On The Round Complexity of Two-Party Quantum Computation

James Bartusek[*] Andrea Coladangelo[†] Dakshita Khurana[‡] Fermi Ma[§]

Abstract

We investigate the round complexity of maliciously-secure two-party quantum computation (2PQC) with setup, and obtain the following results:

- A three-message protocol (two-message if only one party receives output) in the common random string (CRS) model assuming classical two-message oblivious transfer (OT) with post-quantum malicious security. This round complexity is optimal for the sequential communication setting. Under the additional assumption of reusable malicious designated-verifier non-interactive zero-knowledge (MDV-NIZK) arguments for NP, our techniques give an MDV-NIZK for QMA. Each of the assumptions mentioned above is known from the quantum hardness of learning with errors (QLWE).
- A protocol with two simultaneous rounds of communication, in a quantum preprocessing model, assuming sub-exponential QLWE. In fact, we construct a three-round protocol in the CRS model with only two rounds of *online* communication, which implies the above result. Along the way, we develop a new delayed simulation technique that we call "simulation via teleportation," which may be useful in other settings.

In addition, we perform a preliminary investigation into barriers and possible approaches for two-round 2PQC in the CRS model. We provide evidence that protocols admitting a natural class of simulators do not exist, and also give a proof-of-concept construction from a strong form of quantum virtual black-box (VBB) obfuscation.

Prior to our work, maliciously-secure 2PQC required round complexity linear in the size of the quantum circuit.

[*] UC Berkeley. Email: bartusek.james@gmail.com

[†] UC Berkeley. Email: andrea.coladangelo@gmail.com

[‡] UIUC. Email: dakshita@illinois.edu

[§] Princeton University and NTT Research. Email: fermima@alum.mit.edu

Contents

1 Introduction

Secure computation is a cornerstone of modern cryptography. It allows mutually distrusting parties to compute arbitrary functions on their private inputs, revealing only the outputs of the computation while hiding all other private information [Yao86, GMW87, BGW88, CCD88].

With the emergence of quantum computers, it becomes important to understand the landscape of secure *quantum* computation over distributed, private quantum (or classical) states. Specifically, we focus on the two party setting, where Alice and Bob hold (possibly entangled) quantum inputs $\mathbf{x}_A$ and $\mathbf{x}_B$ respectively, and would like to evaluate a quantum circuit Q on their joint input $(\mathbf{x}_A, \mathbf{x}_B)$. The output is of the form $Q(\mathbf{x}_A, \mathbf{x}_B) = (\mathbf{y}_A, \mathbf{y}_B)$, so at the end of the protocol Alice and Bob hold the (possibly entangled) output states $\mathbf{y}_A$ and $\mathbf{y}_B$ respectively. We ask the following natural question:

How many rounds of interaction are necessary for general-purpose two-party quantum computation?

Our work studies this question in the setting of *malicious* attackers. We would like to ensure that a malicious Alice (resp. Bob) who may arbitrarily deviate from the protocol specifications (1) can only hold information that is efficiently computable from either her input or output at any point during the protocol execution, and (2) cannot cause Bob (resp. Alice) to obtain an incorrect outcome of the protocol without being detected. These guarantees are formalized via a simulation-based security notion, which requires that no adversary can recover any information in the real world that it cannot recover in an ideal world where it simply hands its input to a trusted party who then returns the output.

The problem of secure quantum computation on distributed quantum states has a strong tradition in the quantum cryptography literature. It was first studied by [CGS02, BCG$^+$06], who obtained unconditional maliciously-secure general *multi-party* quantum computation with honest majority. The setting where half (or more) of the players are malicious requires computational assumptions due to the impossibility of unconditionally secure quantum bit commitment [May97, LC98, DSWK06]. In the computational setting, [DNS10] gave a two-party quantum computation (2PQC) protocol secure against the quantum analogue of semi-honest adversaries (specious adversaries); this was later extended to the malicious setting by [DNS12]. A recent work of [DGJ$^+$20] constructed maliciously-secure general multi-party quantum computation with *dishonest* majority from any maliciously-secure post-quantum classical MPC. *Importantly, all of the above protocols have round complexity polynomial in the size of the quantum circuit.*

In this work, we show that various flavors of maliciously-secure two-party quantum computation are possible in *two* or *three* rounds, in a setting where parties have access to a common random string (CRS). Along the way, we give a two-message protocol in the setting where only one party receives output — a quantum analogue of Yao's celebrated two-party computation protocol [Yao86] in the malicious setting. Additionally, we study barriers to achieving two-round secure computation where both parties obtain output, which are inherent to the quantum setting.[1]

1.1 Our Results

We consider the setting of two-party quantum computation where parties have access to a trusted setup, like a common random string (CRS). We study two possible models of interaction: (1) the sequential messages model, where only one party speaks in each round (i.e. Bob sends a message to Alice, in the next step Alice sends a message to Bob, and so forth), and (2) the simultaneous messages model, where in each round both parties simultaneously send messages to each other (i.e. Bob's round i message does not depend on Alice's round i message, and vice versa).

Recently, Brakerski and Yuen [BY20] introduced and constructed quantum garbled circuits. As an application, they describe a general-purpose three-message 2PQC and conjecture its security against *specious* (a quantum analogue of semi-honest) adversaries (a formal proof of security was outside the scope of their work). The starting point of our work is a garbling technique sketched in [BY20, §2.5], which the authors present as a simpler alternative to their main quantum garbled circuit construction. While this alternative

[1] Indeed, two-round protocols are known from minimal assumptions in the classical setting.

construction sacrifices some of the efficiency guarantees of the main construction in [BY20], it enables *classical* garbling of quantum circuits. This feature turns out to be crucial for our constructions of maliciously-secure 2PQC.

First Result: A Round-Optimal Protocol in the Sequential Messages Model. Our first result is in the setting of sequential messages, where we obtain a 3-message protocol for two-party quantum computation, assuming post-quantum OT.

Theorem 1.1. *(Informal) There exists a 3-message (resp. 2-message) protocol for two-party quantum computation in the CRS model that delivers an output to both parties (resp. one party). This protocol satisfies simulation-based security against malicious adversaries assuming the existence of post-quantum maliciously-secure two-message oblivious transfer with straight-line simulation in the CRS model (which is known from the quantum hardness of learning with errors (QLWE)).*

We point out that any protocol for two-party quantum computation (in the specious or malicious setting) requires at least three messages of sequential communication when both parties obtain the output, and two messages of sequential communication when only one party obtains the output. Roughly speaking, this is because it takes at least one round for Bob to share his input (properly encoded, for secrecy) with Alice. In the next round, Alice computes on this encrypted input and returns an output to Bob. Crucially, at this point, Alice cannot learn the output herself because she only obtained an encryption of Bob's input. Thus, Bob must send another message to Alice in order for her to obtain her output, resulting in a three round protocol. Therefore, our first result is optimal w.r.t. round complexity in the sequential message setting.

We note that (two-message) secure computation of general two-party quantum functionalities where exactly one party obtains output implies (two-message) zero-knowledge arguments for QMA as a special case. Recall that zero-knowledge arguments for QMA allow a prover to convince a verifier of the validity of a QMA statement while revealing no additional information about the quantum witness. An important goal in the study of zero-knowledge protocols is to minimize interaction; while post-quantum non-interactive zero-knowledge (NIZK) arguments for NP are known in the CRS model [CCH$^+$19, PS19], the analogous task for QMA remains open. Given the apparent difficulty of constructing NIZK arguments for QMA, many recent works have focused on this problem in the *preprocessing* setting [BG19, CVZ20, ACGH19, Shm20]. One such setting considers *designated-verifier* NIZKs, where the prover and verifier share a common *uniformly random* string, and the verifier generates a public key that the prover must use to generate proofs; verification is private and requires the corresponding secret key.

We show that the techniques underlying our malicious 2PQC also imply (reusable) malicious designated verifier (MDV-)NIZKs for QMA in the CRS model. The "malicious" requirement asks that zero knowledge hold against verifiers that generate the public key maliciously, and the "reusable" requirement states that soundness holds for multiple proofs (of potentially different statements) computed with respect to the same setup, even when the prover learns whether or not the verifier accepted each proof. This is referred to as multi-theorem security in [Shm20]. In order to obtain reusable security, we instantiate our protocol with a *reusable* classical two-party computation protocol. Such a reusable 2PC can be based on post-quantum OT (of the type needed for Theorem 1.1) plus a reusable MDV-NIZK for NP, which is known from QLWE [LQR$^+$19].

We therefore also obtain the following.

Theorem 1.2. *(Informal) There exists a reusable MDV-NIZK for* QMA *with a classical CRS and classical proving key assuming the existence of post-quantum maliciously-secure two-message oblivious transfer with straight-line simulation in the CRS model, plus post-quantum reusable MDV-NIZK for NP (both of which are known from QLWE).*

The only two previous results to achieve *reusable* designated-verifier NIZKs for QMA are by Shmueli [Shm20] and Alagic et al [ACGH19]. The former is in the CRS model and assumes sub-exponential security of QLWE, while the latter is in the quantum random oracle model (QROM), and also assumes sub-exponential security of QLWE. Crucially, both of these results require the QMA prover to have access to many copies of the QMA

witness. We achieve reusable MDV-NIZK for QMA that only requires the prover to be in possession of a single copy of the QMA witness.

Furthermore, our general-purpose two-party quantum computation result also has a meaningful interpretation in the plain model with sequential messages (without assuming a CRS or setup), which we state in the following informal theorem, which is optimal in terms of the number of messages/rounds.

Theorem 1.3. *(Informal) Assume any k-message post-quantum maliciously-secure two-party computation protocol for classical functionalities in the plain model that delivers the output to one party only. Then there exists:*

1. *A k-message maliciously-secure two-party quantum computation protocol that delivers the output to one party, and*
2. *A $(k+1)$-message maliciously-secure two-party quantum computation protocol that delivers the output to both parties.*

Second Result: A Two-Round Protocol with (Quantum) Preprocessing in the Simultaneous Messages Model. As discussed above, the other natural model is that parties communicate in rounds, and both players may simultaneously send each other a message in every round. In this simultaneous message model, the three-round lower bound discussed above is inapplicable and two-round protocols may conceivably exist (although one-round protocols cannot).

Our first result in this setting is a two-round protocol in a *preprocessing model*, where both players participate in an "offline" preprocessing step without knowledge of their inputs. Once inputs are available, the "online" phase of the protocol requires just two rounds of interaction (with simultaneous messages). We obtain the following:

Theorem 1.4. *There exists a protocol for two-party quantum computation with two simultaneous rounds of communication in the preprocessing model, assuming the sub-exponential quantum hardness of learning with errors (QLWE).*

In fact, we construct a *three-round protocol* in the CRS model with only two rounds of *online* communication, which implies the above theorem. The crucial ingredient that allows us to remove one round of *online* communication is quantum teleportation (we refer the reader to the technical overview, Section 2.7, for more details). In order to prove security of our protocol, we develop a novel delayed simulation technique, which we call "simulation via teleportation", which may be of independent interest.

Third Result: Barriers and Approaches to Two-Round Protocols with Simultaneous Messages in the CRS Model. A natural next question is whether we can remove the preprocessing step. In other words, we ask: in the simultaneous message model, is it possible to construct a two-round maliciously secure 2PQC protocol with just a common random string (CRS)?

This appears to be a fairly challenging question, and we do not fully resolve it in this work. However, we provide both negative and positive partial results, which we hope will lead to future progress on this question.

First, we give some intuition for why it seems hard to design such a two-round protocol by showing that, under a plausible quantum information-theoretic conjecture, a large class of common simulation techniques would *not* suffice. In more detail, we consider any simulator that learns which player (between Alice and Bob) is corrupted only *after* it has generated the simulated CRS. We call such a simulator an *oblivious simulator*. To the best of our knowledge, all existing classical and quantum two-party computation protocols in the CRS model either (1) already admit oblivious simulation, or (2) can generically be transformed to admit oblivious simulation via post-quantum NIZK proofs of knowledge for NP.

In the quantum setting, we show, roughly, that any two-round 2PQC protocol for general quantum functionalities *with an oblivious simulator* would yield a particular one-round, two-party protocol for distributed computation of general quantum functionalities where:

- each party has access to polynomially-many *arbitrary but input-independent* "resource" qubits (generated in an input-independent pre-processing step), which in particular may be entangled with the other party's resource qubits,
- on any pair of quantum inputs, the two parties must (after one round of communication) produce outputs whose joint state is within negligible trace distance of the correct output, and
- no privacy/security is required of the parties' messages.

This exact setting has been studied in the quantum information literature under the name of *instantaneous nonlocal quantum computation* [Vai03, BK11, Spe16, GC20], and the best known protocols for general functionalities require exponential-size pre-processing [BK11]. Thus, a two-round 2PQC for general functionalities with oblivious simulation would immediately yield progress on this quantum-information-theoretic problem.

Theorem 1.5. *(Informal) Under the conjecture that there exists a quantum functionality that does not admit an instantaneous nonlocal quantum computation protocol with polynomial-size pre-processing, there exists a quantum functionality that cannot be securely computed in two rounds in the classical CRS model with an oblivious simulator.*

Towards getting around this potential barrier, we give a proof-of-concept construction of a protocol with non-oblivious simulation. Specifically, we assume a (strong) form of VBB obfuscation for quantum circuits that contain unitary and measurement gates, where the former may be classically controlled on the outcome of measurement gates. We point out, however, that VBB-obfuscation of circuits with measurement gates is potentially even more powerful than the VBB obfuscation for unitaries that was formalized in [AF16] (see discussion in Section 8.2). Under this assumption, we obtain a two-round two-party secure quantum computation protocol in the CRS model.

Theorem 1.6. *(Informal) Two-round two-party secure quantum computation in the CRS model exists assuming a strong form of VBB or ideal obfuscation for quantum circuits as discussed above.*

We remark that while there exist (contrived) examples of functionalities that cannot be VBB obfuscated [AF16, ABDS20, ALP20], it is still plausible that many quantum functionalities can be obfuscated. However, without any candidate constructions of obfuscation for quantum circuits, we stress that our result should only be taken as a proof-of-concept.

2 Technical Overview

2.1 Quantum Background

We briefly recap some relevant concepts from quantum computation.

Notation. We use bold letters to write (the density matrix of) a quantum state $\mathbf{x}$. We use the shorthand $U(\mathbf{x})$ to mean $U\mathbf{x}U^\dagger$, the result of applying unitary U to $\mathbf{x}$. The notation $(\mathbf{x}, \mathbf{y})$ denotes a state on two registers, where $\mathbf{x}$ and $\mathbf{y}$ are potentially entangled. The k-fold tensor product of a state $\mathbf{x} \otimes \mathbf{x} \otimes \cdots \otimes \mathbf{x}$ will be written as $\mathbf{x}^k$.

The Pauli Group. The Pauli group on a single qubit, denoted by $\mathscr{P}_1$, is generated by the unitary operations X (bit flip) and Z (phase flip), defined as $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. The Pauli group on n qubits, denoted by $\mathscr{P}_n$, is the n-fold tensor product of the single qubit Pauli group. Any unitary in the Pauli group $\mathscr{P}_n$ can be written (up to global phase) as $\bigotimes_{i \in [n]} X^{r_i} Z^{s_i}$ for $r, s \in \{0,1\}^n$.

The Clifford Group. The Clifford group on n qubits, denoted by $\mathscr{C}_n$, is the group of unitaries that normalize $\mathscr{P}_n$, i.e. $C \in \mathscr{C}_n$ if and only if for all $U \in \mathscr{P}_n$, we have $CUC^\dagger \in \mathscr{P}_n$. Alternatively, we can think of a Clifford unitary C as an operation where for any choice of $r, s \in \{0,1\}^n$, there exists a choice of $r', s' \in \{0,1\}^n$ such that

$$C\left(\bigotimes_{i\in[n]} X^{r_i} Z^{s_i}\right) = \left(\bigotimes_{i\in[n]} X^{r'_i} Z^{s'_i}\right) C.$$

Intuitively, this means that with a suitable update of the Pauli operation, one can swap the order in which a Clifford and a Pauli are applied.

Clifford Authentication Codes. We will make extensive use of Clifford authentication codes. Clifford authentication codes are an information-theoretic encoding scheme for quantum states that provides both secrecy and authentication. An n-qubit quantum state $\mathbf{x}$ can be encoded in a Clifford authentication code as follows: prepare a λ-qubit all 0's state which we denote as $\mathbf{0}^\lambda$ (where λ is a security parameter), sample a random Clifford unitary $C \leftarrow \mathscr{C}_{n+\lambda}$, and output $C(\mathbf{x}, \mathbf{0}^\lambda)$. The Clifford C serves as a secret key, while the $\mathbf{0}^\lambda$ qubits enable authentication, and are called "trap" qubits. A party without knowledge of C cannot modify the encoding without modifying the trap qubits (except with negligible probability). Therefore, decoding works by applying $C^\dagger$ and then measuring the λ trap qubits in the computational basis. If these measurements are all 0, this ensures that with all but negligible probability, the n remaining registers hold the originally encoded state $\mathbf{x}$.

Clifford + Measurement Circuits. We will rely heavily on the "Clifford + Measurement" representation of quantum circuits (henceforth "C+M circuits") due to [BK05]. In this representation, a quantum circuit can be decomposed into layers. Each layer consists of a Clifford unitary whose output wires are partitioned into wires that will be fed as inputs into the next layer, and wires that will be measured. The latter group of wires are measured in the computational basis, resulting in a classical bitstring that is used to select the Clifford unitary to be applied in the subsequent layer. The first layer takes in all of the inputs to the quantum circuit, ancilla $\mathbf{0}$ states, and "magic" $\mathbf{T}$ states defined as $\mathbf{T} \coloneqq (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. The final layer only produces output wires (i.e. its output registers have no wires to be measured), which are interpreted as the output of the circuit. [BK05] demonstrate that, with constant multiplicative factor overhead in size, any quantum circuit can be written in a magic state representation.

Therefore, for the purposes of this technical overview, we will assume that any quantum circuit F is written as a C+M circuit F_{CM}, and its evaluation on an input $\mathbf{x}$ is computed as $F(\mathbf{x}) = F_{\mathrm{CM}}(\mathbf{x}, \mathbf{T}^k, \mathbf{0}^k)$. For simplicity, we use the same k to denote the number of $\mathbf{T}$ states and the number of ancilla $\mathbf{0}$ states.

Magic State Distillation. In settings where malicious parties are tasked with providing the $\mathbf{T}$ states, we will use cryptographic techniques such as "cut-and-choose" to ensure that F_{CM} is evaluated on an input of the form $(\mathbf{x}, \widehat{\mathbf{T}^k}, \mathbf{0}^k)$ where $\widehat{\mathbf{T}^k}$ is a state guaranteed to be "somewhat" close to $\mathbf{T}^k$. However, correctness of F_{CM} will require states that are negligibly close to real magic states. To that end, we will make use of a magic state distillation C+M circuit D due to [DGJ$^+$20] which takes in somewhat-close magic states $\widehat{\mathbf{T}^k}$ and outputs states negligibly close to $\mathbf{T}^{k'}$, for $k' < k$. Therefore, the representation of any functionality F will in fact be a C+M circuit $F_{\mathrm{CM},D}$ that first applies D to $\widehat{\mathbf{T}^k}$, and then runs F_{CM}.

2.2 Why is Malicious Security Hard to Achieve?

Before we describe our construction of maliciously secure two-party quantum computation (2PQC), we briefly explain why malicious security does not follow readily from existing techniques. Indeed, a candidate three-message 2PQC with *specious* security (the quantum analogue of classical semi-honest security [DNS10]) was recently proposed in [BY20]. Alternatively, any construction of quantum fully-homomorphic encryption (QFHE) naturally yields a two-message 2PQC protocol (with one-sided output): (1) Alice QFHE-encodes her input and sends it to Bob, (2) Bob evaluates the functionality on his input and Alice's encoded input, and (3) Bob sends Alice the encryption of her output.

One might hope to compile this QFHE-based protocol or the [BY20] protocol into a maliciously secure protocol by having the parties include proofs that their messages are well-formed. Unfortunately, it is unclear how to implement this in the quantum setting. In both of these approaches, the parties would have to prove (in zero-knowledge) statements of the form "$\mathbf{y}$ is the result of evaluating quantum circuit C on $\mathbf{x}$." Crucially, the *statement* the parties need to prove explicitly makes reference to a quantum state. This is beyond the reach of what one can prove with, say, NIZKs for QMA, in which witnesses are quantum but the statements are entirely classical.

Therefore, we design our malicious 2PQC so that parties do not have to prove general statements about quantum states. A core ingredient in our protocol is a quantum garbled circuit construction sketched in [BY20, §2.5], where the circuit garbling procedure is entirely classical.[2] Combining this with a post-quantum maliciously-secure *classical* 2PC, we will ensure valid circuit garbling against malicious quantum adversaries.

2.3 A Garbling Scheme for $\mathsf{C} + \mathsf{M}$ Circuits

Our first step is to formalize the proposal sketched in [BY20, §2.5] for garbling $\mathsf{C} + \mathsf{M}$ circuits. The starting point for the [BY20, §2.5] construction is a simple technique for garbling any quantum circuit that consists of a single Clifford unitary F.[3] The idea is to sample a random Clifford E and give out $FE^\dagger$ as the garbled circuit; note that the description of $FE^\dagger$ will be entirely classical. Since the Clifford unitaries form a group, $FE^\dagger$ is a uniformly random Clifford unitary independent of F. To garble the input quantum state $\mathbf{x}$, simply compute $E(\mathbf{x})$. The construction in [BY20, §2.5] extends this simple construction to any circuit.

To build intuition, we will consider a two-layer $\mathsf{C} + \mathsf{M}$ circuit $Q = (F_1, f)$, where F_1 is the first layer Clifford unitary, and f is a classical circuit that takes as input a single bit meaurement result m, and outputs a classical description of F_2, the second layer Clifford unitary. On input $\mathbf{x}$, the circuit operates as follows:

1. Apply F_1 to $\mathbf{x}$.
2. Measure the last output wire in the computational basis to obtain $m \in \{0, 1\}$, and feed the remaining wires to the next layer. Compute the second layer Clifford unitary $F_2 = f(m)$.
3. Apply F_2 to the non-measured output wires from the first layer. Return the result.

One could try to extend the simple idea for one-layer garbling to this circuit. We still sample a random input-garbling Clifford E_0 and compute $F_1 E_0^\dagger$. To hide the second layer Clifford, a natural idea is to sample yet another random Clifford E_1 to be applied to the non-measured output wires of F_1. That is, we replace $F_1 E_0^\dagger$ with $(E_1 \otimes \mathbb{I}) F_1 E_0^\dagger$, and release the description of a function g such that $g(m) = f(m) E_1^\dagger$.

However, this may in general be insecure. Let $F_2^{(0)}$ be the Clifford output by function f when $m = 0$, and $F_2^{(1)}$ the Clifford output by function f when $m = 1$. Suppose $F_2^{(0)} - F_2^{(1)} = A$ for some invertible matrix A. Then, an attacker with access to g could obtain $F_2^{(0)} E_1^\dagger - F_2^{(1)} E_1^\dagger$, and multiplying the result by A^{-1} yields $A^{-1}(F_2^{(0)} E_1^\dagger - F_2^{(1)} E_1^\dagger) = A^{-1} A E_1^\dagger = E_1^\dagger$.

Therefore, instead of giving out g, the construction of [BY20, §2.7] gives out a classical garbling of g. To accommodate this, the output wire from the first layer that is measured to produce $m_1 \in \{0, 1\}$ must be replaced by a collection of wires that produces the corresponding label lab_{m_1} for the garbled circuit. This can be easily achieved by applying a suitable "label unitary" to the m_1 wire (and ancilla wires) within the garbled gate for the first layer.

[2] We remark that the 2PQC proposed in [BY20] is based on their "main" quantum garbled circuit construction, which crucially does *not* have a classical circuit garbling procedure. The advantage of their main construction is that garbling can be done in low depth, whereas the alternative construction requires an expensive but classical garbling procedure.

[3] [BY20] call this *group-randomizing quantum randomized encoding*.

There is one last issue with this approach: an attacker that chooses not to measure the wires containing lab_{m_1} can obtain a superposition over two valid labels. Recall that the standard definition of security for classical garbled circuits only guarantees simulation of one label, not a quantum superposition of both labels. To ensure the attacker cannot get away with skipping the computational basis measurement, the [BY20, §2.7] construction applies a Z-twirl to m_1 before the "label unitary" is applied. Recall that a Z-twirl is simply a random application of a Pauli Z gate, i.e. Z^b for a uniformly random bit b; applying Z^b to a wire is equivalent to performing a computational basis measurement (without recording the result).

To recap, a garbled 2-layer $\mathsf{C}+\mathsf{M}$ circuit Q consists of three components: an "input garbling" Clifford E_0, an initial Clifford unitary to be applied to the garbled input $D_0 \coloneqq (E_1 \otimes \mathbb{I})F_1 E_0^\dagger$, and a classical garbled circuit $\widetilde{g}$. Extrapolating, we see that in general a garbled $\mathsf{C}+\mathsf{M}$ circuit takes the form

$$(E_0, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d) \coloneqq (E_0, \widetilde{Q}),$$

where the $\widetilde{g}_i$'s are garblings of classical circuits. Crucially, all of these components can be generated by an entirely classical circuit. The only quantum operation involved in the garbling process is the application of E_0 to the input $\mathbf{x}$ to garble the input. Next, we show how we can take advantage of this mostly classical garbling procedure to obtain maliciously-secure 2PQC.

2.4 A Three-Message Protocol with Malicious Security

We begin with a plausible but *insecure* construction of a three-message 2PQC based on the above quantum garbled circuit construction. We will then highlight the ways a malicious attacker might break this construction, and arrive at our final construction by implementing suitable modifications.

Our protocol relies only on a *classical* two-message 2PC with one-sided output that is (post-quantum) secure against malicious adversaries; this can be realized by combining (post-quantum) classical garbled circuits [Yao86] with (post-quantum) two-message oblivious transfer [PVW08] following eg. [IKO+11].

We will consider two parties: Alice with input $\mathbf{x}_A$ and Bob with input $\mathbf{x}_B$. They wish to jointly compute a quantum circuit Q on their inputs whose output is delivered to both players. Q is represented as a Clifford+Measurement circuit that takes input $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{T}^k, \mathbf{0}^k)$. We denote by $(\mathbf{y}_A, \mathbf{y}_B)$ the joint outputs of Alice and Bob. At a high level, the parties will use the first two messages (Bob → Alice, Alice → Bob) to jointly encode their quantum inputs, while in parallel computing a two-message classical 2PC that outputs the classical description of a quantum garbled circuit to Bob. By evaluating the garbled circuit, Bob can learn his own output, as well as Alice's encoded output, which he sends to Alice in the 3rd message.

In more detail, the classical functionality $\mathcal{F}[Q]$ to be computed by the classical 2PC is defined as follows. It takes as input (the classical description of) a Clifford unitary $C_{B,\text{in}}$ from Bob and Clifford unitaries $(C_{A,\text{in}}, C_{A,\text{out}})$ from Alice. Let Q_B be a modification of Q that outputs $(C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{y}_B)$ in place of $(\mathbf{y}_A, \mathbf{y}_B)$; looking ahead, this will enable Bob to evaluate (a garbling of) Q_B on (a garbling of) their joint inputs without learning Alice's output. The functionality computes a garbling $(E_0, \widetilde{Q_B})$ of Q_B. Finally, it computes $W \coloneqq E_0 \cdot (\mathbb{I} \otimes C_{B,\text{in}}^{-1} \otimes \mathbb{I}) \cdot C_{A,\text{in}}^{-1}$ (where the registers implied by the tensor product will become clear below), and outputs $(W, \widetilde{Q_B})$ to Bob.

The (insecure) protocol template is as follows:

- **First Message (Bob → Alice).** Bob picks a random Clifford $C_{B,\text{in}}$ and uses it to encrypt and authenticate his input $\mathbf{x}_B$ as $\mathbf{m}_1 \coloneqq C_{B,\text{in}}(\mathbf{x}_B, \mathbf{0}^\lambda)$. He also computes the first round message m_1 of the classical 2PC, using $C_{B,\text{in}}$ as his input. He sends $(\mathbf{m}_1, m_1)$ to Alice.

- **Second Message (Alice → Bob).** After receiving $(\mathbf{m}_1, m_1)$, Alice picks a random Clifford $C_{A,\text{in}}$ and uses it to encrypt her input $\mathbf{x}_A$ along with Bob's encoding $\mathbf{m}_1$, k copies of a $\mathbf{T}$ state, and $k+\lambda$ copies of a $\mathbf{0}$ state. The result of this is $\mathbf{m}_2 \coloneqq C_{A,\text{in}}(\mathbf{x}_A, \mathbf{m}_1, \mathbf{T}^k, \mathbf{0}^{k+\lambda})$. Alice also samples another random Clifford $C_{A,\text{out}}$ that will serve to encrypt and authenticate her output, and computes the second round message m_2 of the classical 2PC using input $(C_{A,\text{in}}, C_{A,\text{out}})$. She sends $(\mathbf{m}_2, m_2)$ to Bob.

- **Third Message (Bob → Alice).** After receiving $(\mathbf{m}_2, m_2)$, Bob can compute his output of the classical 2PC, which is $(W, \widetilde{Q_B})$. He computes

  $$W(\mathbf{m}_2) = E_0 \cdot (\mathbb{I} \otimes C_{B,\text{in}}^{-1} \otimes \mathbb{I}) \cdot C_{A,\text{in}}^{-1} \left(C_{A,\text{in}}(\mathbf{x}_A, \mathbf{m}_1, \mathbf{T}^k, \mathbf{0}^{k+\lambda})\right) = E_0(\mathbf{x}_A, \mathbf{x}_B, \mathbf{T}^k, \mathbf{0}^{k+\lambda}).$$

  Recall that $E_0(\mathbf{x}_A, \mathbf{x}_B, \mathbf{T}^k, \mathbf{0}^{k+\lambda})$ corresponds to a garbled input for $\widetilde{Q_B}$. He evaluates $\widetilde{Q_B}$ on this garbled input and obtains $(C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{y}_B)$.

  At this point, Bob has his output $\mathbf{y}_B$ in the clear. Next he sets $\mathbf{m}_3 = C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda)$, and sends $\mathbf{m}_3$ to Alice. Upon receiving $\mathbf{m}_3$, Alice can recover her output by computing $C_{A,\text{out}}^{-1}(\mathbf{m}_3)$.

The above protocol can already be shown to be secure against malicious Bob by relying on security of the classical two-party computation protocol against malicious adversaries. But malicious Alice can break security by generating ill-formed auxiliary states. We now describe this issue in some more detail and then present modifications to address the problem.

Malicious Generation of Auxiliary States. In the second message of the protocol, Alice is instructed to send a quantum state $C_{A,\text{in}}(\mathbf{x}_A, \mathbf{m}_1, \mathbf{T}^k, \mathbf{0}^{k+\lambda})$. A malicious Alice can deviate from the honest behavior by submitting arbitrary states in place of the magic $\mathbf{T}$ states and the auxiliary $\mathbf{0}$ states, either of which may compromise security.

We therefore modify the classical 2PC to include randomized checks that will enable Bob to detect if Alice has deviated from honest behavior.

We check validity of $\mathbf{0}$ states using the "random linear map" technique of [DGJ+20]. The classical 2PC will sample a uniformly random matrix $M \in \mathbb{F}_2^{k \times k}$, and apply a unitary U_M that maps the quantum state $\mathbf{v} = |v\rangle\langle v|$ for any $v \in \mathbb{F}_2^k$ to the state $\mathbf{Mv} = |Mv\rangle\langle Mv|$. For any $M \in \mathbb{F}_2^{k \times k}$, there exists an efficient Clifford unitary U_M implementing this map. This check takes advantage of the fact that $U_M(\mathbf{0}^k) = \mathbf{0}^k$ for any M, but on any other pure state $\mathbf{v} = |v\rangle\langle v|$ for non-zero $v \in \mathbb{F}_2^k$, we have $U_M(\mathbf{v}) \neq \mathbf{0}^k$ with overwhelming probability in k.

More precisely, our protocol will now ask Alice to prepare twice ($2k$) the required number of $\mathbf{0}$ states. The classical 2PC will generate a Clifford unitary U_M implementing a random linear map $M \in \mathbb{F}_2^{2k \times 2k}$, and incorporate U_M into its output Clifford W, which is now $W = (E_0 \otimes I) \cdot (\mathbb{I} \otimes C_{B,\text{in}}^{-1} \otimes \mathbb{I}) \cdot (\mathbb{I} \otimes U_M) \cdot C_{A,\text{in}}^{-1}$. Now when Bob applies W to Alice's message $C_{A,\text{in}}(\mathbf{x}_A, C_{B,\text{in}}(\mathbf{x}_B, \mathbf{0}^\lambda), \mathbf{T}^k, \mathbf{0}^{2k})$, it has the effect of stripping off $C_{A,\text{in}}$ by applying $C_{A,\text{in}}^{-1}$, and then applying U_M to the last $2k$ registers. The rest of the application of W has the same effect as before the modification, so it undoes the application of $C_{B,\text{in}}$, and then re-encodes *all but the last k registers* under the input garbling Clifford E_0 to produce a garbled input. Crucially, the last k registers are designated "$\mathbf{0}$-state check registers", which Bob can simply measure in the computational basis to detect if Alice prepared the $\mathbf{0}$ states properly.

Unfortunately, this technique does not extend to checking validity of $\mathbf{T}$ states. To do so, we would have to map $\mathbf{T}$ states to $\mathbf{0}$ states, but there is no Clifford unitary that realizes this transformation.[4] The problem with using a non-Clifford unitary is that security of W relies on the fact that it is the product of a random Clifford $C_{A,\text{in}}$ and some other Clifford W'. Since the Clifford unitaries form a group, multiplication by a random $C_{A,\text{in}}$ perfectly masks the details of W', but only when W' is Clifford.

We will therefore employ the "cut-and-choose" technique from [DGJ+20]. The protocol will now have Alice prepare $\lambda(k+1)$-many $\mathbf{T}$ states instead of just k. The classical 2PC will generate a random permutation π on $[\lambda(k+1)]$, which will move a random selection of λ of the $\mathbf{T}$ states into "$\mathbf{T}$-state check registers." The application of π will be implemented by a unitary U_π incorporated into W. After applying W, Bob will apply a projective measurement onto $\mathbf{T}$ to each of the $\mathbf{T}$-state check registers, and will abort if any of the λ measurements fails.

If all of the λ measurements pass, this means the remaining λk un-tested $\mathbf{T}$ states are "somewhat close" to being real $\mathbf{T}$ states. However, being "somewhat close" will not be sufficient; for instance, an attacker who prepares exactly one completely invalid $\mathbf{T}$ state will only be caught with $1/(k+1)$ probability.

[4] The existence of such a Clifford would imply that Clifford + Measurement circuits *without* magic states are universal for quantum computing, contradicting the Gottesman–Knill theorem (assuming $\mathsf{BPP} \neq \mathsf{BQP}$).

We will therefore need to apply magic-state distillation to transform these into states which are negligibly close to real $\mathbf{T}$ states. For this, we use a magic-state distillation circuit of [DGJ+20, §2.5] (which builds on [BK05]). This circuit consists solely of Clifford gates and computational basis measurements. To apply this circuit we modify our underlying functionality, so that we now give out a garbling of a circuit that first implements magic-state distillation and only then applies Q_B.

This completes an overview of our protocol, and a formal construction and analysis can be found in Section 5.

2.5 Application: Reusable MDV-NIZK for QMA

Now we briefly describe how the above techniques readily give a reusable malicious designated-verifier NIZK for QMA in the CRS model. Note that NIZK for QMA is a special case of two-party quantum computation, where the functionality being computed is the verification circuit $\mathcal{V}$ for some QMA language, the prover (previously Alice) has the quantum witness $\mathbf{w}$ as input, and the verifier (previously Bob) has no input and receives a binary output indicating whether $\mathcal{V}(x, \mathbf{w})$ accepts or rejects, where x is the (classical) description of the instance they are considering.

Since the prover does not receive output, there is no need for the third message in the protocol of Section 2.4. Furthermore, since the verifier has no input, there is no need for any quantum message from him in the first message. The verifier only needs to send a first-round classical 2PC message which then functions as a proving key. The (classical) left-over state is the verifier's secret verification key. After this, the prover just sends one quantum message (the Second Message in the above protocol), proving that $\mathcal{V}(x, \mathbf{w}) = 1$.

In order to make the above template reusable, we can first instantiate the underlying classical 2PC with a reusable 2PC. Once this is in place, the verifier's first-round message is necessarily instance-indepedent. Then, to ensure that a cheating prover cannot break soundness by observing whether the verifier accepts its proofs or not, we modify the classical functionality to take as input a PRF key from the verifier, and generate all required randomness (used for the $\mathbf{0}$ and $\mathbf{T}$ checks, and the quantum garbling procedure) by applying this PRF to the (classical) description of the instance x. By security of the reusable 2PC, a verifier will never accept a maliciously sampled proof for any instance x not in the language. In the main body, when formally establishing adaptive security, we require the reusable 2PC to satisfy a special extractability property. This stipulates the existence of a simulator that efficiently extracts the input of a malicious sender – such that the joint distribution of extracted inputs, the view of the malicious sender, and the output of the receiver – is indistinguishable between the real and ideal experiments. A reusable two-message 2PC protocol satisfying this property was obtained in [LQR+19] by relying on (non-reusable) two-message OT and reusable MDV-NIZKs for NP, which are themselves known from LWE [LQR+19].

2.6 Challenges in Achieving a Two-Round Protocol in the Quantum Setting

The previous sections show that we can achieve the optimal round complexity of three in the sequential message setting. However, if the parties can send simultaneous messages, then it may be possible to reduce the number of rounds to two. Indeed, in the classical setting, there is a natural approach to obtaining a two-simultaneous-round protocol, given a two-sequential-message protocol where one party gets output (which can be obtained as a sub-protocol of any three-sequential message protocol where both parties get output). The parties simply run two parallel executions of the two-sequential-message protocol on the same inputs - one in which Alice speaks first and the functionality only computes her part of the output, and another in which Bob speaks first and the functionality only computes his part of the output. Unfortunately, this natural approach completely fails in the quantum setting, for at least two reasons.

- Running two parallel executions of the same protocol on the same set of inputs seems to require *cloning* those inputs, which is in general impossible if the inputs may be arbitrary quantum states.
- Running two parallel executions of a randomized functionality requires the parties to fix the same random coins to be used in each execution, as otherwise their outputs may not be properly jointly

distributed. This is not possible in the quantum setting, since randomness can come from measurement, and measurement results cannot be fixed and agreed upon beforehand.

These issues motivate the rest of our work, in which we explore potential improvements to the three-message protocol given above when simultaneous messages are allowed. Since running two protocols in parallel on the same inputs is problematic, we take as our guiding principle that one party must be performing the actual computation at some point in the protocol, and then distributing the outputs. The remainder of our paper is devoted to addressing challenges that arise when attempting to accommodate this structure within two rounds of interaction, while maintaining security.

Interestingly, while the first issue mentioned above is unique to the setting of quantum inputs, the second issue applies even if the parties wish to compute a quantum circuit over just *classical* inputs, which we regard as a very natural setting. Thus, while this paper focuses on the most general case of secure quantum computation over potentially quantum inputs, we stress that all the results we achieve are the best known even for the classical input setting. Furthermore, note that both issues also exist in the specious setting, so it doesn't appear to be straightforward to achieve two-round 2PQC even in this setting. While the focus of this paper is on the setting of malicious security, exploring these questions in the specious setting is also an interesting direction.

2.7 A Two-Round Protocol with Pre-Processing

Our main result in the two-simultaneous-round setting is a construction in the *pre-processing* model, where the parties may first jointly compute some *input-independent* (quantum) correlations, and then subsequently compute on their private inputs with only two rounds of interaction.

In fact, we construct a protocol in which the pre-processing phase only consists of a *single* message from Bob to Alice (computed with respect to a CRS). We take our three sequential message protocol as a starting point, and introduce several modifications. The first modification will immediately achieve the goal of removing input-dependence from Bob's first message, and all the subsequent modifications will be necessary to restore correctness and security.

Modification 1: Removing Input-Dependence via Teleportation. Before sending his first message, Bob samples n EPR pairs, where n is the number of qubits of the input $\mathbf{x}_B$. We denote these EPR pairs by $(\mathbf{epr}_1, \mathbf{epr}_2)$, where $\mathbf{epr}_1$ denotes the left n qubits, and $\mathbf{epr}_2$ denotes the right n qubits. In place of sending $C_{B,\text{in}}(\mathbf{x}_B, \mathbf{0}^\lambda)$, Bob sends $\mathbf{m}_{B,1} \coloneqq C_{B,\text{in}}(\mathbf{epr}_1, \mathbf{0}^\lambda)$. Note that the classical 2PC only requires input $C_{B,\text{in}}$, which is a random Clifford that Bob samples for himself, so Bob's entire first round message $(\mathbf{m}_{B,1}, m_{B,1})$ can now be sent *before* Bob receives his input. The idea is that later on, when Bob learns his input $\mathbf{x}_B$, he will perform Bell measurements on $(\mathbf{x}_B, \mathbf{epr}_2)$ to teleport $\mathbf{x}_B$ into $\mathbf{epr}_1$.

Issue: Incorporating Bob's Teleportation Errors. Teleporting $\mathbf{x}_B$ into $\mathbf{epr}_1$ will require Bob to somehow correct $\mathbf{epr}_1$ later in the protocol using the results of his Bell measurements on $(\mathbf{x}_B, \mathbf{epr}_2)$. But enabling Bob to do this in a way that does not compromise security will be tricky, as we now explain.

After receiving the second round message from Alice in our original malicious 2PQC protocol, Bob learns the output of the classical 2PC, which includes (1) a (classical description of a) quantum garbled circuit $\widetilde{Q}$, and (2) a Clifford unitary W. Bob applies W to Alice's quantum message $\mathbf{m}_{A,2}$, performs the appropriate $\mathbf{0}$ and $\mathbf{T}$ state checks, and conditioned on the checks passing, is left with a state of the form $E_0(\mathbf{x}_A, \mathbf{x}_B, \widehat{\mathbf{T}}, \mathbf{0})$, where $\widehat{\mathbf{T}}$ is a state "somewhat close" to $\mathbf{T}^k$. But at this point in our newly modified protocol, Bob is holding the state $E_0(\mathbf{x}_A, \mathbf{epr}_1, \widehat{\mathbf{T}}, \mathbf{0})$. To restore correctness, we somehow need to modify the protocol so that Bob can apply $X^{x_{\text{inp}}} Z^{z_{\text{inp}}}$ to $\mathbf{epr}_1$ "inside" the E_0 mask, where $x_{\text{inp}}, z_{\text{inp}}$ are the result of Bell basis measurements on $(\mathbf{x}_B, \mathbf{epr}_2)$.

Recall that the structure of W is $W = E_0 \cdot U^\dagger_{\text{dec}-\text{check}}$, where E_0 is the input garbling Clifford for the quantum garbled circuit, and $U_{\text{dec}-\text{check}}$ is the matrix that undoes $C_{A,\text{in}}$, undoes $C_{B,\text{in}}$, and then applies a permutation π and a random linear map M, and rearranges all the to-be-checked registers to the last few (rightmost) register slots. The multiplication by E_0 is applied only to the non-checked registers.

Thus, it seems like correctness would have to be restored by inserting the unitary $(\mathbb{I} \otimes X^{x_{\text{inp}}} Z^{z_{\text{inp}}} \otimes \mathbb{I})$ in between E_0 and $U^\dagger_{\text{dec-check}}$. But if Bob can learn $E_0(\mathbb{I} \otimes X^{x_{\text{inp}}} Z^{z_{\text{inp}}} \otimes \mathbb{I})U^\dagger_{\text{dec-check}}$ for even two different values of x_{inp} and z_{inp}, security of the input garbling Clifford E_0 may be lost entirely.

Modification 2: Classical Garbling + Quantum Multi-Key Fully Homomorphic Encryption In order to resolve this issue, we will split up the matrix $E_0(\mathbb{I} \otimes X^{x_{\text{inp}}} Z^{z_{\text{inp}}} \otimes \mathbb{I})U^\dagger_{\text{dec-check}}$ into two matrices

$$U_{x_{\text{inp}},z_{\text{inp}}} := E_0(\mathbb{I} \otimes X^{x_{\text{inp}}} Z^{z_{\text{inp}}} \otimes \mathbb{I})U^\dagger_{\text{rand}}$$
$$U_{\text{check}} := U_{\text{rand}} U^\dagger_{\text{dec-check}}$$

where U_{rand} is a "re-randomizing" Clifford.

The matrix U_{check} is independent of Bob's teleportation errors, and will now be output to Bob by the classical 2PC. But to preserve security, we will have Bob obtain $U_{x_{\text{inp}},z_{\text{inp}}}$ by evaluating a *classical* garbled circuit $\widetilde{f}_{\text{inp}}$ where $f_{\text{inp}}(x_{\text{inp}}, z_{\text{inp}}) := U_{x_{\text{inp}},z_{\text{inp}}}$; the garbled circuit $\widetilde{f}_{\text{inp}}$ is included in the output of the classical 2PC.

But now we are faced with a new problem: how does Bob obtain the (classical) labels for $\widetilde{f}_{\text{inp}}$? Since we only have one round of interaction remaining, Bob won't be able to run an OT to learn the correct labels (Bob could learn the labels by the end of the two online rounds, but then we would still need another round for Bob to send Alice her encrypted output).

We resolve this problem with *quantum multi-key fully-homomorphic encryption* (QMFHE), which we will use in tandem with our classical garbled circuit $\widetilde{f}_{\text{inp}}$ to enable Bob to compute (a homomorphic encryption of) $U_{x_{\text{inp}},z_{\text{inp}}}$ without leaking anything else. Before we continue, we give a brief, intuition-level recap of QMFHE (we refer the reader to Section 3.6 for a formal description). Recall that a standard fully-homomorphic encryption (FHE) allows one to apply arbitrary efficient computation to encrypted data (without needing to first decrypt). *Multi-key* FHE (MFHE) extends FHE to enable computation over multiple ciphertexts encrypted under different keys; the output of such a homomorphic computation is a "multi-key" ciphertext which can only be decrypted given all the secret keys for all of the ciphertexts involved in the computation [LTV12]. Finally, QMFHE extends MFHE a step further to allow arbitrary efficient *quantum* computation over encrypted (classical or quantum) data [Goy18, Bra18, Mah18, ABG⁺20].

We will encrypt each of the garbled circuit labels for $\widetilde{f}_{\text{inp}}$ under an independent QMFHE key. All of these encrypted labels along with the corresponding QMFHE public keys (to enable quantum computations over these ciphertexts) will also be output to Bob as part of the classical 2PC. We remark that this requires a QMFHE scheme where encryptions of classical plaintexts are themselves classical; such schemes are known assuming the quantum hardness of the learning with errors (QLWE) assumption [ABG⁺20].[5]

To recap, Bob obtains from the classical 2PC a collection of QMFHE ciphertexts, one for each of the garbled circuit labels for $\widetilde{f}_{\text{inp}}$. Bob picks out the ciphertexts corresponding to $x_{\text{inp}}, z_{\text{inp}}$ and performs quantum multi-key evaluation of $\widetilde{f}_{\text{inp}}$ over these ciphertexts, obtaining a QMFHE encryption of the output of $\widetilde{f}_{\text{inp}}$, i.e. $\mathsf{QMFHE.Enc}(\mathsf{pk}_{x_{\text{inp}},z_{\text{inp}}}, U_{x_{\text{inp}},z_{\text{inp}}})$ where $\mathsf{pk}_{x_{\text{inp}},z_{\text{inp}}}$ denotes the collection of QMFHE public keys corresponding to $x_{\text{inp}}, z_{\text{inp}}$. The classical 2PC output also includes U_{check} in the clear, which Bob can apply to $\mathbf{m}_{A,2}$ to obtain $U_{\text{rand}}(\mathbf{x}_A, \mathbf{epr}_1, \widehat{\mathbf{T}}, \mathbf{0})$ (after performing appropriate measurement checks). Then Bob can homomorphically compute the ciphertext $\mathsf{QMFHE.Enc}(\mathsf{pk}_{x_{\text{inp}},z_{\text{inp}}}, E_0(\mathbf{x}_A, \mathbf{x}_B, \widehat{\mathbf{T}}, \mathbf{0}))$, and proceed to homomorphically evaluate his quantum garbled circuit to obtain $\mathsf{QMFHE.Enc}(\mathsf{pk}_{x_{\text{inp}},z_{\text{inp}}}, (C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{y}_B))$.

In order for Bob to obtain his final output in the clear, we will have Bob send Alice $x_{\text{inp}}, z_{\text{inp}}$ in the first online round. In response, in the second online round Alice will reply with $\mathsf{sk}_{x_{\text{inp}},z_{\text{inp}}}$; security of the QMFHE will guarantee that Bob cannot decrypt ciphertexts corresponding to any other choice of the teleportation errors. In the second online round, Bob will send Alice $\mathsf{QMFHE.Enc}(\mathsf{pk}_{x_{\text{inp}},z_{\text{inp}}}, (C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda))$, which she can decrypt to obtain $\mathbf{y}_A$. Finally, Bob produces his output by performing QMFHE decryption with $\mathsf{sk}_{x_{\text{inp}},z_{\text{inp}}}$.

[5] We only require *leveled* QMFHE, which can be based solely on the QLWE assumption. Unleveled QMFHE requires an additional circularity security assumption.

Issue: Simulating a Quantum Garbled Circuit with Unknown Output. At this point, we have a correct protocol whose first round is completely input-independent. However, we will run into issues when attempting to prove malicious security.

The problem arises in the security proof for a malicious Bob. In the original three-round maliciously secure protocol, the simulator is able to extract $\mathbf{x}_B$ from Bob's first round message to Alice; this is done by first extracting $C_{B,\text{in}}$ from Bob's first round classical message for the classical 2PC, and then applying $C_{B,\text{in}}^{-1}$ to Bob's first round quantum message. Extracting $\mathbf{x}_B$ from Bob's first round message to Alice is crucial for proving security, since it enables the simulator to query the ideal functionality on $\mathbf{x}_B$, learn the output $\mathbf{y}_B$, and finally simulate the quantum garbled circuit using Bob's output $\mathbf{y}_B$ before computing Alice's simulated second round message to be sent to Bob. This second round message reveals to Bob the quantum garbled circuit, so it is crucial that the quantum garbled circuit simulator has been executed at this point.

Not surprisingly, this simulation strategy runs into a major problem in our newly modified protocol. Bob's first message is independent of $\mathbf{x}_B$, so the simulator cannot query the ideal functionality, and therefore seemingly cannot simulate the quantum garbled circuit before computing Alice's message, which in particular reveals the quantum garbled circuit to Bob. In summary, the simulator must provide Bob with the quantum garbled circuit (part of Alice's first online round message), *before* it has enough information to extract Bob's input. This appears quite problematic since simulating a garbled circuit certainly requires knowing the output. However, since Bob can only obtain an *encryption* of the output of the garbled circuit after receiving Alice's first message, it is still reasonable to expect that the protocol is secure.

Modification 3: Simulation via Teleportation. We fix this problem through a new technique we call *simulation via teleportation.* The idea is as follows. Instead of running the quantum garbled circuit simulator on the output of the circuit (which the simulator does not yet know), the simulator will first prepare fresh EPR pairs $\mathbf{epr}'_1, \mathbf{epr}'_2$ and then run the quantum garbled circuit simulator on $(C_{A,\text{out}}(\mathbf{0}, \mathbf{0}^\lambda), \mathbf{epr}'_1)$ (where $\mathbf{0}$ takes the place of Alice's input $\mathbf{x}_A$ and $\mathbf{epr}'_1$ takes the place of Bob's output $\mathbf{y}_B$). In the following round, after Bob has teleported over his input state $\mathbf{x}_B$, the simulator will query the ideal functionality, learn $\mathbf{y}_B$, and then *teleport* $\mathbf{y}_B$ *into* $\mathbf{epr}'_1$.

Implementing the final teleportation step requires some care. When the simulator learns $\mathbf{y}_B$, it performs Bell measurements on $(\mathbf{y}_B, \mathbf{epr}'_2)$, obtaining measurement outcomes $x_\text{out}, z_\text{out}$. It must then find some way to apply $x_\text{out}, z_\text{out}$ to the state $\mathbf{epr}'_1$ so that Bob can obtain his correct output.

So we further modify the protocol so that the garbled circuit Bob receives from the classical 2PC is modified to output $(C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), X^{x_\text{out}} Z^{z_\text{out}} \mathbf{y}_B)$ instead of $(C_{A,\text{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{y}_B)$, as before. That is, in the real protocol, an honest Alice will sample random $x_\text{out}, z_\text{out}$, and then the 2PC will output the circuit implementing this functionality. Alice will send $x_\text{out}, z_\text{out}$ to Bob in the second online round, and Bob will first apply Pauli corrections $X^{x_\text{out}} Z^{z_\text{out}}$ to his output to obtain $\mathbf{y}_B$. In the simulated protocol, however, $x_\text{out}, z_\text{out}$ are not sampled by the simulator. Instead, they are the result of the simulator's Bell measurements on $(\mathbf{y}_B, \mathbf{epr}'_2)$. The simulator thus simulates a garbled circuit that outputs $(C_{A,\text{out}}(\mathbf{0}, \mathbf{0}^\lambda), \mathbf{epr}'_1)$, and then sends $x_\text{out}, z_\text{out}$ in the second online round. Note that this teleportation step occurs *exclusively within the simulation.*

Modification 4: Alice (Equivocally) Commits to Pauli Corrections. To arrive at a fully secure protocol, we need to address one last issue. As currently described, nothing is stopping a malicious Alice from misreporting her choice of $x_\text{out}, z_\text{out}$. This can introduce arbitrary Pauli errors into Bob's output that he has no way of detecting. However, this can easily be fixed using equivocal commitments (we refer to Section 3.7 for a formal description of equivocal commitments). That is, Alice inputs $x_\text{out}, z_\text{out}$ to the classical 2PC, along with commitment randomness s. Bob obtains the commitment as part of the output of the classical 2PC, and later when Alice sends $x_\text{out}, z_\text{out}$ in the second online round, she must also send along s. The equivocality property enables the simulation strategy to work as before, as the simulator will have the power to send Bob a commitment to an arbitrary value, and after learning $x_\text{out}, z_\text{out}$ from its Bell measurements, use equivocation to produce a valid opening.

2.8 Two Rounds Without Pre-Processing: Challenges and Possibilities

In this section, we explore the possibility of achieving a two-round protocol in the CRS model *without pre-processing.* We stress that this model *does not permit pre-shared entanglement* between the two parties, as we consider sharing of entanglement to be a pre-processing step.

The Challenge of Oblivious Simulation. In the classical setting, all known two-round two-party computation protocols (in the CRS model) can be modified so that security is proven via (what we call) an *oblivious simulator.*[6] That is, the simulator (1) only makes black-box queries to the adversary, (2) is straight-line (meaning it only runs the adversary a single time without rewinding), and (3) it generates the simulated CRS *independently of the choice of corrupted party* (between Alice and Bob).

By focusing on protocols with oblivious simulation, we can highlight an apparent difficulty of building secure two-round protocols for quantum functionalities in the CRS model. Assume without loss of generality that Alice is adversarial (the identical argument applies to Bob). Observe that if the first message that Alice sends is not computationally binding to her input $\mathbf{x}_A$, she can potentially cheat by *equivocating*, i.e. acting as if she had received a different input, and subsequently learn multiple outputs of the functionality. If the simulation is oblivious, then this reasoning applies simultaneously to Alice and Bob — that is, both parties must, in the first round, send computationally-binding commitments to their respective inputs. This is immediately problematic for quantum inputs, since no-cloning implies that their leftover states will have no (computationally) useful information about their original inputs. Thus, it is unclear how a general computation can be performed on their *joint* inputs before the start of the second round, as the parties have effectively swapped their initial states. And somehow, after just one more round of messaging, they must hold their correctly computed output states.

Our negative result formalizes this intuitive difficulty. If the simulator is oblivious, then by roughly following the above reasoning, at the end of the first round:

- Alice holds a computationally binding commitment to Bob's input $\mathbf{x}_B$,
- Bob holds a computationally binding commitment to Alice's input $\mathbf{x}_A$, and
- neither party has information about their original inputs.

Moreover, the correctness of oblivious simulation implies that for a computationally indistinguishable CRS, there exists a "trapdoor" that would enable Alice to extract $\mathbf{x}_B$ and would enable Bob to extract $\mathbf{x}_A$. But now their states can be viewed as the states of two parties at the *beginning of a one-round protocol with polynomial-size pre-processing* in which the parties' inputs are *swapped*; the pre-processing step is necessary to give both parties the trapdoor information of the simulator. The resulting one-round protocol no longer satisfies any meaningful security guarantees, but crucially, it still satisfies correctness. Moreover, the one-round protocol falls into a model of "instantaneous non-local computation" that has been previously studied in the quantum information literature [BK11]. It is currently open whether this model enables general quantum computation with only polynomial-size preprocessing, and a positive result for two-round 2PQC with oblivious simulation would affirmatively answer this question.

A Proof-of-Concept Construction from Quantum VBB Obfuscation. Given the above barrier, one could attempt to construct a two-round protocol whose security relies crucially on a *non-oblivious* simulation strategy. In this work, we take an initial step in this direction by providing a proof-of-concept construction from a strong form of quantum VBB obfuscation that handles obfuscation of quantum circuits that include both unitary gates and measurement gates (see Definition 8.4 and the discussion preceding it).

In our construction, Alice will send an encryption of her input to Bob in round 1, who will then homomorphically compute the functionality over their joint inputs and respond with Alice's encrypted output in round 2. Alice will also send a message in round 2 that allows Bob to decrypt his output. However, the key

[6] Each party will use a NIZK proof of knowledge to prove that their first message is well-formed, using their input and randomness as witness. Then, a simulator programming the CRS may extract either party's input.

is that this interaction will actually be indistinguishable from an interaction in which the *opposite* flow of computation is occuring. In particular, if the CRS if sampled differently (but in an indistinguishable way), it will be the case that Bob is actually sending his encrypted input to Alice in the first round, and then Alice homomorphically computes the functionality and sends Bob's encrypted output back in the second round.

To instantiate this template, we provide a number of quantum obfuscations in the CRS, three per party. First, there are the "input" obfuscations $\mathcal{O}_{A,\mathsf{inp}}$ and $\mathcal{O}_{B,\mathsf{inp}}$. $\mathcal{O}_{A,\mathsf{inp}}$ will take as input Alice's input $\mathbf{x}_A$ along with a "dummy" input $\mathbf{d}_A$, and output Clifford encodings of each. Alice is instructed to send the first output of this obfuscation as her first message, and keep the second output as her state. In the real protocol, the obfuscated functionality will be such that the first output will be the Clifford encoding of the first input (Alice's real input $\mathbf{x}_A$), and the second output will be the Clifford encoding of the second input (Alice's dummy input $\mathbf{d}_A$). On the other hand, $\mathcal{O}_{B,\mathsf{inp}}$ will obfuscate the functionality that does the exact opposite, setting its first output to be a Clifford encoding of its second input, and its second output to be a Clifford encodings of its first input. Thus, in round 1, Alice sends a Clifford encoding of her real input and keeps a Clifford encoding of her dummy input in her state, while Bob sends a Clifford encoding of his dummy input and keeps a Clifford encoding of his real input in his state.

The next obfuscations $\mathcal{O}_{A,\mathsf{cmp}}$ and $\mathcal{O}_{B,\mathsf{cmp}}$ share secret randomness with the input obfuscations (in the form of PRF keys) and can thus decrypt Clifford encodings output by the input obfuscations. They each are defined to decrypt and check the authenticity of their inputs, apply the functionality Q that the parties wish to compute, and then encode the outputs with freshly sampled Cliffords. Each party will run their respective obfuscation on their state and the other party's first round message. Note that then Alice is just using $\mathcal{O}_{A,\mathsf{cmp}}$ to compute Q over dummy inputs, while Bob is using $\mathcal{O}_{B,\mathsf{cmp}}$ to compute Q over their real inputs. Alice will send an encrypted dummy output to Bob in round 2, while Bob will send an encrypted real output to Alice.

Finally, each party applies their respective output obfuscation $\mathcal{O}_{A,\mathsf{out}}$ and $\mathcal{O}_{B,\mathsf{out}}$ to their final state and other party's second round message. $\mathcal{O}_{A,\mathsf{out}}$ will ignore Alice's state (which contains Alice's dummy output) and decrypt and output Bob's second round message (which contains Alice's real output). On the other hand, $\mathcal{O}_{B,\mathsf{out}}$ will ignore Alice's second round message and decrypt and output Bob's state.

Now, it is possible to argue (under the assumption that the obfuscations in the CRS are in fact VBB obfuscations[7]) that, because all intermediate states and messages are Clifford-encoded, "switching the direction" of the input and output obfuscations cannot be noticed by the parties. Note that if each of $\mathcal{O}_{A,\mathsf{inp}}$ and $\mathcal{O}_{B,\mathsf{inp}}$ are re-defined to permute the order of their outputs, then the flow of computation will be completely reversed. In particular, Alice will be computing the functionality over real inputs with $\mathcal{O}_{A,\mathsf{cmp}}$, and Bob will be computing the functionality over dummy inputs with $\mathcal{O}_{B,\mathsf{cmp}}$. Thus, depending on how the simulator programs the CRS, it can either extract directly from Alice's first round message OR it can extract directly from Bob's first round message, but it could never extract from both simultaneously.

Thus, this template represents a potential method for securely computing a quantum functionality in two rounds, where one of the two parties actually performs the computation between rounds 1 and 2 and then distributes the output in round 2. In other words, it is an instantiation of our guiding principle mentioned in Section 2.6 in a model without pre-processing.

Of course, since VBB obfuscation of quantum circuits is in general impossible [AF16], one may wonder how to interpret this result. One may view this construction, in conjunction with our impossibility result for oblivious simulators, as suggesting a particular template for designing two-round 2PQC that with new ideas may eventually be instantiated to give a construction from plausible assumptions. On the other hand, one may view the construction as a potential barrier to obtaining a more general impossibility result. Indeed, showing that it is impossible to securely compute a particular quantum functionality Q in two rounds now requires showing that (strong) VBB obfuscation of certain functionalities is impossible. Currently, we only know that some very specific functionalities are un-obfuscatable [AF16, ABDS20, ALP20].

[7] Attempting to prove this based on just indistinguishability obfuscation runs into issues that arise due to the inherently probabilistic nature of the functionalities obfuscated. In particular, they generate randomness via measurement and then use this randomness to generate Clifford matrices. In the classical setting, one could usually generate the required randomness with a PRF applied to the input, but it is unclear how to do this when the input is a quantum state.

3 Preliminaries

3.1 Notation

Following [BY20], we define a Quantum Random Variable, or QRV, to be a density matrix $\mathbf{x}$ on register X. We will generally refer to QRVs with lowercase bold font and to registers with uppercase gray font. A collection of QRVs $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ on registers $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ is also a QRV, and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ may or may not be entangled with each other.

Let λ denote the security parameter. We will consider non-uniform quantum polynomial-time adversaries, denoted by $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$, where each Adv_λ is the classical description of a $\mathrm{poly}(\lambda)$-size quantum circuit, and each $\boldsymbol{\rho}_\lambda$ is some (not necesarily efficiently computable) non-uniform $\mathrm{poly}(\lambda)$-qubit quantum advice.

We will denote the trace distance between two QRVs $\mathbf{x}$ and $\mathbf{y}$ with $\|\mathbf{x} - \mathbf{y}\|_1$ and for infinite sequences of QRVs $\{\mathbf{x}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathbf{y}_\lambda\}_{\lambda \in \mathbb{N}}$ we write

$$\{\mathbf{x}_\lambda\}_{\lambda \in \mathbb{N}} \approx_s \{\mathbf{y}_\lambda\}_{\lambda \in \mathbb{N}}$$

to indicate that there exists a negligible function $\mu(\cdot)$ such that $||\mathbf{x}_\lambda - \mathbf{y}_\lambda||_1 \leq \mu(\lambda)$. Here, the s refers to "statistical" indistinguishability.

In addition, we write

$$\{\mathbf{x}_\lambda\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{y}_\lambda\}_{\lambda \in \mathbb{N}}$$

to indicate that there exists a negligible function $\mu(\cdot)$ such that for all QPT distinguishers $\mathcal{D} = \{\mathcal{D}_\lambda, \mathbf{d}_\lambda\}_\lambda$,

$$|\Pr[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathbf{x}_\lambda) = 1] - \Pr[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathbf{y}_\lambda) = 1]| \leq \mu(\lambda).$$

Here, the c refers to "computational" indistinguishability.

Let $\mathscr{C}_n$ and $\mathscr{P}_n$ denote the n-qubit Clifford and Pauli groups, respectively. Let $\mathbf{0}$ refer to a 0 state and $\mathbf{T}$ refer to a T state. $\mathbf{0}^n$ denotes n copies of a single qubit 0 state and likewise for $\mathbf{T}^n$. Let X and Z be the Pauli matrices, i.e.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

3.2 Clifford Authentication Code

Definition 3.1 (Clifford Authentication Code)**.** *The n-qubit λ-trap Clifford authentication code consists of the following algorithms, which encode an n-qubit state $\mathbf{x}$ with key $C \in \mathscr{C}_{n+\lambda}$.*

- $\mathsf{Enc}(C, \mathbf{x})$*: Compute* $C(\mathbf{x}, \mathbf{0}^\lambda) := \widehat{\mathbf{x}}$.
- $\mathsf{Dec}(C, \widehat{\mathbf{x}})$*: Compute* $(\mathbf{x}', \mathbf{y}) := C^\dagger(\widehat{\mathbf{x}})$ *(where* $\mathbf{x}'$ *is the first* n *registers of the result and* $\mathbf{y}$ *is the final* λ*) and measure* $\mathbf{y}$ *in the standard basis. If the outcome is* 0^λ*, return* $\mathbf{x}'$*, and otherwise return* $|\bot\rangle\langle\bot|$.

This authentication code satisfies the following property. For any QRV $(\mathbf{x}, \mathbf{z})$ *and any CPTP map* Adv *acting on the encoding and side-information* $\mathbf{z}$*, there exist maps* $\mathcal{B}_0$*,* $\mathcal{B}_1$ *acting on* $\mathbf{z}$ *such that* $\mathcal{B}_0 + \mathcal{B}_1$ *is CPTP (completely positive trace preserving), and*

$$\left\| \mathop{\mathbb{E}}_{C \leftarrow \mathscr{C}_{n+\lambda}} [\mathsf{Dec}(C, \mathsf{Adv}(\mathsf{Enc}(C, \mathbf{x}), \mathbf{z}))] - ((\mathbf{x}, \mathcal{B}_0(\mathbf{z})) + (|\bot\rangle\langle\bot|, \mathcal{B}_1(\mathbf{z}))) \right\|_1 = \mathrm{negl}(\lambda).$$

3.3 Two-Party Quantum Computation

Definition 3.2 (Secure Two-Party Quantum Computation)**.** *We follow the standard real/ideal world paradigm for defining secure computation [Gol04]. Consider a two-party quantum functionality captured by a family of quantum circuits* $\mathcal{Q} = \{Q_\lambda\}_{\lambda \in \mathbb{N}}$ *where* Q_λ *has* $n_A(\lambda) + n_B(\lambda)$ *input qubits,* $n_Z(\lambda)$ *auxiliary* $\mathbf{0}$ *qubits, and* $m_A(\lambda) + m_B(\lambda)$ *output qubits. We will consider a non-uniform QPT adversary* $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$ *that* *corrupts either party A or party B. Below we give the security definition for the case when* Adv *corrupts party A since the definition for corrupted B is symmetric.*

Let Π be a two-party protocol for computing Q. For security parameter λ and any collection of (potentially entangled) quantum states $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$, where $\mathbf{x}_A$ is A's input to Q_λ (on $n_A(\lambda)$ registers), $\mathbf{x}_B$ is B's input to Q_λ (on $n_B(\lambda)$ registers), and $\mathbf{aux}$ is some side information (on an arbitrary number of registers), arbitrarily shared between parties A and B, we define the quantum random variable $\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ *as follows.* $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$ *interacts with an honest party B algorithm $B(1^\lambda, \mathbf{x}_B)$ participating in protocol Π, after which B outputs $\mathbf{y}_B$ (on $m_B(\lambda)$ registers) and* Adv *outputs a final state $\boldsymbol{\rho}_{\mathsf{out}}$ (an arbitrary function computed on an arbitrary subset of the registers that comprise its view). The random variable* $\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ *then consists of $\mathbf{y}_B$ along with $\boldsymbol{\rho}_{\mathsf{out}}$.*

For any Adv*, we require the existence of a simulator* $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda\in\mathbb{N}}$ *that takes as input $(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$, has one-time access to an ideal functionality $\mathcal{I}[\mathbf{x}_B](\cdot)$, and outputs a state $\boldsymbol{\rho}_{\mathsf{out}}$. The ideal functionality $\mathcal{I}[\mathbf{x}_B](\cdot)$ accepts an input $\mathbf{x}'_A$ on $n_A(\lambda)$ registers, prepares $n_Z(\lambda)$ $\mathbf{0}$ qubits, applies Q_λ to $(\mathbf{x}'_A, \mathbf{x}_B, \mathbf{0}^{n_Z(\lambda)})$ to recover $(\mathbf{y}_A, \mathbf{y}_B)$ and returns $\mathbf{y}_A$ to* Sim_λ*. It then waits for either an* abort *or* ok *message from* Sim_λ*. In the case of* ok *it outputs $\mathbf{y}_B$ and in the case of* abort *it outputs $\perp$ (note that this output is not given to* Sim_λ*). Now, we define the quantum random variable* $\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ *to consist of the output of $\mathcal{I}[\mathbf{x}_B](\cdot)$ and the final state $\boldsymbol{\rho}_{\mathsf{out}}$ of* $\mathsf{Sim}_\lambda^{\mathcal{I}[\mathbf{x}_B](\cdot)}(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$.

Finally, we say that Π computes Q with security against malicious party A if for all non-uniform QPT $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$*, there exists a QPT* $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda\in\mathbb{N}}$ *such that,*

$$\begin{aligned}&\big\{\{\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{(\mathbf{x}_A,\mathbf{x}_B,\mathbf{aux})\in\mathcal{S}[Q]_\lambda}\big\}_{\lambda\in\mathbb{N}}\\ &\approx_c \big\{\{\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{(\mathbf{x}_A,\mathbf{x}_B,\mathbf{aux})\in\mathcal{S}[Q]_\lambda}\big\}_{\lambda\in\mathbb{N}},\end{aligned}$$

where $\mathcal{S}[Q]_\lambda$ ranges over all all quantum states $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ where $\mathbf{x}_A$ is on $n_B(\lambda)$ registers, $\mathbf{x}_B$ is on $n_B(\lambda)$ registers, and $\mathbf{aux}$ is on any number of registers.

As mentioned above, we define analogous random variables for the case of a corrupted B, and require that

$$\begin{aligned}&\big\{\{\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{(\mathbf{x}_A,\mathbf{x}_B,\mathbf{aux})\in\mathcal{S}[Q]_\lambda}\big\}_{\lambda\in\mathbb{N}}\\ &\approx_c \big\{\{\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{(\mathbf{x}_A,\mathbf{x}_B,\mathbf{aux})\in\mathcal{S}[Q]_\lambda}\big\}_{\lambda\in\mathbb{N}}.\end{aligned}$$

3.4 Two-Message Two-Party Classical Computation

As a building block, we will use post-quantum maliciously-secure two-message two-party classical computation in the CRS model where one party receives output. We will require that the simulator is straight-line and black-box. We will refer to such a protocol simply as $\mathsf{2PC}$.

Definition 3.3 (Post-Quantum Two-Message Two-Party Computation)**.** $\mathsf{2PC}$ *is defined by four algorithms* $(\mathsf{2PC.Gen}, \mathsf{2PC}_1, \mathsf{2PC}_2, \mathsf{2PC}_{\mathsf{out}})$*. We will keep the convention that party B, with input x_B, first computes* $\mathsf{2PC}_1$*, then party A, with input x_A, computes* $\mathsf{2PC}_2$*, and finally party B recovers its output y with* $\mathsf{2PC}_{\mathsf{out}}$*. The syntax of these algorithms is as follows, where C is the description of the circuit to be computed.*

- $\mathsf{crs} \leftarrow \mathsf{2PC.Gen}(1^\lambda)$.
- $(m_1, \mathsf{st}) \leftarrow \mathsf{2PC}_1(1^\lambda, C, \mathsf{crs}, x_B)$.
- $m_2 \leftarrow \mathsf{2PC}_2(1^\lambda, C, \mathsf{crs}, m_1, x_A)$.
- $y \leftarrow \mathsf{2PC}_{\mathsf{out}}(1^\lambda, \mathsf{st}, m_2)$.

Let $C = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ be a (potentially randomized) family of classical circuits where C_λ takes as input $(x_A, x_B) \in \{0,1\}^{n_A(\lambda)+n_B(\lambda)}$ and outputs y. Consider the case of an adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ *corrupting party A. For every $\lambda \in \mathbb{N}$, the the view of the environment in the real execution is denoted by a random variable* $\mathsf{REAL}_{\mathsf{C},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})$*, where x_A is party A's input, x_B is party B's input, and*

$\mathbf{aux}$ *is some potentially quantum side information. The random variable consists of* $(\boldsymbol{\rho}_{\mathsf{out}}, y_B)$*, where* $\boldsymbol{\rho}_{\mathsf{out}}$ *is* Adv_λ*'s final output after interacting with an honest* B *algorithm* $B(1^\lambda, x_B)$*, and* y_B *is* B*'s output.*

We require the existence of a QPT simulator $\mathsf{2PC.Sim}_A = (\mathsf{2PC.Sim}_A^{(1)}, \mathsf{2PC.Sim}_A^{(2)})$ *that interacts with any non-uniform QPT adversary* $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$ *corrupting party* A*.* $\mathsf{2PC.Sim}_A$ *has the following syntax.*

- $\mathsf{2PC.Sim}_A^{(1)}(1^\lambda)$ *generates* $(\mathsf{crs}, \tau, m_1)$*, sends* (crs, m_1) *to* $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{aux})$*, and receives back* m_2*.*
- $\mathsf{2PC.Sim}_A^{(2)}(1^\lambda, x_A, \tau, m_2)$ *computes either* x'_A *or* $\bot$*, which it forwards to an ideal functionality* $\mathcal{I}_A[x_B](\cdot)$*.*

$\mathcal{I}[x_B](\cdot)$ *operates as follows. It takes an input* x'_A *or* $\bot$*, and in the non-*$\bot$ *case it computes and outputs* $y \leftarrow C_\lambda(x'_A, x_B)$ *(note this is not returned to the simulator), and in the* $\bot$ *case it outputs* $\bot$*. The random variable* $\mathsf{IDEAL}_{\mathsf{C},A}(\mathsf{2PC.Sim}_A, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})$ *consists of the output of* $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{aux})$ *after interacting with the simulator, along with the output of* $\mathcal{I}[x_B](\cdot)$*. We require that for all non-uniform QPT* $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$*, and for all* $(x_A, x_B, \mathbf{aux})$*,*

$$\begin{aligned}&\big\{\{\mathsf{REAL}_{\mathsf{C},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})\}_{(x_A, x_B, \mathbf{aux}) \in \mathcal{S}[C]_\lambda}\big\}_{\lambda \in \mathbb{N}} \\ &\approx_c \big\{\{\mathsf{IDEAL}_{\mathsf{C},A}(\mathsf{2PC.Sim}_A, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})\}_{(x_A, x_B, \mathbf{aux}) \in \mathcal{S}[C]_\lambda}\big\}_{\lambda \in \mathbb{N}},\end{aligned}$$

where $\mathcal{S}[C]$ *ranges over all* $(x_A, x_B, \mathbf{aux})$ *where* $x_A \in \{0,1\}^{n_A(\lambda)}$*,* $x_B \in \{0,1\}^{n_B(\lambda)}$*, and* $\mathbf{aux}$ *is a quantum state on any number of registers.*

We require an analogous security property in the case that Adv *corrupts party* B*. Here, the syntax of* $\mathsf{2PC.Sim} = (\mathsf{2PC.Sim}_B^{(1)}, \mathsf{2PC.Sim}_B^{(2)})$ *is as follows.*

- $\mathsf{2PC.Sim}_B^{(1)}(1^\lambda)$ *generates* (crs, τ)*, sends* crs *to* $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, x_B, \mathbf{aux})$*, and receives back* m_1*.*
- $\mathsf{2PC.Sim}_B^{(2)}(1^\lambda, \tau, m_1)$ *takes the adversary's message* m_1 *and either extracts an input* x'_B *or* $\bot$*, which it forwards to an ideal functionality* $\mathcal{I}[x_A](\cdot)$*. In the non-*$\bot$ *case,* $\mathcal{I}[x_A]$ *computes and returns* $y \leftarrow C(x_A, x'_B)$ *to the simulator and outputs* ok*. In the* $\bot$ *case it outputs* abort*, and the simulator send* $\bot$ *to* Adv *and simulation ends.*
- $\mathsf{2PC.Sim}_B^{(3)}(1^\lambda, \tau, y)$ *receives an output* y *from the ideal functionality and uses it to form a second round message* m_2*, which it sends to* Adv_λ*.*

This defines analogous random variables, and we require that

$$\begin{aligned}&\big\{\{\mathsf{REAL}_{\mathsf{C},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})\}_{(x_A, x_B, \mathbf{aux}) \in \mathcal{S}[C]_\lambda}\big\}_{\lambda \in \mathbb{N}} \\ &\approx_c \big\{\{\mathsf{IDEAL}_{\mathsf{C},B}(\mathsf{2PC.Sim}_B, \boldsymbol{\rho}_\lambda, x_A, x_B, \mathbf{aux})\}_{(x_A, x_B, \mathbf{aux}) \in \mathcal{S}[C]_\lambda}\big\}_{\lambda \in \mathbb{N}}.\end{aligned}$$

A secure two-party computation protocol satisfying Definition 3.3 can be obtained based on any post-quantum maliciously-secure two-message OT with straight-line simulation, via [IPS08, IKO+11]. The non-interactive secure two-party protocol from [IPS08, Appendix B] is based on Yao's garbled circuit technique [Yao86] along with a cut-and-choose mechanism for proving that a garbled circuit is computed correctly. The cut-and-choose is non-interactive in the OT-hybrid model. This can be cast in the simpler setting of two-party computation with a CRS, where we replace the ideal calls to the OT with a post-quantum secure two-message OT with straight-line simulation (that is auxiliary-input secure). The latter can be based on the quantum hardness of the learning with errors (QLWE) problem [PVW08].

In Section 6, we will rely on reusable post-quantum two-party computation with straight-line simulation, in order to obtain reusable malicious designated-verifier NIZKs for QMA. We point out that [LQR+19] build reusable (post-quantum) two-party computation (with straight-line simulation) assuming (post-quantum) malicious MDV-NIZKs for NP, and (post-quantum) oblivious transfer. Both can be obtained from QLWE [LQR+19, PVW08].

3.5 Useful Lemmas

Lemma 3.4 (Magic State Distillation [BK05, DGJ+20])**.** *For any parameters $n, \lambda \in \mathbb{N}$, there exists a* $\mathrm{poly}(n,\lambda)$ *size* $\mathsf{C}+\mathsf{M}$ *circuit Q from $n\lambda$ input qubits to n outputs qubits such that the following holds. Take any state $\mathbf{x}$ on $n\lambda+\lambda$ qubits. Apply a uniformly random permutation to the registers of $\mathbf{x}$ and then measure the final λ qubits in the T-basis. Let $\widetilde{\mathbf{x}}$ be the remaining $n\lambda$ registers. If all measurements returned 0, then* $\left\|Q(\widetilde{\mathbf{x}}) - \mathbf{T}^n\right\|_1 = \mathrm{negl}(\lambda)$.

Proof. This follows from applying [DGJ+20, Lemma I.1] with parameters $k = \lambda$, $\delta = 1/2$ followed by [DGJ+20, Lemma 2.7] with parameters $m = n\lambda$, $w = n\lambda/2, t = n$. □

Lemma 3.5 ([DGJ+20])**.** *For any $n \in \mathbb{N}$ and projector Π on $2n$ qubits, define the quantum channel $\mathcal{L}^{\Pi}$ by*

$$\mathcal{L}^{\Pi}(\mathbf{x}) \coloneqq \Pi\mathbf{x}\Pi + \left|\bot\right\rangle\left\langle\bot\right| \mathrm{Tr}[(\mathbb{I}^{2n} - \Pi)\mathbf{x}],$$

where $\left|\bot\right\rangle$ is a distinguished state on $2n$ qubits with $\Pi\left|\bot\right\rangle = 0$. Let $\Pi_{\mathsf{Full}} \coloneqq \left|0^{2n}\right\rangle\left\langle 0^{2n}\right|$ and let $\Pi_{\mathsf{Half}} \coloneqq \mathbb{I}^n \otimes \left|0^n\right\rangle\left\langle 0^n\right|$. Then for any QRV $\mathbf{x}$ on $2n$ registers,

$$\left\|\mathcal{L}^{\Pi_{\mathsf{Full}}}(\mathbf{x}) - \mathop{\mathbb{E}}_{U \leftarrow \mathsf{GL}(2n,\mathbb{F}_2)}\left[\mathcal{L}^{\Pi_{\mathsf{Half}}}(U(\mathbf{x}))\right]\right\|_1 = \mathrm{negl}(n).$$

3.6 Quantum Multi-Key Fully-Homomorphic Encryption

We use a quantum multi-key fully-homomorphic encryption scheme that supports classical encryption of classical ciphertexts. We do not require compactness or the classicality-preserving property as required by [ABG+20], but we do require a form a circuit-privacy, presented below as ciphertext re-randomization.

Definition 3.6 (Quantum Multi-Key Fully-Homomorphic Encryption [Mah18, ABG+20])**.** *A quantum multi-key fully-homomorphic encryption scheme is given by seven algorithms (*$\mathsf{QMFHE.Gen}, \mathsf{QMFHE.KeyGen}$*,* $\mathsf{QMFHE.CEnc}$*,* $\mathsf{QMFHE.Enc}$*,* $\mathsf{QMFHE.Eval}$*,* $\mathsf{QMFHE.Rerand}$*,* $\mathsf{QMFHE.Dec}$*) with the following syntax.*

- $\mathsf{crs} \leftarrow \mathsf{QMFHE.Gen}(1^\lambda)$*: A PPT algorithm that outputs a classical common reference string.*
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{QMFHE.KeyGen}(1^\lambda, \mathsf{crs})$ *: A PPT algorithm that given a security parameter, samples a classical public key and a classical secret key.*
- $\mathsf{ct} \leftarrow \mathsf{QMFHE.CEnc}(\mathsf{pk}, x)$ *: A PPT algorithm that takes as input a bit x and outputs a classical ciphertext.*
- $\mathbf{ct} \leftarrow \mathsf{QMFHE.Enc}(\mathsf{pk}, \mathbf{x})$ *: A QPT algorithm that takes as input a qubit $\mathbf{x}$ and outputs a quantum ciphertext.*
- $\widehat{\mathbf{ct}} \leftarrow \mathsf{QMFHE.Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_n), Q, (\mathbf{ct}_1, \ldots, \mathbf{ct}_n))$*: A QPT algorithm that takes as input a set of n public keys, a quantum circuit Q, and a set of n (classical or quantum) ciphertexts, and outputs an evaluated ciphertext $\widehat{\mathbf{ct}}$.*
- $\widetilde{\mathbf{ct}} \leftarrow \mathsf{QMFHE.Rerand}((\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mathbf{ct})$*: A QPT algorithm that re-randomizes a ciphertext $\mathbf{ct}$ encrypted under a set of n public keys*
- $\mathbf{x} \leftarrow \mathsf{QMFHE.Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_n), \mathbf{ct})$*: A QPT algorithm that takes as input a set of n secret keys and a quantum ciphertext $\mathbf{ct}$ and outputs a qubit.*

The scheme satisfies the following.

1. **Quantum Semantic Security:** *The encryption algorithm maintains quantum semantic security.*
2. **Quantum Homomorphism:** *For any polynomial-size quantum circuit Q, input state $\mathbf{x}_1, \ldots, \mathbf{x}_n$, crs* $\mathsf{crs} \in \mathsf{QMFHE.Gen}(1^\lambda)$*, and key pairs* $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n) \in \mathsf{QMFHE.KeyGen}(1^\lambda, \mathsf{crs})$*, it holds that $\mathbf{y}_0 \approx_s \mathbf{y}_1$, where $\mathbf{y}_0, \mathbf{y}_1$ are QRVs defined as follows:*

- $\mathbf{y}_0$*: For each* $i \in [n]$*, encrypt each classical bit of* $\mathbf{x}_i$ *with* $\mathsf{QMFHE.CEnc}(\mathsf{pk}_i, \cdot)$ *and the rest with* $\mathsf{QMFHE.Enc}(\mathsf{pk}_i, \cdot)$*. Execute* $\mathsf{QMFHE.Eval}((\mathsf{pk}_1, \dots, \mathsf{pk}_n), Q, \cdot)$ *on the* n *encryptions to obtain* $\widehat{\mathbf{ct}}$*. Then output* $\mathsf{QMFHE.Dec}((\mathsf{sk}_1, \dots, \mathsf{sk}_n), \mathsf{QMFHE.Rerand}(\widehat{\mathbf{ct}}))$*.*
- $\mathbf{y}_1$*: Output* $Q(\mathbf{x}_1, \dots, \mathbf{x}_n)$*.*

3. **Ciphertext Re-randomization:** *For any* $\mathsf{crs} \in \mathsf{QMFHE.Gen}(1^\lambda)$*, key pairs* $(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_n, \mathsf{sk}_n) \in \mathsf{QMFHE.KeyGen}(1^\lambda, \mathsf{crs})$*, and ciphertexts* $\mathbf{ct}_1, \mathbf{ct}_2$ *such that*

$$\mathsf{QMFHE.Dec}((\mathsf{sk}_1, \dots, \mathsf{sk}_n), \mathbf{ct}_1) = \mathsf{QMFHE.Dec}((\mathsf{sk}_1, \dots, \mathsf{sk}_n), \mathbf{ct}_2),$$

it holds that

$$\mathsf{QMFHE.Rerand}((\mathsf{pk}_1, \dots, \mathsf{pk}_n), \mathbf{ct}_1) \approx_s \mathsf{QMFHE.Rerand}((\mathsf{pk}_1, \dots, \mathsf{pk}_n), \mathbf{ct}_2).$$

We now sketch how to add the ciphertext re-randomization property to the QMFHE scheme constructed in [ABG$^+$20] via "noise-flooding". An evaluated ciphertext encrypting the quantum state $\boldsymbol{\rho}$ will have the form

$$\mathsf{MFHE.Enc}((\mathsf{pk}_1, \dots, \mathsf{pk}_n), (x, z)), X^x Z^z \boldsymbol{\rho},$$

where MFHE is a *classical* multi-key fully-homomorphic encryption scheme. Thus, it suffices to show how to add ciphertext re-randomization to the classical multi-key FHE scheme of [MW16].

It is well-known that standard single-key FHE schemes from the literature ([GSW13]) are statistically re-randomizable. Now to construct MFHE with ciphertext re-randomization, we can append to each MFHE public key a freshly sampled GSW encryption of its corresponding secret key. To re-randomize a MFHE ciphertext encrypted under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_n$, one can compute the partial decryption under each corresponding GSW ciphertext, resulting in n ciphertexts whose plaintexts sum to $\mu(q/2) + e$, where μ was the bit encrypted under MFHE. Then, add a random additive secret sharing of e' for a large enough e' under the encryptions and re-randomize each. The result is an random additive sharing of $\mu(q/2) + e + e'$ under re-randomized GSW ciphertexts, where $e + e' \approx_s e''$ for some distribution e'' independent of the computation.

3.7 Non-Interactive Equivocal Commitment

Definition 3.7 (Equivocal Commitment)**.** *A quantum-secure statistically-binding non-interactive equivocal commitment is given by three algorithms* $(\mathsf{Com.Gen}, \mathsf{Com.Enc}, \mathsf{Com.Ver})$ *with the following syntax.*

- $\mathsf{crs} \leftarrow \mathsf{Com.Gen}(1^\lambda)$.
- $\mathsf{cmt} \coloneqq \mathsf{Com.Enc}(1^\lambda, \mathsf{crs}, m; r)$.
- $b \leftarrow \mathsf{Com.Ver}(1^\lambda, \mathsf{crs}, \mathsf{cmt}, m, r)$.

It satisfies the following notion of correctness. For any $m \in \{0,1\}^*$*,*

$$\Pr\left[b = 1 : \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Com.Gen}(1^\lambda), r \leftarrow \{0,1\}^\lambda \\ \mathsf{cmt} \coloneqq \mathsf{Com.Enc}(1^\lambda, \mathsf{crs}, m; r), b \leftarrow \mathsf{Com.Ver}(1^\lambda, \mathsf{crs}, \mathsf{cmt}, m, r) \end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

It satisfies the statistical binding property.

$$\Pr_{\mathsf{crs} \leftarrow \mathsf{Com.Gen}(1^\lambda)}\left[\begin{array}{l} \exists(\mathsf{cmt}, m_0, m_1, r_0, r_1), m_0 \neq m_1 \ \textit{s.t.} \\ \mathsf{Com.Ver}(1^\lambda, \mathsf{crs}, \mathsf{cmt}, m_0, r_0) = 1 = \mathsf{Com.Ver}(1^\lambda, \mathsf{crs}, \mathsf{cmt}, m_1, r_1) \end{array}\right] = \mathrm{negl}(\lambda).$$

Finally, it satisfies the following notion of security (hiding). There exists algorithms $\mathsf{Com.Sim.Gen}, \mathsf{Com.Sim.Open}$ *such that for any* $m \in \{0,1\}^*$*,*

$$\Pr\left[b = 1 : \begin{array}{r} (\mathsf{crs}, \mathsf{cmt}, \tau) \leftarrow \mathsf{Com.Sim.Gen}(1^\lambda) \\ r_m \leftarrow \mathsf{Com.Sim.Open}(1^\lambda, \tau, m), b \leftarrow \mathsf{Com.Ver}(1^\lambda, \mathsf{crs}, \mathsf{cmt}, m, r_m) \end{array}\right] = 1 - \mathrm{negl}(\lambda),$$

and

$$\left\{(\mathsf{crs},\mathsf{cmt}) : \begin{array}{r}\mathsf{crs} \leftarrow \mathsf{Com.Gen}(1^\lambda) \\ \mathsf{cmt} \leftarrow \mathsf{Com.Enc}(1^\lambda,\mathsf{crs},m)\end{array}\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{(\mathsf{crs},\mathsf{cmt}) : (\mathsf{crs},\mathsf{cmt},\tau) \leftarrow \mathsf{Com.Sim.Gen}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

A commitment scheme satisfying the above definition can be based on any quantum-secure one-way function [Nao91].

3.8 Garbled Circuit

Definition 3.8 (Garbled Circuit)**.** *A garbling scheme for circuits is a tuple of PPT algorithms* $(\mathsf{Garble},\mathsf{GEval})$*.* Garble *is the circuit garbling procedure and* GEval *is the corresponding evaluation procedure. More formally:*

- $(\widetilde{C},\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$*:* Garble *takes as input a security parameter* 1^λ*, a classical circuit* C*, and outputs a* garbled circuit $\widetilde{C}$ *along with labels* $\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}$*, where* n *is the length of the input to* C*.*
- $y \leftarrow \mathsf{GEval}\left(\widetilde{C},\{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right)$*: Given a garbled circuit* $\widetilde{C}$ *and a sequence of input labels* $\{\mathsf{lab}_{i,x_i}\}_{i\in[n]}$*,* GEval *outputs a string* y*.*

Correctness. *For correctness, we require that for any classical circuit* C *and input* $x \in \{0,1\}^n$ *we have that:*

$$\Pr\left[C(x) = \mathsf{GEval}\left(\widetilde{C},\{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right)\right] = 1,$$

where $(\widetilde{C},\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$*.*

Security. *For security, we require that there exists a PPT simulator* GSim *such that for any classical circuit* C *and input* $x \in \{0,1\}^n$*, we have that*

$$\left(\widetilde{C},\{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right) \approx_c \mathsf{GSim}\left(1^\lambda, 1^n, 1^{|C|}, C(x)\right),$$

where $(\widetilde{C},\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$*.*

4 A Garbling Scheme for Clifford + Measurement Circuits

In this section, we formalize and prove the security of a method sketched in [BY20, §2.5] for garbling Clifford plus measurement circuits. Note that this is not the main garbling scheme analyzed in [BY20], but it is a scheme that is sketched there informally. We begin by giving the formal definition of a Clifford + measurement circuit, as well as our definition of a garbling scheme for such circuits.

Definition 4.1 (Clifford + Measurement $(\mathsf{C}+\mathsf{M})$ Circuit)**.** *A Clifford + Measurement* $(\mathsf{C}+\mathsf{M})$ *circuit with parameters* $\{n_i,k_i\}_{i\in[d]}$ *operates on* $n_0 := n_1 + k_1$ *input qubits and applies* d *alternating layers of Clifford unitary and computational basis measurements, during which a total of* $k := k_1 + \cdots + k_d$ *of the input qubits are measured. It is specified by* $(F_0, f_1, \dots, f_d)$*, where* F_0 *is a Clifford unitary, and each* f_i *is a classical circuit which takes as input the result of computational basis measurements on the ith layer, and outputs a Clifford unitary* F_i*. In layer* $i \in [d]$*,* k_i *qubits are measured and* n_i *qubits are left over. The circuit is evaluated by first applying* F_0 *to the* n_0 *input qubits. Then the following steps are performed for* $i = 1, \dots, d$*:*

- *Measure the remaining* k_i *qubits in the computational basis, resulting in outcomes* $m_i \in \{0,1\}^{k_i}$*.*
- *Evaluate* $f_i(m_i)$ *to obtain a classical description of a Clifford* $F_i \in \mathscr{C}_{n_i}$*.*

- *Apply F_i to the first n_i registers.*

The output of the circuit is the result of applying F_d to the final n_d registers.

It is well-known ([BK05]) that any polynomial-size quantum circuit can be written as a $\mathsf{C}+\mathsf{M}$ circuit with polynomial-size parameters $\{n_i, k_i\}_{i\in[d]}$. The transformation maintains correctness as along as sufficient $\mathbf{T}$ states are appended to the input during evaluation.

Definition 4.2 (Garbling Scheme for $\mathsf{C}+\mathsf{M}$ Circuits)**.** *A Garbling Scheme for $\mathsf{C}+\mathsf{M}$ Circuits consists of three procedures* $(\mathsf{QGarble}, \mathsf{QGEval}, \mathsf{QGSim})$ *with the following syntax.*

- $(E_0, \widetilde{Q}) \leftarrow \mathsf{QGarble}(1^\lambda, Q)$*: A* classical *PPT procedure that takes as input the security parameter and a* $\mathsf{C}+\mathsf{M}$ *circuit and outputs a Clifford "input garbling" matrix E_0 and a quantum garbled circuit $\widetilde{Q}$.*
- $\mathbf{x}_{\mathsf{out}} \leftarrow \mathsf{QGEval}(\widetilde{\mathbf{x}}_{\mathsf{inp}}, \widetilde{Q})$*: A QPT procedure that takes as input a garbled input $\widetilde{\mathbf{x}}_{\mathsf{inp}}$ and a garbled* $\mathsf{C}+\mathsf{M}$ *circuit $\widetilde{Q}$, and outputs a quantum state $\mathbf{x}_{\mathsf{out}}$.*
- $(\widetilde{\mathbf{x}}_{\mathsf{inp}}, \widetilde{Q}) \leftarrow \mathsf{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, \mathbf{x}_{\mathsf{out}})$*: A QPT procedure that takes as input the security parameter, parameters for a* $\mathsf{C}+\mathsf{M}$ *circuit, and an output state, and outputs a simulated garbled input and garbled circuit.*

Correctness. *For any* $\mathsf{C}+\mathsf{M}$ *circuit Q with parameters $\{n_i, k_i\}_{i\in[d]}$, and n_0-qubit input state $\mathbf{x}_{\mathsf{inp}}$ along with (potentially entangled) auxiliary information $\mathbf{z}$,*

$$\left\{\left(\mathsf{QGEval}\left(E_0\left(\mathbf{x}_{\mathsf{inp}}, \mathbf{0}^{k\lambda}\right), \widetilde{Q}\right), \mathbf{z}\right) : \left(E_0, \widetilde{Q}\right) \leftarrow \mathsf{QGarble}\left(1^\lambda, Q\right)\right\} \approx_s \left(Q\left(\mathbf{x}_{\mathsf{inp}}\right), \mathbf{z}\right).$$

Security. *For any* $\mathsf{C}+\mathsf{M}$ *circuit Q with parameters $\{n_i, k_i\}_{i\in[d]}$, and n_0-qubit input state $\mathbf{x}_{\mathsf{inp}}$ along with (potentially entangled) auxiliary information $\mathbf{z}$,*

$$\left\{\left(E_0\left(\mathbf{x}_{\mathsf{inp}}, \mathbf{0}^{k\lambda}\right), \widetilde{Q}, \mathbf{z}\right) : \left(E_0, \widetilde{Q}\right) \leftarrow \mathsf{QGarble}\left(1^\lambda, Q\right)\right\} \approx_c \left(\mathsf{QGSim}\left(1^\lambda, \{n_i, k_i\}_{i\in[d]}, Q(\mathbf{x}_{\mathsf{inp}})\right), \mathbf{z}\right).$$

Before formally describing the concrete garbling scheme for $\mathsf{C}+\mathsf{M}$ circuits, we give a formal definition of a process LabEnc for sampling a "label encoding" unitary given a set of classical garbled circuit labels. For λ-bit strings s_0, s_1 and a bit b, let $C_b^{s_0,s_1}$ be the Clifford acting on $\lambda+1$ qubits, operating as follows:

- Apply Z_b to the first qubit. Looking ahead, b will be chosen at random so that Z_b will have the effect of a Z-twirl, which is equivalent to a measurement in the computational basis.
- Map $|0, 0^\lambda\rangle$ to $|0, s_0\rangle$, and $|1, 0^\lambda\rangle$ to $|1, s_1\rangle$.

$\mathsf{LabEnc}(\overline{\mathsf{lab}})$**:** Takes as input $\overline{\mathsf{lab}} = \{\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}\}_{i\in[n]}$, where the $\mathsf{lab}_{i,b}$ are λ-bit strings. Draws n random bits $b_1, \ldots, b_n \leftarrow \{0,1\}$, and outputs $\bigotimes_{i\in[n]} C_{b_i}^{\mathsf{lab}_{0,i}, \mathsf{lab}_{1,i}}$,

Lemma 4.3. *Let $m > n$. For any m-qubit state $|\phi\rangle$ and set of labels $\overline{\mathsf{lab}} = \{\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}\}_{i\in[n]}$, where the $\mathsf{lab}_{i,b}$ are λ-bit strings,*

$$L\,|\phi'\rangle\,\langle\phi'|\,L^\dagger = \mathbb{E}_{\mathsf{inp}}\,|\phi'_{\mathsf{inp}}\rangle\,\langle\phi'_{\mathsf{inp}}| \otimes |\mathsf{lab}_{1,\mathsf{inp}_1}, \ldots, \mathsf{lab}_{n,\mathsf{inp}_n}\rangle\,\langle\mathsf{lab}_{1,\mathsf{inp}_1}, \ldots, \mathsf{lab}_{n,\mathsf{inp}_n}|\ ,$$

where $L \leftarrow \mathsf{LabEnc}(\overline{\mathsf{lab}})$, $|\phi'\rangle$ is the $(m+n\lambda)$-qubit state consisting of $|\phi\rangle$ and $n\lambda$ ancillary 0 states, $|\phi'_{\mathsf{inp}}\rangle$ is the post-measurement state on the first $m-n$ qubits, conditioned on measuring the last n qubits and obtaining outcome inp, and the expectation is taken over $\mathsf{inp} \in \{0,1\}^n$ distributed according to the result of measuring the last n qubits of $|\phi\rangle$ in the computational basis.

Proof. We can write $|\phi'\rangle$ as follows:

$$|\phi'\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |\phi_x\rangle |x\rangle \ .$$

for some $\alpha_x \in \mathbb{C}$. Then,

$$\mathbb{E}_{L\leftarrow\mathsf{LabEnc}(\overline{\mathsf{lab}})} L|\phi'\rangle\langle\phi'| L^\dagger = \mathbb{E}_{z\leftarrow\{0,1\}^n} \mathbb{I}{\otimes}Z^z{\otimes}\mathbb{I} \left(\sum_{x\in\{0,1\}^n} \alpha_x |\phi_x\rangle |x\rangle |\mathsf{lab}_x\rangle\right) \left(\sum_{x'\in\{0,1\}^n} \alpha_{x'} \langle\phi_{x'}| \langle x| \langle\mathsf{lab}_{x'}|\right) \mathbb{I}{\otimes}Z^z{\otimes}\mathbb{I}\,,$$

where $|\mathsf{lab}_x\rangle = |\mathsf{lab}_{1,x_1}, \ldots, \mathsf{lab}_{1,x_n}\rangle$.

By a well-known property of the Pauli-Z twirl, the above is equal to:

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 |\phi_x\rangle\langle\phi_x| \otimes |x\rangle\langle x| \otimes |\mathsf{lab}_x\rangle\langle\mathsf{lab}_x| \ ,$$

which implies the desired statement. □

Now, we are ready to describe formally the garbling scheme $(\mathsf{QGarble}, \mathsf{QGEval}, \mathsf{QGSim})$ for $\mathsf{C}+\mathsf{M}$ circuits sketched by [BY20]. Let $(\mathsf{Garble}, \mathsf{GEval}, \mathsf{GSim})$ be a classical garbling scheme.

$\mathsf{QGarble}(1^\lambda, Q)$**:** Takes as input a $\mathsf{C}+\mathsf{M}$ circuit Q with parameters $\{n_i, k_i\}_{i\in[d]}$.

1. For $i \in [0,\ldots,d]$, define $h_i \coloneqq k - \sum_{j=1}^{i} k_j$, so that $h_0 = k, h_1 = k - k_1, h_2 = k - k_1 - k_2$, and so on.
2. For each $i \in [0,\ldots,d]$, sample $E_i \leftarrow \mathscr{C}_{n_i+h_i\lambda}$.
3. For each $i \in [d]$, let f_i be the classical circuit (derived from the description of Q) that takes as input k_i bits interpreted as the outcomes of computational basis measurements in layer i and outputs a Clifford circuit $F_i \in \mathscr{C}_{n_i}$ to be applied on the remaining n_i qubits.
4. Let g_d be a classical circuit outputting descriptions of Clifford circuits, defined so that $g_d(x) = f_d(x)E_d^\dagger$. Compute $(\overline{\mathsf{lab}}_d, \widetilde{g}_d) \leftarrow \mathsf{Garble}(1^\lambda, g_d)$.
5. For each i from $d-1$ to 1, sample $L_{i+1} \leftarrow \mathsf{LabEnc}(\overline{\mathsf{lab}}_{i+1})$ and compute $(\overline{\mathsf{lab}}_i, \widetilde{g}_i) \leftarrow \mathsf{Garble}(1^\lambda, g_i)$, where g_i is a classical circuit that outputs descriptions of Clifford circuits,
$$g_i(x) = (E_{i+1} \otimes L_{i+1})(f_i(x) \otimes \mathbb{I}^{h_i\lambda})E_i^\dagger \ .$$
6. Let F_0 be the initial Clifford to be applied to the input qubits, sample $L_1 \leftarrow \mathsf{LabEnc}(\overline{\mathsf{lab}}_1)$ and output
$$E_0, \widetilde{Q} = \Big((E_1 \otimes L_1)(F_0 \otimes \mathbb{I}^{h_0\lambda})E_0^\dagger, \widetilde{g}_1, \ldots, \widetilde{g}_d\Big)\,.$$

$\mathsf{QGEval}(\widetilde{\mathbf{x}}_{\mathsf{inp}}, \widetilde{Q})$ Takes as input a garbled input $\widetilde{\mathbf{x}}_{\mathsf{inp}}$ and a garbled $\mathsf{C}+\mathsf{M}$ circuit $\widetilde{Q}$.

1. Write $\widetilde{Q} = (D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d)$ and set $\mathbf{x}_0 \coloneqq \widetilde{\mathbf{x}}_{\mathsf{inp}}$. For i from 1 to d, compute $D_{i-1}(\mathbf{x}_{i-1})$, measure the last $k_i\lambda$ qubits to obtain a set of labels $\widetilde{\mathsf{lab}}_i$, compute $D_i \leftarrow \mathsf{GEval}(\tilde{g}_i, \widetilde{\mathsf{lab}}_i)$, and set $\mathbf{x}_i$ to be the remaining $n_i + h_i\lambda$ qubits of the state.
2. Output $D_d(\mathbf{x}_d)$.

$\mathsf{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, \mathbf{x}_{\mathsf{out}})$**:** Takes as input parameters for a $\mathsf{C+M}$ circuit and a state $\mathbf{x}_{\mathsf{out}}$.

1. For each $i \in [0,\dots,d]$, sample $D_i \leftarrow \mathscr{C}_{n_i+h_i\lambda}$, where recall that $h_i \coloneqq k - \sum_{j=1}^{i} k_j$.
2. Let $\mathbf{x}_d = D_d^\dagger(\mathbf{x}_{\mathsf{out}})$. For i from d to 1, compute $\widetilde{\mathsf{lab}}_i, \tilde{g}_i \leftarrow \mathsf{GSim}(1^\lambda, 1^{k_i}, 1^{|g_i|}, D_i^\dagger)$ and set $\mathbf{x}_{i-1} \coloneqq D_{i-1}^\dagger(\mathbf{x}_i, |\widetilde{\mathsf{lab}}_i\rangle\langle\widetilde{\mathsf{lab}}_i|)$.
3. Output $\mathbf{x}_0, D_0, \tilde{g}_1, \dots, \tilde{g}_d$.

Theorem 4.4. *The triple* $(\mathsf{QGarble}, \mathsf{QGEval}, \mathsf{QGSim})$ *defined above satisfies Definition 4.2.*

To prove Theorem 4.4, we need the following additional lemma.

Lemma 4.5. *For any state* $\mathbf{x}$ *on* n *qubits and any Clifford* R *on* n *qubits. The following two states are identical:*

- $\mathbb{E}_{C\leftarrow\mathscr{C}_n}\left(C(\mathbf{x}), RC^\dagger\right)$
- $\mathbb{E}_{D\leftarrow\mathscr{C}_n}\left(D^\dagger R(\mathbf{x}), D\right)$

Proof. The proof is straightforward. Notice first that, because $\mathscr{C}_n$ is a group, we have that

$$\mathbb{E}_{C\leftarrow\mathscr{C}_n}\left(C(\mathbf{x}), RC^\dagger\right) = \mathbb{E}_{C\leftarrow\mathscr{C}_n\cdot R}\left(C(\mathbf{x}), RC^\dagger\right) ,$$

where we denote by $\mathscr{C}_n \cdot R$ the group $\{CR : C \in \mathscr{C}_n\}$. We can equivalently rewrite the RHS as

$$\mathbb{E}_{D\leftarrow\mathscr{C}_n}\left(DR(\mathbf{x}), R(DR)^\dagger\right) ,$$

which, upon simplification, is equal to

$$\mathbb{E}_{D\leftarrow\mathscr{C}_n}\left(DR(\mathbf{x}), D^\dagger\right) .$$

Finally, using again that $\mathscr{C}_n$ is a group, the latter equals

$$\mathbb{E}_{D\leftarrow\mathscr{C}_n}\left(D^\dagger R(\mathbf{x}), D\right) ,$$

as desired. □

Proof of Theorem 4.4. We will show this by induction on the number of measurement layers d in the circuit Q (we use the same notation as above for the components of Q).

Base step ($d = 0$): When $d = 0$, the LHS of the equation defining security in Definition 4.2 is:

$$\left\{E_0\left(\mathbf{x}_{\mathsf{inp}}\right), \widetilde{Q} : \left(E_0, \widetilde{Q} = F_0 E_0^\dagger\right) \leftarrow \mathsf{QGarble}(Q)\right\} \tag{1}$$

By definition of $\mathsf{QGarble}$, the latter is equivalent to:

$$\left\{E_0\left(\mathbf{x}_{\mathsf{inp}}\right), \widetilde{Q} : \quad \widetilde{Q} = F_0 E_0^\dagger,\ E_0 \leftarrow \mathscr{C}_{n_0}\right\} \tag{2}$$

By Lemma 4.5, the latter is identical to:

$$\left\{D_0^\dagger F_0\left(\mathbf{x}_{\mathsf{inp}}\right), D_0 : \ D_0 \leftarrow \mathscr{C}_{n_0}\right\} . \tag{3}$$

Inductive step ($d \Rightarrow d+1$): Suppose that for some d the following two distributions are identical for any $\mathsf{C+M}$ circuit Q with d measurements (where we use the same notation as in definition 4.1 for the components of Q), and any input state $\mathbf{x}_{\mathsf{inp}}$.

- $$\left\{E_0\left(\mathbf{x}_{\mathsf{inp}},\mathbf{0}^{k\lambda}\right),\widetilde{Q}:\left(E_0,\widetilde{Q}=\left((E_1\otimes L_1)(F_0\otimes\mathbb{I}^{h_0\lambda})E_0^{\dagger},\widetilde{g}_1,\ldots,\widetilde{g}_d\right)\right)\leftarrow\mathsf{QGarble}(Q)\right\}$$

- $$\begin{aligned}\Big\{D_0^{\dagger}\Big(D_1^{\dagger}\otimes\mathbb{I}\Big)\Big(\cdots\Big(D_{d-1}^{\dagger}\otimes\mathbb{I}\Big)\Big((D_d^{\dagger}\otimes\mathbb{I})\left(Q\left(\mathbf{x}_{\mathsf{inp}}\right)\otimes\mathsf{lab}_d\right)\otimes\mathsf{lab}_{d-1}\Big)\otimes\cdots\otimes\mathsf{lab}_1\Big),D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d:&\\ D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d\},&\\ (\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{ for } i\in[d]\Big\}&\end{aligned}$$

Let Q be a $\mathsf{C+M}$ circuit with $d+1$ measurements, and let $\mathbf{x}_{\mathsf{inp}}$ be an input to the circuit. Consider the distribution of input encoding + garbled circuit:

$$\left\{E_0\left(\mathbf{x}_{\mathsf{inp}},\mathbf{0}^{k\lambda}\right),\widetilde{Q}:\left(E_0,\widetilde{Q}=\left((E_1\otimes L_1)(F_0\otimes\mathbb{I}^{h_0\lambda})E_0^{\dagger},\widetilde{g}_1,\ldots,\widetilde{g}_{d+1}\right)\right)\leftarrow\mathsf{QGarble}(Q)\right\}$$

Let Q_d be the circuit that runs Q up to (and including) the adaptive Clifford controlled on the d-th measurement outcome. For ease of notation, we simply write $\mathsf{lab}_{i,x}$ to denote the encoding label for measurement outcome x at the i-th layer. More precisely, $\mathsf{lab}_{i,x}=(\mathsf{lab}_{i,x_1},\ldots,\mathsf{lab}_{i,x_n})$ for an appropriate n. Since $\widetilde{g}_{d+1}$ is independent of the random Clifford E_d, we can apply the inductive hypothesis to the d-measurement circuit $(E_{d+1}\otimes L_{d+1})\,Q_d$ on input $\mathbf{x}_{\mathsf{inp}}$). We deduce that the above distribution is computationally indistinguishable from:

$$\begin{aligned}\Big\{D_0^{\dagger}\Big(D_1^{\dagger}\otimes\mathbb{I}\Big)\Big(\cdots\Big(D_d^{\dagger}\otimes\mathbb{I}\Big)\Big(\left(E_{d+1}\otimes L_{d+1}\right)\left(Q_d\left(\mathbf{x}_{\mathsf{inp}}\right)\right)\otimes\mathsf{lab}_d\Big)\otimes\cdots\otimes\mathsf{lab}_1\Big),D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d,\widetilde{g}_{d+1}:&\\ &D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d\},\ E_{d+1}\leftarrow\mathscr{C}_{n_{d+1}}\\ &(\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{ for } i\in[d],\\ &\left(\mathsf{lab}_{d+1}=\{\mathsf{lab}_{d+1,x}\}_{x\in\{0,1\}^{\lambda}},\widetilde{g}_{d+1}\right)\leftarrow\mathsf{Garble}\left(g_{d+1}:g_{d+1}(x)=f_{d+1}(x)E_{d+1}^{\dagger}\right),\\ &L_{d+1}\leftarrow\mathsf{LabEnc}(\mathsf{lab}_{d+1})\Big\}\end{aligned}$$

Let $Q_d^x\left(\mathbf{x}_{\mathsf{inp}}\right)$ be the post-measurement state upon executing circuit Q up to the d-th measurement, conditioned on the d-th measurement outcome being x. By Lemma 4.3, the above distribution is identical to:

$$\begin{aligned}\Big\{D_0^{\dagger}\Big(D_1^{\dagger}\otimes\mathbb{I}\Big)\Big(\cdots\Big(D_d^{\dagger}\otimes\mathbb{I}\Big)\Big(\mathbb{E}_{x\leftarrow\mathsf{Meas}(Q_d(\mathbf{x}_{\mathsf{inp}}))}\left[\left(E_{d+1}\otimes\mathbb{I}\right)\left(Q_d^x\left(\mathbf{x}_{\mathsf{inp}}\right)\otimes\mathsf{lab}_{d+1,x}\right)\right]\otimes\mathsf{lab}_d\Big)\otimes\cdots\otimes\mathsf{lab}_1\Big),&\\ &D_0,\widetilde{g}_1,..,\widetilde{g}_{d+1}:\\ &D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d\},\ E_{d+1}\leftarrow\mathscr{C}_{n_{d+1}},\\ &(\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{ for } i\in[d],\\ &\left(\mathsf{lab}_{d+1}=\{\mathsf{lab}_{d+1,x}\}_{x\in\{0,1\}^{\lambda}},\widetilde{g}_{d+1}\right)\leftarrow\mathsf{Garble}\left(g_{d+1}:g_{d+1}(x)=f_{d+1}(x)E_{d+1}^{\dagger}\right)\Big\}\end{aligned}$$

We apply the simulation property of the classical garbling scheme (for each x), and deduce that the latter is computationally indistinguishable from:

$$\begin{aligned}\Big\{\mathbb{E}_{x\leftarrow\mathsf{Meas}(Q_d(\mathbf{x}_{\mathsf{inp}}))}\Big[\Big\{D_0^\dagger\Big(D_1^\dagger\otimes\mathbb{I}\Big)\Big(\cdot\cdot\Big(D_d^\dagger\otimes\mathbb{I}\Big)\Big((E_{d+1}\otimes\mathbb{I})\left(Q_d^x\left(\mathbf{x}_{\mathsf{inp}}\right)\otimes\mathsf{lab}_{d+1,x}\right)\otimes\mathsf{lab}_d\Big)\otimes\cdot\cdot\otimes\mathsf{lab}_1\Big),&\\ D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d,\widetilde{g}_{d+1,x}\Big]:&\\ D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d\},\ E_{d+1}\leftarrow\mathscr{C}_{n_{d+1}},&\\ (\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{for } i\in[d],&\\ (\mathsf{lab}_{d+1,x},\widetilde{g}_{d+1,x})\leftarrow\mathsf{GSim}\left(f_{d+1}(x)E_{d+1}^\dagger\right),\ \text{for } x\in\{0,1\}^\lambda\Big\}&\end{aligned}$$

We apply Lemma 4.5 (for each x) to deduce that the latter is identical to:

$$\begin{aligned}\Big\{\mathbb{E}_{x\leftarrow\mathsf{Meas}(Q_d(\mathbf{x}_{\mathsf{inp}}))}\Big[D_0^\dagger\Big(D_1^\dagger\otimes\mathbb{I}\Big)\Big(\cdot\cdot\Big(D_d^\dagger\otimes\mathbb{I}\Big)\Big((D_{d+1,x}^\dagger\otimes\mathbb{I})\left(f_{d+1}(x)Q_d^x\left(\mathbf{x}_{\mathsf{inp}}\right)\otimes\mathsf{lab}_{d+1,x}\right)\otimes\mathsf{lab}_d\Big)\otimes\cdot\cdot\otimes\mathsf{lab}_1\Big),&\\ D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d,\widetilde{g}_{d+1,x}\Big]:&\\ D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d\},&\\ D_{d+1,x}\leftarrow\mathscr{C}_{n_{d+1}},\ x\in\{0,1\}^\lambda,&\\ (\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{for } i\in[d],&\\ (\mathsf{lab}_{d+1,x},\widetilde{g}_{d+1,x})\leftarrow\mathsf{GSim}\left(D_{d+1,x}\right),\ \text{for } x\in\{0,1\}^\lambda\Big\}&\end{aligned}$$

It is straightforward to see that latter is the same distribution as:

$$\begin{aligned}\Big\{D_0^\dagger\Big(D_1^\dagger\otimes\mathbb{I}\Big)\Big(\cdot\cdot\Big(D_d^\dagger\otimes\mathbb{I}\Big)\Big((D_{d+1}^\dagger\otimes\mathbb{I})\left(\mathbb{E}_{x\leftarrow\mathsf{Meas}(Q_d(\mathbf{x}_{\mathsf{inp}}))}\left[f_{d+1}(x)Q_d^x\left(\mathbf{x}_{\mathsf{inp}}\right)\right]\otimes\mathsf{lab}_{d+1}\right)\otimes\mathsf{lab}_d\Big)\otimes\cdot\cdot\otimes\mathsf{lab}_1\Big),&\\ D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d,\widetilde{g}_{d+1}\Big]:&\\ D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d+1\},&\\ (\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{for } i\in[d+1]\Big\}&\end{aligned}$$

i.e. sampling the same D_{d+1} and simulated garbling output for all x results in the same distribution. Finally, we can rewrite the latter as:

$$\begin{aligned}\Big\{D_0^\dagger\Big(D_1^\dagger\otimes\mathbb{I}\Big)\Big(\cdot\cdot\Big(D_d^\dagger\otimes\mathbb{I}\Big)\Big((D_{d+1}^\dagger\otimes\mathbb{I})\left(Q\left(\mathbf{x}_{\mathsf{inp}}\right)\otimes\mathsf{lab}_{d+1}\right)\otimes\mathsf{lab}_d\Big)\otimes\cdot\cdot\otimes\mathsf{lab}_1\Big),&\\ D_0,\widetilde{g}_1,\ldots,\widetilde{g}_d,\widetilde{g}_{d+1}\Big]:&\\ D_i\leftarrow\mathscr{C}_{n_i+h_i\lambda},\ i\in\{0,\ldots,d+1\},&\\ (\mathsf{lab}_i,\widetilde{g}_i)\leftarrow\mathsf{GSim}(D_i),\ \text{for } i\in[d+1]\Big\},&\end{aligned}$$

as desired.

□

5 Two-Party Quantum Computation in Three Messages

5.1 The Protocol

Ingredients. Our protocol will make use of the following cryptographic primitives: (1) Quantum-secure two-message two-party classical computation in the CRS model $(\mathsf{2PC.Gen}, \mathsf{2PC}_1, \mathsf{2PC}_2, \mathsf{2PC}_{\mathsf{out}})$ with a straight-line black-box simulator (Section 3.4), and (2) a garbling scheme for $\mathsf{C}+\mathsf{M}$ circuits $(\mathsf{QGarble}, \mathsf{QGEval}, \mathsf{QGSim})$.

Notation. The protocol below computes a two-party quantum functionality represented by a $\mathsf{C}+\mathsf{M}$ circuit Q that takes n_A+n_B input qubits, produces m_A+m_B output qubits, and requires n_Z auxiliary $\mathbf{0}$ states and n_T auxiliary $\mathbf{T}$ states. Let λ be the security parameter. The total number of quantum registers used will be $s = n_A + (n_B+\lambda) + (2n_Z+\lambda) + (n_T+1)\lambda$, and we'll give a name to different groups of these registers.

In round 1, B operates on $n_B+\lambda$ registers, partitioned as $(\mathsf{B}, \mathsf{Trap_B})$, and sends these registers to A. In round 2, A operates on these registers, along with A of size n_A, $\mathsf{Z_A}$ of size $2n_Z$, $\mathsf{Trap_A}$ of size λ, and $\mathsf{T_A}$ of size $(n_T+1)\lambda$. An honest party A will return all registers to B in the order $(\mathsf{A}, \mathsf{B}, \mathsf{Trap_B}, \mathsf{Z_A}, \mathsf{Trap_A}, \mathsf{T_A})$. During party B's subsequent computation, the register $\mathsf{Z_A}$ will be partitioned into two registers $(\mathsf{Z_{inp}}, \mathsf{Z_{check}})$, where each has size n_Z, and register $\mathsf{T_A}$ will be partitioned into two registers $(\mathsf{T_{inp}}, \mathsf{T_{check}})$, where $\mathsf{T_{inp}}$ has size $n_T\lambda$ and $\mathsf{T_{check}}$ has size λ.

Given a $\mathsf{C}+\mathsf{M}$ circuit Q and a Clifford $C_{\mathsf{out}} \in \mathscr{C}_{m_A+\lambda}$, we define another $\mathsf{C}+\mathsf{M}$ circuit $Q[\mathsf{dist}, C_{\mathsf{out}}]$. This circuit takes as input $n_A+n_B+n_Z+\lambda+n_T\lambda$ qubits $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}})$ on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z_{inp}}, \mathsf{Trap_A}, \mathsf{T_{inp}})$. It will first apply the magic state distillation circuit from Lemma 3.4 with parameters $(n_T\lambda, \lambda)$ to $\mathbf{t}_{\mathsf{inp}}$ to produce QRV $\mathbf{t}$ of size n_T. It will then run Q on $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{z}_{\mathsf{inp}}, \mathbf{t})$ to produce $(\mathbf{y}_A, \mathbf{y}_B)$. Finally, it will output $(C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A), \mathbf{y}_B)$.

Protocol 1: Classical Functionality $\mathcal{F}[Q]$

Common Information: Security parameter λ, and $\mathsf{C}+\mathsf{M}$ circuit Q to be computed with $n_A + n_B$ input qubits, $m_A + m_B$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary $\mathbf{T}$ states. Let $s = n_A + (n_B + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$.

Party A Input: Classical descriptions of $C_A \in \mathscr{C}_s$ and $C_{\mathsf{out}} \in \mathscr{C}_{m_A+\lambda}$.
Party B Input: Classical description of $C_B \in \mathscr{C}_{n_B+\lambda}$.

The Functionality:

1. Sample the unitary $U_{\mathsf{dec-check}}$ as follows:
   - Sample a random permutation π on $(n_T + 1)\lambda$ elements.
   - Sample a random element $M \leftarrow \mathsf{GL}(2n_T, \mathbb{F}_2)$.
   - Compute a description of the Clifford U_{check} that operates as follows on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Trap_B}, \mathsf{Z_A}, \mathsf{Trap_A}, \mathsf{T_A})$.
     (a) Rearrange the registers of $\mathsf{T_A}$ according to the permutation π and then partition the registers into $(\mathsf{T_{inp}}, \mathsf{T_{check}})$.
     (b) Apply the linear map M to the registers $\mathsf{Z_A}$ and then partition the registers into $(\mathsf{Z_{inp}}, \mathsf{Z_{check}})$.
     (c) Re-arrange the registers to $(\mathsf{A}, \mathsf{B}, \mathsf{Z_{inp}}, \mathsf{Trap_A}, \mathsf{T_{inp}}, \mathsf{Z_{check}}, \mathsf{Trap_B}, \mathsf{T_{check}})$.
   - Define $U_{\mathsf{dec-check}}$ as:

$$U_{\mathsf{dec-check}} \coloneqq U_{\mathsf{check}} \left(\mathbb{I}^{n_A} \otimes C_B^\dagger \otimes \mathbb{I}^{(2n_Z+\lambda)+(n_T+1)\lambda} \right) C_A^\dagger.$$

2. Sample $(E_0, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d) \leftarrow \mathsf{QGarble}(1^\lambda, Q[\mathsf{dist}, C_{\mathsf{out}}])$.
3. Compute a description of $U_{\mathsf{dec-check-enc}} \coloneqq \left(E_0 \otimes \mathbb{I}^{(n_Z+\lambda)+\lambda}\right) U_{\mathsf{dec-check}}^\dagger$.

Party B Output: (1) A unitary $U_{\mathsf{dec-check-enc}}$ on s qubits (to be applied on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Trap_B}, \mathsf{Z_A}, \mathsf{Trap_A}, \mathsf{T_A})$), and (2) A QGC $(D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d)$ (to be applied to registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z_{inp}}, \mathsf{Trap_A}, \mathsf{T_{inp}})$).

Figure 1: Classical functionality to be used in Protocol 2.

Protocol 2: Three-message two-party quantum computation

Common Information: (1) Security parameter λ, and (2) a $\mathsf{C}+\mathsf{M}$ circuit Q over $n_A + n_B$ input qubits, $m_A + m_B$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary $\mathbf{T}$ states. Let $s = n_A + (n_B + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$.

Party A Input: $\mathbf{x}_A$
Party B Input: $\mathbf{x}_B$

<u>The Protocol:</u>
Setup. Run classical 2PC setup: $\mathsf{crs} \leftarrow \mathsf{2PC.Gen}(1^\lambda)$.

Round 1. *Party B:*

1. Sample $C_B \leftarrow \mathscr{C}_{n_B+\lambda}$ and compute $\mathbf{m}_{B,1} \coloneqq C_B(\mathbf{x}_B, \mathbf{0}^\lambda)$.
2. Compute $(m_{B,1}, \mathsf{st}) \leftarrow \mathsf{2PC}_1(1^\lambda, \mathcal{F}[Q], \mathsf{crs}, C_B)$.
3. Send to Party A: $(m_{B,1}, \mathbf{m}_{B,1})$.

Round 2. *Party A:*

1. Sample $C_A \leftarrow \mathscr{C}_s$ and $C_{\mathsf{out}} \leftarrow \mathscr{C}_{m_A+\lambda}$.
2. Compute $\mathbf{m}_{A,2} \coloneqq C_A(\mathbf{x}_A, \mathbf{m}_{B,1}, \mathbf{0}^{2n_Z}, \mathbf{0}^\lambda, \mathbf{T}^{(n_T+1)\lambda})$.
3. Compute $m_{A,2} \leftarrow \mathsf{2PC}_2(1^\lambda, \mathcal{F}[Q], \mathsf{crs}, m_{B,1}, (C_A, C_{\mathsf{out}}))$.
4. Send to Party B: $(m_{A,2}, \mathbf{m}_{A,2})$.

Round 3. *Party B:*

1. Compute $(U_{\mathsf{dec}-\mathsf{check}-\mathsf{enc}}, D_0, \tilde{g}_1, \dots, \tilde{g}_d) \leftarrow \mathsf{2PC}_{\mathsf{out}}(1^\lambda, \mathsf{st}, m_{A,2})$.
2. Compute $(\mathbf{m}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) \coloneqq U_{\mathsf{dec}-\mathsf{check}-\mathsf{enc}}(\mathbf{m}_2)$, where
   - $\mathbf{m}_{\mathsf{inp}}$ is on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z}_{\mathsf{inp}}, \mathsf{Trap}_{\mathsf{A}}, \mathsf{T}_{\mathsf{inp}})$,
   - $(\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}})$ is on registers $(\mathsf{Z}_{\mathsf{check}}, \mathsf{Trap}_{\mathsf{B}}, \mathsf{T}_{\mathsf{check}})$.
3. Measure each qubit of $(\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B)$ in the standard basis and abort if any measurement is not zero.
4. Measure each qubit of $\mathbf{t}_{\mathsf{check}}$ in the T-basis and abort if any measurement is not zero.
5. Compute $(\widehat{\mathbf{y}}_A, \mathbf{y}_B) \leftarrow \mathsf{QGEval}((D_0, \tilde{g}_1, \dots, \tilde{g}_d), \mathbf{m}_{\mathsf{inp}})$, where $\widehat{\mathbf{y}}_A$ consists of $m_A + \lambda$ qubits and $\mathbf{y}_B$ consists of m_B qubits.
6. Send to Party A: $\widehat{\mathbf{y}}_A$.

Output Reconstruction.

- *Party A:* Compute $(\mathbf{y}_A, \mathsf{trap}_A) \coloneqq C^\dagger_{\mathsf{out}}(\widehat{\mathbf{y}}_A)$, where $\mathbf{y}_A$ consists of m_A qubits and trap_A consists of λ qubits. Measure each qubit of trap_A in the standard basis and abort if any measurement is not zero. Otherwise, output $\mathbf{y}_A$.
- *Party B:* Output $\mathbf{y}_B$.

Figure 2: Three-message two-party quantum computation.

5.2 Security Against Malicious A

The simulator. Consider any QPT adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ corrupting party A. The simulator Sim is defined as follows. Whenever we say that the simulator aborts, we mean that it sends $\bot$ to the ideal functionality and to the adversary.

$\mathsf{Sim}^{\mathcal{I}[\mathbf{x}_B](\cdot)}(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$:

- Compute $(\mathsf{crs}, \tau, m_{B,1}) \leftarrow \mathsf{2PC.Sim}_A^{(1)}(1^\lambda)$, sample $C_B \leftarrow \mathscr{C}_{n_B+\lambda}$, compute $\mathbf{m}_{B,1} := C_B(\mathbf{0}^{n_B}, \mathbf{0}^\lambda)$, and send $(\mathsf{crs}, m_{B,1}, \mathbf{m}_{B,1})$ to $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$.

- Receive $(m_{A,2}, \mathbf{m}_{A,2})$ from Adv_λ and compute $\mathsf{out} \leftarrow \mathsf{2PC.Sim}_A^{(1)}(1^\lambda, \tau, m_{A,2})$. If $\mathsf{out} = \bot$ then abort. Otherwise, parse out as (C_A, C_{out}).

- Using (C_A, C_B), sample $U_{\mathsf{dec-check}}$ as in the description of $\mathcal{F}[Q]$. Compute
$$(\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) := U_{\mathsf{dec-check}}(\mathbf{m}_{A,2}).$$
Measure each qubit of $\mathbf{z}_{\mathsf{check}}$ and trap_B in the standard basis and each qubit of $\mathbf{t}_{\mathsf{check}}$ in the T-basis. If any measurement is non-zero, then abort.

- Forward $\mathbf{x}'_A$ to $\mathcal{I}[\mathbf{x}_B](\cdot)$ and receive back $\mathbf{y}_A$. Compute $\widehat{\mathbf{y}}_A := C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A)$, send $\widehat{\mathbf{y}}_A$ to Adv_λ, send ok to $\mathcal{I}[\mathbf{x}_B]$, and output the output of Adv_λ.

Lemma 5.1. *Let Π be the protocol described in Protocol 2 computing some quantum circuit Q. For any adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ corrupting party A, and any QRV $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$,*[8]

$$\{\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}}.$$

Proof. We consider a sequence of hybrid distributions, where the first hybrid $\mathcal{H}_0$ is $\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$, i.e. the real interaction between the adversary $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$ and an honest party $B(1^\lambda, \mathbf{x}_B)$. In each hybrid, we describe the differences from the previous hybrid.

- $\mathcal{H}_1$: Simulate 2PC as described in Sim, using $\mathsf{2PC.Sim}_A^{(1)}$ to compute $m_{B,1}$ and $\mathsf{2PC.Sim}_A^{(2)}$ to extract an input (C_A, C_{out}) (or abort). Use (C_A, C_{out}) to sample an output $(U_{\mathsf{dec-check-enc}}, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d)$ of the classical functionality. Use this output to run party B's honest Message 3 algorithm.

- $\mathcal{H}_2$: In this hybrid, we change how B's third round message $\widehat{\mathbf{y}}_A$ is sampled. In particular, rather than evaluating the quantum garbled circuit on $\mathbf{m}_{\mathsf{inp}}$, we will directly evaluate $Q[\mathsf{dist}, C_{\mathsf{out}}]$ on the input. In more detail, given $\mathbf{m}_{A,2}$ returned by Adv_λ, (C_A, C_{out}) extracted from Adv_λ, and C_B sampled in Message 1, $\widehat{\mathbf{y}}_A$ is sampled as follows. Sample $U_{\mathsf{dec-check}}$ as in Step 1 of $\mathcal{F}[Q]$. Compute
$$(\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) := U_{\mathsf{dec-check}}(\mathbf{m}_{A,2})$$
and carry out the checks on $\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}$ as described in Steps 3.(c) and 3.(d) of Protocol 2, aborting if needed. Then, compute
$$(\widehat{\mathbf{y}}_A, \mathbf{y}_B) \leftarrow Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}})$$
and return $\widehat{\mathbf{y}}_A$ to Adv_λ.

- $\mathcal{H}_3$: Compute $\mathbf{m}_{B,1}$ as $C_B(\mathbf{0}^{n_B}, \mathbf{0}^\lambda)$, and substitute $\mathbf{x}_B$ for $\mathbf{x}'_B$ before applying $Q[\mathsf{dist}, C_{\mathsf{out}}]$ to the registers described above in $\mathcal{H}_2$.

[8] Technically, we are considering any infinite sequence $((\mathbf{x}_A)_\lambda, (\mathbf{x}_B)_\lambda, (\mathbf{aux})_\lambda)_{\lambda\in\mathbb{N}}$, but we suppress the indexing by λ for readability.

- $\mathcal{H}_4$: Rather than directly computing $Q[\mathsf{dist}, C_{\mathsf{out}}]$, query the ideal functionality with $\mathbf{x}'_A$, receive $\mathbf{y}_A$, and send $\widehat{\mathbf{y}}_A := C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A)$ to Adv_λ. This hybrid is $\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$.

We show indistinguishability between each pair of hybrids.

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: This follows from the security against corrupted A of $\mathsf{2PC}$.
- $\mathcal{H}_1 \approx_s \mathcal{H}_2$: This follows from the statistical correctness of QGC.
- $\mathcal{H}_2 \approx_s \mathcal{H}_3$: First, by the security of the Clifford authentication code, conditioned on all measurements of qubits in trap_B returning 0, we have that $\mathbf{x}'_B \approx_s \mathbf{x}_B$. Next, switching $\mathbf{x}_B$ to $\mathbf{0}^{n_B}$ in B's first message is perfectly indistinguishable due to the perfect hiding of the Clifford authentication code.
- $\mathcal{H}_3 \approx_s \mathcal{H}_4$: First, by Lemma 3.5, conditioned on all measurements of qubits in $\mathbf{z}_{\mathsf{check}}$ returning 0, we have that $\mathbf{z}_{\mathsf{inp}} \approx_s \mathbf{0}^{n_Z}$.

  Next, the above observation, along with Lemma 3.4, implies that, conditioned on all T-basis measurements of qubits in $\mathbf{t}_{\mathsf{check}}$ returning 0, it holds that the output of $Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}'_A, \mathbf{x}_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}})$ is statistically close to the result of computing $(\mathbf{y}_A, \mathbf{y}_B) \leftarrow Q(\mathbf{x}'_A, \mathbf{x}_B, \mathbf{0}^{n_Z}, \mathbf{T}^{n_T})$ and returning $(C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A), \mathbf{y}_B)$. This is precisely what is being computed in $\mathcal{H}_4$.

□

5.3 Security Against Malicious B.

The simulator. Consider any QPT adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$ corrupting party B. The simulator Sim is defined as follows.

$\mathsf{Sim}^{\mathcal{I}[\mathbf{x}_A](\cdot)}(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$:

- Simulate CRS and extract from adversary's round 1 message:
  - Compute $(\mathsf{crs}, \tau) \leftarrow \mathsf{2PC}.\mathsf{Sim}_B^{(1)}(1^\lambda)$ and send crs to the adversary $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_B, \mathbf{aux})$.
  - Receive $(m_{B,1}, \mathbf{m}_{B,1})$ from Adv_λ and compute $\mathsf{inp} \leftarrow \mathsf{2PC}.\mathsf{Sim}_B^{(2)}(1^\lambda, \tau, m_{B,1})$. If $\mathsf{inp} = \bot$ then abort. Otherwise, parse inp as C_B and compute $(\mathbf{x}'_B, \mathsf{trap}_B) := C_B^\dagger(\mathbf{m}_{B,1})$.
- Query ideal functionality and compute simulated round 2 message:
  - Forward $\mathbf{x}'_B$ to $\mathcal{I}[\mathbf{x}_A](\cdot)$ and receive back $\mathbf{y}_B$.
  - Sample $C_{\mathsf{out}} \leftarrow \mathscr{C}_{m_A+\lambda}$ and compute $\widehat{\mathbf{y}}'_A := C_{\mathsf{out}}(\mathbf{0}^{m_A+\lambda})$.
  - Compute $(\widetilde{\mathbf{m}}_{\mathsf{inp}}, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d) \leftarrow \mathsf{QGSim}\left(1^\lambda, \{n_i, k_i\}_{i\in[d]}, (\widehat{\mathbf{y}}'_A, \mathbf{y}_B)\right)$, where $\widetilde{\mathbf{m}}_{\mathsf{inp}}$ is the simulated quantum garbled input on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z}_{\mathsf{inp}}, \mathsf{Trap}_\mathsf{A}, \mathsf{T}_{\mathsf{inp}})$, and $\{n_i, k_i\}_{i\in[d]}$ are the parameters of $\mathsf{C}+\mathsf{M}$ circuit $Q[\mathsf{dist}, C_{\mathsf{out}}]$.
  - Sample $U_{\mathsf{dec-check-enc}} \leftarrow \mathscr{C}_s$ and compute $\mathbf{m}_{A,2} := U^\dagger_{\mathsf{dec-check-enc}}(\widetilde{\mathbf{m}}_{\mathsf{inp}}, \mathbf{0}^{n_Z}, \mathsf{trap}_B, \mathbf{T}^\lambda)$.
  - Compute $m_{A,2} \leftarrow \mathsf{2PC}.\mathsf{Sim}_B^{(3)}(1^\lambda, \tau, (U_{\mathsf{dec-check-enc}}, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d))$.
  - Send $(m_{A,2}, \mathbf{m}_{A,2})$ to Adv_λ.
- Check for abort:
  - Receive $\widehat{\mathbf{y}}_A$ from Adv_λ and measure the last λ qubits of $C^\dagger_{\mathsf{out}}(\widehat{\mathbf{y}}_A)$. If any measurement is not zero, send abort to the ideal functionality and otherwise send ok.
  - Output the output of Adv_λ.

Lemma 5.2. *Let Π be the protocol described in Protocol 2 computing some quantum circuit Q. For any adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ corrupting party B, and any QRV $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$,*

$$\{\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}}.$$

Proof. We consider a sequence of hybrid distributions, where $\mathcal{H}_0$ is $\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$, i.e. the real interaction between $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_B, \mathbf{aux})$ and an honest party $A(1^\lambda, \mathbf{x}_A)$. In each hybrid, we describe the differences from the previous hybrids.

- $\mathcal{H}_1$: Simulate 2PC, using $\mathsf{2PC.Sim}_B^{(1)}$ to sample $\mathsf{2PC.crs}$, $\mathsf{2PC.Sim}_B^{(2)}$ to extract the adversary's input C_B, and $\mathsf{2PC.Sim}_B^{(3)}$ to sample party A's message $m_{A,2}$. Use C_B and freshly sampled (C_A, C_{out}) to sample the output of the classical functionality that is given to $\mathsf{2PC.Sim}_B^{(3)}$.

- $\mathcal{H}_2$: In this hybrid, we make a (perfectly indistinguishable) switch in how $\mathbf{m}_{A,2}$ is computed and how $U_{\mathsf{dec-check-enc}}$ (part of the classical 2PC output) is sampled. Define $(\mathbf{x}'_B, \mathsf{trap}_B) \coloneqq C_B^\dagger(\mathbf{m}_{B,1})$, where C_B was extracted from $m_{B,1}$. Note that in $\mathcal{H}_1$, by the definition of $\mathcal{F}[Q]$,

  $$U_{\mathsf{dec-check-enc}}(\mathbf{m}_{A,2}) \coloneqq (E_0(\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

  Moreover, there exists a Clifford unitary U such that $U_{\mathsf{dec-check-enc}} = UC_A^\dagger$, where C_A was sampled uniformly at random from $\mathscr{C}_s$. Thus, since the Clifford matrices form a group, an equivalent sampling procedure would be to sample $U_{\mathsf{dec-check-enc}} \leftarrow \mathscr{C}_s$ and define

  $$\mathbf{m}_{A,2} \coloneqq U^\dagger_{\mathsf{dec-check-enc}}(E_0(\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

  This is how $\mathcal{H}_2$ is defined.

- $\mathcal{H}_3$: In this hybrid, we simulate the quantum garbled circuit. In particular, compute

  $$(\widehat{\mathbf{y}}_A, \mathbf{y}_B) \leftarrow Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda}),$$

  followed by

  $$(\widetilde{\mathbf{m}}_{\mathsf{inp}}, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d) \leftarrow \mathsf{QGSim}(1^\lambda, \{n_i, k_i\}_{i\in[d]}, (\widehat{\mathbf{y}}_A, \mathbf{y}_B)).$$

  Finally, substitute $\widetilde{\mathbf{m}}_{\mathsf{inp}}$ for $E_0(\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda})$ in the computation of $\mathbf{m}_{A,2}$ so that

  $$\mathbf{m}_{A,2} \coloneqq U^\dagger_{\mathsf{dec-check-enc}}(\widetilde{\mathbf{m}}_{\mathsf{inp}}, \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

- $\mathcal{H}_4$: Note that $Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda})$ may be computed in two stages, where the first outputs $(\mathbf{y}_A, \mathbf{y}_B, \mathbf{0}^\lambda, C_{\mathsf{out}})$ and the second outputs $(C_{\mathsf{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{y}_B)$. In this hybrid, compute only the first stage, set $\mathbf{y}_A$ aside and re-define the final output to be $(\widehat{\mathbf{y}}'_A, \mathbf{y}_B) \coloneqq (C_{\mathsf{out}}(\mathbf{0}^{m_A+\lambda}), \mathbf{y}_B)$.

  Now, during A's output reconstruction step, if the check (step 4.(b) of Protocol 2) passes, output $\mathbf{y}_A$, and otherwise abort.

- $\mathcal{H}_5$: Instead of directly computing $\mathbf{y}_B$ from the first stage of $Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda})$, forward $\mathbf{x}'_B$ to $\mathcal{I}[\mathbf{x}_A](\cdot)$ and receive back $\mathbf{y}_B$. Now, during party A's output reconstruction step, if the check passes, send ok to the ideal functionality, and otherwise send abort to the ideal functionality. This is $\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$.

We show indistinguishability between each pair of hybrids.

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: This follows directly from the security against corrupted B of 2PC.

- $\mathcal{H}_1 \equiv \mathcal{H}_2$: This is argued above.

- $\mathcal{H}_2 \approx_c \mathcal{H}_3$: This follows directly from the security of the QGC.
- $\mathcal{H}_3 \approx_s \mathcal{H}_4$: This follows directly from the (perfect) hiding and (statistical) authentication of the Clifford code.
- $\mathcal{H}_4 \equiv \mathcal{H}_5$: This follows from the definition of $\mathcal{I}[\mathbf{x}_A](\cdot)$.

□

6 Application: Reusable Malicious Designated Verifier NIZK for QMA

In this section, we show how a small tweak to the protocol from last section gives a reusable MDV-NIZK for QMA. Features of our construction differ from those of [Shm20] in several ways.

- It assumes post-quantum OT and reusable MDV-NIZK for NP, whereas [Shm20] assumed (levelled) fully-homomorphic encryption (note that both assumptions are known from QLWE).
- It achieves adaptive soundness with only *polynomial* hardness, whereas [Shm20] assumed sub-exponential hardness of QLWE to obtain adaptive soundness.
- The prover only requires a single copy of the witness state, whereas [Shm20] required the prover to have access to polynomially-many identical copies of the witness.

Definition 6.1 (MDV-NIZK Argument for QMA)**.** *A non-interactive computational zero-knowlege argument for a language $\mathcal{L} = (\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}}) \in \mathsf{QMA}$ in the malicious designated-verifier model consists of 4 algorithms* $(\mathsf{Setup}, \mathsf{VSetup}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax.*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*: A classical PPT algorithm that on input the security parameter samples a common uniformly random string* crs*.*
- $(\mathsf{pvk}, \mathsf{svk}) \leftarrow \mathsf{VSetup}(\mathsf{crs})$*: A classical PPT algorithm that on input* crs *samples a pair of public and secret verification keys.*
- $\boldsymbol{\pi} \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{pvk}, x, \mathbf{w})$*: A QPT algorithm that on input* crs*, the public verification key, an instance* $x \in \mathcal{L}_{\mathsf{yes}}$*, and a quantum witness* $\mathbf{w}$*, outputs a quantum state* $\boldsymbol{\pi}$*.*
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{svk}, x, \boldsymbol{\pi})$*: A QPT algorithm that on input* crs*, secret verification key* svk*, and instance* $x \in \mathcal{L}$*, and a quantum proof* $\boldsymbol{\pi}$*, outputs a bit indicating acceptance or rejection.*

The protocol satisfies the following properties.

- ***Statistical Completeness:*** *There exists a negligible function $\mu(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $x \in \mathcal{L}_{\mathsf{yes}} \cap \{0,1\}^\lambda$, $\mathbf{w} \in \mathcal{R}_{\mathcal{L}}(x)$, $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pvk}, \mathsf{svk}) \in \mathsf{VSetup}(\mathsf{crs})$,*

$$\Pr_{\boldsymbol{\pi} \leftarrow \mathsf{Prove}(\mathsf{crs},\mathsf{pvk},x,\mathbf{w})} [\mathsf{Verify}(\mathsf{crs}, \mathsf{svk}, x, \boldsymbol{\pi})] \geq 1 - \mu(\lambda).$$

- ***Reusable Soundness:*** *For every quantum polynomial-size adversarial prover $\mathcal{P}^* = \{\mathcal{P}^*_\lambda, \mathbf{p}_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function $\mu(\cdot)$ such that for every $\lambda \in \mathbb{N}$,*

$$\Pr_{\substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pvk},\mathsf{svk}) \leftarrow \mathsf{VSetup}(\mathsf{crs}) \\ (x,\boldsymbol{\pi}) \leftarrow \mathcal{P}^*_\lambda(\mathbf{p}_\lambda, \mathsf{crs}, \mathsf{pvk})^{\mathsf{Verify}(\mathsf{crs},\mathsf{svk},\cdot,\cdot)}}} [(x \in \mathcal{L}_{\mathsf{no}}) \wedge (1 = \mathsf{Verify}(\mathsf{crs}, \mathsf{svk}, x, \boldsymbol{\pi}))] \leq \mu(\lambda).$$

- ***Malicious Zero-Knowledge:*** *There exists a QPT simulator* Sim *such that for every QPT distinguisher* $\mathcal{D} = \{\mathcal{D}_\lambda, \mathbf{d}_\lambda\}_{\lambda\in\mathbb{N}}$*, there exists a negligible function* $\mu(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*,*

  $$\left| \Pr_{\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda)} \left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{crs})^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot,\cdot)}\right] - \Pr_{(\mathsf{crs},\tau)\leftarrow\mathsf{Sim}(1^\lambda)} \left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{crs})^{\mathsf{Sim}(\tau,\cdot,\cdot)}\right] \right| \leq \mu(\lambda),$$

  where,

  - *Every query* $\mathcal{D}_\lambda$ *makes to the oracle is of the form* $(\mathsf{pvk}^*, x, \mathbf{w})$*, where* pvk^* *is arbitrary,* $x \in \mathcal{L}_{\mathsf{yes}} \cup \{0,1\}^\lambda$*, and* $\mathbf{w} \in \mathcal{R}_{\mathcal{L}}(s)$*.*
  - $\mathsf{Prove}(\mathsf{crs},\cdot,\cdot,\cdot)$ *is the honest prover algorithm and* $\mathsf{Sim}(\tau,\cdot,\cdot)$ *acts only on* τ, pvk^**, and* x*.*

Theorem 6.2. *Assuming post-quantum maliciously-secure two-message oblivious transfer with straight-line simulation in the CRS model and post-quantum reusable MDV-NIZK for NP (see the discussion following Definition 3.3), there exists a reusable MDV-NIZK satisfying Definition 6.1.*

Proof. (Sketch) For $x \in \mathcal{L}$, let $\mathcal{V}_{\mathcal{L}}[x](\cdot)$ be the QMA verification circuit that takes as input a potential witness $\mathbf{w}$ and outputs a bit indicating acceptance or rejection. For any x, we will use Protocol 2 to compute the functionality $\mathcal{V}_{\mathcal{L}}[x](\cdot)$ (where Alice has input $\mathbf{w}$ and only Bob obtains output) in two messages. Note that Bob has no input, and thus his first message is entirely classical, only consisting of the first message of the classical 2PC. This already gives a one-time MDV-NIZK.

Now, we sketch how to achieve reusability, while maintaining soundness and zero-knowledge. First, we will instantiate the classical 2PC with one that is post-quantum secure, *reusable and extractable.* The latter property requires the existence of an efficient extractor that extracts the implicit input used by Alice in any given 2PC session, such that the joint distribution of the extracted input remains indistinguishable between the real and ideal experiments. We note that Lombardi et. al. [LQR+19] build a post-quantum 2PC protocol that is reusable and extractable, under the assumption that post-quantum OT with straight-line simulation in the CRS model exists, and post-quantum MDV-NIZKs for NP exist.

Given such a 2PC protocol, Bob can compute his first message independently of the statement to be proven by Alice, and Alice can subsequently re-use this first message to prove any number of statements. This already satisfies zero-knowledge, as the MDV-NIZK simulator can always just query the 2PQC simulator with output 1.

To achieve reusable soundness, we alter the classical functionality $\mathcal{F}[\mathcal{V}_{\mathcal{L}}[x]]$ computed by the 2PC. It now takes as input a PRF key k from Bob and generates the auxiliary state checking randomness (permutation π and linear map M) via $\mathsf{PRF}(k,x)$, i.e., the PRF applied to the (classical) instance x. It also now generates the quantum garbled circuit randomness using $\mathsf{PRF}(k,x)$, as well as randomness input by the prover (in order to preserve zero-knowledge). When proving soundness, one can first simulate Bob's first message with the 2PC simulator, and then replace the PRF in $\mathcal{F}[\mathcal{V}_{\mathcal{L}}[x]]$ with a truly random function. In this process, one can argue due to the simulation security of 2PC and extractability of Alice's private input, that the joint distribution of the instances and witnesses used by Alice, and the output of Bob, remains indistinguishable between these experiments. Thus, if Bob accepts proofs for (adaptively chosen) no instances, the modified challenger that simulates Bob's first message with the 2PC simulator, and replaces the PRF in $\mathcal{F}[\mathcal{V}_{\mathcal{L}}[x]]$ with a truly random function, also accepts proofs for no instances. Now, whenever $\mathcal{P}^*$ queries the Verify oracle with an $x \in \mathcal{L}_{\mathsf{no}}$, the oracle will return a rejection with overwhelming probability. Moreover, any queries for $x \in \mathcal{L}_{\mathsf{yes}}$ will not be useful to $\mathcal{P}^*$ since the corresponding 2PC message is sampled with independent randomness. Thus, $\mathcal{P}^*$ will not be able to produce an x and a proof $\boldsymbol{\pi}$ (i.e. a second round 2PQC) message that will cause the verifier (Bob) to accept. □

7 Two-Party Quantum Computation with Two Online Rounds

This section presents a three-round protocol that only requires two rounds of online communication. This protocol can be equivalently interpreted as a two-round protocol with (quantum) pre-processing.

7.1 The Protocol

Ingredients. Our protocol will make use of the following cryptographic primitives, which are all assumed to be sub-exponentially secure (i.e. there exists ϵ such that the primitive is $(2^{-\lambda^{\epsilon}})$-secure).

- Quantum-secure two-message two-party classical computation in the CRS model where one party receives output $(\mathsf{2PC.Gen}, \mathsf{2PC}_1, \mathsf{2PC}_2, \mathsf{2PC}_{\mathsf{out}})$ and with a straight-line black-box simulator (Section 3.4).
- A garbling scheme for $\mathsf{C}+\mathsf{M}$ circuits $(\mathsf{QGarble}, \mathsf{QGEval}, \mathsf{QGSim})$.
- A quantum multi-key FHE scheme $\mathsf{QMFHE} = (\mathsf{KeyGen}, \mathsf{CEnc}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Rerand}, \mathsf{Dec})$ with ciphertext re-randomization and classical encryption of classical messages.
- A quantum-secure equivocal commitment $\mathsf{Com} = (\mathsf{Com.Gen}, \mathsf{Com.Enc}, \mathsf{Com.Ver})$.
- A quantum-secure classical garbled circuit $(\mathsf{Garble}, \mathsf{GEval}, \mathsf{GSim})$.

Notation. The circuit $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}]$ is defined like $Q[\mathsf{dist}, C_{\mathsf{out}}]$ from Section 5.1 except that $X^{x_{\mathsf{out}}}Z^{z_{\mathsf{out}}}$ is applied to B's output $\mathbf{y}_B$. $f_{\mathsf{inp-cor}}[E_0, U_{\mathsf{rerand}}]$ is a classical "input correction" circuit that takes as input $x_{\mathsf{inp}}, z_{\mathsf{inp}} \in \{0,1\}^{n_B}$ and outputs $U_{\mathsf{rerand-enc}} := E_0\left(\mathbb{I}^{n_A} \otimes X^{x_{\mathsf{inp}}}Z^{z_{\mathsf{inp}}} \otimes \mathbb{I}^{n_Z+\lambda+n_T\lambda}\right)U_{\mathsf{rerand}}^{\dagger}$.

For a $2 \times n$ set of elements $\{a_{i,b}\}_{i\in[n],b\in\{0,1\}}$, and a string $x \in \{0,1\}^n$, we let $a^{(x)} := \{a_{i,x_i}\}_{i\in[n]}$. We will use this notation below to refer to sets of public keys $\mathsf{pk}^{(x_{\mathsf{out}},z_{\mathsf{out}})}$, secret keys $\mathsf{sk}^{(x_{\mathsf{out}},z_{\mathsf{out}})}$, labels $\mathsf{lab}^{(x_{\mathsf{out}},z_{\mathsf{out}})}$, and random strings $r^{(x_{\mathsf{out}},z_{\mathsf{out}})}$. Let c_{lev} be a constant satisfying $c_{\mathsf{lev}} > 1/\epsilon$.

Protocol 3: Classical Functionality $\mathcal{G}[Q, \mathsf{Com.crs}]$

Common Information: (1) Security parameter λ, (2) a $\mathsf{C}+\mathsf{M}$ circuit Q on $n_A + n_B$ input qubits, $m_A + m_B$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary $\mathbf{T}$ states, and (3) a crs $\mathsf{Com.crs}$ for an equivocal commitment. Let $s = n_A + (n_B + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$. Let $\lambda_{\mathsf{lev}} = \max\{\lambda, (2n_B)^{c_{\mathsf{lev}}}\}$.

Party A Input: Classical descriptions of $C_A \in \mathscr{C}_s$, $C_{\mathsf{out}} \in \mathscr{C}_{m_A+\lambda}$, $\{r_{i,b}\}_{i\in[2n_B],b\in\{0,1\}} \in (\{0,1\}^{\lambda_{\mathsf{lev}}})^{4n_B}$, $x_{\mathsf{out}}, z_{\mathsf{out}} \in \{0,1\}^{m_B}$, $s \in \{0,1\}^{\lambda_{\mathsf{lev}}}$.
Party B Input: Classical description of $C_B \in \mathscr{C}_{n_B+\lambda}$.

The Functionality:

1. Sample $U_{\mathsf{dec-check}}$ as in $\mathcal{F}[Q]$, using C_A and C_B.
2. Sample $(E_0, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d) \leftarrow \mathsf{QGarble}(1^{\lambda_{\mathsf{lev}}}, Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}])$.
3. Sample $U_{\mathsf{rerand}} \leftarrow \mathscr{C}_{n_A+n_B+n_Z+\lambda+n_T}$.
4. Compute a description of $U_{\mathsf{dec-check-rerand}} \coloneqq \left(U_{\mathsf{rerand}} \otimes \mathbb{I}^{(n_Z+\lambda)+\lambda}\right) U_{\mathsf{dec-check}}$.
5. Compute $(\{\mathsf{lab}_{i,b}\}_{i\in[2n_B],b\in\{0,1\}}, \widetilde{f}_{\mathsf{inp-cor}}) \leftarrow \mathsf{Garble}(1^{\lambda_{\mathsf{lev}}}, f_{\mathsf{inp-cor}}[E_0, U_{\mathsf{rerand}}])$.
6. For each $i \in [2n_B], b \in \{0,1\}$, compute $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \coloneqq \mathsf{QMFHE.Gen}(1^{\lambda_{\mathsf{lev}}}; r_{i,b})$ and $\mathsf{ct}_{i,b} \leftarrow \mathsf{QMFHE.CEnc}(\mathsf{pk}_{i,b}, \mathsf{lab}_{i,b})$.
7. Compute $\mathsf{cmt} \coloneqq \mathsf{Com.Enc}(\mathsf{Com.crs}, (x_{\mathsf{out}}, z_{\mathsf{out}}); s)$.

Party B Output: (1) A unitary $U_{\mathsf{dec-check-rerand}}$ to be applied to s qubits, partitioned as registers $(\mathsf{A}, \mathsf{B}, \mathsf{Trap}_\mathsf{B}, \mathsf{Z}_\mathsf{A}, \mathsf{Trap}_\mathsf{A}, \mathsf{T}_\mathsf{A})$, (2) a classical garbled circuit along with encryptions of its labels $\{\mathsf{pk}_{i,b}, \mathsf{ct}_{i,b}\}_{i\in[2n_B],b\in\{0,1\}}, \widetilde{f}_{\mathsf{inp-cor}}$, (3) a QGC $(D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d)$ to be applied to registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z}_{\mathsf{inp}}, \mathsf{Trap}_\mathsf{A}, \mathsf{T}_{\mathsf{inp}})$, and (4) a commitment cmt.

Figure 3: Classical functionality to be used in Protocol 5.

Protocol 5: Two-party quantum computation with two online rounds

Common Information: Security parameter λ, and $\mathsf{C}+\mathsf{M}$ circuit Q to be computed with $n_A + n_B$ input qubits, $m_A + m_B$ output qubits, n_Z auxiliary $\mathbf{0}$ states, and n_T auxiliary $\mathbf{T}$ states. Let $s = n_A + (n_B + \lambda) + (2n_Z + \lambda) + (n_T + 1)\lambda$. Let $\lambda_{\mathsf{lev}} = \max\{\lambda, (2n_B)^{c_{\mathsf{lev}}}\}$.

Party A input: $\mathbf{x}_A$
Party B input: $\mathbf{x}_B$

The Protocol:
Setup. Run classsical 2PC setup: $\mathsf{2PC.crs} \leftarrow \mathsf{2PC.Gen}(1^{\lambda_{\mathsf{lev}}}), \mathsf{Com.crs} \leftarrow \mathsf{Com.Gen}(1^{\lambda_{\mathsf{lev}}})$.

Round 0 (pre-processing).
Party B:

1. Prepare n_B EPR pairs $\left\{\left(\mathbf{e}_1^{(i)}, \mathbf{e}_2^{(i)}\right)\right\}_{i\in[n_B]}$. Let $\mathbf{e}_1$ denote $(\mathbf{e}_1^{(i)})_{i\in[n_B]}$ and $\mathbf{e}_2$ denote $(\mathbf{e}_2^{(i)})_{i\in[n_B]}$.
2. Sample $C_B \leftarrow \mathscr{C}_{n_B+\lambda}$ and compute $\mathbf{m}_{B,1} \coloneqq C_B(\mathbf{e}_1, \mathbf{0}^\lambda)$.
3. Compute $(m_{B,1}, \mathsf{st}) \leftarrow \mathsf{2PC}_1(1^{\lambda_{\mathsf{lev}}}, \mathcal{G}[Q, \mathsf{Com.crs}], \mathsf{2PC.crs}, C_B)$.
4. Send to Party A: $(m_{B,1}, \mathbf{m}_{B,1})$.

Round 1.
Party A:

1. Sample the following:
   - a random Clifford $C_A \leftarrow \mathscr{C}_s$,
   - a random Clifford $C_{\mathsf{out}} \leftarrow \mathscr{C}_{m_A+\lambda}$,
   - $4n_B$ random length-λ_{lev} bitstrings $\{r_{i,b}\}_{i\in[2n_B],b\in\{0,1\}}$,
   - one random length-λ_{lev} bitstring s,
   - two random length-m_B bitstrings $x_{\mathsf{out}}, z_{\mathsf{out}}$.
2. Compute $\mathbf{m}_{A,2} \coloneqq C_A(\mathbf{x}_A, \mathbf{m}_{B,1}, \mathbf{0}^{2n_Z}, \mathbf{0}^\lambda, \mathbf{T}^{(n_T+1)\lambda})$.
3. Compute
$$m_{A,2} \leftarrow \mathsf{2PC}_2(1^{\lambda_{\mathsf{lev}}}, \mathcal{G}[Q, \mathsf{Com.crs}], \mathsf{2PC.crs}, m_{B,1}, (C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s)).$$
4. Send to Party B: $(m_{A,2}, \mathbf{m}_{A,2})$.

Party B:

1. Perform Bell measurements on each pair of corresponding qubits in $(\mathbf{x}_B, \mathbf{e}_2)$, obtaining measurement outcomes $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$.
2. Send to Party A: $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$.

Figure 4: Two-party quantum computation with two online rounds.

Protocol 5: Two-party quantum computation with two online rounds

Round 2.

Party A:

1. Send to Party B: $\left(r^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, x_{\mathsf{out}}, z_{\mathsf{out}}, s\right)$.

Party B:

1. Compute
$$\begin{pmatrix} U_{\mathsf{dec-check-rerand}}, \{\mathsf{pk}_{i,b}, \mathsf{ct}_{i,b}\}_{i,b}, \\ \widetilde{f}_{\mathsf{inp-cor}}, D_0, \tilde{g}_1, \dots, \tilde{g}_d, \mathsf{cmt} \end{pmatrix} \leftarrow \mathsf{2PC}_{\mathsf{out}}(1^{\lambda_{\mathsf{lev}}}, \mathsf{st}, m_{A,2}).$$
2. Compute
$$(\mathbf{m}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) \coloneqq U_{\mathsf{dec-check-rerand}}(\mathbf{m}_{A,2}),$$
where
   - $\mathbf{m}_{\mathsf{inp}}$ is on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z}_{\mathsf{inp}}, \mathsf{Trap}_{\mathsf{A}}, \mathsf{T}_{\mathsf{inp}})$,
   - $(\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}})$ is on registers $(\mathsf{Z}_{\mathsf{check}}, \mathsf{Trap}_{\mathsf{B}}, \mathsf{T}_{\mathsf{check}})$.
3. Measure $(\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B)$ in the standard basis and abort if any measurement is not zero.
4. Measure each qubit of $\mathbf{t}_{\mathsf{check}}$ in the T-basis and abort if any measurement is not zero.
5. Compute a ciphertext $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, U_{\mathsf{rerand-enc}})$ via homomorphic evaluation, where $U_{\mathsf{rerand-enc}} \leftarrow \mathsf{GEval}(\widetilde{f}_{\mathsf{inp-cor}}, \mathsf{lab}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})})$.
6. Compute a ciphertext $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, (\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B))$ via homomorphic evaluation, where $(\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B) \leftarrow \mathsf{QGEval}((D_0, \tilde{g}_1, \dots, \tilde{g}_d), U_{\mathsf{rerand-enc}}(\mathbf{m}_{\mathsf{inp}}))$.
7. Apply $\mathsf{QMFHE.Rerand}$ to the encryption of $\widehat{\mathbf{y}}_A$ and send the result $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$.

Output Reconstruction.

- *Party A*: Use $\mathsf{sk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}$ to decrypt $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$. If decryption fails, then abort. Compute $(\mathbf{y}_A, \mathsf{trap}_A) \coloneqq C_{\mathsf{out}}^{\dagger}(\widehat{\mathbf{y}}_A)$, where $\mathbf{y}_A$ consists of m_A qubits and trap_A consists of λ qubits. Measure each qubit of trap_A in the standard basis and abort if any measurement is not zero. Otherwise, output $\mathbf{y}_A$.
- *Party B*: Use $r^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}$ to generate $\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, \mathsf{sk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}$ and check that these public keys match the public keys obtained from the output of $\mathsf{2PC}$ in Round 2. If not, then abort. Use $\mathsf{sk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}$ to decrypt $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}},z_{\mathsf{inp}})}, \overline{\mathbf{y}}_B)$. If $\mathsf{Com.Ver}(1^{\lambda_{\mathsf{lev}}}, \mathsf{Com.crs}, \mathsf{cmt}, (x_{\mathsf{out}}, z_{\mathsf{out}}), s) = 1$, then compute and output $\mathbf{y}_B \coloneqq X^{x_{\mathsf{out}}} Z^{z_{\mathsf{out}}} \overline{\mathbf{y}}_B$, and otherwise abort.

Figure 5: Two-party quantum computation with two online rounds (continued).

7.2 Security Against Malicious A

The simulator. Consider any QPT adversary $\{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ corrupting party A. The simulator Sim is defined as follows.

$\mathsf{Sim}^{\mathcal{I}[\mathbf{x}_B](\cdot)}(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$**:**

- Compute $(\mathsf{crs}, \tau, m_{B,1}) \leftarrow \mathsf{2PC.Sim}_A^{(1)}(1^{\lambda_{\mathsf{lev}}})$, sample $C_B \leftarrow \mathscr{C}_{n_B+\lambda}$, compute $\mathbf{m}_{B,1} := C_B(\mathbf{0}^{n_B}, \mathbf{0}^\lambda)$, sample $x_{\mathsf{inp}}, z_{\mathsf{inp}} \leftarrow \{0,1\}^{n_B}$, and send $(m_{B,1}, \mathbf{m}_{B,1}), (x_{\mathsf{inp}}, z_{\mathsf{inp}})$ to the adversary $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$.
- Receive $(m_{A,2}, \mathbf{m}_{A,2})$ from Adv_λ and compute $\mathsf{out} \leftarrow \mathsf{2PC.Sim}_A^{(1)}(1^\lambda, \tau, m_{A,2})$. If $\mathsf{out} = \bot$ then abort. Otherwise, parse out as $(C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s)$.
- Using (C_A, C_B), sample $U_{\mathsf{dec-check}}$ as in the description of $\mathcal{F}[Q]$. Compute
$$(\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) := U_{\mathsf{dec-check}}(\mathbf{m}_{A,2}).$$
Measure each qubit of $\mathbf{z}_{\mathsf{check}}$ and trap_B in the standard basis and each qubit of $\mathbf{t}_{\mathsf{check}}$ in the T-basis. If any measurement is non-zero, then abort.
- Forward $\mathbf{x}'_A$ to $\mathcal{I}[\mathbf{x}_B](\cdot)$ and receive back $\mathbf{y}_A$. Compute $\widehat{\mathbf{y}}_A := C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A)$, and send a re-randomized $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$ to Adv_λ, where $\mathsf{pk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}$ are generated from $r^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}$.
- Receive $\left(\{r'_i\}_{i\in[2n_B]}, x'_{\mathsf{out}}, z'_{\mathsf{out}}, s'\right)$ from Adv and check that:
  - For all $i \in [2n_B], \mathsf{pk}'_i$ is equal to pk_i, where $(\mathsf{pk}'_i, \mathsf{sk}'_i) := \mathsf{QMFHE.Gen}(1^{\lambda_{\mathsf{lev}}}; r'_i)$ and $(\mathsf{pk}_i, \mathsf{sk}_i) := \mathsf{QMFHE.Gen}(1^{\lambda_{\mathsf{lev}}}; r_{i,(x_{\mathsf{inp}}, z_{\mathsf{inp}})_i})$.
  - $\mathsf{Com.Ver}(1^{\lambda_{\mathsf{lev}}}, \mathsf{Com.crs}, \mathsf{cmt}, (x'_{\mathsf{out}}, z'_{\mathsf{out}}), s') = 1$, where $\mathsf{cmt} := \mathsf{Com.Enc}(1^{\lambda_{\mathsf{lev}}}, \mathsf{Com.crs}, (x_{\mathsf{out}}, z_{\mathsf{out}}); s)$.

  If the checks pass send ok to $\mathcal{I}[\mathbf{x}_B]$ and otherwise send abort.

Lemma 7.1. *Let* Π *be the protocol described in Protocol 5 computing some quantum circuit* Q*. For any adversary* $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ *corrupting party* A*, and any QRV* $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$*,*

$$\{\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}}.$$

Proof. We consider a sequence of hybrid distributions, where $\mathcal{H}_0$ is $\mathsf{REAL}_{\Pi,\mathsf{Q},A}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$, i.e. the real interaction between $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{aux})$ and an honest party $B(1^\lambda, \mathbf{x}_B)$. In each hybrid, we describe the differences from the previous hybrid.

- $\mathcal{H}_1$: Simulate 2PC as described in Sim, using $\mathsf{2PC.Sim}_A^{(1)}$ to compute $m_{B,1}$ and $\mathsf{2PC.Sim}_A^{(2)}$ to extract an input $(C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s)$ (or abort). Use $(C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s)$ to sample an output $(U_{\mathsf{dec-check-rerand}}, D_0, \widetilde{g}_1, \ldots, \widetilde{g}_d)$ of the classical functionality. Use this output to run party B's honest Round 2 algorithm.
- $\mathcal{H}_2$: In this hybrid, we change how B's second round message $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$ is sampled. In particular, rather than evaluating the classical garbled circuit and quantum garbled circuit under QMFHE, we will directly evaluate $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}]$ on the input. In more detail, given $\mathbf{m}_{A,2}$ returned by Adv_λ, $(C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s)$ extracted from Adv_λ, and C_B sampled in Message 0, $\widehat{\mathbf{y}}_A$ is sampled as follows. Sample $U_{\mathsf{dec-check}}$ as in Step 1 of $\mathcal{F}[Q]$. Compute
$$(\mathbf{x}'_A, \overline{\mathbf{x}}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}}, \mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}) := U_{\mathsf{dec-check}}(\mathbf{m}_{A,2})$$
and carry out the checks on $\mathbf{z}_{\mathsf{check}}, \mathsf{trap}_B, \mathbf{t}_{\mathsf{check}}$ as described in Steps 3 and 4 of Protocol 5, aborting if needed. Then, set $\mathbf{x}'_B := X^{x_{\mathsf{inp}}} Z^{z_{\mathsf{inp}}} \overline{\mathbf{x}}'_B$ and compute
$$(\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B) \leftarrow Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}](\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}})$$
and return a re-randomized $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$ to Adv_λ.

- $\mathcal{H}_3$: Compute $\mathbf{m}_{B,1}$ as $C_B(\mathbf{0}^{n_B}, \mathbf{0}^\lambda)$, and sample $x_{\mathsf{inp}}, z_{\mathsf{inp}} \leftarrow \{0,1\}^{n_B}$ rather than computing them based on Bell measurement outcomes. Furthermore, substitute $\mathbf{x}_B$ for $\mathbf{x}'_B$ before applying $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}]$ to the registers described above in $\mathcal{H}_2$.
- $\mathcal{H}_4$: Do not compute $\overline{\mathbf{y}}_B$ followed by $\mathbf{y}_B := X^{x_{\mathsf{inp}}} Z^{z_{\mathsf{inp}}} \overline{\mathbf{y}}_B$ (in party B's reconstruction). Rather, compute

$$(\widehat{\mathbf{y}}_A, \mathbf{y}_B) \leftarrow Q[\mathsf{dist}, C_{\mathsf{out}}, 0^{m_B}, 0^{m_B}](\mathbf{x}'_A, \mathbf{x}_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}}).$$

- $\mathcal{H}_5$: Rather than directly computing $Q[\mathsf{dist}, C_{\mathsf{out}}, 0^{m_B}, 0^{m_B}]$, query the ideal functionality with $\mathbf{x}'_A$, receive $\mathbf{y}_A$, and send $\mathsf{QMFHE.Enc}(\mathsf{pk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A))$ to Adv_λ. After receiving $\left(\{r'_i\}_{i \in [2n_B]}, x'_{\mathsf{out}}, z'_{\mathsf{out}}, s'\right)$ from Adv, carry out the checks described in Sim and send ok or abort to $\mathcal{I}[\mathbf{x}_B]$. This hybrid is $\mathsf{IDEAL}_{\Pi,\mathsf{Q},A}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$.

We show indistinguishability between each pair of hybrids.

- $\mathcal{H}_0 \approx_c \mathcal{H}_1$: This follows directly from the security against corrupted A of $\mathsf{2PC}$.
- $\mathcal{H}_1 \approx_s \mathcal{H}_2$: This follows directly from the statistical correctness of the classical garbled circuit, the statistical correctness of the quantum garbled circuit, and the statistical ciphrerext re-randomization of QMFHE.
- $\mathcal{H}_2 \approx_s \mathcal{H}_3$: First, by the correctness of teleportation, and by the security of the Clifford authentication code, conditioned on all measurements of qubits in trap_B returning 0, we have that $\mathbf{x}'_B \approx_s \mathbf{x}_B$. Next, switching $\mathbf{e}_1$ to $\mathbf{0}^{n_B}$ in B's first message is perfectly indistinguishable due to the perfect hiding of the Clifford authentication code.
- $\mathcal{H}_3 \approx_s \mathcal{H}_4$: This follows from the statistical binding of Com.
- $\mathcal{H}_4 \approx_s \mathcal{H}_5$: First, by Lemma 3.5, conditioned on all measurements of qubits in $\mathbf{z}_{\mathsf{check}}$ returning 0, we have that $\mathbf{z}_{\mathsf{inp}} \approx_s \mathbf{0}^{n_Z}$.

  Next, the above observation, along with Lemma 3.4, implies that, conditioned on all T-basis measurements of qubits in $\mathbf{t}_{\mathsf{check}}$ returning 0, it holds that the output of $Q[\mathsf{dist}, C_{\mathsf{out}}](\mathbf{x}'_A, \mathbf{x}_B, \mathbf{z}_{\mathsf{inp}}, \mathsf{trap}_A, \mathbf{t}_{\mathsf{inp}})$ is statistically close to the result of computing $(\mathbf{y}_A, \mathbf{y}_B) \leftarrow Q(\mathbf{x}'_A, \mathbf{x}_B, \mathbf{0}^{n_Z}, \mathbf{T}^{n_T})$ and returning $(C_{\mathsf{out}}(\mathbf{y}_A, \mathsf{trap}_A), \mathbf{y}_B)$. This is precisely what is being computed in $\mathcal{H}_4$.

□

7.3 Security Against Malicious B

The simulator. Consider any QPT adversary $\{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda \in \mathbb{N}}$ corrupting party B. The simulator Sim is defined as follows.

$\mathsf{Sim}^{\mathcal{I}[\mathbf{x}_A](\cdot)}(\boldsymbol{\rho}_\lambda, \mathbf{x}_B, \mathbf{aux})$**:**

- Simulate CRS and extract from adversary's round 0 message:
  - Compute $(\mathsf{2PC.crs}, \mathsf{2PC}.\tau) \leftarrow \mathsf{2PC.Sim}_B^{(1)}(1^{\lambda_{\mathsf{lev}}})$, $(\mathsf{Com.crs}, \mathsf{Sim.cmt}, \mathsf{Com}.\tau) \leftarrow \mathsf{Com.Sim.Gen}(1^{\lambda_{\mathsf{lev}}})$, and send $(\mathsf{2PC.crs}, \mathsf{Com.crs})$ to $\mathsf{Adv}_\lambda(\boldsymbol{\rho}_\lambda, \mathbf{x}_B, \mathbf{aux})$.
  - Receive $(m_{B,1}, \mathbf{m}_{B,1})$ and then compute $\mathsf{inp} \leftarrow \mathsf{2PC.Sim}_B^{(2)}(1^{\lambda_{\mathsf{lev}}}, \mathsf{2PC}.\tau, m_1)$. If $\mathsf{inp} = \bot$, then abort, if not parse inp as C_B and compute $(\overline{\mathbf{x}}_B, \mathsf{trap}_B) := C_B^\dagger(\mathbf{m}_{B,1})$.
- Compute quantum part of simulated round 1 message:
  - Sample $C_{\mathsf{out}} \leftarrow \mathscr{C}_{m_A+\lambda}$ and compute $\widehat{\mathbf{y}}'_A := C_{\mathsf{out}}(\mathbf{0}^{m_A+\lambda})$.

  - Prepare m_B EPR pairs $\left\{\left(\mathbf{e}_{\mathsf{Sim},1}^{(i)}, \mathbf{e}_{\mathsf{Sim},1}^{(i)}\right)\right\}_{i\in[m_B]}$, and let
$$\mathbf{e}_{\mathsf{Sim},1} \coloneqq \left(\mathbf{e}_{\mathsf{Sim},1}^{(1)}, \dots, \mathbf{e}_{\mathsf{Sim},1}^{(m_B)}\right), \mathbf{e}_{\mathsf{Sim},2} \coloneqq \left(\mathbf{e}_{\mathsf{Sim},2}^{(1)}, \dots, \mathbf{e}_{\mathsf{Sim},1}^{(m_B)}\right).$$
  - Compute $(\widetilde{\mathbf{m}}_{\mathsf{inp}}, D_0, \widetilde{g}_1, \dots, \widetilde{g}_d) \leftarrow \mathsf{QGSim}\left(1^{\lambda_{\mathsf{lev}}}, \{n_i, k_i\}_{i\in[d]}, (\widehat{\mathbf{y}}'_A, \mathbf{e}_{\mathsf{Sim},1})\right)$, where $\widetilde{\mathbf{m}}_{\mathsf{inp}}$ is the simulated quantum garbled input on registers $(\mathsf{A}, \mathsf{B}, \mathsf{Z}_{\mathsf{inp}}, \mathsf{Trap}_{\mathsf{A}}, \mathsf{T}_{\mathsf{inp}})$, and $\{n_i, k_i\}_{i\in[d]}$ are the parameters of $\mathsf{C}+\mathsf{M}$ circuit $Q[\mathsf{dist}, \cdot, \cdot, \cdot]$.
  - Sample $U_{\mathsf{rerand-enc}} \leftarrow \mathscr{C}_{n_A+n_B+n_Z+\lambda+n_T\lambda}$.
  - Sample $U_{\mathsf{dec-check-rerand}} \leftarrow \mathscr{C}_S$ and compute
$$\mathbf{m}_{A,2} \coloneqq U_{\mathsf{dec-check-rerand}}^{\dagger}(U_{\mathsf{rerand-enc}}^{\dagger}(\widetilde{\mathbf{m}}_{\mathsf{inp}}), \mathbf{0}^{n_Z}, \mathsf{trap}_B, \mathbf{T}^{\lambda}).$$

- Compute classical part of simulated round 1 message:
  - Compute $(\{\widetilde{\mathsf{lab}}_i\}_{i\in[2n_B]}, \widetilde{f}_{\mathsf{inp-cor}}) \leftarrow \mathsf{GSim}(1^{\lambda_{\mathsf{lev}}}, 1^{2n_B}, 1^{|f_{\mathsf{inp-cor}}|}, U_{\mathsf{rerand-enc}})$.
  - Sample $\{r_{i,b}\}_{i\in[2n_B],b\in\{0,1\}} \leftarrow (\{0,1\}^{\lambda_{\mathsf{lev}}})^{4n_B}$.
  - For $i \in [2n_B], b \in \{0,1\}$, compute $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \coloneqq \mathsf{QMFHE.Gen}(1^{\lambda_{\mathsf{lev}}}; r_{i,b})$ and $\mathsf{ct}_{i,b} \leftarrow \mathsf{QMFHE.CEnc}(\mathsf{pk}_{i,b}, \widetilde{\mathsf{lab}}_i)$.
  - Compute
$$m_{A,2} \leftarrow \mathsf{2PC.Sim}_B^{(3)}\left(1^{\lambda_{\mathsf{lev}}}, \mathsf{2PC}.\tau, \left(\begin{array}{c} U_{\mathsf{dec-check-rerand}}, \{\mathsf{pk}_{i,b}, \mathsf{ct}_{i,b}\}_{i,b}, \\ \widetilde{f}_{\mathsf{inp-cor}}, D_0, \widetilde{g}_1, \dots, \widetilde{g}_d, \mathsf{Sim.cmt} \end{array}\right)\right).$$

- Send round 1 message and extract adversary's input:
  - Send $(m_{A,2}, \mathbf{m}_{A,2})$ to Adv_λ.
  - Receive $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$ from Adv_λ and compute $\mathbf{x}'_B \coloneqq X^{x_{\mathsf{inp}}}Z^{z_{\mathsf{inp}}}\overline{\mathbf{x}}_B$.

- Query ideal functionality and send simulated round 2 message:
  - Forward $\mathbf{x}'_B$ to $\mathcal{I}[\mathbf{x}_A](\cdot)$ and receive back $\mathbf{y}_B$.
  - Perform Bell measurements on each pair of corresponding qubits in $(\mathbf{y}_B, \mathbf{e}_{\mathsf{Sim},2})$ and let $x_{\mathsf{out}}, z_{\mathsf{out}} \in \{0,1\}^{m_B}$ be the measurement outcomes.
  - Compute $s \leftarrow \mathsf{Com.Sim.Open}(1^{\lambda_{\mathsf{lev}}}, \mathsf{Com}.\tau, (x_{\mathsf{out}}, z_{\mathsf{out}}))$.
  - Send $\left(r^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, x_{\mathsf{out}}, z_{\mathsf{out}}, s\right)$ to Adv_λ.

- Check for abort:
  - Receive $\mathsf{QMFHE.Dec}(\mathsf{sk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}, \widehat{\mathbf{y}}_A)$ from Adv_λ and use $\mathsf{sk}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}$ to decrypt the ciphertext. If decryption fails, then abort.
  - Measure the last λ qubits of $C_{\mathsf{out}}^{\dagger}(\widehat{\mathbf{y}}_A)$ in the standard basis. If any measurement is not zero, send abort to the ideal functionality and otherwise send $\mathsf{continue}$.
  - Output the output of Adv_λ.

Notation. For any adversary $\{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ and inputs $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$, we partition $\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ and $\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ by the first round message $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$ sent by the adversary. That is, we define the distribution $\mathsf{REAL}_{\Pi,\mathsf{Q},B}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ to be the distribution $\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ except that the output of the distribution is replaced with $\perp$ if the adversary did *not* send $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$ in round 1. We define $\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ analogously. We also define $\mathsf{REAL}_{\Pi,\mathsf{Q},B}^{(\mathsf{abort})}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ and $\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}^{(\mathsf{abort})}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ to be the respective distribution replaced with $\perp$ if the adversary sent anything at all in round 1 (i.e. if the adversary did not abort after round 0).

We now prove the following lemma, which is the main part of the proof of security against malicious B.

Lemma 7.2. *For any QPT* $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$*, QPT distinguisher* $\mathcal{D} = \{\mathcal{D}_\lambda, \mathbf{d}_\lambda\}_{\lambda\in\mathbb{N}}$ *inputs* $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ *and* $(x_{\mathsf{inp}}, z_{\mathsf{inp}}) \in (\{0,1\}^{n_B})^2 \cup \{\mathsf{abort}\}$*, there exists a negligible function* μ *such that*

$$\Bigg| \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{out}) = 1 : \mathsf{out} \leftarrow \mathsf{REAL}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\right]$$
$$- \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{out}) = 1 : \mathsf{out} \leftarrow \mathsf{IDEAL}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\right]\Bigg| \le \frac{\mu(\lambda)}{2^{2n_B}}.$$

Proof. First note that by the definition of λ_{lev}, a $\mathcal{D}$ violating the lemma distinguishes with probability at least $\left(\frac{1}{\mathrm{poly}(\lambda)}\right) 2^{-\lambda_{\mathsf{lev}}{}^{(1/c_{\mathsf{lev}})}} \ge \frac{1}{2^{\lambda_{\mathsf{lev}}{}^{\epsilon}}}$.

Now fix any collection $\mathcal{D}, \mathsf{Adv}, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux}, x_{\mathsf{inp}}, z_{\mathsf{inp}}$, and say that $(x_{\mathsf{inp}}, z_{\mathsf{inp}}) \neq \mathsf{abort}$, since the abort case is subsumed by the non-abort case given below. We show the indistinguishability via a sequence of hybrids, where $\mathcal{H}_0$ is the distribution $\mathsf{REAL}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$. In each hybrid, we describe the differences from the previous hybrid.

- $\mathcal{H}_1$: Simulate 2PC, using $\mathsf{2PC.Sim}^{(1)}_B$ to sample $\mathsf{2PC.crs}$, $\mathsf{2PC.Sim}^{(2)}_B$ to extract the adversary's input C_B, and $\mathsf{2PC.Sim}^{(3)}_B$ to sample party A's message $m_{A,2}$. Use C_B and freshly sampled $C_A, C_{\mathsf{out}}, \{r_{i,b}\}_{i,b}, x_{\mathsf{out}}, z_{\mathsf{out}}, s$ to sample the output of the classical functionality that is given to $\mathsf{2PC.Sim}^{(3)}_B$.
- $\mathcal{H}_2$: Simulate Com, using $\mathsf{Com.Sim.Gen}$ to sample $\mathsf{Com.crs}$ and the commitment $\mathsf{Sim.cmt}$. Note that $\mathsf{Sim.cmt}$ is now used directly in computing the output of 2PC, and s is no longer sampled by party A. Open the commitment in the second round to $(x_{\mathsf{out}}, z_{\mathsf{out}})$ using $\mathsf{Com.Sim.Open}$.
- $\mathcal{H}_3$: In this hybrid, we make a (perfectly indistinguishable) switch in how $\mathbf{m}_{A,2}$ is computed and how $U_{\mathsf{dec-check-rerand}}$ (part of the 2PC output) is sampled. Define $(\overline{\mathbf{x}}'_B, \mathsf{trap}_B) := C_B^\dagger(\mathbf{m}_{B,1})$, where C_B was extracted from $m_{B,1}$. Note that in $\mathcal{H}_2$, by the definitions of $\mathcal{F}[Q]$ and $\mathcal{G}[Q]$,

  $$U_{\mathsf{dec-check-rerand}}(\mathbf{m}_{A,2}) := (U_{\mathsf{rerand}}(\mathbf{x}_A, \overline{\mathbf{x}}'_B, \mathbf{0}^{nz}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

  Moreover, there exists a Clifford unitary U such that $U_{\mathsf{dec-check-rerand}} = UC_A^\dagger$, where C_A was sampled uniformly at random from $\mathscr{C}_s$. Thus, since the Clifford matrices form a group, an equivalent sampling procedure would be to sample $U_{\mathsf{dec-check-rerand}} \leftarrow \mathscr{C}_s$ and define

  $$\mathbf{m}_{A,2} := U^\dagger_{\mathsf{dec-check-rerand}}(U_{\mathsf{rerand}}(\mathbf{x}_A, \overline{\mathbf{x}}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

  This is how $\mathcal{H}_3$ is defined.
- $\mathcal{H}_4^{(1)}, \ldots, \mathcal{H}_4^{(2n_B)}$: In $\mathcal{H}_4^{(i)}$, let $\mathsf{ct}_{i,1-(x_{\mathsf{inp}},z_{\mathsf{inp}})_i} \leftarrow \mathsf{QMFHE.CEnc}(\mathsf{pk}_{i,1-(x_{\mathsf{inp}},z_{\mathsf{inp}})_i}, 0)$.
- $\mathcal{H}_5$: Simulate the classical garbled circuit. In particular, let

  $$U_{\mathsf{rerand-enc}} := E_0\left(\mathbb{I}^{n_A} \otimes X^{x_{\mathsf{inp}}} Z^{z_{\mathsf{inp}}} \otimes \mathbb{I}^{nz+\lambda+n_T\lambda}\right) U^\dagger_{\mathsf{rerand}},$$

  and compute $(\{\widetilde{\mathsf{lab}}_i\}_{i\in[2n_B]}, \widetilde{f}_{\mathsf{inp-cor}}) \leftarrow \mathsf{GSim}(1^{\lambda_{\mathsf{lev}}}, 1^{2n_B}, 1^{|f_{\mathsf{inp-cor}}|}, U_{\mathsf{rerand-enc}})$. Now, each $\mathsf{ct}_{i,(x_{\mathsf{inp}},z_{\mathsf{inp}})_i}$ be will an encryption of $\widetilde{\mathsf{lab}}_i$.
- $\mathcal{H}_6^{(1)}, \ldots, \mathcal{H}_6^{(2n_B)}$: In $\mathcal{H}_6^{(i)}$, let $\mathsf{ct}_{i,1-(x_{\mathsf{inp}},z_{\mathsf{inp}})_i} \leftarrow \mathsf{QMFHE.CEnc}(\mathsf{pk}_{i,1-(x_{\mathsf{inp}},z_{\mathsf{inp}})_i}, \widetilde{\mathsf{lab}}_i)$.
- $\mathcal{H}_7$: In this hybrid, we make another perfectly indistinguishable switch in how $\mathbf{m}_{A,2}$ is computed. Let $\mathbf{x}'_B := X^{x_{\mathsf{inp}}} Z^{z_{\mathsf{inp}}} \overline{\mathbf{x}}'_B$, and compute $U_{\mathsf{rerand-enc}} := E_0 U^\dagger_{\mathsf{rerand}}$ and

  $$\mathbf{m}_{A,2} := U^\dagger_{\mathsf{dec-check-rerand}}(U_{\mathsf{rerand}}(\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{nz+\lambda}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{nz}, \mathsf{trap}_B, \mathbf{T}^\lambda).$$

- $\mathcal{H}_8$: Simulate the quantum garbled circuit. In particular, compute
$$(\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B) \leftarrow Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{n_Z}, \mathbf{T}^{n_T\lambda}),$$
followed by
$$(\widetilde{\mathbf{m}}_{\mathsf{inp}}, D_0, \widetilde{g}_1, \dots, \widetilde{g}_d) \leftarrow \mathsf{QGSim}(1^{\lambda_{\mathsf{lev}}}, \{n_i, k_i\}_{i\in[d]}, (\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B)),$$
where $\{n_i, k_i\}_{i\in[d]}$ are the parameters of the $\mathsf{C+M}$ circuit $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}]$. Sample $U_{\mathsf{rerand-enc}} \leftarrow \mathscr{C}_{n_A+n_B+n_Z+\lambda+n_T\lambda}$ and compute
$$\mathbf{m}_{A,2} \coloneqq U^\dagger_{\mathsf{dec-check-rerand}}(U^\dagger_{\mathsf{rerand-enc}}(\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{n_Z+\lambda}, \mathbf{T}^{n_T\lambda}), \mathbf{0}^{n_Z}, \mathsf{trap}_B, \mathbf{T}^{\lambda}).$$

- $\mathcal{H}_{10}$: Note that $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{n_Z+\lambda}, \mathbf{T}^{n_T\lambda})$ may be computed in two stages, where the first outputs $(\mathbf{y}_A, \mathbf{y}_B, \mathbf{0}^\lambda, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}})$ and the second outputs $(\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B) \coloneqq (C_{\mathsf{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), X^{x_{\mathsf{out}}} Z^{z_{\mathsf{out}}} \mathbf{y}_B)$. In this hybrid, we make the following perfectly indistinguishable switch to the second part of this computation. Prepare m_B EPR pairs $\left\{\left(\mathbf{e}^{(i)}_{\mathsf{Sim},1}, \mathbf{e}^{(i)}_{\mathsf{Sim},1}\right)\right\}_{i\in[m_B]}$, and let $\mathbf{e}_{\mathsf{Sim},1} \coloneqq \left(\mathbf{e}^{(1)}_{\mathsf{Sim},1}, \dots, \mathbf{e}^{(m_B)}_{\mathsf{Sim},1}\right)$ and $\mathbf{e}_{\mathsf{Sim},2} \coloneqq \left(\mathbf{e}^{(1)}_{\mathsf{Sim},2}, \dots, \mathbf{e}^{(m_B)}_{\mathsf{Sim},1}\right)$. Then set $(\widehat{\mathbf{y}}_A, \overline{\mathbf{y}}_B) = (C_{\mathsf{out}}(\mathbf{y}_A, \mathbf{0}^\lambda), \mathbf{e}_{\mathsf{Sim},1})$ and let $x_{\mathsf{out}}, z_{\mathsf{out}}$ be the result of Bell measurements applied to corresponding pairs of qubits of $(\mathbf{y}_B, \mathbf{e}_{\mathsf{Sim},2})$. Note that these Bell measurements do not have to be performed until the simulator sends its simulated round 2 message.

- $\mathcal{H}_{11}$: After computing the first stage of $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{n_Z+\lambda}, \mathbf{T}^{n_T\lambda})$, set $\mathbf{y}_A$ aside and re-define the final output to be $(\widehat{\mathbf{y}}'_A, \overline{\mathbf{y}}_B) = (C_{\mathsf{out}}(\mathbf{0}^{m_A+\lambda}), \mathbf{e}_{\mathsf{Sim},1})$. Now, during A's output reconstruction step, if the check (step 3) passes, output $\mathbf{y}_A$, and otherwise abort.

- $\mathcal{H}_{12}$: Rather than directly computing $\mathbf{y}_A$ from the first stage of $Q[\mathsf{dist}, C_{\mathsf{out}}, x_{\mathsf{out}}, z_{\mathsf{out}}](\mathbf{x}_A, \mathbf{x}'_B, \mathbf{0}^{n_Z+\lambda}, \mathbf{T}^{n_T\lambda})$, forward $\mathbf{x}'_B$ to $\mathcal{I}[\mathbf{x}_A](\cdot)$ and receive back $\mathbf{y}_B$, which gives the same distribution as $\mathcal{H}_{11}$. Now, during A's reconstruction step, if the check passes, send ok to the ideal functionality, and otherwise send abort. This is $\mathsf{IDEAL}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$.

□

Lemma 7.3. *Let Π be the protocol described in Protocol 5 computing some quantum circuit Q. For any adversary $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$ corrupting party B, and any QRV $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$,*
$$\{\mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\}_{\lambda\in\mathbb{N}}.$$

Proof. Assume towards contradiction the existence of a QPT $\mathcal{D} = \{\mathcal{D}_\lambda, \mathbf{d}_\lambda\}_{\lambda\in\mathbb{N}}$, a QPT $\mathsf{Adv} = \{\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda\}_{\lambda\in\mathbb{N}}$, and $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ such that
$$\Big| \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{out}) = 1 : \mathsf{out} \leftarrow \mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\right]$$
$$- \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{out}) = 1 : \mathsf{out} \leftarrow \mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})\right] \Big| \geq 1/\mathrm{poly}(\lambda).$$

Define $\mathsf{REAL} \coloneqq \mathsf{REAL}_{\Pi,\mathsf{Q},B}(\mathsf{Adv}_\lambda, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$ and $\mathsf{IDEAL} \coloneqq \mathsf{IDEAL}_{\Pi,\mathsf{Q},B}(\mathsf{Sim}, \boldsymbol{\rho}_\lambda, \mathbf{x}_A, \mathbf{x}_B, \mathbf{aux})$. Furthermore, let $\mathbf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{REAL}}$ be the event that Adv sends $(x_{\mathsf{inp}}, z_{\mathsf{inp}})$ as its first round message in REAL and define $\mathbf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{IDEAL}}$, $\mathbf{E}^{(\mathsf{abort})}_{\mathsf{REAL}}$, and $\mathbf{E}^{(\mathsf{abort})}_{\mathsf{IDEAL}}$ analogously. The above implies that there exists some $(x_{\mathsf{inp}}, z_{\mathsf{inp}}) \in (\{0,1\}^{n_B})^2 \cup \{\mathsf{abort}\}$ such that
$$\Big| \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{REAL}) = 1 \middle| \mathsf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{REAL}}\right] \Pr\left[\mathsf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{REAL}}\right]$$
$$- \Pr\left[\mathcal{D}_\lambda(\mathbf{d}_\lambda, \mathsf{IDEAL}) = 1 \middle| \mathsf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{IDEAL}}\right] \Pr\left[\mathsf{E}^{(x_{\mathsf{inp}}, z_{\mathsf{inp}})}_{\mathsf{IDEAL}}\right] \Big| \geq \frac{1}{\mathrm{poly}(\lambda)(2^{2n_B}+1)}.$$

However, such a distinguisher immediately contradicts Lemma 7.2. □

8 Two Rounds Without Pre-Processing: Challenges and Possibilities

8.1 An Oblivious Simulation Barrier for Two Round Protocols

We begin with our negative result showing that any two-round 2PQC protocol with an *oblivious simulator* supporting general quantum functionalities would imply new protocols for the setting of *instantaneous non-local quantum computation* [Vai03, BK11, Spe16, GC20].

Instantaneous Non-local Quantum Computation. Instantaneous non-local quantum computation of a unitary U on $n_A + n_B$ qubits is an information-theoretic task where parties A and B, who may share some initial entangled quantum state, receive as input quantum states $\mathbf{x}_A, \mathbf{x}_B$ and wish to compute the functionality $U(\mathbf{x}_A, \mathbf{x}_B) = (\mathbf{y}_A, \mathbf{y}_B)$ with only one round of simultaneous communication. For a family of unitaries $\{U_\lambda\}_{\lambda\in\mathbb{N}}$ on $\{n_{A,\lambda}+n_{B,\lambda}\}_{\lambda\in\mathbb{N}}$ qubits, we say that an instantaneous non-local quantum computation protocol must satisfy the following properties:

- **Correctness.** For all input states $(\mathbf{x}_{A,\lambda}, \mathbf{x}_{B,\lambda})$, the joint outputs $(\mathbf{y}'_{A,\lambda}, \mathbf{y}'_{B,\lambda})$ obtained by A and B after engaging in the protocol are such that $(\mathbf{y}'_{A,\lambda}, \mathbf{y}'_{B,\lambda}) \approx_s (\mathbf{y}_{A,\lambda}, \mathbf{y}_{B,\lambda})$, where $(\mathbf{y}_{A,\lambda}, \mathbf{y}_{B,\lambda}) \coloneqq U_\lambda(\mathbf{x}_{A,\lambda}, \mathbf{x}_{B,\lambda})$.
- **Efficiency.** The size of the entangled quantum state initially shared by A and B in the protocol for computing U_λ is bounded by some polynomial in λ (note that the running time of A or B in the protocol does not need to be polynomial).

Conjecture 1. *There exists a family of efficiently-computable unitaries $\{U_\lambda\}_{\lambda\in\mathbb{N}}$ for which no* correct *and* efficient *instantaneous non-local quantum computation protocol exists.*

As noted in the introduction, the best known instantaneous non-local quantum computation protocols for general functionalities on n-qubit inputs for $n > 2$ require exponentially many EPR pairs in both n and in $\log(1/\epsilon)$, where ϵ is the desired correctness error [BK11]. Moreover, there has been recent progress on proving lower bounds for particular classes of unitaries [GC20]. While current lower bounds on the size of input-independent pre-processing are linear in the number of input qubits, the current state of the art leaves open the possibility that exponentially-many EPR pairs are necessary for general functionalities. To the best of the authors' knowledge, known results give no indication as to whether Conjecture 1 is more likely to be true or false. Nevertheless, the fact that it remains unresolved provides some indication that positive progress on two-round 2PQC with oblivious simulation will require new ideas.

Two-Round 2PQC in the CRS Model. Consider a generic two-round two-party protocol for computing an arbitrary functionality U in the (classical) CRS model assuming simultaneous messages. Such a protocol is described by the algorithms $(A_1, A_2, A_3, B_1, B_2, B_3)$ where A_1, A_2, A_3 are (respectively) Alice's first message algorithm, second message algorithm, and output reconstruction algorithm (and likewise for Bob with B_1, B_2, B_3). As usual, Alice's input is $\mathbf{x}_A$ and Bob's input is $\mathbf{x}_B$. They compute a unitary U and obtain $U(\mathbf{x}_A, \mathbf{x}_B) = (\mathbf{y}_A, \mathbf{y}_B)$ where $\mathbf{y}_A$ and $\mathbf{y}_B$ are their respective outputs. We stress that since this model does not allow pre-processing, Alice and Bob *may not share entanglement* before receiving their inputs.

An execution of such a two-round protocol proceeds as follows:

1. **Setup.** Run $\mathsf{crs} \leftarrow \mathsf{Gen}$.
2. **Round 1.** Alice and Bob generate their first round messages and leftover states as $(\mathbf{m}_1^{(A)}, \mathbf{st}_1^{(A)}) \leftarrow A_1(\mathsf{crs}, \mathbf{x}_A)$ and $(\mathbf{m}_1^{(B)}, \mathbf{st}_1^{(B)}) \leftarrow B_1(\mathsf{crs}, \mathbf{x}_B)$. They send their messages to each other, which has the effect of interchanging/swapping $\mathbf{m}_1^{(A)}$ and $\mathbf{m}_1^{(B)}$.

3. **Round 2.** Alice and Bob generate their second round message and leftover states as $(\mathbf{m}_2^{(A)}, \mathbf{st}_2^{(A)}) \leftarrow A_2(\mathbf{st}_1^{(A)}, \mathbf{m}_1^{(B)})$ and $(\mathbf{m}_2^{(B)}, \mathbf{st}_2^{(B)}) \leftarrow B_2(\mathbf{st}_1^{(B)}, \mathbf{m}_1^{(A)})$. They send their messages to each other, which swaps $\mathbf{m}_2^{(A)}$ and $\mathbf{m}_2^{(B)}$.

4. **Output.** $\mathbf{y}_A \leftarrow A_3(\mathbf{st}_2^{(A)}, \mathbf{m}_2^{(B)})$ and $\mathbf{y}_B \leftarrow B_3(\mathbf{st}_2^{(B)}, \mathbf{m}_2^{(A)})$.

Oblivious Simulation. We now define a natural class of black-box, straight-line simulators that we call *oblivious* simulators. Recall that a simulator is *black-box* if it only makes query access to the attacker (and does not need the code/state of the attacker), and is *straight-line* if it only runs a single time in the forward direction. The defining property of an oblivious simulator is that it learns which player (out of A or B) is corrupted only *after* it has generated (and committed to) a simulated CRS. No matter which party is corrupted, such a simulator must use its committed CRS to generate a view for the corrupt party that is computationally indistinguishable from the party's view in the real world.

As discussed in Section 2.8, a negative result for oblivious simulation demonstrates that a natural strategy for constructing two-round two-party computation in the *classical* setting does not extend to the quantum setting.

The following definition specifies the *additional requirements* for a simulator to be "oblivious"; an oblivious simulator must still satisfy the standard real/ideal indistinguishability notion in Definition 3.2, which we will not repeat here.

Definition 8.1 (Syntactic Requirements for Oblivious Simulation)**.** *A simulator for a two-round two-party quantum computation protocol in the classical CRS model is* oblivious *if it can be described by a tuple of algorithms* $(\mathsf{Sim}_0, \mathsf{Sim}^{(A)}, \mathsf{Sim}^{(B)})$ *where* $\mathsf{Sim}^{(A)} = (\mathsf{Sim}_1^{(A)}, \mathsf{Sim}_2^{(A)}, \mathsf{Sim}_3^{(A)})$ *and* $\mathsf{Sim}^{(B)} = (\mathsf{Sim}_1^{(B)}, \mathsf{Sim}_2^{(B)}, \mathsf{Sim}_3^{(B)})$*, simulation proceeds as follows.*

1. *The simulator runs* $(\mathsf{crs}, \mathbf{st}_0^{(\mathsf{Sim})}) \leftarrow \mathsf{Sim}_0$ *to generate the CRS and leftover simulator state* $\mathbf{st}_0^{(\mathsf{Sim})}$*.*

Next, the simulator "learns" whether it should simulate the view of party A or party B. If the simulator is simulating the view of party A, it proceeds using $\mathsf{Sim}^{(A)} = (\mathsf{Sim}_1^{(A)}, \mathsf{Sim}_2^{(A)}, \mathsf{Sim}_3^{(A)})$*, and if it is simulating the view of party B, it proceeds with* $\mathsf{Sim}^{(B)} = (\mathsf{Sim}_1^{(B)}, \mathsf{Sim}_2^{(B)}, \mathsf{Sim}_3^{(B)})$*. We write out the case for simulating the view of party A below (the case for party B is identical).*

2. $(\mathbf{m}_1^{(B)}, \mathbf{st}_1^{(\mathsf{Sim})}) \leftarrow \mathsf{Sim}_1^{(A)}(\mathbf{st}_0^{(\mathsf{Sim})})$
   Then query A_1 on $(\mathsf{crs}, \mathbf{m}_1^{(B)})$ *and receive* $\mathbf{m}_1^{(A)}$*.*

3. $(\mathbf{x}_A, \mathbf{st}_2^{(\mathsf{Sim})}) \leftarrow \mathsf{Sim}_2^{(A)}(\mathbf{st}_1^{(\mathsf{Sim})}, \mathbf{m}_1^{(A)})$
   Then query the ideal functionality on $\mathbf{x}_A$ *and receive* $\mathbf{y}_A$

4. $\mathbf{m}_2^{(B)} \leftarrow \mathsf{Sim}_3^{(A)}(\mathbf{st}_2^{(\mathsf{Sim})}, \mathbf{y}_A)$
   Then query A_2 on $\mathbf{m}_2^{(B)}$*.*

In short, for an oblivious simulator, the distribution of the simulated CRS is completely independent of whether A is corrupt or B is corrupt. Moreover, because the simulator is straight-line, it is possible to define a (possibly inefficient) algorithm $\mathsf{Sim}_{\mathsf{comb}}$ that computes $\left(\mathsf{crs}, \mathsf{st}_1^{(\mathsf{Sim},A)}, \mathsf{st}_1^{(\mathsf{Sim},B)}\right)$, where each of the two simulator states computed is with respect to the *same* classical crs. This can be done for example by running many iterations of Sim_0 until two of them output the same classical crs. Thus, one would obtain a "first-round-only" simulator with the following syntax for the first round:

1. $\left(\mathsf{crs}, \mathsf{st}_1^{(\mathsf{Sim},A)}, \mathsf{st}_1^{(\mathsf{Sim},B)}\right) \leftarrow \mathsf{Sim}_{\mathsf{comb}}$.
   (send crs *to A_1 and receive* $\mathbf{m}_1^{(A)}$ *and send* crs *to B_1 and receive* $\mathbf{m}_1^{(B)}$*)*

2. $(\mathbf{x}_A, \mathsf{st}_2^{(\mathsf{Sim},A)}) \leftarrow \mathsf{Sim}_2^{(A)}(\mathsf{st}_1^{(\mathsf{Sim},A)}, \mathbf{m}_1^{(A)})$.
   Then send $\mathbf{x}_A$ to the ideal functionality and receive $\mathbf{y}_A$

3. $(\mathbf{x}_B, \mathsf{st}_2^{(\mathsf{Sim},B)}) \leftarrow \mathsf{Sim}_2^{(B)}(\mathsf{st}_1^{(\mathsf{Sim},B)}, \mathbf{m}_1^{(B)})$.
   Then send $\mathbf{x}_B$ to the ideal functionality and receive $\mathbf{y}_B$

Non-Local Computation from Two-Round 2PQC with Oblivious Simulation. We now describe how to turn two-round 2PQC for general functionalities with oblivious simulation (in the CRS model) into an instantaneous non-local quantum communication protocol. In the following theorem, we will only make use of the "first-round-only" simulator discussed above.

Theorem 8.2. *Assuming Conjecture 1, there does not exist a two-round two-party quantum computation protocol for general functionalities in the classical CRS model, with an oblivious simulator.*

Proof. Given any family of unitaries $U = \{U_\lambda\}_{\lambda\in\mathbb{N}}$ on $\{n_{A,\lambda}+n_{B,\lambda}\}_{\lambda\in\mathbb{N}}$ qubits promised by Conjecture 1, we define the functionality C-SWAP-U $= \{\text{C-Swap-U}_\lambda\}_{\lambda\in\mathbb{N}}$ as follows. C-SWAP-U$_\lambda$ takes a $(n_{A,\lambda}+n_{B,\lambda})$-qubit state $(\mathbf{x}_A, \mathbf{x}_B)$ as input along with an additional two classical bits of input z_A, z_B. If $z_A \oplus z_B = 0$, it applies U_λ to $(\mathbf{x}_A, \mathbf{x}_B)$ to produce $(\mathbf{y}_A, \mathbf{y}_B)$, and then swaps the output states, outputting $(\mathbf{y}_B, \mathbf{y}_A)$. If $z_A \oplus z_B = 1$, it simply swaps the input states, outputting $(\mathbf{x}_B, \mathbf{x}_A)$. In what follows, we will show that any two-round two-party quantum computation protocol for C-SWAP-U implies a correct and efficient instantaneous non-local quantum computation protocol for U, violating Conjecture 1.

Consider the oblivious simulator for the protocol computing C-SWAP-U. We will only be interested in the simulated first round and subsequent input extraction. We will not be concerned with simulating the second round at all. Furthermore, we will only care about simulating the view of a specific type of adversary: one that simply runs the honest A (resp. B) algorithm. Such an "adversary" does not rush, i.e. the first message algorithm of the (honestly behaving) adversary is independent of B's first round message $\mathbf{m}_1^{(B)}$. Therefore for simplicity we will suppress mention of this message being generated by the simulator (since we are also not concerned with simulation of the second round).

Now, we will combine the "first-round-only" simulator discussed above with the first-message algorithms of parties A and B to produce the following algorithm U_{extract} (which can be written as a unitary), which will be applied to $(\mathbf{x}_A, \mathbf{x}_B)$ (tensored with sufficiently many $\mathbf{0}$ states, which we write as $\mathbf{0}^*$). Technically, U_{extract} is a family of unitaries parameterized by the security parameter λ, and the inputs $(\mathbf{x}_A, \mathbf{x}_B)$ are families of input states, though for simplicity we will drop the explicit indexing by λ.

U_{extract}, on input $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{0}^*)$ works as follows:

1. Compute $\left(\mathsf{crs}, \mathsf{st}_1^{(\mathsf{Sim},A)}, \mathsf{st}_1^{(\mathsf{Sim},B)}\right) \leftarrow \mathsf{Sim}_{\mathsf{comb}}$.
2. Compute $(\mathbf{m}_1^{(A)}, \mathbf{st}_1^{(A)}) \leftarrow A_1(\mathbf{x}_A, \mathsf{crs})$.
3. Compute $(\mathbf{m}_1^{(B)}, \mathbf{st}_1^{(B)}) \leftarrow B_1(\mathbf{x}_B, \mathsf{crs})$.
4. Compute $(\mathbf{x}'_A, \mathbf{st}_2^{(\mathsf{Sim},A)}) \leftarrow \mathsf{Sim}_2^{(A)}(\mathsf{st}_1^{(\mathsf{Sim},A)}, \mathbf{m}_1^{(A)})$.
5. Compute $(\mathbf{x}'_B, \mathbf{st}_2^{(\mathsf{Sim},B)}) \leftarrow \mathsf{Sim}_2^{(B)}(\mathsf{st}_1^{(\mathsf{Sim},B)}, \mathbf{m}_1^{(B)})$.
6. Output $(\mathbf{x}'_A, \mathbf{x}'_B, \mathbf{st}_1^{(A)}, \mathbf{st}_1^{(B)}, \mathbf{st}_2^{(\mathsf{Sim},A)}, \mathbf{st}_2^{(\mathsf{Sim},B)})$.

Now, we show that for any pair of pure states $(\mathbf{x}_A, \mathbf{x}_B)$ that can be deterministically efficiently generated (i.e. can be generated by applying an efficient unitary to $\mathbf{0}$ states), the (traced out) portion of $U_{\mathsf{extract}}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{0}^*)$ consisting of $(\mathbf{x}'_A, \mathbf{x}'_B)$ is statistically close to $(\mathbf{x}_A, \mathbf{x}_B)$. First, we argue that $\mathbf{x}'_A \approx_s \mathbf{x}_A$. Recall that regardless of A's classical input bit z_A, there is always a possibility that, depending on B's classical input bit z_B, the functionality computed will simply be swapping $\mathbf{x}_A$ and $\mathbf{x}_B$ (from the definition of our C-SWAP-U unitary). In this case, the value $\mathbf{x}'_A$ queried by Sim to the ideal functionality will be forwarded to B as its output in the simulated world. B's output in the real world is $\mathbf{x}_A$, and thus $\mathbf{x}'_A \approx_s \mathbf{x}_A$, since otherwise the real and ideal worlds would be distinguishable by the measurement $\{\mathbf{x}_A, \mathbb{I} - \mathbf{x}_A\}$; note that projecting onto $\mathbf{x}_A$ can be performed efficiently since $\mathbf{x}_A$ is a (deterministically) efficiently generated pure state. An identical argument shows that $\mathbf{x}'_B \approx_s \mathbf{x}_B$.

Now, we can apply Lemma 8.3 below to U_{extract}; since the above argument applies to the case where $\mathbf{x}_A, \mathbf{x}_B$ are deterministically efficiently generated pure states, it in particular applies to the states required by Lemma 8.3 (i.e. all computational basis states and all uniform superpositions of two computational basis states). Lemma 8.3 applied to U_{extract} allows us to conclude that for *any* input state $(\mathbf{x}_A, \mathbf{x}_B)$, the states $(\mathbf{st}_1^{(A)}, \mathbf{st}_1^{(B)}, \mathbf{st}_2^{(\mathsf{Sim},A)}, \mathbf{st}_2^{(\mathsf{Sim},B)})$ are (statistically) independent of $(\mathbf{x}_A, \mathbf{x}_B)$. This fact can be used to design a correct and efficient instantaneous non-local quantum computation protocol for U, as described below.

- Setup: Execute U_{extract} on all $\mathbf{0}$ states to produce $(\mathbf{0}', \mathbf{0}', \mathbf{st}_1^{(A)}, \mathbf{st}_1^{(B)}, \mathbf{st}_2^{(\mathsf{Sim},A)}, \mathbf{st}_2^{(\mathsf{Sim},B)})$, where $\mathbf{0}'$ denotes a state that is statistically indistinguishable from $\mathbf{0}$. Discard $(\mathbf{0}', \mathbf{0}')$, send $(\mathbf{st}_1^{(B)}, \mathbf{st}_2^{(\mathsf{Sim},A)})$ to party A, and send $(\mathbf{st}_1^{(A)}, \mathbf{st}_2^{(\mathsf{Sim},B)})$ to party B.
- Party A, on input $\mathbf{x}_A$, does the following.
  1. Compute $(\mathsf{st}_1^{(\mathsf{Sim},A)}, \mathbf{m}_1^{(A)}) := \mathsf{Sim}_2^{(A)^\dagger}(\mathbf{x}_A, \mathbf{st}_2^{(\mathsf{Sim},A)})$.
  2. Compute $(\mathbf{m}_2^{(B)}, \mathbf{st}_2^{(B)}) \leftarrow B_2(\mathbf{st}_1^{(B)}, \mathbf{m}_1^{(A)})$.
  3. Send $\mathbf{m}_2^{(B)}$.
- Party B, on input $\mathbf{x}_B$, does the following.
  1. Compute $(\mathsf{st}_1^{(\mathsf{Sim},B)}, \mathbf{m}_1^{(B)}) := \mathsf{Sim}_2^{(B)^\dagger}(\mathbf{x}_A, \mathbf{st}_2^{(\mathsf{Sim},B)})$.
  2. Compute $(\mathbf{m}_2^{(A)}, \mathbf{st}_2^{(A)}) \leftarrow A_2(\mathbf{st}_1^{(A)}, \mathbf{m}_1^{(B)})$.
  3. Send $\mathbf{m}_2^{(A)}$.
- Party A computes and outputs $\mathbf{y}_B \leftarrow B_3(\mathbf{st}_2^{(B)}, \mathbf{m}_2^{(A)})$.
- Party B computes and outputs $\mathbf{y}_A \leftarrow A_3(\mathbf{st}_2^{(A)}, \mathbf{m}_2^{(B)})$.

Observe that the above protocol produces a transcript that is statistically close to the transcript between an honest A and B. Fix A and B's classical inputs z_A, z_B to be such that $z_A \oplus z_B = 0$. Since A is receiving B's output and B is receiving A's output, the parties are then computing a statistically close approximation to $U(\mathbf{x}_A, \mathbf{x}_B)$ in one round of online communication. The size of the initial entangled state held by A and B is bounded by the size of the honest A and B algorithms and the size of the simulator algorithms, which are all polynomial-size. Thus, the above is a correct and efficient instantaneous non-local quantum computation protocol for computing U.

□

Lemma 8.3. *Let* U^{AB} *be a unitary over registers* A, B, *and suppose that there exists* ϵ *between* 0 *and* 1 *such that for any* $|x\rangle^{\mathsf{A}}$ *that is (the density matrix of) either a computational basis state* $|i\rangle$, *or a uniform superposition of two computational basis states* $\frac{1}{\sqrt{2}}(|0\rangle + |i\rangle)$,

$$\left| \mathrm{Tr}_{\mathsf{B}}(U^{\mathsf{AB}}(|x\rangle^{\mathsf{A}} \otimes |0\rangle^{\mathsf{B}})) - |x\rangle^{\mathsf{A}} \right|_1 \leq \epsilon.$$

Then there exists a constant $\delta > 0$ *and a pure state* $|y\rangle^{\mathsf{B}}$ *such that for* every *state* $|x\rangle^{\mathsf{A}}$,

$$\left| U^{\mathsf{AB}}(|x\rangle^{\mathsf{A}} \otimes |0\rangle^{\mathsf{B}}) - (|x\rangle^{\mathsf{A}} \otimes |y\rangle^{\mathsf{B}}) \right|_1 \leq \epsilon^{\delta}.$$

Proof. If U^{AB} satisfies the conditions of the lemma statement, then for any computational basis state $|i\rangle^{\mathsf{A}}$ on the A registers, there exists a pure state $|y_i\rangle^{\mathsf{B}}$ and a polynomial poly$(\cdot)$ such that

$$\left|U^{\mathsf{AB}}(|i\rangle^{\mathsf{A}} \otimes |0\rangle^{\mathsf{B}}) - (|i\rangle^{\mathsf{A}} \otimes |y_i\rangle^{\mathsf{B}})\right|_1 \leq \mathrm{poly}(\epsilon).$$

Moreover, for each i there must exist a state $|y_{0,i}\rangle^{\mathsf{B}}$ such that

$$\left|U^{\mathsf{AB}}\left(\frac{1}{\sqrt{2}}(|0\rangle^{\mathsf{A}} + |i\rangle^{\mathsf{A}}) \otimes |0\rangle^{\mathsf{B}}\right) - \left(\frac{1}{\sqrt{2}}(|0\rangle^{\mathsf{A}} + |i\rangle^{\mathsf{A}}) \otimes |y_{0,i}\rangle^{\mathsf{B}}\right)\right|_1 \leq \mathrm{poly}(\epsilon).$$

By linearity, we also have that

$$\left|U^{\mathsf{AB}}\left(\frac{1}{\sqrt{2}}(|0\rangle^{\mathsf{A}} + |i\rangle^{\mathsf{A}}) \otimes |0\rangle^{\mathsf{B}}\right) - \left(\frac{1}{\sqrt{2}}(|0\rangle^{\mathsf{A}} \otimes |y_0\rangle^{\mathsf{B}}) + \frac{1}{\sqrt{2}}(|i\rangle^{\mathsf{A}} \otimes |y_i\rangle^{\mathsf{B}})\right)\right|_1 \leq \mathrm{poly}(\epsilon).$$

This implies that for each i, $|y_{0,i}\rangle^{\mathsf{B}}$ is within poly(ϵ) trace distance of both $|y_0\rangle^{\mathsf{B}}$ and $|y_i\rangle^{\mathsf{B}}$, which means that $|y_0\rangle^{\mathsf{B}}$ is within poly(ϵ) trace distance of $|y_i\rangle^{\mathsf{B}}$. Thus, $|y_0\rangle^{\mathsf{B}}$ satisfies the condition in the lemma statement, since for an arbitrary state $|x\rangle^{\mathsf{A}} = \sum_i \alpha_i |i\rangle^{\mathsf{A}}$, we have that

$$\left|U^{\mathsf{AB}}\left(\sum_i \alpha_i |i\rangle^{\mathsf{A}} \otimes |0\rangle^{\mathsf{B}}\right) - \sum_i \alpha_i(|i\rangle^{\mathsf{A}} \otimes |y_i\rangle^{\mathsf{B}})\right|_1 \leq \mathrm{poly}(\epsilon),$$

which implies that

$$\left|U^{\mathsf{AB}}\left(\sum_i \alpha_i |i\rangle^{\mathsf{A}} \otimes |0\rangle^{\mathsf{B}}\right) - (|x\rangle^{\mathsf{A}} \otimes |y_0\rangle^{\mathsf{B}})\right|_1 \leq \mathrm{poly}(\epsilon).$$

□

8.2 A Two-Round Protocol from Quantum VBB Obfuscation

In what follows, we describe a two-round protocol in the CRS model, assuming the existence of a (strong form of) VBB obfuscation of quantum circuits.

8.2.1 VBB Obfuscation of Quantum Circuits

We consider virtual black-box obfuscation of quantum circuits, which was defined (and shown to be impossible in general) by [AF16]. In fact, we consider a potentially stronger version than that given by [AF16], who only consider VBB obfuscation of unitaries. We consider quantum functionalities Q from n qubits to n qubits that include not just unitary gates, but also *measurement* gates, and unitary gates that may be *classically controlled* on the outcome of the measurement gates. While one can always push any measurement to the end of the computation so that the circuit becomes unitary, doing so would not necessarily preserve the security of obfuscation, as it would introduce new auxiliary input registers that a malicious evaluator may initialize in a non-zero state. Thus, obfuscation for unitary + measurement circuits is potentially stronger than obfuscation for unitaries.

We model black-box access to a unitary+measurement circuit as an oracle that accepts a quantum state on n registers, manipulates it according to Q, and returns those same n registers. We allow the obfuscation itself to be either a quantum circuit with a purely classical description, or a quantum circuit along with some quantum state. We refer to this obfuscation as $\mathcal{O}(Q)$, and write $\mathcal{O}(Q)(\mathbf{x})$ to indicate evaluation of the obfuscation on an n-qubit input $\mathbf{x}$, with the understanding that this operation may either be directly applying a quantum circuit to $\mathbf{x}$, or first augmenting $\mathbf{x}$ with additional registers, applying a circuit to the expanded system, and then discarding the extra registers.

Definition 8.4 (Quantum VBB Obfuscation)**.** *Let $\{\mathcal{Q}_n\}_{n\in\mathbb{N}}$ be a family of polynomial-size quantum circuits, where each $Q \in \mathcal{Q}_n$ maps n qubits to n qubits. A quantum black-box obfuscator $\mathcal{O}$ is a quantum algorithm that takes as input an input length $n \in \mathbb{N}$, a security parameter $\lambda \in \mathbb{N}$, and a quantum circuit Q, and outputs an obfuscated quantum circuit. $\mathcal{O}$ should satisfy the following properties.*

- *Polynomial expansion: for every $n, \lambda \in \mathbb{N}$ and $Q \in \mathcal{Q}_n$, the size of $\mathcal{O}(1^n, 1^\lambda, Q)$ is at most* $\mathrm{poly}(n, \lambda)$.
- *Functional equivalence: for every $n, \lambda \in \mathbb{N}$, $Q \in \mathcal{Q}_n$, and $\mathbf{x}$ on n qubits, $\mathcal{O}(1^n, 1^\lambda, Q)(\mathbf{x}) \approx_s Q(\mathbf{x})$.*
- *Virtual black-box: for every (non-uniform) QPT* Adv, *there exists a (non-uniform) QPT $\mathcal{S}$ such that for each $n \in \mathbb{N}$ and $Q \in \mathcal{Q}_n$,*

$$\left|\Pr[\mathsf{Adv}(\mathcal{O}(1^n, 1^\lambda, Q)) = 1] - \Pr[\mathcal{S}^Q(1^n, 1^\lambda) = 1]\right| = \mathrm{negl}(\lambda).$$

We now make a few remarks on the definition that will allow us to simplify the constructions given in the next section.

- We will consider functionalities that discard, or trace out, some subset of registers. In order to implement this with a circuit from n qubits to n qubits, we can have the functionality measure the subset of qubits to be traced out and then "randomize" the outcomes (since we don't want the evaluator to know these measurement results) by applying Hadamard to each register and measuring again. Thus, we will consider obfuscation of functionalities from n qubits to $m \leq n$ qubits.
- We will consider functionalities represented by quantum circuits that require the use of auxiliary $\mathbf{0}$ states. We do not want the evaluator to be able to run such a functionality using non-zero auxiliary states, so we'll have the functionality first measure any auxiliary states input by the evaluator. If all measurements are 0, then the circuit will be run on the all registers, otherwise the functionality can "abort" by discarding all registers as explained above. Thus, we will suppress mention of auxiliary input registers, and assume the functionality has access to any auxiliary $\mathbf{0}$ states that it needs.
- We will consider functionalities that can sample classical bits uniformly at random. This can be accomplished by applying Hadamard to a $\mathbf{0}$ state and measuring. Note that this is a uniquely quantum phenomenon - one cannot obfuscate classical circuits that produce their own randomness.

8.2.2 The Protocol

We present a two-round protocol for two-party quantum computation in the common reference string model. Let Q be the two-party quantum functionality to be computed, and assume for simplicity that it takes n qubits from each party and outputs n qubits to each party. The common reference string will consist of obfuscations of six quantum functionalities

$$\mathcal{F}^{(b)}_{A,\mathsf{inp}}, \mathcal{F}^{(b)}_{B,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}, \mathcal{F}_{B,\mathsf{cmp}}, \mathcal{F}^{(b)}_{A,\mathsf{out}}, \mathcal{F}^{(b)}_{B,\mathsf{out}},$$

three to be used by each party. Each functionality has hard-coded some subset of 8 PRF keys

$$k^{(A,A)}_{\mathsf{inp}}, k^{(A,B)}_{\mathsf{inp}}, k^{(B,A)}_{\mathsf{inp}}, k^{(B,B)}_{\mathsf{inp}}, k^{(A,A)}_{\mathsf{out}}, k^{(A,B)}_{\mathsf{out}}, k^{(B,A)}_{\mathsf{out}}, k^{(B,B)}_{\mathsf{out}}.$$

We take each $\mathsf{PRF}(k^{(\cdot,\cdot)}_{\mathsf{inp}}, \cdot)$ to be a mapping from a λ-bit string to a classical description of a Clifford $C \in \mathscr{C}_{n+\lambda}$. Each PRF key is used in two of the six obfuscated circuits, and the pair of letters in the superscript refers to the identity of the party associated with the first obfuscation it is used in, followed by the identify of the party associated with the second obfuscation it is used in.

Below we describe only $\mathcal{F}^{(b)}_{A,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}$, and $\mathcal{F}^{(b)}_{A,\mathsf{out}}$ since $\mathcal{F}^{(b)}_{B,\mathsf{inp}}, \mathcal{F}_{B,\mathsf{cmp}}$, and $\mathcal{F}^{(b)}_{B,\mathsf{out}}$ are defined exactly the same with A and B switched.

- $\mathcal{F}^{(b)}_{A,\mathsf{inp}}\left[k^{(A,A)}_{\mathsf{inp}}, k^{(A,B)}_{\mathsf{inp}}\right]$:

1. Take as input $(\mathbf{x}_A, \mathbf{d}_A)$ which consists of A's input $\mathbf{x}_A$ on n qubits and a "dummy" input $\mathbf{d}_A$ on n qubits.
2. Sample classical strings $r_{\mathsf{inp}}^{(A,A)}, r_{\mathsf{inp}}^{(A,B)} \leftarrow \{0,1\}^\lambda$.
3. Compute $C_{\mathsf{inp}}^{(A,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{inp}}^{(A,A)}, r_{\mathsf{inp}}^{(A,A)}), C_{\mathsf{inp}}^{(A,B)} \coloneqq \mathsf{PRF}(k_{\mathsf{inp}}^{(A,B)}, r_{\mathsf{inp}}^{(A,B)})$.
4. Output
$$\begin{cases} \left(r_{\mathsf{inp}}^{(A,A)}, C_{\mathsf{inp}}^{(A,A)}(\mathbf{x}_A, \mathbf{0}^\lambda), r_{\mathsf{inp}}^{(A,B)}, C_{\mathsf{inp}}^{(A,B)}(\mathbf{d}_A, \mathbf{0}^\lambda)\right) & \text{if } b = 0 \\ \left(r_{\mathsf{inp}}^{(A,A)}, C_{\mathsf{inp}}^{(A,A)}(\mathbf{d}_A, \mathbf{0}^\lambda), r_{\mathsf{inp}}^{(A,B)}, C_{\mathsf{inp}}^{(A,B)}(\mathbf{x}_A, \mathbf{0}^\lambda)\right) & \text{if } b = 1 \end{cases}.$$

- $\mathcal{F}_{A,\mathsf{cmp}}\left[Q, k_{\mathsf{inp}}^{(A,A)}, k_{\mathsf{inp}}^{(B,A)}, k_{\mathsf{out}}^{(A,A)}, k_{\mathsf{out}}^{(A,B)}\right]$:

  1. Take as input $\left(r_{\mathsf{inp}}^{(A,A)}, \widehat{\mathbf{x}}_A, r_{\mathsf{inp}}^{(B,A)}, \widehat{\mathbf{x}}_B\right)$, where $\widehat{\mathbf{x}}_A$ and $\widehat{\mathbf{x}}_B$ are $(n+\lambda)$-qubit states.
  2. Compute $C_{\mathsf{inp}}^{(A,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{inp}}^{(A,A)}, r_{\mathsf{inp}}^{(A,A)}), C_{\mathsf{inp}}^{(B,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{inp}}^{(B,A)}, r_{\mathsf{inp}}^{(B,A)})$.
  3. Compute $C_{\mathsf{inp}}^{(A,A)}(\widehat{\mathbf{x}}_A)$ and measure the final λ qubits. If each is zero, let $\mathbf{x}_A$ be the remaining n-qubit state. Otherwise, abort.
  4. Compute $C_{\mathsf{inp}}^{(B,A)}(\widehat{\mathbf{x}}_B)$ and measure the final λ qubits. If each is zero, let $\mathbf{x}_B$ be the remaining n-qubit state. Otherwise, abort.
  5. Compute $(\mathbf{y}_A, \mathbf{y}_B) \coloneqq Q(\mathbf{x}_A, \mathbf{x}_B)$.
  6. Sample classical strings $r_{\mathsf{out}}^{(A,A)}, r_{\mathsf{out}}^{(A,B)} \leftarrow \{0,1\}^\lambda$.
  7. Compute $C_{\mathsf{out}}^{(A,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{out}}^{(A,A)}, r_{\mathsf{out}}^{(A,A)}), C_{\mathsf{out}}^{(A,B)} \coloneqq \mathsf{PRF}(k_{\mathsf{out}}^{(A,B)}, r_{\mathsf{out}}^{(A,B)})$.
  8. Output
  $$\left(r_{\mathsf{out}}^{(A,A)}, C_{\mathsf{out}}^{(A,A)}(\mathbf{y}_A, \mathbf{0}^\lambda), r_{\mathsf{out}}^{(A,B)}, C_{\mathsf{out}}^{(A,B)}(\mathbf{y}_B, \mathbf{0}^\lambda)\right).$$

- $\mathcal{F}_{A,\mathsf{out}}^{(b)}\left[k_{\mathsf{out}}^{(A,A)}, k_{\mathsf{out}}^{(B,A)}\right]$:

  1. Take as input $\left(r_{\mathsf{out}}^{(A,A)}, \widehat{\mathbf{y}}_A^{(0)}, r_{\mathsf{out}}^{(B,A)}, \widehat{\mathbf{y}}_A^{(1)}\right)$, where $\widehat{\mathbf{y}}_A^{(0)}$ and $\widehat{\mathbf{y}}_A^{(1)}$ are $(n+\lambda)$-qubit states.
  2. Compute $C_{\mathsf{out}}^{(A,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{out}}^{(A,A)}, r_{\mathsf{out}}^{(A,A)}), C_{\mathsf{out}}^{(B,A)} \coloneqq \mathsf{PRF}(k_{\mathsf{out}}^{(B,A)}, r_{\mathsf{out}}^{(B,A)})$.
  3. Compute $C_{\mathsf{out}}^{(A,A)}(\widehat{\mathbf{y}}_A^{(0)})$ and measure the final λ qubits. If each is zero, let $\mathbf{y}_A^{(0)}$ be the remaining n-qubit state. Otherwise, abort.
  4. Compute $C_{\mathsf{out}}^{(B,A)}(\widehat{\mathbf{y}}_A^{(1)})$ and measure the final λ qubits. If each is zero, let $\mathbf{y}_A^{(1)}$ be the remaining n-qubit state. Otherwise, abort.
  5. Output $\mathbf{y}_A^{(b)}$.

Now we are ready to describe the protocol.

Protocol 6

Common Information: Quantum circuit Q to be computed with $2n$ input qubits and $2n$ output qubits.

Party A Input: $\mathbf{x}_A$
Party B Input: $\mathbf{x}_B$

The Protocol:
Setup.

1. Sample 8 PRF keys $k_{\mathsf{inp}}^{(A,A)}, k_{\mathsf{inp}}^{(A,B)}, k_{\mathsf{inp}}^{(B,A)}, k_{\mathsf{inp}}^{(B,B)}, k_{\mathsf{out}}^{(A,A)}, k_{\mathsf{out}}^{(A,B)}, k_{\mathsf{out}}^{(B,A)}, k_{\mathsf{out}}^{(B,B)}$.
2. Publish the following obfuscations:

$$\begin{gathered}
\mathcal{O}_{A,\mathsf{inp}} \coloneqq \mathcal{O}\left(\mathcal{F}^{(1)}_{A,\mathsf{inp}}\left[k_{\mathsf{inp}}^{(A,A)}, k_{\mathsf{inp}}^{(A,B)}\right]\right), \mathcal{O}_{B,\mathsf{inp}} \coloneqq \mathcal{O}\left(\mathcal{F}^{(0)}_{B,\mathsf{inp}}\left[k_{\mathsf{inp}}^{(B,B)}, k_{\mathsf{inp}}^{(B,A)}\right]\right) \\
\mathcal{O}_{A,\mathsf{cmp}} \coloneqq \mathcal{O}\left(\mathcal{F}_{A,\mathsf{cmp}}\left[Q, k_{\mathsf{inp}}^{(A,A)}, k_{\mathsf{inp}}^{(B,A)}, k_{\mathsf{out}}^{(A,A)}, k_{\mathsf{out}}^{(A,B)}\right]\right), \\
\mathcal{O}_{B,\mathsf{cmp}} \coloneqq \mathcal{O}\left(\mathcal{F}_{B,\mathsf{cmp}}\left[Q, k_{\mathsf{inp}}^{(B,B)}, k_{\mathsf{inp}}^{(A,B)}, k_{\mathsf{out}}^{(B,B)}, k_{\mathsf{out}}^{(B,A)}\right]\right), \\
\mathcal{O}_{A,\mathsf{out}} \coloneqq \mathcal{O}\left(\mathcal{F}^{(1)}_{A,\mathsf{out}}\left[k_{\mathsf{out}}^{(A,A)}, k_{\mathsf{out}}^{(B,A)}\right]\right), \mathcal{O}_{B,\mathsf{out}} \coloneqq \mathcal{O}\left(\mathcal{F}^{(0)}_{B,\mathsf{out}}\left[k_{\mathsf{out}}^{(B,B)}, k_{\mathsf{out}}^{(A,B)}\right]\right).
\end{gathered}$$

Round 1.
Party A:

1. Compute $(st_{A,1}, \mathbf{st}_{A,1}, m_{A,1}, \mathbf{m}_{A,1}) \leftarrow \mathcal{O}_{A,\mathsf{inp}}(\mathbf{x}_A, \mathbf{0}^n)$, where $\mathbf{0}^n$ is the "dummy input".
2. Send to Party B: $(m_{A,1}, \mathbf{m}_{A,1})$.

Party B:

1. Compute $(st_{B,1}, \mathbf{st}_{B,1}, m_{B,1}, \mathbf{m}_{B,1}) \leftarrow \mathcal{O}_{B,\mathsf{inp}}(\mathbf{x}_B, \mathbf{0}^n)$, where $\mathbf{0}^n$ is the "dummy input".
2. Send to Party A: $(m_{B,1}, \mathbf{m}_{B,1})$.

Round 2.
Party A:

1. Compute $(st_{A,2}, \mathbf{st}_{A,2}, m_{A,2}, \mathbf{m}_{A,2}) \leftarrow \mathcal{O}_{A,\mathsf{cmp}}(st_{A,1}, \mathbf{st}_{A,1}, m_{B,1}, \mathbf{m}_{B,1})$.
2. Send to Party B: $(m_{A,2}, \mathbf{m}_{A,2})$.

Party B:

1. Compute $(st_{B,2}, \mathbf{st}_{B,2}, m_{B,2}, \mathbf{m}_{B,2}) \leftarrow \mathcal{O}_{B,\mathsf{cmp}}(st_{B,1}, \mathbf{st}_{B,1}, m_{A,1}, \mathbf{m}_{A,1})$.
2. Send to Party A: $(m_{B,2}, \mathbf{m}_{B,2})$.

Output Reconstruction.

- *Party A:* Compute $\mathbf{y}_A \leftarrow \mathcal{O}_{A,\mathsf{out}}(st_{A,2}, \mathbf{st}_{A,2}, m_{B,2}, \mathbf{m}_{B,2})$.
- *Party B:* Compute $\mathbf{y}_B \leftarrow \mathcal{O}_{B,\mathsf{out}}(st_{B,2}, \mathbf{st}_{B,2}, m_{A,2}, \mathbf{m}_{A,2})$.

Figure 6: Two-round two-party quantum computation.

Lemma 8.5. *Assuming quantum VBB obfuscation (Definition 8.4), the protocol described in Protocol 6 satisfies security against a malicious A and malicious B.*

Proof. (Sketch) First observe where the computation is taking place in an honest execution of the protocol. In the first round, A executes $\mathcal{O}_{A,\mathsf{inp}}$ and sends a Clifford encoding of its input $\mathbf{x}_A$ to B, while B executes $\mathcal{O}_{B,\mathsf{inp}}$ and sends a Clifford encoding of its dummy input $\mathbf{0}^n$ to A while keeping a Clifford encoding of its input $\mathbf{x}_B$ in its state. In the second round, B executes $\mathcal{O}_{B,\mathsf{cmp}}$ to produce a Clifford encoding of the output $(\mathbf{y}_A, \mathbf{y}_B)$, while A executes $\mathcal{O}_{A,\mathsf{cmp}}$ to produce a Clifford encoding of a dummy output. B sends the encoding of $\mathbf{y}_A$, which is decrypted by A using $\mathcal{O}_{A,\mathsf{out}}$ and B decrypts its output $\mathbf{y}_B$ using $\mathcal{O}_{B,\mathsf{out}}$.

Now note that it is straightforward to perfectly simulate a malicious A. The simulator can sample the CRS (so in particular it knows all the PRF keys) and then emulate an honest B, receiving A's encoded input, decrypting it, evaluating the circuit, and finally encoding A's output and sending it back.

Now consider sampling the CRS to be obfuscations of the functionalities

$$\mathcal{F}^{(0)}_{A,\mathsf{inp}}, \mathcal{F}^{(1)}_{B,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}, \mathcal{F}_{B,\mathsf{cmp}}, \mathcal{F}^{(0)}_{A,\mathsf{out}}, \mathcal{F}^{(1)}_{B,\mathsf{out}}.$$

This only differs from the real protocol in the superscript b values of the inp and out functionality. However, this completely reverses the flow of computation in an honest execution of the protocol. Now, A is computing the functionality on the actual inputs, while B is computing on the dummy inputs. Thus with this sampling of the CRS, it is straightforward to perfectly simulate a malicious B. It remains to show that these two methods of sampling the CRS are computationally indistinguishable.

Consider an adversary that can distinguish obfuscations of

$$\mathcal{F}^{(1)}_{A,\mathsf{inp}}, \mathcal{F}^{(0)}_{B,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}, \mathcal{F}_{B,\mathsf{cmp}}, \mathcal{F}^{(1)}_{A,\mathsf{out}}, \mathcal{F}^{(0)}_{B,\mathsf{out}}$$

from obfuscations of

$$\mathcal{F}^{(0)}_{A,\mathsf{inp}}, \mathcal{F}^{(1)}_{B,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}, \mathcal{F}_{B,\mathsf{cmp}}, \mathcal{F}^{(0)}_{A,\mathsf{out}}, \mathcal{F}^{(1)}_{B,\mathsf{out}}.$$

By the security of (our strong form of) VBB obfuscation, such an adversary implies a distinguisher that is only given oracle access to these functionalities. Now, since this distinguisher only has oracle access to the PRFs, one can replace them with truly random functions. At this point, due to the perfect hiding of the Clifford code, an adversary cannot obtain any information from the outputs of $(\mathcal{F}^{(0)}_{A,\mathsf{inp}}, \mathcal{F}^{(1)}_{B,\mathsf{inp}}, \mathcal{F}_{A,\mathsf{cmp}}, \mathcal{F}_{B,\mathsf{cmp}})$, except with negligible probability (this non-zero probability is due to the possibility of collision in sampling the r values used as inputs to the PRFs / random functions). Furthermore, due to the statistical authentication of the Clifford code, an adversary can only obtain an output from $(\mathcal{F}^{(b)}_{A,\mathsf{out}}, \mathcal{F}^{(1-b)}_{B,\mathsf{out}})$ with non-negligible probability if it emulates an honest execution of the protocol, starting with arbitrary inputs $(\mathbf{x}_A, \mathbf{d}_A)$ to $\mathcal{F}^{(b)}_{A,\mathsf{inp}}$ and $(\mathbf{x}_B, \mathbf{d}_B)$ to $\mathcal{F}^{(1-b)}_{B,\mathsf{inp}}$. But regardless of the value of b, the outputs of $(\mathcal{F}^{(b)}_{A,\mathsf{out}}, \mathcal{F}^{(1-b)}_{B,\mathsf{out}})$ will be $(\mathbf{y}_A, \mathbf{y}_B) \coloneqq Q(\mathbf{x}_A, \mathbf{x}_B)$. Thus switching the value of b at this point will be statistically indistinguishable, completing the proof.

□

Acknowledgements

This material is based on work supported in part by DARPA under Contract No. HR001120C0024 (for DK). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. AC acknowledges support of the DoD. Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing in Spring 2020.

References

[ABDS20] Gorjan Alagic, Zvika Brakerski, Yfke Dulek, and Christian Schaffner. Impossibility of quantum virtual black-box obfuscation of classical circuits. *arXiv preprint arXiv:2005.06432*, 2020.

[ABG$^+$20] Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Post-quantum multi-party computation in constant rounds. *ArXiv*, abs/2005.12904, 2020.

[ACGH19] Gorjan Alagic, Andrew M Childs, Alex B Grilo, and Shih-Han Hung. Non-interactive classical verification of quantum computation. *arXiv*, pages arXiv–1911, 2019.

[AF16] Gorjan Alagic and Bill Fefferman. On quantum obfuscation. *ArXiv*, abs/1602.01771, 2016.

[ALP20] Prabhanjan Ananth and Rolando L La Placa. Secure software leasing. *arXiv preprint arXiv:2005.05289*, 2020.

[BCG$^+$06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *47th FOCS*, pages 249–260. IEEE Computer Society Press, October 2006.

[BG19] Anne Broadbent and Alex B Grilo. Zero-knowledge for qma from locally simulatable proofs. *arXiv preprint arXiv:1911.07782*, 2019.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[BK05] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.

[BK11] Salman Beigi and Robert Koenig. Simplified instantaneous non-local quantum computation with applications to position-based cryptography. *New Journal of Physics*, 13(9):093036, sep 2011.

[Bra18] Zvika Brakerski. Quantum FHE (almost) as secure as classical. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 67–95. Springer, Heidelberg, August 2018.

[BY20] Zvika Brakerski and Henry Yuen. Quantum garbled circuits. *arXiv preprint arXiv:2006.01085*, 2020.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract) (informal contribution). In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, page 462. Springer, Heidelberg, August 1988.

[CCH$^+$19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *34th ACM STOC*, pages 643–652. ACM Press, May 2002.

[CVZ20] Andrea Coladangelo, Thomas Vidick, and Tina Zhang. Non-interactive zero-knowledge arguments for qma, with preprocessing. In *Annual International Cryptology Conference*, pages 799–828. Springer, 2020.

[DGJ$^+$20] Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 729–758. Springer, Heidelberg, May 2020.

[DNS10] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 685–706. Springer, Heidelberg, August 2010.

[DNS12] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012. Proceedings*, volume 7417 of *LNCS*, pages 794–811. Springer, 2012.

[DSWK06] Giacomo Mauro D'Ariano, D Schlingemann, RF Werner, and D Kretschmann. Quantum bit commitment revisited: the possible and the impossible. Technical report, 2006.

[GC20] Alvin Gonzales and Eric Chitambar. Bounds on instantaneous nonlocal quantum computation. *IEEE Trans. Inf. Theory*, 66(5):2951–2963, 2020.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[Goy18] Rishab Goyal. Quantum multi-key homomorphic encryption for polynomial-sized circuits. Cryptology ePrint Archive, Report 2018/443, 2018. https://eprint.iacr.org/2018/443.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[LC98] Hoi-Kwong Lo and Hoi Fung Chau. Why quantum bit commitment and ideal quantum coin tossing are impossible. *Physica D: Nonlinear Phenomena*, 120(1-2):177–187, 1998.

[LQR$^+$19] Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier NIZKs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 670–700. Springer, Heidelberg, August 2019.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[Mah18] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In Mikkel Thorup, editor, *59th FOCS*, pages 332–338. IEEE Computer Society Press, October 2018.

[May97] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical review letters*, 78(17):3414, 1997.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[Shm20] Omri Shmueli. Multi-theorem (malicious) designated-verifier nizk for qma, 2020.

[Spe16] Florian Speelman. Instantaneous non-local computation of low t-depth quantum circuits. In Anne Broadbent, editor, *11th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2016, September 27-29, 2016, Berlin, Germany*, volume 61 of *LIPIcs*, pages 9:1–9:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[Vai03] Lev Vaidman. Instantaneous measurement of nonlocal variables. *Phys. Rev. Lett.*, 90:010402, Jan 2003.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.