicious prover can apply a secret-CNOT gadget honestly but report fabricated outcomes $\hat{y}$ and $\hat{d}$ to the verifier. This may cause the verifier to compute the exponents of the Pauli padding on the affected qubits incorrectly, but (under the security assumptions of TCFs) confers no advantage to the prover in computing the exponents of the padding induced by either the honest outcomes or the fabricated ones.[3] We can model this attack as the prover performing the gadget honestly and reporting the honest outcomes, but then applying a probabilistic mixture of Pauli attacks. In particular, the prover applies Z to the control qubit with probabiliy $1/2$ and applies X to the control qubit with probability $1/2$. If the server deviates in this way many times, the overall effect of the attack can be modelled as some probabilistic mixture of Pauli attacks on the committed state.

If the total weight of a Pauli attack is w, it must apply X to at least $w/2$ qubits or Z to at least $w/2$ qubits. In Protocol 1 (Figure 4.3), a single X applied to a random qubit of the commited state escapes detection whenever it is not applied to a computational basis qubit selected to be tested in the computational basis. This happens with probability $\frac{3}{4} + \frac{1}{8\ell}$. We can thus upper-bound the probability that an attack applying X to at least $w/2$ qubits is undetected during the test phase by $(\frac{3}{4}+\frac{1}{8\ell})^{w/2}$; the analysis for Z is analogous. In Protocol 2 (Figure 4.4), a pauli attack must have weight at least 3 to have a nontrivial effect on Steane-encoded data.

The prover can also report fabricated outcomes to trick the verifier into thinking a secret-CNOT gadget has been applied when the server has actually done nothing at all. If a prover fabricates all gadget measurement outcomes during a Shell Game protocol without ever actually applying a gadget, then the server will also have to fabricate test measurement outcomes. This kind of deviation is detected during the test phase with probability $1-(1/4)^T$, where T is the number of test measurements performed ($2\ell^2-\ell$ in Protocol 1 or $42\ell^2-21\ell$ in Protocol 2).

[3] Strictly speaking, this is only true if the verifier aborts when the prover reports that d is the all-zero string too often. If the prover is allowed to always (falsely) report that d is all-zero, then the prover knows the Z exponent induced by the fabricated outcome is always 0.

5.3.2 Re-Running a Gadget

A prover cannot re-use a magic box without the verifier's knowledge, since the box outputs are sent to the verifier from the box itself. However, it is possible to use a public key to re-run secret-CNOT gadgets an arbitrary number of times. That is, once a prover knows how to implement $\hat{U}_{\mathsf{Enc}(s)}$, a prover can use this unitary to run the secret-CNOT gadget induced by that encryption of s as many times as they want later in the protocol.[4] For example, if a prover is instructed to use $\hat{U}_{\mathsf{Enc}(s)}$ to run a secret-CNOT gadget on a pair of qubits, the prover could instead run the gadget on the pair twice (reporting the outcomes only once) to undo the secret-CNOT and change the Pauli padding. Although it is not immediately obvious how an adversary could gain advantage in the Shell Game by doing this, the fact that it is possible prevents us from treating secret-SWAP gadgets as a black box directly.

5.3.3 Implementing Other Secret Unitaries

At a certain point in the honest implementation of a secret-CNOT gadget (see subsection 3.2.1), the prover holds the state

$$\sum_{a,b} \alpha_{ab} \left|a\right\rangle_A \left|b\right\rangle_B \left|\mu_a, r_a\right\rangle_X .$$

The honest prover then performs a CNOT controlled by the first bit of the X register (call it the M register) and targeting the B register. The promise that $\mu_0 \oplus \mu_1 = s$ ensures that this is equivalent to performing $I_A \otimes X_B^{\mu_0} \mathrm{CNOT}_{AB}^{s}$. However, if a prover is dishonest, they could implement some other secret unitary at this step. For example, by performing a cZ_{MB} (controlled Z) instead of CNOT_{MB}, the prover would effectively implement the unitary $I_A \otimes Z_B^{\mu_0} cZ_{AB}^{s}$. More generally, for a Hermitian single-qubit unitary U with an associated two-qubit controlled unitary cU, a prover can use cU_{MB} instead of CNOT_{MB} to produce the state

$$Z_A^{\bar{d}} \otimes U_B^{\mu_0} cU_{AB}^{s} \left|\phi\right\rangle_{AB}$$

on an initial state $\left|\phi\right\rangle_{AB}$. These secret-U gadgets are interesting and potentially useful in cryptographic protocols; for example, one could implement a secret basis-flip on the qubit in the B register by using an ancillary $|1\rangle$ in register A and applying a controlled Hadamard cH at the crucial step.[5] However, the ability to implement secret unitaries other than CNOT using the gadget further complicates the analysis of Magic SWAP Boxes built from secret-SWAP gadgets.

[4] In a Shell Game implemented using secret-SWAP gadgets for magic boxes, the verifier would sample a new TCF pair for each secret-CNOT.

[5] This "secret" basis-flip isn't quite secret as described, since a QHBC prover knows that the M register contains the pure state $|\mu_1\rangle$ and can just measure it to determine whether the flip will be performed. Some refinement is required.

5.4 More Secure Authentication Schemes

In the Shell Game as explained in chapter 4, the Steane code was chosen for use in Protocol 2 mostly as a proof of concept. After all, authenticated output is not of high value in a world without noise, where all adversaries are merely too curious but not particularly malicious. However, against active adversaries, authentication is vital, and security of a trap code based on the Steane code with distance $d = 3$ may not be sufficient for all purposes. Thankfully, we can achieve arbitrary security against Pauli attacks by concatenating the Steane code with itself many times. Since the Steane code can be implemented using only (secret) CNOTs (given enough extra $|0\rangle$ and $|+\rangle$ qubits), so can any concatenated Steane code. The only extra requirements to concatenate the Steane code c times are a larger initial quantum state $|+\rangle^{\otimes 3 \cdot 7^c \ell^2} |0\rangle^{\otimes 3 \cdot 7^c \ell^2}$ and $2c$ additional rounds of secret transpositions (2 rounds for each Steane encoding circuit).

Chapter 6

Conclusion and Future Work

We have presented a novel protocol for verifiable remote state preparation in the Shell Game. We have demonstrated that in the QHBC setting the magic box resources necessary to preform the Shell Game protocol can be implemented from TCFs, and we have proved the security of the protocol in this setting. The Shell Game distinguishes itself from state-of-the-art protocols for verifiable remote state preparation QFactory and BRSP in that the Shell Game requires only constantly many rounds of communication to prepare an arbitrary number of qubits and can optionally prepare authenticated qubits. However, in order to fairly compare the Shell Game with QFactory and BSRP—indeed, for the Shell Game to be of practical use for remote state preparation—it is necessary to demonstrate security of the Shell Game against active adversaries. This thesis has unfortunately not accomplished this task, although it has begun addressing it. Two major gaps remain: First, we must explicitly relate the magic box games presented in chapter 5 to the test phase of the Shell Game in order to derive a guarantees about the state prepared by an adversarial server in the commitment phase. Second, we must find a way to implement magic boxes in the active security setting, or else replace magic boxes in the Shell Game protocol with some other resource that can be implemented in that setting. Future research on the Shell Game must address these gaps.

Other improvements to the Shell Game can also be addressed in future work. The Shell Game as presented in this thesis is not particularly fault-tolerant, in the sense that a single physically inconsistent measurement outcome will always cause the client to abort during the test phase. One possible solution is to change the test protocol to more closely resemble the blind self-testing introduced with QFactory [CCKW19], which evaluates test measurement outcomes in terms of something like physical "plausibility" (within some tolerance) than physical consistency. The disadvantage in this kind of testing is that its security is also conjectural at this time.

An additional improvement would be to enlarge the set of states that can be randomly prepared by the Shell Game. The Shell Game protocols prepare random BB84 states. BB84 states are useful for many quantum cryptographic protocols, but they aren't sufficient for verifiable blind quantum computation (i.e., the original motivation for developing verifiable remote state preparation). Thus future work with the Shell Game should explicitly address secure preparation of a suitable set of states, e.g., those used in the Fitzsimons and Kashefi BQC protocol [FK17].

Bibliography

[ABOEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive Proofs for Quantum Computations. *arXiv:1704.04487 [quant-ph]*, April 2017. arXiv: 1704.04487.

[ACGK17] Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. On the implausibility of classical client blind quantum computing. *arXiv:1704.08482 [quant-ph]*, April 2017. arXiv: 1704.08482.

[ADSS17] Gorjan Alagic, Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum Fully Homomorphic Encryption With Verification. *arXiv:1708.09156 [quant-ph]*, 10624:438–467, 2017. arXiv: 1708.09156.

[ALMO08] Andris Ambainis, Debbie Leung, Laura Mancinska, and Maris Ozols. Quantum Random Access Codes with Shared Randomness. *arXiv:0810.2937 [quant-ph]*, October 2008. arXiv: 0810.2937.

[AS06] Pablo Arrighi and Louis Salvail. Blind Quantum Computation. *International Journal of Quantum Information*, 04(05):883–898, October 2006.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. A Cryptographic Test of Quantumness and Certifiable Randomness from a Single Quantum Device. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 320–331, Paris, October 2018. IEEE.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526, October 2009. arXiv: 0807.4154.

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. *arXiv:1211.1080 [quant-ph]*, 8043:344–360, 2013. arXiv: 1211.1080.

[BJ15] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. *arXiv:1412.8766 [quant-ph]*, 9216:609–629, 2015. arXiv: 1412.8766.

[Bra18] Zvika Brakerski. Quantum FHE (Almost) As Secure As Classical. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10993, pages 67–95. Springer International Publishing, Cham, 2018.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum Complexity Theory. *SIAM Journal on Computing*, 26(5):1411–1473, October 1997.

[CCKW18] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. On the possibility of classical client blind quantum computing. *arXiv:1802.08759 [quant-ph]*, February 2018. arXiv: 1802.08759.

[CCKW19] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. QFactory: classically-instructed remote secret qubits preparation. *arXiv:1904.06303 [quant-ph]*, April 2019. arXiv: 1904.06303.

[Chi05] Andrew Childs. Secure assisted quantum computation. *Quantum Information and Computation*, 5(6):456–466, September 2005.

[DK16] Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states. *arXiv:1604.01586 [quant-ph]*, April 2016. arXiv: 1604.01586.

[DKL12] Vedran Dunjko, Elham Kashefi, and Anthony Leverrier. Universal Blind Quantum Computing with Weak Coherent Pulses. *Physical Review Letters*, 108(20):200502, May 2012. arXiv: 1108.5571.

[DNS10] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. *arXiv:1009.2096 [quant-ph]*, 6223:685–706, 2010. arXiv: 1009.2096.

[DS18] Yfke Dulek and Florian Speelman. Quantum ciphertext authentication and key recycling with the trap code. *arXiv:1804.02237 [quant-ph]*, April 2018. arXiv: 1804.02237.

[dW19] Ronald de Wolf. Quantum Computing: Lecture Notes. *arXiv:1907.09415 [quant-ph]*, July 2019. arXiv: 1907.09415.

[FK17] Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind computation. *Physical Review A*, 96(1):012303, July 2017. arXiv: 1203.5217.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing - STOC '89*, pages 25–32, Seattle, Washington, United States, 1989. ACM Press.

[GMMR13] Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G. Rudolph. Efficient universal blind computation. *Physical Review Letters*, 111(23):230501, December 2013. arXiv: 1306.2724.

[GV19] Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. *arXiv:1904.06320 [quant-ph]*, April 2019. arXiv: 1904.06320.

[HM15] Masahito Hayashi and Tomoyuki Morimae. Verifiable measurement-only blind quantum computing with stabilizer testing. *Physical Review Letters*, 115(22):220502, November 2015. arXiv: 1505.07535.

[Lia15] Min Liang. Quantum fully homomorphic encryption scheme based on universal quantum circuit. *Quantum Information Processing*, 14(8):2749–2759, August 2015. arXiv: 1410.2435.

[Mah18a] Urmila Mahadev. Classical Homomorphic Encryption for Quantum Circuits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–338, Paris, October 2018. IEEE.

[Mah18b] Urmila Mahadev. Classical Verification of Quantum Computations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–267, Paris, October 2018. IEEE.

[MDK10] Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground state blind quantum computation on AKLT state. *arXiv:1009.3486 [cond-mat, physics:quant-ph]*, September 2010. arXiv: 1009.3486.

[MF12] Tomoyuki Morimae and Keisuke Fujii. Blind topological measurement-based quantum computation. *Nature Communications*, 3(1), January 2012.

[MPDF13] Atul Mantri, Carlos A. Perez-Delgado, and Joseph F. Fitzsimons. Optimal Blind Quantum Computation. *Physical Review Letters*, 111(23):230502, December 2013. arXiv: 1306.3677.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, page 187, Victoria, British Columbia, Canada, 2008. ACM Press.

[Reg05] 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*, page 84, Baltimore, MD, USA, 2005. ACM Press.

[Ste96] Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, November 1996.