Round Efficient Secure Multiparty Quantum Computation with Identifiable Abort

Bar Alon[*1], Hao Chung[†2], Kai-Min Chung[‡3], Mi-Ying Huang[§3,4], Yi Lee[¶3], and Yu-Ching Shen[‖3]

[1]Department of Computer Science, Ariel University, Israel
[2]Department of Electrical and Computer Engineering, Carnegie Mellon University, USA
[3]Institute of Information Science, Academia Sinica, Taiwan
[4]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan

June 25, 2021

Abstract

A recent result by Dulek et al. (EUROCRYPT 2020) showed a secure protocol for computing any quantum circuit even without the presence of an honest majority. Their protocol, however, is susceptible to a "denial of service" attack and allows even a single corrupted party to force an abort. We propose the first quantum protocol that admits *security-with-identifiable-abort*, which allows the honest parties to agree on the identity of a corrupted party in case of an abort. Additionally, our protocol is the first to have the property that the number of rounds where quantum communication is required is *independent of the circuit complexity*. Furthermore, if there exists a post-quantum secure classical protocol whose round complexity is independent of the circuit complexity, then our protocol has this property as well. Our protocol is secure under the assumption that classical quantum-resistant fully homomorphic encryption schemes with decryption circuit of logarithmic depth exist. Interestingly, our construction also admits a reduction from quantum fair secure computation to classical fair secure computation.

Contents

1   Introduction

In the setting of secure multiparty computation (MPC), the goal is to allow a set of mutually distrustful parties to compute some function of their private inputs in a way that preserves some security properties, even in the face of adversarial behavior by some of the parties. Some of the desired properties of a secure protocol include correctness (cheating parties can only affect the output by choosing their inputs), privacy (nothing but the specified output is learned), fairness (all parties receive an output or none do), and even guaranteed output delivery (meaning that all honestly behaving parties always learn an output). Informally speaking, a protocol π computes a functionality f with full-security if it provides all of the above security properties.

It is well-known that, assuming an honest majority and a broadcast channel, any functionality can be computed with full-security [RBO89]. However, achieving fairness, and hence full-security, is impossible in general assuming no honest majority [Cle86]. Instead, one usually settles on a weaker notion called *security-with-abort*, which completely disregards fairness. Roughly, security-with-abort guarantees that either the protocol terminates successfully, in which case the honest parties receive their outputs, or the protocol aborts, in which case all honest parties learn that there was an attack. Note that since fairness is not guaranteed, it might be the case where the adversary learns the output of the corrupted parties. In many setting, however, security-with-abort is not enough, as an adversary can cause a *denial-of-service* attack by repeatedly aborting the protocol. Thus, it is highly desirable to consider the stronger security notion called *security-with-identifiable-abort* (SWIA) [IOZ14]. Here, if the protocol is aborted, then all honest parties additionally agree on an identity of a corrupted party. It is well-known that there are protocols admitting SWIA for any number of corrupted parties, e.g., the GMW protocol [GMW87].

In this work we consider the quantum version of MPC. In the *fully quantum* setting, the functionality – including the inputs and outputs – is quantum. As such, the parties, as well as the adversary attacking the protocol, are quantum. Secure multiparty quantum computation (MPQC) in the fully quantum setting, was first studied by [CGS02], who constructed a fully secure n-party protocol tolerating strictly less than $n/6$. The threshold $n/6$ was subsequently improved the more general honest majority setting [BOCG+06], assuming the availability of a classical broadcast channel. Similarly to the classical setting, if there is no honest majority, then full-security is impossible to achieve in general [ABDR04, Kit].[1] Moreover, [DNS12] presented a secure-with-abort protocol in the two-party case, and recently [DGJ+20] extended it to the multiparty case, tolerating any number of corrupted parties.

The protocol of [DGJ+20], however, does not admit *identifiable abort*. This follows from the fact that it is impossible to broadcast a quantum state. Therefore a corrupted party can accuse an honest party of not sending it a message, thus, not only is the quantum state lost, but the other parties cannot identify the corrupted party. When compared to the classical setting, this raises the following natural question.

> *Can any multiparty quantum circuit be computed with security-with-identifiable-abort, tolerating any number of corrupted parties?*

1.1   Our Results

In this paper, we answer the above question affirmatively. Additionally, our protocol is the first to have the property that the number of rounds where quantum communication is required is

---

[1]The impossibility proof is in the information theoretic setting, where the adversary is unbounded. However, even though Cleve's impossibility result is stated for classical protocols, the proof can still be applied for quantum protocols.

independent of the circuit complexity. Furthermore, if there exists a post-quantum secure classical protocol whose round complexity is independent of the circuit complexity, then our protocol has this property as well.

Similarly to [DGJ+20, DNS12], we present the results and the protocol, assuming the availability of a reactive trusted party, called cMPC, that is able to compute any *classical* multiparty functionality. We refer to this as the cMPC-*hybrid model*. Furthermore, we assume that the parties are able to broadcast classical messages. The implementation of cMPC can be done by first removing the reactive assumption using standard techniques, and then implement each call using a post-quantum secure-with-identifiable-abort protocol. We refer the reader to Section 3.2.1 for more details. We prove the following.

Theorem 1.1 (Informal)**.** *Assume the existence of a classical quantum-resistant fully homomorphic encryption scheme with decryption circuit of logarithmic depth. Then any multiparty quantum circuit can be computed with security-with-identifiable-abort tolerating any number of corrupted parties in the cMPC-hybrid model. Moreover, the round complexity of the quantum communication of the protocol is independent of the circuit complexity.*

The formal statement of the theorem appears in Section 4. A few notes are in place. First, Brakerski and Vaikuntanathan [BV11] showed that the existence of a fully-homomorphic encryption satisfying the conditions stated in Theorem 1.1 can be reduced to the *learning with errors* assumption.

Second, we note that the protocol can be split into an online phase and an offline phase, where the parties have yet to receive their inputs. In the offline phase, the parties prepare auxiliary magic states in order to compute quantum gates later in the online phase. In fact, it suffices that the parties know only an upper bound on the number of gates in the circuit before interacting in the offline phase.

Third, although the number of rounds requiring quantum information to be sent is independent of the circuit complexity (i.e., independent of the number of gates), it still depends on the number of parties, the number of input-qubits and output-qubits of the circuit, and the security parameter. Specifically, the offline phase consists of $O(n^4 \cdot \kappa)$ rounds, and the online phase consists of $O(n^3 \cdot (\ell_{\mathsf{in}} + \ell_{\mathsf{out}}))$ rounds, where n is the number of parties, κ is the security parameter, and ℓ_{in} and ℓ_{out} upper-bound the number of input-qubits and output-qubits of the circuit, respectively.

Fourth, although our protocol admits security-with-identifiable-abort, any single corrupted party can cause it to abort. It is arguably more desirable and an interesting open problem to have a protocol that requires the adversary to corrupt more parties to cause an abort.

Finally, an interesting consequence of our construction is that quantum fair secure computation can be implemented assuming the hybrid functionality cMPC is fair.[2]

In the following sections, we present the ideas behind the construction.

1.2 Our Techniques

In this section, we present the main ideas behind the construction of our protocol.

1.2.1 A Warm-Up: Reliable Transmission of Quantum States

Before presenting the general construction, let us consider the following simple task. Suppose that there are n parties $P_1, \ldots, P_n$, where P_1 – called the sender – holds a quantum state ρ. The

---

[2]Intuitively, fair computation means that either all parties receive their respective outputs, or none of them do.

goal of the parties is to send ρ to P_n – called the receiver – such that if either the sender or the receiver is corrupted and deviate from the protocol, then the other parties can identify which of them is corrupted. Moreover, this should hold even the corrupted party collude with some of the other parties in $\{P_2, \ldots, P_{n-1}\}$.

As stated before, simply having P_1 send ρ to P_n, and have P_n broadcast a complaint in case it did not receive a message, does not work. Indeed, it could be the case where the receiver is corrupted, and falsely accuse the sender of not sending ρ. Since broadcasting a quantum state is impossible, to the other parties, this scenario is identical to the case where a corrupted sender did not send ρ. Thus, the desired security property is not met. Moreover, due to the no-cloning theorem, the state ρ is now permanently lost, making it unclear as to how to proceed the protocol.

Dealing with false accusations. As such "packet loss" seems unavoidable, our first idea is to not send ρ directly, but rather to encode ρ using a *quantum error-correcting code* (QECC), that can tolerate d deletions, where d will be determined below. This generates an q-qubit codeword $(\sigma_k)_{k=1}^{q}$, for some q, which will then be transmitted qubit-by-qubit as explained below.[3] By doing so, P_n can still recover ρ as long as it receives enough qubits of the codeword.

We next explain how the parties can transmit the codeword's qubits in such a way that will allow them to identify the corrupted party, if such exists. For simplicity of the current discussion, let us assume that the adversary can perform one of the following two attacks. Either it does not send a message, or it can falsely accuse a party of not sending a message. Below we will explain how to remove this assumption and how to resist general malicious attackers. Under these simplifying assumptions, we can make the following observation. If P_n accused P_1 of not sending a message, then all parties know that at least one of them is corrupted. Therefore, they can agree to remove the channel between them, and have P_1 send the next qubit of the codeword via a different path. The parties continue in this fashion until either enough σ_k's where successfully transmitted to the receiver, or until there is no path from the sender to the receiver. Formally, the parties keep track of a simple and undirected graph G, which represents *trust* between parties, i.e., an edge between two vertices exists if and only if there was no accusation between the two parties that the vertices represent. Observe that in the above protocol, all honest parties form a clique in G. Thus, if G becomes disconnected, the honest parties can agree on a corrupted party *not* connected to them. Therefore, using a QECC that can tolerate $d = \Theta(n^2)$ deletions results in a secure protocol.

Dealing with general malicious behavior. Next, we show to remove the simplifying assumption of the behavior of the adversary, and allow it to tamper with the messages arbitrarily. Here, we utilize *quantum authentication codes* [BCG+02], that allow a party to verify if a quantum state was tampered with. However, in our protocol the parties must know where on the path the message had been tampered with (if any tampered occurred), in order to later remove the corresponding edge. To achieve this, we define a new primitive, which we call *sequential authentication* (SA), that allows the sender to transmit a qubit to the receiver along some path, so that if the qubit was tampered with, all parties know where on the path the tampering occurred. We then combine SA with the previous protocol that dealt with false accusations, to construct a secure-with-identifiable-abort protocol for the transmission of a quantum state. One subtlety in the final construction, is that any path from P_1 to P_n must go through all parties, so as to ensure that at least one honest party can verify the integrity of the message.

---

[3]Here we abuse the notation that we denote the kth qubit of the codeword σ_k, while these q qubits may be entangled.

We now describe the construction of a protocol for sequential authentication. The construction is inspired by the swaddling notion from [DNS12] and the public authentication test from [DGJ+20], which are both based on *Clifford authentication codes*. Let us first recall Clifford codes [ABOE10]. Given a m-qubit state ρ and a security parameter κ, the Clifford encryption[4] appends an auxiliary register $|0^\kappa\rangle\langle 0^\kappa|$, called *traps*. Then, a random Clifford operator E is sampled from the Clifford group acting on $m + \kappa$ qubits. Finally, the encryption outputs the ciphertext $E(\rho \otimes |0^\kappa\rangle\langle 0^\kappa|)E^\dagger$, where E serves as the secret key. The decryption of a Clifford ciphertext σ, simply applies $E^\dagger$ to σ and measures the last κ trap qubits. If the measurement outcome is all-zero, then the decoding algorithm outputs the resulting state of the first m qubits. Otherwise, it rejects. The security of Clifford codes stems from the fact that any operation that is applied to the ciphertext, will flip each qubit in the trap with noticeable probability upon measurement. Moreover, the secret key of the Clifford code can be sampled efficiently by a classical algorithm [DLT02].

Constructing a sequential authentication protocol. We utilize these property to build a protocol for SA. Suppose that a message ρ is going to be transmitted through ℓ parties. Let us first present a naïve solution. The first party on the path will append $\ell\kappa$ qubits of $|0\rangle$ to ρ. Then, using the classical MPC functionality cMPC, the parties will securely sample for P_1 a Clifford key E_1 to encrypt its state. It then sends the encrypted message to P_2. To verify the authenticity of the state, the parties will again use cMPC for sampling a Clifford $V_2 = E_2 E_1^\dagger$, where E_2 acts only on the first $(\ell - 1)\kappa$ qubits. We then let P_2 receive V_2 and apply it to the encrypted message it received from P_1. This allows P_2 to measure the last κ qubits and compare them to zero. For each party P_i on the transmitting path, P_i measures κ qubits of traps. The parties can then continue in this fashion. Notice, however, that a corrupt P_1 might only append the last κ qubits honestly, which will not be immediately detected by P_2. This could later result in an honest party accusing another honest party. To overcome this issue, we use a similar trick to the public authentication test [DGJ+20], and have the Clifford V_2 that cMPC sampled include a random invertible linear transformation over $\mathbb{F}_2$ acting on all traps. Specifically, we let $V_2 = E_2 G_2 E_1^\dagger$, where we abuse notations and let $G_2 |x\rangle = |G_2(x)\rangle$. Observe that if P_1 did not prepare the traps correctly, then upon measurement with high probability P_2 will not obtain all-zero.

1.2.2 Security With Packet Drops

With the above technique, it is natural to incorporate it into the construction of [DGJ+20]. This naïve solution, however, does not work. Towards explaining the issue, let us first briefly describe the protocol of [DGJ+20]. Roughly, their protocol starts with an input encoding phase, such that at the end of the phase each party's input is encrypted under a Clifford code with cMPC holding the secret Clifford key. This is done similarly to the sequential authentication protocol described earlier. The parties then proceed to perform computation over the encrypted inputs. Computation over single-qubit Clifford gates can be done by simply letting cMPC update its key, while CNOT gates require communication since the inputs to CNOT gates are encrypted separately under different Clifford keys.

While the input encoding phase can be modified to admit security-with-identifiable-abort, it is unclear how to modify the computation phase of the protocol. This follows from the fact that the parties are required to use QECC over their inputs in the input encoding phase, thus at the end

---

[4]It is more common to use the term Clifford encoding. However, in the quantum setting authentication implies encryption. Thus, we refer to these as encryptions to remove confusion with the QECC encoding.

of this phase, each party will hold a Clifford encoding of each qubit of its input's codeword. As a result, the parties have to either perform the computation over QECC codewords in some way, or decode the Clifford encrypted codewords and perform computation similarly to [DGJ⁺20]. We next give an intuitive explanation as to why both solutions fail.

Let us first argue why the second solution fails. That is, suppose the parties decode all QECC encodings before starting to perform any computation. The issue here is that once the parties decode the QECC they lose its protection, hence the protocol cannot tolerate losing quantum states after this step. Since the protocol of [DGJ⁺20] requires communicating quantum messages to compute CNOT gates, this causes inevitable packet drops *during computation*, causing the honest parties to output incorrect values.

The former solution fails due to the fact that a corrupted party might not encode its qubit correctly using the QECC. Observe that our sequential authentication protocol will not be able to detect such error, since it is able to detect an attack only after a Clifford had been applied. Furthermore, this error might propagate into the evaluation. Indeed, consider the following example.

Suppose that the parties use repetition code as an implementation of the QECC.[5] In repetition code, a logical zero $|\bar{0}\rangle$ is encoded as $|000\rangle$ and a logical one $|\bar{1}\rangle$ is encoded as $|111\rangle$. The decoding is done by taking the majority, e.g., $|000\rangle$, $|001\rangle$, $|010\rangle$ and $|100\rangle$ are all decoded to $|\bar{0}\rangle$. Suppose three parties wish to compute the following circuit, where the CNOTs are applied transversally, and where the inputs $|\psi_i\rangle$ are repetition codes of logical $|\bar{0}\rangle$.
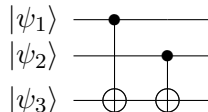
Clearly, in an honest execution the value of $|\psi_3\rangle$ becomes $|000\rangle$ which decodes to $|0\rangle$. Now, suppose the two parties holding $|\psi_1\rangle$ and $|\psi_2\rangle$ are corrupted and prepares $|\psi_1\rangle = |001\rangle$ and $|\psi_2\rangle = |010\rangle$. Then the value of $|\psi_3\rangle$ under such an attack becomes $|011\rangle$. Consequently, even if all codewords are of logical 0 at the beginning, the decoding would result in a logical 1.

A possible way to try and fix this issue, would be to try to *correct* the QECC codewords. However, this in particular would require the parties to compute a multi-qubit gate (e.g., CNOT), which as stated before, cannot be done without losing the quantum states due to a potential attack.

With this state of affairs, we aim to construct a protocol that has the property that no adversary can cause qubits to be "dropped" during the computation of the circuit. Thus, we first propose an abstraction of a security notion that allows the adversary to "drop" some of the input-states and output-states. We call this security notion secure-with-identifiable-abort-and-packet-drop (IDPD-security). We then show how to reduce the problem of constructing a secure-with-identifiable-abort protocol to the problem of constructing an IDPD-secure protocol.

Defining IDPD-security.   Let us now define IDPD-security. Similarly to other notions of security in multiparty computation, here we follow the standard ideal vs. real paradigm. Roughly, the ideal-world follows similar instructions to that of the security-with-identifiable-abort ideal-world, with the following two additions. First, when the parties send their inputs to the trusted party, the adversary additionally sends it a bounded-sized set, representing which input-qubits are to be replaced with $|0\rangle$ (modelling "packet drop"). Note that it might be the case where a single party holds several qubits as inputs, and the adversary changes only a subset of them to the 0 state. The second change we make is done after the adversary receives its output from the trusted party. Here,

---

[5]Repetition codes only resist bit-flip error (i.e., Pauli X attack). However, it is sufficient for the purposes of demonstration here.

the adversary either instructs the trusted party to abort while revealing the identity of a corrupted party, or it instructs the trusted party to continue and drop some qubits from the output.[6] In case the adversary instructed to continue, the trusted party then sends to all other parties their respective outputs that remained. Additionally, the trusted party reveals which input-qubits and which output-qubits were dropped. The formal definition of IDPD-security can be found in Section 3.1.

Reducing SWIA to IDPD-security. We now show a simple reduction from SWIA to IDPD-security. The reduction makes use of a QECC. Let C be the circuit that the parties wish to compute. First, each party encodes its input using the QECC. The parties then use an IDPD-secure protocol in order to compute the circuit C' that first decodes its inputs using the QECC, then applies C, and finally re-encodes each output using the QECC. Upon receiving their encoded outputs, each party locally decodes it to obtain their output. To see why this reduction works, observe that the adversary can only drop some of the qubits in the input to C' and some of the qubits in the output. Therefore, by the properties of the QECC and IDPD-security, either the original state can be reconstructed, or the adversary has revealed the identity of a corrupted party.

1.2.3 Securely Computing A General Circuit

We next explain how to achieve a secure protocol for computing a general circuit. With the above reduction, it suffices to construct an IDPD-secure protocol. Unfortunately, previous approaches, such as that of [DGJ+20], for constructing secure protocols fail to achieve IDPD-security. Indeed, as stated before, the protocol of [DGJ+20] requires communicating quantum messages to compute CNOT gates, which causes inevitable packet drops during computation and thus fails to achieve IDPD-security.

Our approach. To circumvent the aforementioned issue, the parties need a way to perform computation *without* quantum communication. To do so, our main idea is to delegate the computation to some designated party, say P_1, and let it perform computation under *verifiable quantum fully homomorphic encryption* (VQFHE) [ADSS17]. More precisely, the first step of our protocol will encrypt all parties' input using the VQFHE scheme of [ADSS17], called TrapTP, send their encrypted inputs to P_1, and store the VQFHE classical secret key sk in cMPC. We refer to this step as the *pre-computation* step. This allows us to let P_1 perform the computation homomorphically to obtain encrypted output without any quantum communication. Furthermore, the verification of the evaluation can be done using the help of cMPC holding sk. If the verification passes, P_1 delivers the output to each party. Note that an additional advantage of our approach is that the round complexity of our protocol is independent of the circuit complexity.

VQFHE scheme TrapTP. We first review some useful facts about the TrapTP scheme. In TrapTP, the encryption of a 1-qubit state $|\psi\rangle$ consists of a quantum part and a classical part. The quantum part is a *trap code* encryption of $|\psi\rangle$

$$\Pi X^x Z^z (\mathsf{QECC.Enc}(|\psi\rangle) \otimes |0\rangle^{\otimes\kappa} \otimes |+\rangle^{\otimes\kappa}),$$

where Π is a random permutation over 3κ qubits (which is part of the secret key sk) and $x, z \leftarrow \{0,1\}^{3\kappa}$ are sampled independent and uniformly at random. The classical part is a classical FHE

---

[6]Formally, the ideal-world is parametrized by two polynomial in the security parameter that bound the number input-qubits and number of output-qubits that can be dropped.

encryption of the Pauli key x, z. Homomorphic evaluation requires a quantum evaluation key ρ_{evk}, which consists of multiple TrapTP encryptions of magic states, including ancilla zero states, phase (P) states $|P\rangle := P|+\rangle$, Hadamard (H) states $|H\rangle := (H \otimes I)\mathsf{CNOT}(|+\rangle \otimes |0\rangle)$, T states $|T\rangle := T|+\rangle$, and a special gadget state $|\gamma\rangle$ (see Section 7.3 for a more detailed definition of $|\gamma\rangle$). These (encrypted) states are used to perform computation homomorphically over the underlying trap codes.

The pre-computation step. Recall that the goal is to send TrapTP encrypted inputs to P_1, with the secret key stored in cMPC. The first step is to let each party send their input to P_1 using the technique we developed in Section 1.2.1. Namely, we let each party to send Clifford encryptions of their input qubits using sequential authentication protocol through paths determined by a trust graph G. We formalize this as an *authenticated routing* (AR) protocol that achieves the following functionality with IDPD-security.

> **Authenticated Routing (AR):** As input, each sender P_i holds multiple quantum messages $\rho_1, \ldots, \rho_\ell$ (the "packets") to send to P_1. As output, the receiver P_1 receives Clifford ciphertexts $\sigma_j = E_j(\rho_j \otimes |0^t\rangle\langle 0^t|)E_j^\dagger$ with trap size t and cMPC receives the Clifford keys E_j for $j \in [\ell]$ with at most n^2 packet drop.

We note that in AR, a packet ρ_j can consist of multiple qubits and the trap size can be set arbitrarily; these properties will be useful later. Here, we let each P_i send their input qubit-by-qubit to P_1 using AR with trap size $3\kappa - 1$. After that, P_1 holds Clifford encodings of all parties' input (with certain packet drops). Note that AR allows to drop at most n^2 input states, while it is acceptable in IDPD-security.

However, in TrapTP, the quantum messages are encrypted under trap code instead of Clifford code. We next use the following simple *re-encrypt* protocol to turn Clifford codes into trap codes: Let $\sigma = E(\rho \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|)E^\dagger$ be a Clifford encoding of ρ held by P_1 with the corresponding Clifford key E held by cMPC. We simply let cMPC send to P_1 the Clifford operator

$$V = X^x Z^z \Pi (U_{\mathsf{Enc}} \otimes I^{\otimes \kappa} \otimes H^{\otimes \kappa}) E^\dagger,$$

where U_{Enc} is an unitary operator maps $\rho \otimes |0^{\kappa-1}\rangle\langle 0^{\kappa-1}|$ into an QECC codeword. Observe that if P_1 applies V to σ, the result would be a trap-code encryption of ρ, which is also the quantum part of the TrapTP encryption of ρ. Also note that since the Clifford key E is uniformly random to P_1, it serves as a one-time pad, hence P_1 learns nothing about the trap code secret Π, x, z from V. After that, we can let cMPC generate and send the classical part of the TrapTP encryption of ρ to P_1 so that it obtains a complete TrapTP encryption of ρ.

It is worth mentioning that a natural alternative is to use trap code to construct SA in AR to avoid using two different codes with re-encryption. However, this does not provide a secure protocol since, unlike Clifford codes, in trap codes each qubit is encrypted individually. If only one qubit has been tampered with, then there is no guarantee that the adversary would be immediately caught.

To conclude the pre-computation step, it is left to prepare the evaluation key ρ_{evk} for P_1, which consists of multiple TrapTP encryptions of auxiliary magic states and a special gadget state. Preparing such states turns out to be involved, which we discuss next.

Magic state preparation (except T). We first note that it suffices to generate Clifford encryption of these states, and we can apply the above re-encryption protocol to turn them into TrapTP encryption.

Let us start with the simplest case of ancilla zero state $|0\rangle$. For this, we can use the AR protocol to send the the empty state, denoted ε, with trap size 3κ to prepare it. Indeed, the Clifford encoding outputs

$$E(\varepsilon \otimes |0^{3\kappa}\rangle\langle 0^{3\kappa}|)E^\dagger = E(|0\rangle\langle 0| \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|)E^\dagger,$$

as required. Note that AR protocol takes as input a list of "packets," where n^2 packets may be dropped. Since magic state preparation is independent to parties' private states, the parties actually call AR protocol with $n^2 + 1$ packets to make sure that at least one packet can be delivered. Then, the server and cMPC keep the lexicographically first remaining packet. For simplicity, we omit the number of initial packets.

Next, consider preparing a $|P\rangle$ magic state. Since a P gate is a Clifford, we can generate it by preparing encoding of $|0\rangle$ and update the Clifford key held by cMPC. Specifically, if cMPC updates its Clifford E to $E(PH)^\dagger$ (where PH is applied only to the first qubit of the codeword), then decrypting the ciphertext with the updated key would result in

$$(E(PH)^\dagger)^\dagger E(|0\rangle \otimes |0^{3\kappa-1}\rangle) = PH(|0\rangle \otimes |0^{3\kappa-1}\rangle) = |P\rangle \otimes |0^{3\kappa-1}\rangle.$$

The $|H\rangle$ magic state, is also generated by a Clifford, but consists of two qubits. To generate this, we first use AR to send the the empty state with trap size 6κ and view it as

$$E(|0\rangle^{M_1} \otimes |0\rangle^{M_2} \otimes |0^{3\kappa-1}\rangle^{T_1} \otimes |0^{3\kappa-1}\rangle^{T_2}),$$

where the gray superscript denote the registers the qubits are stored in. Then, we let cMPC send to P_1 the Clifford operator

$$V = (E_1^{M_1 T_1} \otimes E_2^{M_2 T_2})(H \otimes I)\mathsf{CNOT}(H \otimes I)^{M_1 M_2} E^\dagger,$$

where E_1 and E_2 are two Clifford sampled uniformly at random and independently, and where the gray superscript denote the registers on which each operator acts. Observe that upon applying V to its codeword, P_1 will obtain an encrypted H state. Additionally, as V is distributed like a uniform random Clifford operator, it follows that a corrupted P_1 will gain no new information.

More generally, the above examples suggest that we can prepare any ℓ-qubit state in the Clifford group by first preparing Clifford encoding of $3\ell\kappa$ qubits $E|0^{3\ell\kappa}\rangle$ using AR, and letting cMPC send Clifford operator V to instruct P_1 to prepare the Clifford state and split it into ℓ Clifford encodings of each qubit. We note that the special gadget state $|\gamma\rangle$ is of this type and therefore can be prepared in this way.

T **magic state preparation.** Among all magic states, the preparation of $T := T|+\rangle$ magic state is the most difficult, since T is not a Clifford operator. We follow a similar approach to that of [DGJ+20], but with modifications to achieve security-with-identifiable-abort. Here, we give a brief overview of their construction and discuss the required modifications.

At a high-level, the protocol asks a party, say P_1, to prepare a large number N of (supposedly) $|T\rangle$ states under Clifford encoding with Clifford keys stored in cMPC. This can be done by, e.g., letting P_1 send these states using AR in our context. Then, the parties randomly distribute these encoded states among themselves, and have $\mathrm{P}_2, \ldots, \mathrm{P}_n$ verify that they are indeed $|T\rangle$ states. This is done by sending the Clifford keys to P_i, and having P_i measure the decoded states in the $\{|T\rangle, |T^\perp\rangle\}$-basis. If any $|T^\perp\rangle$ outcome is detected, the protocol aborts. If not, then we know that the states held in P_1 contains only a small number of errors with high probability. The protocol then apply a T state distillation circuit (over the encoded states) to distill the desired T magic

states.

To achieve security-with-identifiable-abort, we cannot let the protocol be aborted when an error is detected, since the parties cannot distinguish the case where the error was due to a malicious P_1 preparing incorrect states, or a malicious party P_i falsely reporting the error. Thus, to identify the malicious party, we let each party P_i report its *error rate* ϵ_i, i.e., the fraction of $|T^\perp\rangle$ outcomes it obtained, to cMPC with ϵ_1 set to 0. cMPC then sort these numbers, and check if there are two consecutive numbers with difference greater than a certain threshold δ that is larger than expected sampling errors. If so, cMPC finds the smallest such pairs, say, they are $\epsilon_i < \epsilon_j$ reported by P_i and P_j, respectively, and publish the result. The parties then abort, with an honest party P_k identifying P_i (resp., P_j) as the malicious party if $\epsilon_k \geq \epsilon_j$ (resp., $\epsilon_k \leq \epsilon_i$). Intuitively, this works since all honest parties should obtain roughly the same error rate up to a small sampling error, and hence they will belong to the same side and accuse the same party being the malicious party. Also, if the protocol does not abort, it means that all reported error rates are small, since $\epsilon_1 = 0$ and we still have the guarantee that the error rate of the states held in P_1 is small.

The second issue is that we need to be able to apply the T state distillation circuit to the (Clifford encrypted) states held by P_1, which is a classically-controlled Clifford circuit (A circuit consists of Clifford gates and measurements, and which Clifford gates should be applied depends on all previous measurement outcomes.). If these states are encrypted separately, then we do not know how to compute the distillation circuit without quantum communication, as this is the problem we want to solve to begin with. Fortunately, as discussed above, if these states are encrypted as a single Clifford ciphertext of a multi-qubit message, then we can perform Clifford operation on the underlying message and split it into multiple Clifford ciphertexts of smaller messages by letting cMPC sending proper Clifford instruction to P_1. We can further extend it to evaluate classically-controlled Clifford circuit. Based on this observation, we let P_1 to prepare the N copies of $|T\rangle$ states and send it as a N-qubit quantum message $\rho = |T\rangle^{\otimes N}$ in AR (with a sufficiently large trap size). This allows us to distribute the states to all parties (by splitting the ciphertexts) and apply the T state distillation circuit to the states held by P_1 later.

Final issue: re-encryption to Clifford codes. The computation step is rather straightforward, so we do not discuss the details here but just state that as a result, P_1 holds trap code encoding of the output. All that is left is to show how it can distribute each output to its corresponding party. The idea is to reverse the operations done until now. That is, to first re-encrypt the trap codes back to Clifford codes, and then use AR to distribute the outputs. The final issue is that re-encrypting trap code to Clifford cannot be done in the same way as it was done in the other direction. This is because before, we use the randomness of the Clifford key as one-time pad to protect the trap code key, but now the randomness in the trap code key is not enough to protect the Clifford key.

To resolve the issue, we again use AR. Let us say σ is a trap code that P_1 needs to send to a party P_i. We let P_1 send ρ as a 3κ-qubit message to itself using AR. As a result, P_1 will receive a Clifford encoding $\sigma = E(\rho \otimes |0^t\rangle\langle 0^t|)E^\dagger$ (with a sufficiently large trap size t) for which we can let P_1 perform Clifford operation on the underlying message ρ. Note that if P_1 is malicious, the underlying message ρ of σ may not be a valid trap code. Thus, we let P_1 and cMPC verify and decode the supposedly trap code ρ. Specifically, cMPC will check the classical parts of the computation. If the verification rejects, we abort and identify P_1 as the malicious party. If it passes, then we obtain a Clifford encoding of the qubit underlying the trap code as desired. Finally, we remind the reader that some of the trap codes in ρ may be dropped by AR, but this is allowed since IDPD-security allows to drop part of the output qubits.

1.3 Roadmap

In Section 2 we provide the required preliminaries. In Section 3 we explain in detail the model of our computation. Then, in Section 4 we state our main theorem and show the reduction to IDPD-security. In Section 5 we give the construction of sequential authentication, and in Section 6 we use it to construct authenticated routing. These constructions admits information theoretic security. Following that, in Section 7 we show how to prepare all required magic states. In Section 8 we show how to securely compute the pre-computation protocol, Section 9 is dedicated to performing the computation of the circuit, and finally, in Section 10 we show how the parties can distribute the output securely. We note that only the computation protocol from Section 9 has computational security.

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, let $[n] = \{1, 2 \ldots n\}$. We also let Sym_n to denote the symmetric group over n symbols, and let $\mathrm{GL}(n, \mathbb{F}_2)$ denote the general linear group over $\mathbb{F}_2^n$. Given a binary string x, we write $|x|$ to denote the length of x, and $w(x)$ to denote the *relative Hamming weight* of x which equals to Hamming weight of x divided by $|x|$. For a string x and a subset $\mathcal{S} \subseteq [|x|]$, we use $x_{\mathcal{S}}$ to denote the substring of x indicated by $\mathcal{S}$.

Given a set $\mathcal{S}$, we write $|\mathcal{S}|$ to denote the cardinality of $\mathcal{S}$, and write $s \leftarrow \mathcal{S}$ to indicate that s is selected uniformly at random from $\mathcal{S}$. Given a random variable (or a distribution) X, we write $x \leftarrow X$ to indicate that x is selected according to X. A function $\mu \colon \mathbb{N} \to [0, 1]$ is called negligible, if for every positive polynomial $p(\cdot)$ and all sufficiently large n, it holds that $\mu(n) < 1/p(n)$. We use $\mathrm{neg}(\cdot)$ to denote an unspecified negligible function.

For $n \in \mathbb{N}$, we use $\mathcal{H}^n$ to denote the Hilbert space of n qubits. A density matrix is a positive semidefinite operator with unit trace. We write $D(\mathcal{H})$ to denote the set of density matrices over the Hilbert space $\mathcal{H}$, and let $\mathcal{D}^\ell := D(\mathcal{H}^\ell)$. We define $\mathcal{D}^* := \bigcup_{\ell=0}^{\infty} D(\mathcal{H}^\ell)$ to denote the set of the density matrices acting on the Hilbert space of arbitrary number of qubits. We use lowercase Greek alphabets, e.g., ρ, σ, τ, to denote quantum state.

A *quantum register* or quantum system is a physical object that can store quantum information. We use capital Latin alphabets, e.g., A, B, M, T, to denote quantum registers. For a quantum register A, we write $|A|$ to denote the number of qubit in it. The content of a quantum register is called a *quantum state* and is modelled by a density matrix. We denote the Hilbert space of a quantum register A by $\mathcal{H}_A$. The Hilbert space $\mathcal{H}_{AB}$ of a joint quantum register AB is the tensor product of the Hilbert spaces of each subsystems, that is, $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$. It will be convenient to denote by $\varepsilon \in \mathcal{D}^0$ the empty state

The trace distance between two quantum states ρ and σ, denoted as $\Delta(\rho, \sigma)$, is define by

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

where $\|M\|_1 = \mathrm{tr}\left(\sqrt{M^\dagger M}\right)$ is the trace norm of a matrix. Let $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. An EPR pair is the two-qubit state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

A *state ensemble* $\rho = \{\rho_{a,\kappa}\}_{a \in \mathcal{D}_\kappa, \kappa \in \mathbb{N}}$ is an infinite sequence of quantum states indexed by $a \in \mathcal{D}_\kappa$ and $\kappa \in \mathbb{N}$, where $\mathcal{D}_\kappa$ is a domain that might depend on κ. When the domains of a and κ are clear from context, we remove them for brevity. We write $\rho \approx_{\mathrm{neg}(\kappa)} \sigma$ if there exists a negligible function

μ, such that for all $\kappa \in \mathbb{N}$ and $a \in \mathcal{D}_\kappa$, it holds that

$$\Delta(\rho_{a,\kappa}, \sigma_{a,\kappa}) \leq \mu(\kappa).$$

We sometimes abuse notations and write $\rho_{a,\kappa} \approx_{\text{neg}(\kappa)} \sigma_{a,\kappa}$.

Let QPT stand for quantum polynomial time. Computational indistinguishability is defined as follows.

Definition 2.1. Let $\rho = \{\rho_{a,\kappa}\}_{a \in \mathcal{D}_\kappa, \kappa \in \mathbb{N}}$ and $\sigma = \{\sigma_{a,\kappa}\}_{a \in \mathcal{D}_\kappa, \kappa \in \mathbb{N}}$ be two ensembles. We say that ρ and σ are *computationally indistinguishable*, denoted $\rho \overset{\text{C}}{\equiv} \sigma$, if for every non-uniform QPT distinguisher D, there exists a negligible function $\mu(\cdot)$, such that for all $\kappa \in \mathbb{N}$ and $a \in \mathcal{D}_\kappa$, it holds that

$$|\Pr[\mathsf{D}(\rho_{a,\kappa}) = 1] - \Pr[\mathsf{D}(\sigma_{a,\kappa}) = 1]| \leq \mu(\kappa).$$

2.2 Quantum Computation

The Pauli X gate, Pauli Z gate, Hadamard gate H, phase gate P, $\pi/8$ gate T and CNOT gate are defined as

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \; H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix},$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}, \text{ and } \mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

We let I denote identity operator.

For a quantum operator U, we write U^A to specify that the quantum operator U acts on register A. Similarly, we write ρ^A to specify that the quantum state ρ lies in register A. Here, the register written in gray on the superscript is only for reminder, and whether it is written does not change the meaning of the operator or the state. That is, $U^A = U$ and $\rho^A = \rho$. We write χ^A to denote the maximally mixed state $I^A/|A|$ of register A.

For an ℓ-bit string $r = r_1 r_2 \ldots r_\ell$ and a quantum operator U, we let $U^r = U^{r_1} \otimes U^{r_2} \otimes \cdots \otimes U^{r_\ell}$. When $\Pi \in \text{Sym}_\ell$ is used as an unitary operator, we say it is to permute ℓ qubits according to Π, for example,

$$\Pi(|\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_\ell\rangle) = |\phi_{\Pi(1)}\rangle \otimes |\phi_{\Pi(2)}\rangle \otimes \cdots \otimes |\phi_{\Pi(\ell)}\rangle.$$

Let $Y = iXZ$. We use $\mathcal{P}_\ell$ to denote the set of ℓ-qubit Pauli operators:

$$\mathcal{P}_\ell = \{P = P_1 \otimes P_2 \otimes \cdots \otimes P_\ell : P_1, P_2, \ldots P_\ell \in \{I, X, Y, Z\}\}.$$

The ℓ-qubit Pauli set $\mathcal{P}_\ell$ forms a complete basis of all $2^\ell \times 2^\ell$ complex matrix. That is, any $U \in \mathbb{C}^{2^\ell \times 2^\ell}$ can be written as

$$U = \sum_{a,b \in \{0,1\}^\ell} \alpha_{a,b} X^a Z^b, \tag{1}$$

where each $\alpha_{a,b} \in \mathbb{C}$.

A Clifford operator maps a Pauli operator to a Pauli operator, up to a phase of ± 1 or $\pm i$. We use $\mathcal{C}_\ell$ to denote the set of Clifford operators acting on ℓ qubits. For any $P \in \mathcal{P}_\ell$ and $C \in \mathcal{C}_\ell$,

14

it holds that $\alpha CPC^\dagger \in \mathcal{P}_\ell$ for some $\alpha \in \{\pm 1, \pm i\}$. The Clifford group can be generated by $\{X, Z, H, P, \mathsf{CNOT}\}$ [Got98]. The universal quantum circuit can be implemented by Clifford group, T gate and the computational-basis measurement [BMP$^+$00]. It is well known that a uniform random Clifford group element over n qubits can be sampled by a classical algorithm efficiently [DLT02].

The Pauli and Clifford group satisfy the following properties.

Lemma 2.2 (Quantum one time pad [Chi05]). *Let $\rho^{AR} \in D(\mathcal{H}_A \otimes \mathcal{H}_R)$ be a quantum state lying on a composite system AR, and let $n = |A|$. Then it holds that*

$$\frac{1}{|\mathcal{P}_n|} \sum_{P \in \mathcal{P}_n} P^A \rho^{AR} P^\dagger = \chi^A \otimes \operatorname{tr}_A \rho^{AR}. \tag{2}$$

Lemma 2.3. *Let A be a quantum register of size n. Then for all $\rho \in D(\mathcal{H}_A)$, it holds that*

$$\frac{1}{|\mathcal{C}_A|} \sum_{C \in \mathcal{C}_{|A|}} C\rho C^\dagger = \chi.$$

A quantum channel is a completely positive trace-preserving (CPTP) linear map $\Xi : D(\mathcal{H}_A) \to D(\mathcal{H}_B)$. Similarly to operators, we let $\Xi^{A \to B}$ denote that Ξ maps from register A to register B. We can also define twirling on a quantum channel. If Ξ is a quantum channel, we say twirling over $\mathcal{S}$ on Ξ is choosing an unitary operator U uniformly at random from $\mathcal{S}$ and applying it on Ξ. It is equivalently to apply U on the input state ρ of Ξ, then to apply the quantum channel Ξ, and finally to apply the inverse of U.

The diamond norm of a quantum channels is the trace norm of its output state, maximized over all input-states (which can include an reference system).

Definition 2.4 (Diamond norm). Let $\Xi : D(\mathcal{H}_A) \to D(\mathcal{H}_B)$ be a quantum channel. The diamond norm of Ξ, denoted $\|\Xi\|_\diamond$, is defined as

$$\|\Xi\|_\diamond := \max_\rho \|(\Xi^{A \to B} \otimes I^R)(\rho^{AR})\|_1, \tag{3}$$

where the identity operator I acts on an auxiliary Hilbert space $\mathcal{H}_R$ and $\rho \in D(\mathcal{H}_A \otimes \mathcal{H}_R)$.

2.3 Useful Concentration Inequalities

The following are some useful concentration bounds.

Lemma 2.5 (Hoeffding's inequality for the hypergeometric distribution). *Let $x \in \{0,1\}^n$ be an arbitrary string with relative Hamming weight $\mu = w(x)$. Let $S \subseteq [n]$ be a random subset of size k sampled uniformly at random. Let $X_1, \ldots, X_k$ denote the values of x_S. Then for all $\delta > 0$, it holds that*

$$\Pr \left[\left| \frac{1}{k} \sum_{i=1}^{k} X_i - \mu \right| \geq \delta \right] \leq 2e^{-2\delta^2 k}.$$

A simple corollary of Hoeffding's bound states that a random substring of half the length as the relative Hamming weight as the complement string.

Corollary 2.6. *Let $x \in \{0,1\}^{2n}$ and let $S \subset [2n]$ be a subset of size n sampled uniformly at random. Let $S^\complement = [2t] \setminus S$. Then for all $\delta > 0$, it holds that*

$$\Pr \left[|w(x_S) - w(x_{S^\complement})| \geq \delta \right] \leq 4e^{-2(\frac{\delta}{2})^2 n}.$$

15

Proof. Let $\mu = w(x)$. By Lemma 2.5, we have that

$$\Pr\left[|w(x_S) - \mu| \geq \frac{\delta}{2}\right] \leq 2e^{-2(\frac{\delta}{2})^2 n}.$$

Moreover, observe that S and $S^{\complement}$ are identically distributed. Therefore, applying Lemma 2.5 again yields

$$\Pr\left[|w(x_{S^{\complement}}) - \mu| \geq \frac{\delta}{2}\right] \leq 2e^{-2(\frac{\delta}{2})^2 n}.$$

Since $|w(x_S) - w(x_{S^{\complement}})| \leq |w(x_S) - \mu| + |w(x_{S^{\complement}}) - \mu|$, it follows that the event $|w(x_S) - w(x_{S^{\complement}})| \geq \delta$ implies the event $|w(x_S) - \mu| \geq \delta/2 \lor |w(x_{S^{\complement}}) - \mu| \geq \delta/2$. Thus, by union bound we get

$$\Pr\left[|w(x_S) - w(x_{S^{\complement}})| \geq \delta\right] \leq \Pr\left[|w(x_S) - \mu| \geq \frac{\delta}{2}\right] + \Pr\left[|w(x_{S^{\complement}}) - \mu| \geq \frac{\delta}{2}\right] \leq 4e^{-2(\frac{\delta}{2})^2 n}.$$

$\square$

Bouman and Fehr [BF10] generalized the above to the quantum setting.

Lemma 2.7 (Application of Theorem 3 of [BF10]). *Let $\mathcal{H}_A$ be an $2n$-qubit system, let $\mathcal{H}_R$ be the reference system, let $|\phi\rangle_{AR} \in \mathcal{H}_A \otimes \mathcal{H}_R$ be an arbitrary state, and let $|v_0\rangle, |v_1\rangle \in \mathcal{H}_A$ be orthonormal quantum states. Suppose we randomly sample $S \subseteq [2n]$ of size n uniformly at random, measure the n qubits in register A indicated by S in the $\{|v_0\rangle, |v_1\rangle\}$-basis, and get the outcome $y \in \{0,1\}^n$. Denote by $|\tilde{\phi}\rangle_{AR}$ the post-measurement state. For all $0 < \delta < 1$, we define*

$$B^{\delta} = \left\{x \in \{0,1\}^{2n} : x_S = y \land |w(x_{S^{\complement}}) - w(y)| \leq \delta\right\}.$$

Then for all $0 < \delta < 1$, there exists $|\psi\rangle \in \mathrm{span}\left(\{|x\rangle : x \in B^{\delta}\}\right) \otimes \mathcal{H}_R$ such that

$$\Delta\left(|\tilde{\phi}\rangle_{AR}, |\psi\rangle\right) \leq 2e^{-(\frac{\delta}{2})^2 n}.$$

2.4 Magic State Distillation

The T magic state $|T\rangle$ is defined as $|T\rangle := T|+\rangle$. Bravyi and Kitaev [BK05] proposed a algorithm that generates a single qubit $|T\rangle$, given sufficiently many noisy copies of T magic states. Specifically, to get a δ-close $|T\rangle$ state, the algorithm requires $\mathrm{poly}(\log(1/\delta))$ copies of T magic states with a constant fraction error.

The algorithm of [BK05] consists of classically controlled Clifford gates and computational basis measurement, and requires that all qubits of the input state to be identical.

[DNS12] modified the distillation circuit such that this assumption can be weakened. Specifically, they require the states to be close to a subspace spanned by a basis, as stated in Theorem D.1 of [DNS12]. As a corollary, we get the following.

Theorem 2.8 (Application of Theorem D.1 of [DNS12]). *Let $n \in \mathbb{N}$, let $\ell = \lfloor 0.041n \rfloor$, and let*

$$V = \mathrm{span}\left(\left\{\Pi(|T\rangle^{\otimes(n-w)} \otimes |T^{\perp}\rangle^{\otimes w}) : 0 \leq w \leq \ell, \Pi \in \mathrm{Sym}_n\right\}\right)$$

to be a n-qubit Hilbert space, where $|T^{\perp}\rangle := T|-\rangle$. There exists a quantum circuit C that consists of classically controlled Clifford gates and computational basis measurement, such that for any n-qubit

state $|\phi\rangle \in V$, if η is the quantum state of the first qubit of $C(|\phi\rangle)$, then it holds that

$$\Delta\left(\eta, |T\rangle\langle T|\right) \leq O(n \cdot 0.259^{n^{0.4}}).$$

2.5 Quantum Error-Correcting Codes

A *quantum error correction code* (QECC) allows to encode a quantum state using sufficiently many entangled quantum state, in such a way that any bounded number of tampering or deletions can be corrected. Formally, an $[[m, 1, d]]$-QECC is a pair of QPT algorithms QECC.Enc and QECC.Dec, where QECC.Enc : $\mathcal{D}^1 \to \mathcal{D}^m$ is called the encoding algorithm, and where QECC.Dec : $\mathcal{D}^m \to \mathcal{D}^1$ is called the decoding algorithm, such that the following holds. For all quantum channel Λ acting on $\lfloor \frac{d-1}{2} \rfloor$ qubits and for all $\rho \in \mathcal{D}^1$, it holds that

$$(\mathsf{QECC.Dec} \circ \Lambda \circ \mathsf{QECC.Enc})(\rho) = \rho.$$

That is, the codeword QECC.Enc(ρ) can tolerate arbitrary errors on at most $\lfloor \frac{d-1}{2} \rfloor$ qubits.

We remark that evaluating homomorphically Pauli gates, P, H, and CNOT over the codewords of the self-dual CSS code, can be done by applying those gates transversally. Additionally, the homormorphic measurement of a codeword of a CSS code can be done by measuring the codeword transversally, followed by running ECC.Dec(m), where ECC.Dec is the classical decoding algorithm of the linear code associated with the CSS code, and where m is the outcome of the measurement on the codeword. Finally, the encoding and the decoding circuit of a CSS code only consists of Clifford gates.

In this paper, we fix QECC to be a self-dual $[[m, 1, d]]$ CSS code, where m is a polynomial in d. This can be achieved by, for example, using $[[7, 1, 3]]$ Steane code in a concatenated structure.

2.6 Quantum Authentication Schemes

A quantum authentication scheme (QAS) is a way to ensure that quantum state was not tampered with. We restate the definition of QAS in [ABOE10] as follows. To remove confusion with the encoding and decoding of QECC, we view QAS as an encryption scheme.

Definition 2.9. A *quantum authentication scheme* is a pair of polynomial time algorithms Enc and Dec together with a set of classical keys $\mathcal{K}$ such that for any key $k \in \mathcal{K}$ the following holds.

- $\mathsf{Enc}_k : D(\mathcal{H}_M) \to D(\mathcal{H}_C)$ maps from the message register M to the ciphertext register C.

- $\mathsf{Dec}_k : D(\mathcal{H}_C) \to D(\mathcal{H}_{MF})$ takes a (possibly altered) ciphertext from register C, and outputs a message from register M, and a single qubit from the register F to specify whether the message was tampered with. The basis states of F are called $|\mathsf{Acc}\rangle$ and $|\mathsf{Rej}\rangle$.

Additionally, we require the following correctness property. For all keys $k \in \mathcal{K}$ and all messages $\rho \in \mathcal{D}^*$, it holds that

$$\mathsf{Dec}_k(\mathsf{Enc}_k((\rho)) = \rho \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|.$$

We now present the security definition for a QAS.

Definition 2.10 (Security of QAS)**.** Let M be the message resister, C be the QAS ciphertext register, R be a reference system, and let $\varepsilon > 0$. A QAS (Enc, Dec) is said to be *ε-secure*, if for all

U^{CR} there exists two CP maps $\mathscr{U}^R_{\mathsf{Acc}}$ and $\mathscr{U}^R_{\mathsf{Rej}}$ satisfying $\mathscr{U}^R_{\mathsf{Acc}} + \mathscr{U}^R_{\mathsf{Rej}} = I^R$, such that for any input ρ^{MR} it holds that

$$\Delta(\frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \mathsf{Dec}_k(U^{CR}\mathsf{Enc}_k(\rho^{MR})U^\dagger), \ \mathscr{U}^R_{\mathsf{Acc}}(\rho^{MR}) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|$$
$$+ |\bot\rangle\langle\bot|^M \otimes \mathrm{tr}_M\left(\mathscr{U}^R_{\mathsf{Rej}}(\rho^{MR})\right) \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|) \le \varepsilon, \quad (4)$$

where $|\bot\rangle\langle\bot|$ is a predetermined fixed state.

2.6.1 Clifford Authentication Code

Clifford codes are defined as follows.

Definition 2.11 (Clifford code [ABOE10])**.** Let M be the message register, and let $C = MT$ be the ciphertext register, where T is a t-qubit register called the *trap register*. The contents of T are called traps. The set of keys are the Clifford group $\mathcal{C}_{|C|}$. Define the projectors $P_{\mathsf{Acc}} := |0^t\rangle\langle 0^t|$ and $P_{\mathsf{Rej}} := I^{\otimes t} - P_{\mathsf{Acc}}$. The Clifford authentication code $\mathsf{CAuth} = (\mathsf{CAuth.Enc}, \mathsf{CAuth.Dec})$ is defined as follows.

- **Encryption:** Append t-qubits from register T in the state $|0^t\rangle\langle 0^t|$. Then apply E on register MT. That is
$$\mathsf{CAuth.Enc}_E(\rho^M) := E(\rho^M \otimes |0^t\rangle\langle 0^t|^T)E^\dagger.$$

- **Decryption:** Apply $E^\dagger$ on register MT, then measure register T in the computational basis. If the outcome is all-zero string 0^t, set $|\mathsf{Acc}\rangle\langle\mathsf{Acc}|$ in F. Otherwise, set $|\mathsf{Rej}\rangle\langle\mathsf{Rej}|$ in F and replace the state with Ω in register M. That is,
$$\mathsf{CAuth.Dec}_E(\sigma^{MT}) := \mathrm{tr}_T\left(P^T_{\mathsf{Acc}}E^\dagger\sigma^{MT}E\right) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|$$
$$+ \mathrm{tr}_{MT}\left(P^T_{\mathsf{Rej}}E^\dagger\sigma^{MT}E\right)|\bot\rangle\langle\bot|^M \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|.$$

Theorem 2.12 (Security of Clifford code, Theorem 3.1 of [ABOE10])**.** *A Clifford code* $(\mathsf{CAuth.Enc}, \mathsf{CAuth.Dec})$ *with key set* $\mathcal{C}_{|MT|}$ *is a quantum authentication scheme* $(2^{-|T|})$*-secure.*

Clifford code supports homomorphic evaluation of Clifford gate by a simple key update. Given a Clifford encryption $\mathsf{CAuth.Enc}_E(\rho^M)$, observe that by decrypting with the key $E(G^\dagger)^M$ results in

$$\mathsf{CAuth.Dec}_{EG^\dagger}(\mathsf{CAuth.Enc}_E(\rho)) = G\rho G^\dagger \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|,$$

thus applying G on ρ.

2.6.2 Trap Authentication Code

We also use *trap authentication code* [BGS13]. A trap code is a QAS built using a QECC.

Definition 2.13 (Trap code [BGS13])**.** For a trap code using $[[t, 1, d]]$-QECC, the key set of trap code is $\mathrm{Sym}_{3t} \times \{0,1\}^{3t} \times \{0,1\}^{3t}$. Let M be a single-qubit register, let $\tilde{M}$ be the register that stores the QECC codewords, and let $C = \tilde{M}T_XT_Z$ be the ciphertext register, where T_X and T_Z are two

t-qubit registers called *trap registers*. Similarly to Clifford codes, we call the contents of T_X and T_Z traps. Define the projectors $P_{\mathsf{Acc}} := (|0\rangle\langle 0|)^{\otimes t} \otimes (|+\rangle\langle +|)^{\otimes t}$ and $P_{\mathsf{Rej}} := I^{\otimes 2t} - P_{\mathsf{Acc}}$. For a single-qubit input message $\rho \in D(\mathcal{H}_M)$, the trap authentication code $\mathsf{TAuth} = (\mathsf{TAuth.Enc}, \mathsf{TAuth.Dec})$ is defined as follows.

- **Encryption:** Apply $\mathsf{QECC.Enc}$ on register M, and append t-qubits from register T_X in the state $(|0\rangle\langle 0|)^{\otimes t}$, and append n-qubits from register T_Z in the states $(|+\rangle\langle +|)^{\otimes t}$. Then permute the qubits of $\tilde{M}T_X T_Z$ according to Π. Finally, apply $X^x Z^z$ on register $\tilde{M}T_X T_Z$. That is,

$$\mathsf{TAuth.Enc}_{\Pi,x,z}(\rho) := X^x Z^z \Pi \left(\mathsf{QECC.Enc}(\rho) \otimes (|0\rangle\langle 0|)^{\otimes n} \otimes (|+\rangle\langle +|)^{\otimes n} \right) (X^x Z^z \Pi)^\dagger.$$

- **Decryption:** Apply $(X^x Z^z \Pi)^\dagger$ onto register $\tilde{M}T_X T_Z$, and measure the register T_X in computational basis, and measure the register T_Z in Hadamard $\{|+\rangle, |-\rangle\}$-basis. If the outcome of T_X is all zeros and the outcome of T_Z is all $+$, then apply $\mathsf{QECC.Dec}$ on register $\tilde{M}$ and set $|\mathsf{Acc}\rangle\langle\mathsf{Acc}|$ in F. Otherwise, replace the state in M with $|\bot\rangle\langle\bot|$ and set $|\mathsf{Rej}\rangle\langle\mathsf{Rej}|$ in F. That is,

$$\mathsf{TAuth.Dec}_{\Pi,x,z}(\sigma) := \mathsf{QECC.Dec} \left(\mathrm{tr}_{T_X T_Z} \left(I^{\otimes t} \otimes P_{\mathsf{Acc}} (X^x Z^z \Pi)^\dagger \sigma X^x Z^z \Pi \right) \right) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|$$
$$+ |\bot\rangle\langle\bot| \, \mathrm{tr}_{\tilde{M}T_X T_Z} \left(I^{\otimes t} \otimes P_{\mathsf{Rej}} (X^x Z^z \Pi)^\dagger \sigma X^x Z^z \Pi \right) \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|.$$

Since $X^x Z^z$ is a Pauli operator up to a phase ± 1 or $\pm i$, we sometimes write $\mathsf{TAuth.Enc}_{\Pi,P}$ and $\mathsf{TAuth.Dec}_{\Pi,P}$, where P is a Pauli operator.

Theorem 2.14 (Security of trap code [BGS13])**.** *A trap code that uses a $[[t, 1, d]]$-QECC is a $(2/3)^{d/2}$-secure quantum authentication scheme.*

We define the trap code *partial decryption* operation $\mathsf{TAuth.PDec}$, as the unitary part of $\mathsf{TAuth.Dec}$. That is, it decodes the permutation and quantum one time pad, perform the (unitary part of) QECC decoding on the first t qubits, and then apply Hadamards on the last t qubits to map $|+\rangle$ to $|0\rangle$.

Definition 2.15. Let $x, z \in \{0,1\}^m$, $\Pi \in \mathrm{Sym}_m$. Then,

$$\mathsf{TAuth.PDec}_{\Pi,x,z} := (U_{\mathsf{Dec}} \otimes I^{\otimes 2t})(I^{\otimes 2t} \otimes H^{\otimes t})(X^x Z^z \Pi)^\dagger,$$

where U_{Dec} is the unitary operator corresponding to the $\mathsf{QECC.Dec}$ circuit.

Notice that when applied to a $\mathsf{TAuth.Enc}$ encoding of ρ using the same keys, the result is $\rho \otimes |0^{2t}\rangle\langle 0^{2t}|$. Similarly, we define the trap code *partial encryption* operation $\mathsf{TAuth.PEnc}$ as the unitary part of $\mathsf{TAuth.Enc}$.

Definition 2.16. Let $x, z \in \{0,1\}^m$ and $\Pi \in \mathrm{Sym}_m$. Then,

$$\mathsf{TAuth.PEnc}_{\Pi,x,z} := X^x Z^z \Pi(U_{\mathsf{Enc}} \otimes I^{\otimes t} \otimes H^{\otimes t}),$$

where U_{Enc} is the unitary operator corresponding to the $\mathsf{QECC.Enc}$ circuit.

2.6.3 Homomorphic Evaluation over Trap Codes

In this section we explain how to homomorphically evaluate any quantum circuit under trap codes.

Evaluating Pauli and CNOT gates. First observe that, a the $[[t, 1, d]]$-QECC we use evaluates X, Z and CNOT by transversally applying these gates, it follows that they can be evaluated homomorphically under trap code. Indeed, let (Π, x, z) be the key used, where $\Pi \in \text{Sym}_m$, and $x, z \in \{0, 1\}^m$, and let $\oplus$ denote addition modular 2. Then applying an X gate can be done by updating x to $x \oplus \Pi(1^t 0^{2t})$. That is, apply permute $(1^t 0^{2t})$ according to Π and XOR with x. Similarly, applying a Z gate can be done by updating z to $z \oplus \Pi(1^t 0^{2t})$.

Next, if two trap code ciphertext $\sigma_1 = \text{TAuth.Enc}_{\Pi, x_1, z_1}(\rho_1)$ and $\sigma_2 = \text{TAuth.Enc}_{\Pi, x_2, z_2}(\rho_2)$ share the same permutation $\Pi \in \text{Sym}_m$, then applying CNOT on ρ_1 and ρ_2 can be done as follows. First, execute CNOT transversally on the two trap code ciphertext. Then, update z_1 to $z_1 \oplus z_2$ and update x_2 to $x_1 \oplus x_2$. Correctness follows from the identity

$$\text{CNOT}(X^{x_1} Z^{z_1} \otimes X^{x_2} Z^{z_2}) = (X^{x_1} Z^{z_1 \oplus z_2} \otimes X^{x_1 \oplus x_2} Z^{z_2})\text{CNOT},$$

for all $x_1, x_2, z_1, z_2 \in \{0, 1\}$. Furthermore, the traps will not be affected by this operation since $\text{CNOT} |0\rangle |0\rangle = |0\rangle |0\rangle$ and $\text{CNOT} |+\rangle |+\rangle = |+\rangle |+\rangle$.

Performing measurements. To perform measurements, we use the fact that the $[[t, 1, d]]$-QECC we use can homomorphically measure the encoded qubit by first measuring it and then decoding *classically*. Thus, the measurement outcome of the trap code plaintext can be obtained by measuring before executing a classical decryption procedure as well. Moreover, authenticating the measurement can still be done by checking register T_X. We next define the classical decryption procedure TAuth.VerM that is used after measurement.

Definition 2.17. Let $s \in \{0, 1\}^{3t}$, let $\Pi \in \text{Sym}_{3t}$, and let $x \in \{0, 1\}^{3t}$. The classical procedure $\text{TAuth.VerM}_{\Pi, x}(s)$ is defined as follows.

1. Compute $s' = \Pi(x \oplus s)$.

2. If there exists $i \in \{t + 1, \ldots, 2t\}$ such that $s'_i \neq 0$ then return $(\perp, \text{Rej})$.

3. Otherwise, return $(\text{ECC.Dec}(s'_1, \ldots, s'_t), \text{Acc})$.

We extend TAuth.VerM to operate over density matrices in the natural way. That is, for a diagonal density matrix $\sigma = \sum_{s \in \{0,1\}^{3t}} p_s |s\rangle\langle s|$, let $\tilde{M}$, T_X and T_Z be the first, second, and third t-qubit registers containing σ. We extend the classical procedure $\text{TAuth.VerM}_{\Pi, x}$ to be applied on a quantum input σ as follows.

$$\text{TAuth.VerM}_{\Pi, x}(\sigma) = \text{ECC.Dec}(\text{tr}_{T_X T_Z}(P^{T_X}_{\text{Acc}}(X^x \Pi)^\dagger \sigma(X^x \Pi))) \otimes |\text{Acc}\rangle\langle\text{Acc}|$$
$$+ |\perp\rangle\langle\perp| (\text{tr}_{\tilde{M} T_X T_Z}(P^{T_X}_{\text{Rej}}(X^x \Pi)^\dagger \sigma(X^x \Pi))) \otimes |\text{Rej}\rangle\langle\text{Rej}|,$$

where $P_{\text{Acc}} = (|0\rangle\langle 0|)^t$, $P_{\text{Rej}} = I^{\otimes t} - P_{\text{Acc}}$, and where

$$\text{ECC.Dec}\left(\sum_{m \in \{0,1\}^t} p_m |m\rangle\langle m|\right) := \sum_{m \in \{0,1\}^t} p_m |\text{ECC.Dec}(m)\rangle\langle\text{ECC.Dec}(m)|.$$

It is showed that the measuring then decoding procedure is ε-secure if the trap code is ε-secure in Section B of [BGS13]. We have the following lemma.

Lemma 2.18. *Let M, C, and R be the message, the ciphertext, and the reference register of a trap code respectively. For $A \in \{M, C\}$ and $\rho \in D(\mathcal{H}_A)$ let $\Lambda^A(\rho) = \sum_{m \in \{0,1\}^{|A|}} |m\rangle\langle m| \rho |m\rangle\langle m|$ be the*

measurement in the computational basis on register A. Then for all unitary attacks U^{CR}, there exists two CP maps $\mathscr{U}^R_{\mathsf{Acc}}$ and $\mathscr{U}^R_{\mathsf{Rej}}$ with $\mathscr{U}^R_{\mathsf{Acc}} + \mathscr{U}^R_{\mathsf{Rej}} = I^R$ such that for all states ρ^{MR} it holds that

$$\mathbb{E}_{\Pi,x,z} \left[\mathsf{TAuth.VerM}^C_{\Pi,x} \circ \Lambda^C(U^{CR}(\mathsf{TAuth.Enc}^M_{\Pi,x,z}(\rho^{MR}))U^\dagger) \right]$$
$$\approx_{\mathrm{neg}(\kappa)} \left(\Lambda^M \circ \mathscr{U}^R_{\mathsf{Acc}}(\rho^{MR}) \right) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| + |\bot\rangle\langle\bot|^M \, \mathrm{tr}_M(\mathscr{U}^R_{\mathsf{Rej}}(\rho^{MR})) \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|. \quad (5)$$

For any attack U and ρ^{MR}, we can get the accept and reject probability by tracing out MR register on the right hand side of Equation (5). This results in $q\,|\mathsf{Acc}\rangle\langle\mathsf{Acc}| + (1-q)\,|\mathsf{Rej}\rangle\langle\mathsf{Rej}|$, where q is the accept probability.

Similarly, we can perform Hadamard measurement homomorphically under trap code. This is done by executing above steps, using the Hadamard basis, rather than the computational basis, and consider the trpas in T_Z rather than T_X.

Evaluating P and H gates. Though the QECC in the trap code evaluates P and H gate transversally, we cannot directly homomorphically evaluate P and H by applying them transversally and updating keys. This is due to the fact that the registers T_X and T_Z will be affected.

Instead, we use the so called magic states to computed these gates [BGS13]. The P magic state is defined by $|P\rangle := P\,|+\rangle$ and the H magic state is defined by $|H\rangle := (H \otimes I)\,|\Phi^+\rangle$. The phase gate P and Hadamard gate H can then be implemented by a quantum circuit using these magic states and the procedures described previously to compute Paulis, CNOT, and measurements. The quantum circuit for implementing P and H gate using magic states is showed in Figure 1.
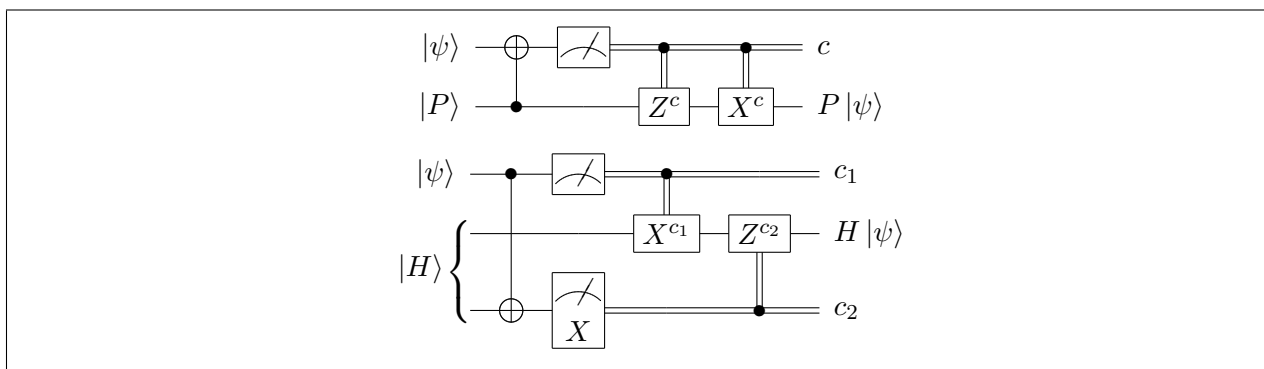
Figure 1: Applying Hadamard and phase gates using magic states

Evaluating T gates. With P and H states, we can homomorphically evaluate Clifford gates under trap code. To achieve universal quantum computation, we need to implement T gates. [BMP+99] Showed that a T gate can be implemented by a quantum circuit that consists only of classical-controlled Clifford gates and computational measurements, using the T magic state $|T\rangle := T\,|+\rangle$ as resource [BMP+99]. The circuit is showed in Figure 2.
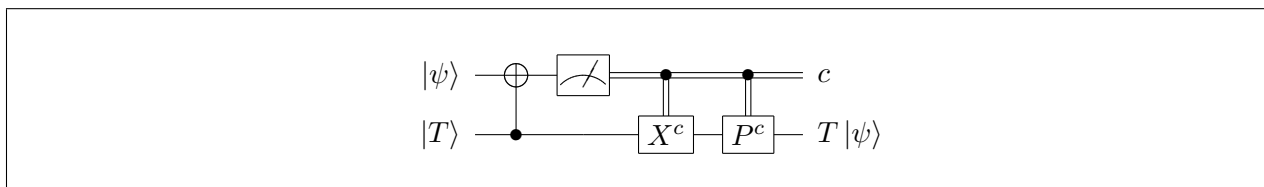
Figure 2: Applying T gates using magic states

2.7 Verifiable Quantum Fully Homormorphic Encryption

In this section, we restate the definition and the construction of verifiable quantum fully homormorphic encryption (VQFHE) introduced in [ADSS17]. The original definition characterizes families of circuits that can be homomorphically evaluated. We rephrase this model in the language of leveled homomorphic encryption schemes; that is, said family is determined solely by circuit size. This is also consistent with the construction in [ADSS17], and is enough for our purposes.

Definition 2.19 (VQFHE scheme)**.** A leveled VQFHE scheme is a 4-tuple of QPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ such that the following hold.

1. **Key Generation.** The algorithm $(\mathsf{sk}, \rho_{\mathsf{evk}}) \leftarrow \mathsf{KeyGen}(1^\kappa, 1^L)$ takes as input the security parameter and the level parameter, and outputs a classical symmetric key sk and a quantum evaluation key ρ_{evk}.

2. **Encryption.** The algorithm $\hat{\rho} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(\rho)$ takes a classical secret key sk and an input state ρ, and outputs a quantum ciphertext $\hat{\rho}$.

3. **Homomorphic Evaluation.** The algorithm $(\hat{\sigma}, \mathsf{log}) \leftarrow \mathsf{Eval}(C, \rho_{\mathsf{evk}}, \hat{\rho})$ is given a quantum circuit C of size at most L, an evaluation key ρ_{evk}, and a ciphertext $\hat{\rho}$. It outputs a evaluted quantum ciphertext $\hat{\sigma}$ and a classical string log.

4. **Verified Decryption.** The algorithm $(\sigma, \mathtt{flag}) \leftarrow \mathsf{VerDec}_{\mathsf{sk}}(C, \hat{\sigma}, \mathsf{log})$ is given a secret key sk, a ciphertext $\hat{\sigma}$, a circuit description C, and a classical string log. It outputs a quantum message σ and $\mathtt{flag} \in \{\mathsf{Acc}, \mathsf{Rej}\}$.

Furthermore, the scheme is required to be correct. That is, for $(\mathsf{sk}, \rho_{\mathsf{evk}}) \leftarrow \mathsf{KeyGen}(1^\kappa, 1^L)$, for all quantum circuit C of size at most L, and for all $\rho \in D(\mathcal{H}_M \otimes \mathcal{H}_R)$ it holds that

$$\Delta\left(\mathsf{VerDec}_{sk}(C, \mathsf{Eval}(C, \rho_{\mathsf{evk}}, \mathsf{Enc}_{\mathsf{sk}}(\rho))), \; C(\rho) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|\right) \leq \mathrm{neg}(\kappa).$$

Remark 1. *In [ADSS17], a complete definition for a VQFHE scheme should also satisfies the* compactness, *which restricts the complexity of* VerDec*. For brevity, we omit this requirement. However, since our protocol implements the construction given in [ADSS17], our protocol also achieves compactness.*

The security of a VQFHE is defined by the following indistinguishability game. Given an VQFHE scheme S, an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a security parameter κ, and a level parameter L, the game $\mathsf{VerGame}_{S,\mathcal{A}}(\kappa, L)$ is defined using the following game between the adversary and a challenger $\mathfrak{C}$ (see Figure 3).

Game 1 $\mathsf{VerGame}_{S,\mathcal{A}}(\kappa, L)$

---

1. The challenger $\mathfrak{C}$ computes $(\mathsf{sk}, \rho_{\mathsf{evk}}) \leftarrow S.\mathsf{KeyGen}(1^\kappa, 1^L)$ and sends ρ_{evk} to the adversary.

2. The adversary chooses an input ρ^{MR} along with its reference system R, and sends only the content of register M to the challenger.

3. The challenger samples a random bit $r \leftarrow \{0, 1\}$. If $r = 1$ the challenger encrypts $|0^n\rangle\langle 0^n|$ using sk. Otherwise, if $r = 0$ it encrypts ρ. Let $\hat{\rho}$ denote the resulting ciphertext. The challenger then sends $\hat{\rho}$ to the adversary.
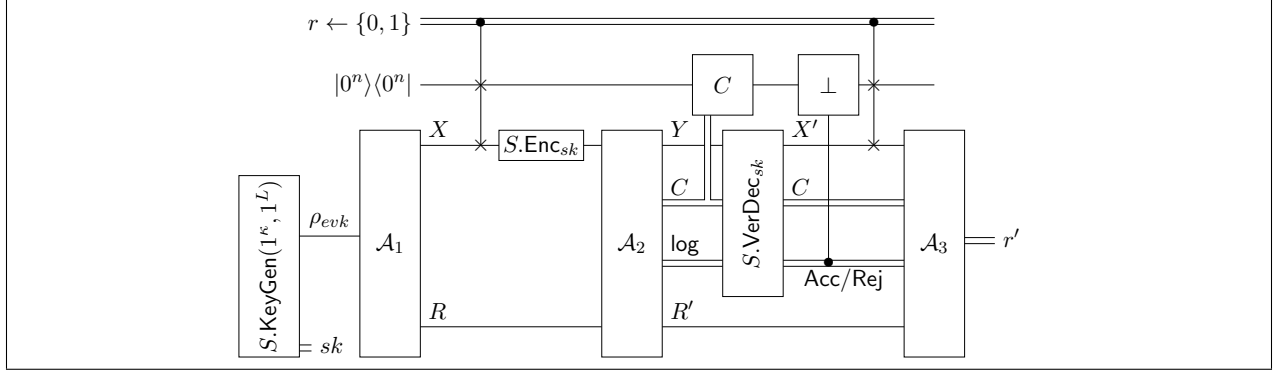
Figure 3: The indistinguishability game $\mathsf{VerGame}_{S,\mathcal{A}}(\kappa, L)$, restated from [ADSS17].

4. The adversary is now supposed to compute $S.\mathsf{Eval}(C, \hat{\rho}, \rho_{\mathsf{evk}})$, for some circuit C of size at most L. Let $(\hat{\sigma}, \mathsf{log})$ be the purported output of $S.\mathsf{Eval}$. The adversary then sends $(C, \hat{\sigma}, \mathsf{log})$ to the challenger.

5. The challenger decrypts $(\sigma, \mathtt{flag}) \leftarrow \mathsf{TrapTP.VerDec}_{\mathsf{sk}}(\hat{\sigma}, C, \mathsf{log})$, and sends $\mathtt{flag}$ and

$$\sigma' := \begin{cases} |\bot\rangle\langle\bot| & \text{if } \mathtt{flag} = \mathsf{Rej} \\ C(\rho^M) & \text{if } \mathtt{flag} = \mathsf{Acc} \text{ and } r = 1 \\ \sigma & \text{if } \mathtt{flag} = \mathsf{Acc} \text{ and } r = 0 \end{cases}$$

to the adversary.

6. The adversary outputs a bit $r' \in \{0, 1\}$.

Intuitively, a VQFHE scheme is deemed secure if the adversary cannot guess r with probability that is significantly higher than $1/2$. Formally, security is defined as follows.

Definition 2.20 (IND-VER security [ADSS17])**.** A VQFHE scheme S is *IND-VER secure*, if for all QPT adversaries $\mathcal{A}$, it holds that

$$\Pr[\mathsf{VerGame}_{S,\mathcal{A}}(\kappa, L) = 1] \leq \frac{1}{2} + \mathsf{neg}(\kappa),$$

where we abuse notation and let $\mathsf{VerGame}_{S,\mathcal{A}}(\kappa, L) = 1$ denote the event that $\mathcal{A}$ outputs r in the corresponding game.

2.7.1 A Construction for VQFHE

In this section, we briefly introduce the VQFHE scheme TrapTP proposed by [ADSS17]. The construction uses the fact that trap codes support homormorphic evaluation of a universal circuit with the help of some magic states. The key secret key sk used in TrapTP consists of the classical strings, defined to be a collection of permutations – one is called *global permutation* and the rest are called *local permutations* – the secret key of a message authentication code MAC, and both the public and secret key of a homomorphic encryption HE. Thus, $\mathsf{TrapTP.KeyGen}$ first generate these values. Towards homomorphic evaluation, the key generation algorithm must also prepare magic

states, which will be part of the evaluation key ρ_{evk}. Therefore, TrapTP.KeyGen will further encrypt the magic states using TrapTP.Enc$_{\mathsf{sk}}$. Specifically, the global permutation will be used to encrypt all magic states described in Section 2.6.3, and the local permutation will be used to encrypt special gadget states. We next explain encryption algorithm.

The encryption algorithm TrapTP.Enc will first sample a random Pauli and use it to encrypt the input state. Next, to get variability in the decryption, the random Pauli will be encrypted using a the classical HE, and then sign it using MAC.

To preform evaluation, TrapTP.Eval uses the techniques we described in Section 2.6.3. This is particular updates the Pauli keys that were sampled in the encryption algorithm. To update the Paulis, TrapTP.Eval simply uses the properties of HE. Additionally, to achieve variability during the decoding later, TrapTP.Eval also generates a classical string log of the of the computation, that includes all the classical messages including randomness, computation steps, and all intermediate results during evaluation. The only issue remained, is the phase introduced by computing T gate. To overcome this, the algorithm uses the special gadget states prepared during TrapTP.KeyGen.

We now explain the decryption algorithm TrapTP.VerDec, that also verifies the evaluation. First, it checks log, using a classical algorithm, denoted CheckLogs. Next, it decrypts all HE ciphertexts, which produces the Paulis. Now, given also the secret key, TrapTP.VerDec can now decrypt all ciphertexts by decrypting the trap codes (note that this could also cause TrapTP.VerDec to reject the evaluation). We refer the reader to [ADSS17] for a detailed construction. They proved that TrapTP is secure with respect to VerGame.

Theorem 2.21 (Theorem 5 of [ADSS17]). *Assuming the existence of a classical fully homomorphic encryption scheme, the VQFHE scheme* TrapTP *is IND-VER secure.*

3 The Model of Computation

The security of multiparty computation protocols is defined using the real vs. ideal paradigm. In this paradigm, we consider the real-world model, in which protocols are executed. Here, an n-party quantum protocol π for computing a quantum circuit family $C = \{C_\kappa\}_{\kappa \in \mathbb{N}}$ is defined by a set of n interactive uniform QPT circuits $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_n\}$. To alleviate notation, we simply write C for the circuit. We then formulate an ideal model for executing the task. This ideal model involves a trusted party whose functionality captures the security requirements of the task. Finally, we show that the real-world protocol "emulates" the ideal-world protocol, i.e., for any real-world adversary $\mathcal{A}$ there exists an ideal-world adversary Sim (called the simulator) such that the global output of an execution of the protocol with $\mathcal{A}$ in the real-world is distributed similarly to the global output of running Sim in the ideal model.

In the following section we present the formal definitions. We describe the relevant variants of the ideal-world and real-world for the quantum, stand alone, definition of secure multiparty computation. We focus on the definition of *security-with-identifiable-abort* for static quantum adversaries. Additionally, in Section 3.1 we present a variant of security-with-identifiable-abort that allows the adversary to block some of the inputs and outputs states.

The Real Model

An n-party quantum protocol π for computing a quantum circuit family $C = \{C_\kappa\}_{\kappa \in \mathbb{N}}$ is defined by a set of n interactive uniform QPT circuits $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_n\}$. To alleviate notation, we simply write C for the circuit. Each circuit (party) P_i holds the security parameter 1^κ as a joint input and a private input quantum state ρ_i. Throughout the paper, we assume that all of the inputs

to C are of the same size, and all of its outputs are of the same size as well, respectively. This is without loss of generality as we can simply pad the shorter inputs and outputs with 0's. We denote by $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\cdot)$ and $\ell_{\mathsf{out}} = \ell_{\mathsf{out}}(\cdot)$ the number of input-qubits and output-qubits, respectively, each party has.

The parties execute the protocol over a synchronous network. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their messages for this round) followed by a *receive phase* (where they receive messages from other parties). We consider a fully connected point-to-point network, where every pair of parties is connected by both a quantum and a classical communication line.

In the setting of secure multiparty computation, we consider a single adversary $\mathcal{A}$ that controls a subset $\mathcal{I} \subset \mathcal{P}$. The adversary has access to the full view of all corrupted parties. A malicious adversary may instruct the corrupted parties to deviate from the protocol in any way it chooses. Furthermore, the adversary is given an auxiliary input ρ_{aux}. The adversary is static; that is, it chooses the subset it corrupts prior to the execution of the protocol.

We denote by $\textsc{real}_{\pi,\mathcal{A}(\rho_{\mathsf{aux}})}(\kappa, \rho_1, \ldots, \rho_n)$ the joint output of the adversary $\mathcal{A}$ and of the honest parties in an execution of π on security parameter $\kappa \in \mathbb{N}$, inputs $\rho_1, \ldots, \rho_n \in \mathcal{D}^*$, and auxiliary input $\rho_{\mathsf{aux}} \in \mathcal{D}^*$.

The Ideal Model

We now describe the interaction in the ideal model, which specifies the requirements for a secure-with-identifiable-abort computation of a circuit C with security parameter κ. Let $\mathcal{A}$ be an adversary in the ideal-world, which is given an auxiliary quantum state ρ_{aux} and corrupts a subset $\mathcal{I}$ of the parties.

Security with identifiable abort

Inputs: Each party P_i holds the security parameter 1^κ and an input ρ_i. The adversary $\mathcal{A}$ is given an auxiliary quantum state ρ_{aux}, and ρ_i for every P_i controlled by it. The trusted party T holds 1^κ.

Parties send inputs: Each honest party $\mathrm{P}_j \in \mathcal{P} \setminus \mathcal{I}$ sends ρ_j as its input to T. The malicious adversary sends a value ρ_i^* as the input for party $\mathrm{P}_i \in \mathcal{I}$. Write $\rho' = (\rho_1', \ldots, \rho_n')$ for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T prepares ancilla zero states ρ_0 and computes $C(\rho', \rho_0)$. Let $(\sigma_1, \ldots, \sigma_n, \sigma_{\mathsf{discard}})$ be the resulting output-states, where σ_i is the output associated with party P_i. The trusted party sends $\sigma_{\mathcal{I}}$ to $\mathcal{A}$.

Adversary instructs trusted party to continue or halt: The adversary sends to T either `continue` or $(\texttt{abort}, \mathrm{P}_i)$ for some $\mathrm{P}_i \in \mathcal{I}$. If it sent `continue`, then for every honest party P_j the trusted party sends it σ_j. Otherwise, if $\mathcal{A}$ sent $(\texttt{abort}, \mathrm{P}_i)$, then T sends $(\texttt{abort}, \mathrm{P}_i)$ to the each honest party P_j.

Outputs: Each honest party outputs whatever output it received from T, the parties in $\mathcal{I}$ output nothing, and the adversary outputs some function of its view.

We denote by $\textsc{ideal}_{C,\mathcal{A}(\rho_{\mathsf{aux}})}(\kappa, \rho_1, \ldots, \rho_n)$ the joint output of the adversary $\mathcal{A}$ and the honest parties in an execution of the above ideal-world computation of C, on security parameter κ, inputs $\rho_1, \ldots, \rho_n$, and auxiliary input ρ_{aux}.

Having defined the real and ideal models, we can now define security of protocols according to the real/ideal paradigm.

Definition 3.1 (security with identifiable abort)**.** Let π be a protocol for computing a circuit C. We say that π computes C with computational *security-with-identifiable-abort*, if the following holds. For every non-uniform QPT adversary $\mathcal{A}$, controlling a set $\mathcal{I} \subset \mathcal{P}$ in the real-world, there exists a non-uniform QPT adversary $\mathsf{Sim}_{\mathcal{A}}$, controlling $\mathcal{I}$ in the ideal-world, such that

$$\left\{ \text{IDEAL}_{C,\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} \left(\kappa, (\rho_i)_{i=1}^n \right) \right\}_{\kappa \in \mathbb{N}, \rho_1, \dots, \rho_n, \rho_{\mathsf{aux}} \in \mathcal{D}^*} \stackrel{\mathrm{C}}{\equiv} \left\{ \text{REAL}_{\pi,\mathcal{A}(\rho_{\mathsf{aux}})} \left(\kappa, (\rho_i)_{i=1}^n \right) \right\}_{\kappa \in \mathbb{N}, \rho_1, \dots, \rho_n, \rho_{\mathsf{aux}} \in \mathcal{D}^*}.$$
(6)

Statistical and perfect security are defined similarly, by replacing $\stackrel{\mathrm{C}}{\equiv}$ in Equation (6) with $\approx_{\mathrm{neg}(\kappa)}$ and $=$, respectively, and assuming unbounded adversaries and simulators.

3.1 Security With Packet Drops

We now introduce a relaxed security notion of security-with-identifiable-abort that allows the adversary to drop some of the input-qubits and some of the output-qubits. We call this security notion *IDPD-security*. This security notion is parameterized with two polynomials $d_{\mathsf{in}} = d_{\mathsf{in}}(\kappa)$ and $d_{\mathsf{out}} = d_{\mathsf{out}}(\kappa)$ representing an upper bound on the number of input-qubits and output-qubits, respectively, the adversary is allowed to drop from the computation. The definition follows the standard ideal vs. real paradigm.

Informally, in the ideal world, in addition to sending inputs, the adversary also instructs the trusted party which single qubits are to be replaced with 0. Then, upon receiving the output, the adversary can decide to either abort the protocol while revealing the identity of a corrupted party, or to instruct the trusted party to discard some of the qubits in the output and distribute it.

We now formally describe the $(d_{\mathsf{in}}, d_{\mathsf{out}})$-IDPD ideal model, which specifies the requirements for an IDPD-secure computation of a circuit C with security parameter κ. Unlike the informal discussion from Section 1.2.2, it will be more convenient to have the adversary send to the trusted party the set of *remaining* qubits. Let $\mathcal{A}$ be an adversary in the ideal-world, which is given an auxiliary quantum state ρ_{aux} and corrupts a subset $\mathcal{I}$ of the parties.

Security with identifiable abort and packet drops

Inputs: Each honest party P_i holds the security parameter 1^κ where $\kappa \in \mathbb{N}$ and an input $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$ where each $\rho_{ij} \in \mathcal{D}^1$ is single-qubit. The adversary is given 1^κ, input ρ_i of every corrupted party $\mathrm{P}_i \in \mathcal{I}$, and an auxiliary input ρ_{aux}. Finally, the trusted party T is given the security parameter 1^κ.

Parties send inputs: Each honest P_i sends ρ_i to T. For every corrupted party P_i, the adversary sends a state ρ_i^* to T as the input of P_i.

The adversary instructs T to drop some input-qubits: The adversary sends to T a set $\mathcal{R}_{\mathsf{in}} \subseteq \left\{ (i,j) \in \mathbb{N}^2 \mid i \in [n], j \in [\ell_{\mathsf{in}}] \right\}$ of size $|\mathcal{R}_{\mathsf{in}}| \geq n\ell_{\mathsf{in}} - d_{\mathsf{in}}$ (note that it could be the

---

case that $n\ell_{\text{in}} < d_{\text{in}}$, in which case no restriction are imposed on $\mathcal{R}_{\text{in}}$). Denote

$$\rho'_{ij} = \begin{cases} |0\rangle\langle 0| & \text{if } (i,j) \notin \mathcal{R}_{\text{in}} \\ \rho_{ij} & \text{if } (i,j) \in \mathcal{R}_{\text{in}} \text{ and } i \notin \mathcal{I} \\ \rho^*_{ij} & \text{if } (i,j) \in \mathcal{R}_{\text{in}} \text{ and } i \in \mathcal{I} \end{cases}$$

and let $\rho' = (\hat{\rho}_{ij})_{i \in [n], j \in [\ell_{\text{in}}]}$.

The trusted party performs the computation: The trusted party T prepares ancilla zero states ρ_0 and computes $C(\rho', \rho_0)$. Let $(\sigma_1, \ldots, \sigma_n, \sigma_{\text{discard}})$ be the resulting output-states, where σ_i is the output associated with party P_i. The trusted party sends $\sigma_{\mathcal{I}}$ to $\mathcal{A}$.

Adversary instructs T to drop some output-qubits or halt: For every $i \in [n]$ write $\sigma_i = (\sigma_{ij})_{j=1}^{\ell_{\text{out}}}$, where each $\sigma_{ij} \in \mathcal{D}^1$ is single-qubit. The adversary $\mathcal{A}$ sends to T either $(\texttt{continue}, \mathcal{R}_{\text{out}})$ where $\mathcal{R}_{\text{out}} \subseteq \left\{(i,j) \in \mathbb{N}^2 \mid i \in [n], j \in [\ell_{\text{out}}]\right\}$ is of size $|\mathcal{R}_{\text{out}}| \geq \ell_{\text{out}} - d_{\text{out}}$, or $(\texttt{abort}, \mathrm{P}_i)$ for some $\mathrm{P}_i \in \mathcal{I}$. If the adversary sent $(\texttt{continue}, \mathcal{R}_{\text{out}})$, then for every honest party $\mathrm{P}_i \notin \mathcal{I}$, the trusted party sends it $(\mathcal{R}_{\text{in}}, \mathcal{R}_{\text{out}}, \sigma'_i)$, where $\sigma'_i = \left(\sigma'_{ij}\right)_{j=1}^{\ell_{\text{out}}}$ are defined as

$$\sigma'_{ij} = \begin{cases} \sigma_{ij} & \text{if } (i,j) \in \mathcal{R}_{\text{out}} \\ \perp & \text{if } (i,j) \notin \mathcal{R}_{\text{out}} \end{cases}$$

Otherwise, if $\mathcal{A}$ sent $(\texttt{abort}, \mathrm{P}_i)$, then T sends $(\texttt{abort}, \mathrm{P}_i)$ to all honest parties.

Outputs: Each honest party outputs whatever it received from the trusted party, the parties in $\mathcal{I}$ output nothing, and the adversary outputs some function of its view.

Observe that if $d_{\text{in}} = d_{\text{out}} = 0$ then the above process is identical to the security-with-identifiable-abort process. We denote by $\text{IDEAL}_{C, \mathcal{A}(\rho_{\text{aux}})}^{(d_{\text{in}}, d_{\text{out}})\text{-IDPD}} (\kappa, (\rho_i)_{i=1}^n)$ the joint output of the adversary $\mathcal{A}$ and the honest parties in an execution of the above ideal-world computation of C, on security parameter κ, inputs $(\rho_i)_{i=1}^n$, auxiliary input ρ_{aux}, and packet-drop bounds d_{in} and d_{out}. When d_{in} and d_{out} are clear from context, we remove them from the notations.

We next give the definition of IDPD-security.

Definition 3.2 (IDPD-security)**.** Let π be a protocol for computing a circuit C, and let $d_{\text{in}} = d_{\text{in}}(\cdot)$ and $d_{\text{out}} = d_{\text{out}}(\cdot)$ be two polynomials. We say that π computes C with computational $(d_{\text{in}}, d_{\text{out}})$-*IDPD-security*, if the following holds. For every non-uniform QPT adversary $\mathcal{A}$, controlling a set $\mathcal{I} \subset \mathcal{P}$ in the real-world, there exists a non-uniform QPT adversary $\mathsf{Sim}_{\mathcal{A}}$, controlling $\mathcal{I}$ in the IDPD ideal-world, such that

$$\left\{\text{IDEAL}_{C, \mathsf{Sim}_{\mathcal{A}}(\rho_{\text{aux}})}^{(d_{\text{in}}, d_{\text{out}})\text{-IDPD}} (\kappa, (\rho_i)_{i=1}^n)\right\}_{\kappa \in \mathbb{N}, \rho_1, \ldots, \rho_n, \rho_{\text{aux}} \in \mathcal{D}^*} \overset{\text{C}}{\equiv} \left\{\text{REAL}_{\pi, \mathcal{A}(\rho_{\text{aux}})} (\kappa, (\rho_i)_{i=1}^n)\right\}_{\kappa \in \mathbb{N}, \rho_1, \ldots, \rho_n, \rho_{\text{aux}} \in \mathcal{D}^*}$$
(7)

Statistical and perfect security are defined similarly to Definition 3.1.

In Section 4, we reduce the problem of constructing a secure-with-identifiable-abort protocol, to the problem of constructing an IDPD-secure protocol.

3.2 The Hybrid Model

The *hybrid model* is a model that extends the real model with a trusted party that provides ideal computation for specific circuits. The parties communicate with this trusted party as specified by the ideal model.

Let C be a quantum circuit. Then, an execution of a protocol π computing a circuit C' in the C-hybrid model involves the parties sending normal messages to each other (as in the real model) and in addition, having access to a trusted party computing C. It is essential that the invocations of C are done sequentially, meaning that before an invocation of C begins, the preceding invocation must finish. In particular, there is at most a single call to C per round, and no other messages are sent during any round in which C is called.

Let type be an ideal world. Let $\mathcal{A}$ be a non-uniform QPT machine with auxiliary input ρ_{aux} controlling a subset $\mathcal{I} \subset \mathcal{P}$ of the parties. We denote by $\mathrm{HYBRID}^{C,\mathsf{type}}_{\pi,\mathcal{A}(\rho_{\mathsf{aux}})}(\kappa, \rho_1, \ldots, \rho_n)$ the joint output of the adversary and of the honest parties, following an execution of π with ideal calls to a trusted party computing C according to the ideal model "type," on inputs $\rho_1, \ldots, \rho_n$, auxiliary input ρ_{aux} given to $\mathcal{A}$, and security parameter κ. We call this the (C, type)-hybrid model. When type is clear from context we remove it for brevity.

3.2.1 The Classical MPC Hybrid Model

Following [DNS12, DGJ+20], throughout the paper, we assume the availability of a *trusted party*, denoted cMPC, that is able to compute any efficiently computable *classical* multiparty functionality. Furthermore, we assume cMPC is a *reactive* functionality, i.e., it is allowed to have an internal state that may be taken into account the next time it is invoked. One particular classical functionality we employ is the broadcast functionality. Thus, we implicitly assume that each party can broadcast a classical message at any given round of the protocol.

Similarly to [DGJ+20], we can implement cMPC using a post-quantum secure protocol. Specifically, we first remove the assumption that cMPC is reactive via standard techniques. To maintain security-with-identifiable-abort, this is done as follows. At the end of each call to cMPC, its state s will be shared in an additive n-out-of-n secret sharing scheme. Let s_i denote the ith share. The functionality then uses a post-quantum secure signature scheme to sign each share. Let σ_i denote the signature of s_i. The output of P_i will now additionally include s_i, σ_i, and the verification key of the signature scheme (which is the same for all parties). Note that the parties do not keep the signing key. In the next call to cMPC, the parties will additionally send their signed shares and keys to cMPC. If the keys are not all equal, then cMPC sends to party P_i the output $(\mathsf{abort}, \mathrm{P})$, where P is the lexicographically smallest party whose key differs from the key of P_i. Otherwise, if all the keys are the same, cMPC verifies all shares, sending the identity of a party whose verification failed if such a party exists, and reconstruct the state s and continue with the computation otherwise. Note that since the honest parties forward the output they received from the previous call, in case of abort they all agree on the identity of a corrupted party.

Finally, we can implement each call to cMPC assuming a correlated randomness setup, using the information theoretic UC-secure protocol of [IOZ14] and apply [Unr10]'s lifting theorem, to obtain post-quantum security. Furthermore, pre-computing the randomness in an off-line phase yields a protocol in the pre-processing model [DPSZ12]. Such protocols have an off-line phase which admits computational security, however, assuming no attack was successful during this phase, their online-phase admit information theoretic security.

Furthermore, for the sake of presentation, we sometimes abuse the existence of cMPC, and

construct some of the ideal worlds with the ability to interact with it. Although this cannot happen in the standalone model, such an assumption can be removed using the techniques described above, i.e., each party will hold a signed share of cMPC's input and receive a signed share of its output.

We denote by $\text{HYBRID}_{\pi, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, (\rho_i)_{i=1}^n)$ the joint output of the adversary $\mathcal{A}$, cMPC, and of the honest parties in a random execution of π in the cMPC-hybrid model on security parameter $\kappa \in \mathbb{N}$, inputs $\rho_1, \ldots, \rho_n$, and an auxiliary input ρ_{aux}.

4 Statement of Our Main Result

In this section we present the main theorem of the paper, namely that any multiparty quantum functionality can be computed with security-with-identifiable-abort against any number of corrupted parties.

Theorem 4.1. *Assume the existence of a classical quantum-resistant fully homomorphic encryption scheme with decryption circuit of logarithmic depth. Let C be an n-ary quantum circuit. Then C can be computed with computational security-with-identifiable-abort in the cMPC-hybrid model. Moreover, the round complexity of the protocol is independent of the circuit depth.*

Toward proving Theorem 4.1 we first show how to reduce the problem to the problem of constructing an IDPD-secure protocol for a related circuit. The following lemma states the existence of such an IDPD-secure protocol.

Lemma 4.2. *Assume the existence of a classical quantum-resistant fully homomorphic encryption scheme with decryption circuit of logarithmic depth. Let C be an n-ary quantum circuit. Then C can be computed with computational $(n^2, 2n^2)$-IDPD security in the cMPC-hybrid model. Moreover, the round complexity of the protocol is independent of the circuit depth.*

The proof of Lemma 4.2 is given in Section 10. Toward proving it, in the following sections we construct several building blocks used in the construction of the final protocol. We now use it to prove Theorem 4.1. It suffices to prove the following claim, asserting that security-with-identifiable-abort can be reduced to IDPD-security.

Claim 4.3. *Let C be an n-ary quantum circuit and let $d_{\mathsf{in}} = d_{\mathsf{in}}(\kappa)$ and $d_{\mathsf{out}} = d_{\mathsf{out}}(\kappa)$ be two polynomials. Additionally, let QECC denote a quantum error-correcting code that can tolerate $\max\{d_{\mathsf{in}}, d_{\mathsf{out}}\}$ errors. Then C can be computed with perfect security-with-identifiable-abort in the $(C', (d_{\mathsf{in}}, d_{\mathsf{out}})\text{-IDPD})$-hybrid model, where*

$$C' = \mathsf{QECC.Enc}^{\otimes n\ell_{\mathsf{out}}} \circ C \circ \mathsf{QECC.Dec}^{\otimes n\ell_{\mathsf{in}}}.$$

That is, C' transversely decodes each of its inputs using the QECC, computes C, and then re-encode each output.

Proof. The idea is as follows. First, each party encode each of its input-qubits using the QECC. The parties then call $(C', (d_{\mathsf{in}}, d_{\mathsf{out}})\text{-IDPD})$. If the functionality did not abort, then by the properties of the QECC, each party can locally decode its output.

We now present a formal proof. Let $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$ be the input of P_i, where each $\rho_{ij} \in \mathcal{D}^1$ is a single qubit. Similarly, we denote by $\sigma_i = (\sigma_{ij})_{j=1}^{\ell_{\mathsf{out}}}$ the ith output of $C(\rho_1, \ldots, \rho_n)$, where each $\sigma_{ij} \in \mathcal{D}^1$ is a single qubit. Additionally, let q be the number of qubits in each codeword of the QECC.

29

We construct a protocol in the $(C', (d_{\mathsf{in}}, d_{\mathsf{out}})$-IDPD)-hybrid model. For $k \in [q]$ we let ρ_{ijk} be the kth qubit of the QECC encoding of ρ_{ij}. Additionally, we let σ_{ijk} be the output of C' given to party P_i on input $(\rho_{ijk})_{i \in [n], j \in [\ell_{\mathsf{in}}], k \in [q]}$, being the kth qubit of the QECC applied by C'. We next formally describe our protocol π.

Protocol 1 Protocol π for computing the circuit C

Inputs: Each party P_i holds an input $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, where each $\rho_{ij} \in \mathcal{D}^1$
Common input: The parties hold the security parameter 1^κ.

1. Each party P_i encodes each qubit of its input to obtain $(\rho_{ijk})_{k=1}^q \leftarrow \mathsf{QECC.Enc}(\rho_{ij})$, for all $j \in [\ell_{\mathsf{in}}]$.

2. The parties call $(C', (d_{\mathsf{in}}, d_{\mathsf{out}})$-IDPD).

3. If the functionality aborted, then the parties hold the identity of a corrupted party. They output it and halt.

4. Otherwise, for all $i \in [n]$, party P_i receives $(\sigma_{ijk})_{(j,k) \in [\ell_{\mathsf{out}}] \times [q]}$.[7] It then decodes the codeword to obtain $\sigma_{ij} \leftarrow \mathsf{QECC.Dec}((\sigma_{ijk})_{k=1}^q)$ for all $j \in [\ell_{\mathsf{out}}]$, i.e., the jth qubit of its output. It then outputs $(\sigma_{ij})_{j \in \ell_{\mathsf{in}}}$ and halt.

We now show that π admits security-with-identifiable-abort. Fix an adversary $\mathcal{A}$ corrupting a subset $\mathcal{I}$ of the parties. The simulator $\mathsf{Sim}_\mathcal{A}$ in the ideal world, does the following.

Simulator 1 Simulator $\mathsf{Sim}_\mathcal{A}$

---

1. Query the adversary $\mathcal{A}$ for its inputs to $(C', (d_{\mathsf{in}}, d_{\mathsf{out}})$-IDPD). Let $(\rho_{ijk}^*)_{(i,j,k) \in \mathcal{I} \times [\ell_{\mathsf{in}}] \times [q]}$ be the inputs of the corrupted parties and let $\mathcal{R}_{\mathsf{in}}$ be the set of remaining input-qubits $\mathcal{A}$ sent. Note that each qubit of the ith party is indexed with two numbers j and k to include which of the qubits in each codeword are kept, i.e., $\mathcal{R}_{\mathsf{in}} \subseteq \{(i, (j,k)) : i \in [n], j \in [\ell_{\mathsf{in}}], k \in [q]\}$.

2. For all $i \in \mathcal{I}$ and $(i, (j,k)) \notin \mathcal{R}_{\mathsf{in}}$, update ρ_{ijk}^* to $|0\rangle$.

3. For all $i \in \mathcal{I}$ and $j \in \ell_{\mathsf{in}}$ compute $\rho_{ij}^* \leftarrow \mathsf{QECC.Dec}((\rho_{ijk}^*)_{k=1}^q)$. Send $(\rho_{ij}^*)_{(i,j) \in \mathcal{I} \times [\ell_{\mathsf{in}}]}$ to the trusted party T, and receive back $(\sigma_{ij})_{(i,j) \in \mathcal{I} \times [\ell_{\mathsf{out}}]}$.

4. For each $(i,j) \in \mathcal{I} \times [\ell_{\mathsf{out}}]$, encode each output-qubit to obtain $(\sigma_{ijk})_{k=1}^q \leftarrow \mathsf{QECC.Enc}(\sigma_{ij})$. Send $(\sigma_{ijk})_{(i,j,k) \in \mathcal{I} \times [\ell_{\mathsf{out}}] \times [q]}$ to $\mathcal{A}$ as the output of C'.

5. The adversary sends back either $(\mathtt{continue}, \mathcal{R}_{\mathsf{out}})$ or $(\mathtt{abort}, \mathrm{P})$ where $\mathrm{P} \in \mathcal{I}$. If it sent $(\mathtt{abort}, \mathrm{P})$, then forward it to the trusted party T. Otherwise send $\mathtt{continue}$ to T (note that we discard $\mathcal{R}_{\mathsf{out}}$).

6. Output whatever $\mathcal{A}$ outputs and halt.

---

[7]Observe that an adversary might cause some of the σ_{ijk}'s to be dropped during the call in Step 2. As this is a QECC codeword, for the sake of simplicity we ignore such technicalities.

First, consider the corrupted parties' inputs to C'. Clearly, in both the ideal and real worlds, the adversary chooses then same ρ^*_{ijk} and $\mathcal{R}_{\mathsf{in}}$. Therefore, $\mathsf{Sim}_{\mathcal{A}}$ and C' will decode the corrupted parties' inputs to the same values. Furthermore, as the QECC can tolerate d_{in} errors, C' will decode the inputs of the honest parties correctly. Thus, the outputs $\mathsf{Sim}_{\mathcal{A}}$ sends to $\mathcal{A}$ upon re-encoding, is the same as whatever C' sent to $\mathcal{A}$ in the real world. Therefore, $\mathcal{A}$ replied with the same message. If it sent $(\mathtt{continue}, \mathcal{R}_{\mathsf{out}})$, then due to the fact that the QECC can tolerate d_{out} errors, it follows that the output of the honest parties in the real world will be the same as in the ideal world. Finally, conditioned on the output of the honest parties' being the same in both worlds, it is clear that the output of $\mathcal{A}$ in the real world is the same as $\mathsf{Sim}_{\mathcal{A}}$ in the ideal world. Thus

$$\mathrm{IDEAL}_{C,\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})}\left(\kappa, (\rho_i)_{i=1}^n\right) = \mathrm{HYBRID}_{\pi, \mathcal{A}(\rho_{\mathsf{aux}})}^{(C', (d_{\mathsf{in}}, d_{\mathsf{out}})\text{-IDPD})}\left(\kappa, (\rho_i)_{i=1}^n\right).$$

□

5 Sequential Authentication

In this section, we present a protocol, called *sequential authentication* (SA), that allows a party – called the sender – to send an encryption of its input along a predetermined path known to everyone, to a designated party called the receiver (not necessarily different from the sender). The security achieved by this protocol roughly guarantees that in case the protocol is aborted, the parties will identify two parties, one of which is guaranteed to be corrupted. Later, in Section 6, using sequential calls to SA we show how to use it in order to augment the security to obtain an IDPD-secure protocol.

Let us first formally define the ideal world of SA. To simplify the presentation, we assume that cMPC is an additional party that will receive an output. Additionally, the parties have two common inputs, in addition to the security parameter. These are a path $\mathtt{PATH}$ and number of traps t. The domain of $\mathtt{PATH}$ is the set of all (non-simple) paths that goes through all parties, whose length is exactly[8] $\ell := n^2$. To remove confusion with the parties themselves, we call the parties along the path *relays* and denote by Q_i the ith party along the path (note that a single party may be multiple relays on the path). Furthermore, we call Q_1 the sender, and call Q_ℓ the receiver.

Ideal world of sequential authentication

Inputs: Each party P_i and cMPC holds the security parameter 1^κ, a path description $\mathtt{PATH} = (\mathsf{Q}_1, \cdots, \mathsf{Q}_\ell)$ that goes through every party at least once, and the number of traps $t = t(\kappa)$ required for the output ciphertext. The sender Q_1 holds an m-qubit input state ρ. The adversary $\mathcal{A}$ is given an auxiliary quantum state ρ_{aux}.

The sender sends input: If $\mathsf{Q}_1 \notin \mathcal{I}$, then it sends ρ as its input to T. Otherwise, the adversary chooses an input ρ^* to be given to T. Let ρ' be the input received by the trusted party.

The trusted party encodes the state and sends $\mathcal{A}$ its output: T samples a Clifford $E \leftarrow \mathcal{C}_{m+t}$ and encode ρ' to obtain $\sigma \leftarrow \mathsf{CAuth.Enc}_E(\rho')$. If $\mathsf{Q}_\ell \in \mathcal{I}$, then T sends σ to $\mathcal{A}$.

---

[8]The reason for the fixed length is due to a technicality that follows from the way SA is used.

The adversary instructs trusted party to continue or to abort: The adversary $\mathcal{A}$ sends to T either `continue` or $(\texttt{abort}, \mathsf{Q}_i, \mathsf{Q}_{i+1})$ where $1 \leq i < \ell$ and where either Q_i or Q_{i+1} is corrupted. If $\mathcal{A}$ sent `continue`, then T sends E to cMPC. Additionally, if the receiver $\mathsf{Q}_\ell \notin \mathcal{I}$ is honest, then T sends it σ. Otherwise, if $\mathcal{A}$ sends $(\texttt{abort}, \mathsf{Q}_i, \mathsf{Q}_{i+1})$, then T forwards it to the cMPC and all honest parties.

Outputs: The honest parties output whatever they received from T, the corrupted parties in $\mathcal{I}$ output nothing, and the adversary outputs some function of its view.

We denote by $\mathsf{SA}_{\mathcal{A}(\rho_{\mathsf{aux}})}(\kappa, \texttt{PATH}, t, \rho)$ the joint output of $\mathcal{A}$, the honest parties, and cMPC, in an execution of the above ideal world, on security parameter κ, input ρ, auxiliary input ρ_{aux}, path PATH, and the number of traps t.

As mentioned in Section 1.2.1, our construction is similar to swaddling from [DNS12] and the public authentication test from [DGJ$^+$20], both of which are based on Clifford code. Swaddling in [DNS12], is a technique for two parties to check whether each other's ciphertexts are evaluated correctly or not. Public authentication test in [DGJ$^+$20] extends it to the multiparty. Concretely, the ciphertext is transmitted through multiple parties and is only checked at the very end by the final receiver, which only guaranteed that the protocol will abort if the ciphertext was tampered with. However, since the ciphertext is only checked by the last party, the parties will not be able to know who performed the attack.

We next present a high level idea of our protocol, which follows similar ideas to the protocol of [DGJ$^+$20]. Let us first show a naïve construction. First, cMPC will generate a secret Clifford key E_1 and send it to the sender Q_1, who use it to encrypt its input ρ. Let $\sigma_1 = E_1 \left(\rho \otimes |0^{(\ell+1)t}\rangle\langle 0^{(\ell+1)t}| \right) E_1^\dagger$ be the resulting Clifford ciphertext. It then forwards it to the next relay Q_2. Following this, cMPC generates another Clifford $V_2 := E_2 E_1^\dagger$ for Q_2 used to check the authenticity of the ciphertext. Here, E_2 acts only on ρ and the first t traps. Then, upon applying V_2 to its message σ_1, Q_2 will obtain $E_2 \left(\rho \otimes |0^{\ell t}\rangle\langle 0^{\ell t}| \right) E_2^\dagger \otimes |0^t\rangle\langle 0^t|$, which can be verified by measuring the last t traps and compare them to 0^t. If Q_2 does not aborts, it sends the post-measurement state $E_2 \left(\rho \otimes |0^{\ell t}\rangle\langle 0^{\ell t}| \right) E_2^\dagger$ to Q_3, which repeats the same process.

This construction, however, has the following issue. If Q_1 is corrupted, then it may generate the initial traps incorrectly. Such an attack will not be immediately caught by an honest Q_2, as long as Q_1 prepares the last t traps correctly. To prevent such an attack, cMPC will slightly modify each V_i as follows. Let $t_i = (\ell - i + 2) \cdot t$ be the number of traps in the state that Q_{i-1} sent to Q_i. Then at round i, cMPC will sample an invertible linear function over t_i elements from $\mathbb{F}_2$, i.e., $G_i \leftarrow \mathrm{GL}_{t_i}(\mathbb{F}_2)$, and set $V_i := E_i G_i E_{i-1}^\dagger$. Here, G_i will act only on the traps to be measured by Q_i (we abuse notation and allow G_i to act on qubits as $G_i |x\rangle = |G_i(x)\rangle$). Observe that if $x \in \mathbb{F}_2^{t_i}$ is not the all-zero vector, then each coordinate of $G_i(x)$ has probability $1/2$ to be 0. Thus, if Q_{i-1} did not prepare the traps correctly, it will be caught by the next party on the path with probability at least $1 - 1/2^n$. Moreover, each G_i can be realized using Clifford operators. Indeed, one such way is through the LU-decomposition, and implementing the triangular matrices using CNOTs.

We next present our protocol π_{SA} for computing SA. In the following we let M be the m-qubit register holding the input-message ρ. For all $2 \leq i \leq \ell$, let T_i be a t-qubit register for storing the traps that Q_i will verify. Let $\overline{T}_\ell$ be a t-qubit register for the traps on the output ciphertext, and denote $\overline{T}_i = \overline{T}_\ell T_\ell \ldots T_{i+1}$ for all $i \in [\ell - 1]$. Finally, for $i \in \{2, \ldots, \ell\}$ let $t_i = (\ell - i + 2) \cdot t$ be the size of register $\overline{T}_i T_i$.

Protocol 2 Protocol π_{SA} for computing SA

Inputs: The sender Q_1 holds an input $\rho \in \mathcal{D}^m$.

Common input: All parties hold the security parameter 1^κ, a path PATH that goes through every party, and the trap-size t.

1. cMPC samples a Clifford $E_1 \leftarrow \mathcal{C}_{m+\ell t}$ uniformly at random and sends it to the sender Q_1.

2. Q_1 computes $\sigma_1^{M\overline{T}_1} = E_1 \left(\rho^M \otimes |0^{(\ell+1)t}\rangle\langle 0^{(\ell+1)t}|^{\overline{T}_1} \right) E_1^\dagger$ and sends it to Q_2.

3. For $i = 2$ to ℓ:

   (a) cMPC samples a Clifford $E_i \leftarrow \mathcal{C}_{m+t_i-t}$ and an invertible linear function $G_i \leftarrow \mathrm{GL}_{t_i}(\mathbb{F}_2)$ independently and uniformly at random. It sets
   $$V_i := \left(E_i^{M\overline{T}_i} \otimes I^{\otimes t T_i} \right) \left(I^{\otimes m^M} \otimes G_i^{\overline{T}_i T_i} \right) E_{i-1}^{\dagger}{}^{M\overline{T}_i T_i}$$
   and sends it to Q_i.

   (b) Q_i evaluates $\tau_i^{M\overline{T}_i T_i} = V_i \sigma_{i-1} V_i^\dagger$ and measures register T_i in the computational basis. If the measurement result is 0^t then Q_i sends Acc to cMPC. Otherwise it sends Rej to cMPC. Let $\sigma_i^{M\overline{T}_i}$ denote the post-measurement state.

   (c) If cMPC received Rej, then it sends $(\texttt{abort}, Q_{i-1}, Q_i)$ to all parties. They then output this message and halt.

   (d) If $i < \ell$ then Q_i sends σ_i to the next relay Q_{i+1}.

4. The receiver Q_ℓ outputs σ_ℓ, cMPC outputs E_ℓ, and the other parties output nothing.

The following theorem states that π_{SA} securely computes the functionality SA.

Lemma 5.1. *For every non-uniform* QPT *adversary* $\mathcal{A}$, *controlling a set* $\mathcal{I} \subset \mathcal{P}$ *in the real-world, there exists a non-uniform* QPT *adversary* $\mathsf{Sim}_{\mathcal{A}}$, *controlling* $\mathcal{I}$ *in the ideal-world for* SA, *such that*

$$\left\{ \mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \texttt{PATH}, t, \rho) \right\}_{\kappa, t, \texttt{PATH}, \rho, \rho_{\mathsf{aux}}} \approx_{\mathrm{neg}(\kappa)} \left\{ \mathrm{REAL}^{\mathsf{cMPC}}_{\pi_{\mathsf{SA}}, \mathcal{A}(\rho_{\mathsf{aux}})} (\kappa, \rho) \right\}_{\kappa, t, \texttt{PATH}, \rho, \rho_{\mathsf{aux}}}. \quad (8)$$

Before proving the security of π_{SA}, we first state a useful property of the linear operators G_i. The proof of Lemma 5.1 is deferred to Section 5.1. It is shown in [DGJ$^+$20] that, if T_i and $\overline{T}_i$ are of the same size, then applying a random element of $\mathrm{GL}(m, \mathbb{F}_2)$ on the register $\overline{T}_i T_i$ followed by measuring T_i results in the following. If the outcome of the measurement is 0^t, then the post-measurement state on $\overline{T}_i T_i$ is 0^m with overwhelming probability. In other words, if a corrupted party did not tampered with register T_i, then with high probability the next party can deduce that the rest of the traps were not tampered with as well. We observe that this can be generalized in a straightforward way to the case where T_i and $\overline{T}_i$ are possibly of different sizes. Moreover, [DGJ$^+$20] considered a general measurement result that is compared to an arbitrary string r. For our purposes, it suffices to consider only the case where $r = 0$. To make the above statement precise, we first define the following operators. First, for each projector P, define $\mathcal{L}^P$ as the quantum channel that replaces the component removed in P with $\perp$, that is

$$\mathcal{L}^P(\rho) := P\rho P^\dagger + \mathrm{tr}(Q\rho Q^\dagger) |\perp\rangle\langle\perp|$$

where $Q = I - P$ and where $|\bot\rangle$ represents a rejection state that satisfies $P|\bot\rangle = 0$. Now define the *partial projector* on 0^t as

$$P_{\mathsf{part},i} := I^{\overline{T}_i} \otimes |0\rangle\langle 0|^{T_i},$$

which gives the post-measurement state after measuring 0^t on register T_i. Observe that applying a random G_i followed by checking whether or not $r_i = 0$, can be written as $\mathcal{L}^{P_{\mathsf{part},i}} \circ \mathcal{T}_{\mathrm{GL}(tn, \mathbb{F}_2)}$, where $\mathcal{T}_{\mathcal{S}}$ is the *twirling* operation over a set of operators $\mathcal{S}$, defined *twirling* as

$$\mathcal{T}_{\mathcal{S}}(\rho) = \frac{1}{|\mathcal{S}|} \sum_{U \in \mathcal{S}} U \rho U^\dagger. \tag{9}$$

Additionally, define the *full projector* as

$$P_{\mathsf{full},i} := |0^{t_i}\rangle\langle 0^{t_i}|^{\overline{T}_i T_i}$$

which describes checking all the traps. Using these notations, we can now formally describe the main property of the linear operators that we use.

Lemma 5.2. *Applying a random element of* $\mathrm{GL}(tn, \mathbb{F}_2)$ *followed by* $\mathcal{L}^{P_{\mathsf{part},i}}$ *is equivalent to applying* $\mathcal{L}^{P_{\mathsf{full},i}}$ *up to a negligible difference. That is,*

$$\left\| \mathcal{L}^{P_{\mathsf{full},i}} - \mathcal{L}^{P_{\mathsf{part},i}} \circ \mathcal{T}_{\mathrm{GL}(tn,\mathbb{F}_2)} \right\|_\diamond \leq \mathrm{neg}(t).$$

The proof follows similar argument to the proof of Lemma 4.3 of [DGJ+20] and is therefore omitted.

5.1 Proof of Lemma 5.1

We are now ready to prove the security of π_{SA}.

Proof of Lemma 5.1. Observe that the correctness directly follows from the correctness of the Clifford encryption and decryption.

Now, fix an adversary $\mathcal{A}$ corrupting a subset $\mathcal{I}$ of the parties holding auxiliary input ρ_{aux}. We next construct its simulator $\mathsf{Sim}_{\mathcal{A}}$. Roughly, it will emulate the honest parties and cMPC, replacing Clifford encoding of unknown states with the encoding of the 0 state. In more detail, we first partition the protocol into three parts according to the identities of the corrupted parties. Let i_F and i_L be the indexes of the first and last honest parties on the path, respectively (note that it could be the case where $i_F = i_L$). The first part of the protocol, is when the state is held by the corrupted relays Q_i where $i < i_F$. The second part is when $i_F \leq i \leq i_L$, and the third part is when $i > i_L$. Note that the first and third parts may be empty, depending on whether the sender or the receiver are corrupted.

For the first part, if the sender Q_1 is corrupted, then $\mathsf{Sim}_{\mathcal{A}}$ can reconstruct the adversary's input once the path reaches the honest relay Q_{i_F}. Indeed, once the state is held by an honest party, the adversary is *committed* to the state being encoded, since any tampering done to it by a subsequent corrupted party will be caught. Simulating the second part is done in the natural way, by emulating the honest parties and cMPC. As for the final part, if the receiver Q_ℓ is corrupted, then the simulator will receive the output σ from the trusted party. To ensure that σ will be the last encoding $\mathcal{A}$ receives from the simulator, $\mathsf{Sim}_{\mathcal{A}}$ will undo the operations done in the last $\ell - i_L$ steps of the protocol. That is, it will append to σ the traps $|0^{t_{i_L}}\rangle$ to be measured by Q_{i_L+1} to

Q_ℓ, and send to $\mathcal{A}$ the state obtained from applying $V_{i_L+1}^\dagger \ldots V_\ell^\dagger$ to $\sigma \otimes |0^{t_{i_L}}\rangle\langle 0^{t_{i_L}}|$, as the state it expects to receive from Q_{i_L}.

Simulator 2 Simulator $\mathsf{Sim}_\mathcal{A}$ for SA

---

1. Let i_F, i_L be the indexes of the first and last honest relays on PATH, respectively.

2. If $i_F \neq 1$, then send to $\mathcal{A}$ a random uniform Clifford $E_1 \leftarrow \mathcal{C}_{m+(\ell+1)t}$.

3. For $i = 2$ to i_F do the following.

   (a) Set the Clifford

   $$V_i = \left(E_i^{M\overline{T}_i} \otimes I^{\otimes t^{T_i}} \right) \left(I^{\otimes mM} \otimes G_i^{\overline{T}_i T_i} \right) E_{i-1}^{\dagger}{}^{M\overline{T}_i T_i}$$

   and send it to $\mathcal{A}$ as what Q_i expects to receive from cMPC, where $G_i \leftarrow \mathrm{GL}_{t_i}(\mathbb{F}_2)$ and $E_i \leftarrow \mathcal{C}_{m+t_i-t}$ are sampled independently and uniformly at random.

   (b) Query $\mathtt{flag}_i$ from $\mathcal{A}$. If $\mathtt{flag}_i = \mathsf{Rej}$, send $(\mathtt{abort}, Q_{i-1}, Q_i)$ to the trusted party T, halt, and output whatever $\mathcal{A}$ outputs. Otherwise, the simulator continues.

   (c) If $i = i_F - 1$, then $\mathcal{A}$ also sends a state σ^* from Q_{i-1}. The simulator decodes it to obtain $(\rho^*, \mathtt{flag}^*) = \mathsf{CAuth.Dec}_{E_{i-1}}(\sigma^*)$. If $\mathtt{flag}^* = \mathsf{Rej}$, then send $(\mathtt{abort}, Q_{i_F-1}, Q_{i_F})$ to T, output whatever $\mathcal{A}$ outputs, and halt. Otherwise, if $\mathtt{flag}^* = \mathsf{Acc}$, send ρ^* to T.

4. Sample $E_{i_F} \leftarrow \mathcal{C}_{m+t_{i_F}-t}$ and compute $\sigma_{i_F} = \mathsf{CAuth.Enc}_{E_{i_F}}(|0^m\rangle\langle 0^m|)$.

5. For $i = i_F + 1$ to i_L, do the following.

   - If $Q_i \in \mathcal{I}$ is corrupted:

     (a) If $Q_{i-1} \notin \mathcal{I}$ is honest, send σ_{i-1} to $\mathcal{A}$.

     (b) Set the Clifford

     $$V_i = \left(E_i^{M\overline{T}_i} \otimes I^{\otimes t^{T_i}} \right) \left(I^{\otimes mM} \otimes G_i^{\overline{T}_i T_i} \right) E_{i-1}^{\dagger}{}^{M\overline{T}_i T_i}$$

     and send it to $\mathcal{A}$ as what Q_i expects to receive from cMPC, where $G_i \leftarrow \mathrm{GL}_{t_i}(\mathbb{F}_2)$ and $E_i \leftarrow \mathcal{C}_{m+t_i-t}$.

     (c) Query $\mathtt{flag}_i$ from $\mathcal{A}$. If $\mathtt{flag}_i = \mathsf{Rej}$, send $(\mathtt{abort}, Q_{i-1}, Q_i)$ to the trusted party T, halt, and output whatever $\mathcal{A}$ outputs. Otherwise, the simulator continues.

   - Otherwise, if $Q_i \notin \mathcal{I}$ is honest:

     (a) If $Q_{i-1} \in \mathcal{I}$ is corrupted, receive σ_{i-1} from $\mathcal{A}$.

     (b) The simulator decodes σ_{i-1} to obtain $(\rho_i, \mathtt{flag}_i) = \mathsf{CAuth.Dec}_{E_{i-1}}(\sigma_{i-1})$. If $\mathtt{flag}_i = \mathsf{Rej}$, then send $(\mathtt{abort}, Q_{i-1}, Q_i)$ to T, output whatever $\mathcal{A}$ outputs, and halt. Otherwise if $\mathtt{flag}_i = \mathsf{Acc}$, compute $\sigma_i = \mathsf{CAuth.Enc}_{E_i}(\rho_i)$ where $E_i \leftarrow \mathcal{C}_{m+t_i-t}$.

6. If $i_L = \ell$, then send $\mathtt{continue}$ to T, output whatever $\mathcal{A}$ outputs, and halt.

7. Otherwise, if $i_L < \ell$, then do the following.

   (a) Receive σ from T.

(b) For all $i \in \{i_L + 1, \ldots, \ell\}$, sample $V_i \leftarrow \mathcal{C}_{m+t_i}$ independently and uniformly at random.

(c) Send

$$\sigma_{i_L+1} = V_{i_L+1}^\dagger \cdots V_\ell^\dagger \left(\sigma \otimes |0^{t_{i_L}}\rangle\langle 0^{t_{i_L}}| \right) V_\ell \cdots V_{i_L+1}$$

to $\mathcal{A}$ as what it expects to receive from Q_{i_L}.

(d) For $i = i_L + 1$ to ℓ do the following.

    i. Send V_i to $\mathcal{A}$.

    ii. Query $\mathtt{flag}_i$ from $\mathcal{A}$. If $\mathtt{flag}_i = \mathsf{Rej}$, send $(\mathtt{abort}, \mathsf{Q}_{i-1}, \mathsf{Q}_i)$ to the trusted party T, halt, and output whatever $\mathcal{A}$ outputs. Otherwise, the simulator continues.

(e) Send $\mathtt{continue}$ to T, output whatever $\mathcal{A}$ outputs, and halt.

We now analyze the simulator. We denote by R the real-world adversary's internal reference register. We additionally assume without loss of generality that each of the adversary's attacks can be fully described by a unitary A_i as follows. Upon receiving V_i, it first writes $|V_i\rangle\langle V_i|$ into its reference system. It then applies V_i honestly before applying its chosen attack A_i. This is also without loss of generality, as any other attack A_i' applied before V_i, can be expressed as $A_i' = V_i^\dagger A_i V_i$, for some unitary A_i. Finally, to generate $\mathtt{flag}_i$, $\mathcal{A}$ measures a qubit F in its register R. This is without loss of generality since we can assume that everything the adversary holds is quantum, so a measurement is required to generate a classical bit. Moreover, if the adversary wanted to measure some other qubit or measure in a different basis, this behavior can also be expressed as part of A_i instead.

Analysis of the first part. We start with analyzing the first part of the protocol, showing that the simulator successfully simulates the $\mathcal{A}$'s view. Note that if Q_1 is honest then this part is empty. Now, assuming $\mathsf{Q}_1 \in \mathcal{I}$ is corrupted, the adversary in the real world only receives from cMPC the Cliffords E_1 and $V_1, \ldots, V_{i_F-1}$. Clearly, the simulator samples these Cliffords with exactly the same distribution. Thus, the adversary sends the same $\mathtt{flag}_i$ for all $i \in [i_F - 1]$, so both cMPC in the real world and $\mathsf{Sim}_{\mathcal{A}}$ in the ideal world, will abort at every round with exactly the same probability. Therefore, we may condition on the adversary sending $\mathtt{flag}_i = \mathsf{Acc}$ for all $i \in [i_F - 1]$. Next, when $i = i_F - 1$ the $\mathcal{A}$ additionally holds the same state $\sigma^{* M \overline{T}_{i_F-1} R}$ in both worlds, and sends $\sigma^{* M \overline{T}_{i_F-1}}$. Write the result of correctly decrypting σ^* as

$$\mathsf{CAuth.Dec}_{E_{i-1}}(\sigma^*) = q_{\mathsf{Acc}} \left(\rho^{* MR} \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| \right) + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot|^M \otimes \sigma_{\mathsf{Rej}}^R \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}| \right),$$

where

$$q_{\mathsf{Acc}} = \mathrm{tr} \left(\langle 0^{t_i-t}| E_{i-1}^\dagger \sigma^* E_{i-1} |0^{t_i-t}\rangle \right)$$

is the accept probability of Q_{i_F}, where

$$\rho^* = \frac{1}{q_{\mathsf{Acc}}} \langle 0^{t_i-t}| E_{i-1}^\dagger \sigma^* E_{i-1} |0^{t_i-t}\rangle$$

is the decrypted value conditioned on success, where

$$\begin{aligned} q_{\mathsf{Rej}} &= 1 - q_{\mathsf{Acc}} \\ &= \mathrm{tr} \left(\left(I - |0^{t_i-t}\rangle\langle 0^{t_i-t}| \right) \left(E_{i-1}^\dagger \sigma^* E_{i-1} \right) \left(I - |0^{t_i-t}\rangle\langle 0^{t_i-t}| \right) \right) \end{aligned}$$

is the reject probability, and where

$$\sigma_{\mathsf{Rej}}^{R} = \frac{1}{q_{\mathsf{Rej}}} \left(\sum_{s \neq 0^{t_i - t}} \langle s^{\overline{T}_{i-1}} | E_{i-1}^{\dagger} \sigma^* E_{i-1} | s \rangle \right)$$

is the post-measurement state held by $\mathcal{A}$ conditioned on reject.

We now focus on the real world and claim that

$$\mathcal{L}^{P_{\mathsf{part},i}} (V_i \sigma^* V_i^{\dagger}) \approx_{\mathrm{neg}(\kappa)} q_{\mathsf{Acc}} \left(\mathsf{CAuth.Enc}_{E_i}(\rho^*) \otimes |0^t\rangle\langle 0^t| \right) + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right). \qquad (10)$$

Indeed, observe that

$$\begin{aligned}
\mathcal{L}^{P_{\mathsf{part},i}} \left(V_i \sigma^* V_i^{\dagger} \right) &= \mathcal{L}^{P_{\mathsf{part},i}} \left(E_i^{M\overline{T}_i} G_i^{\overline{T}_i T_i} \left(E_{i-1}^{\dagger} \right)^{M\overline{T}_i T_i} \sigma E_{i-1} G_i^{\dagger} E_i^{\dagger} \right) \\
&= \mathcal{L}^{P_{\mathsf{part},i}} \left(E_i \left(\mathcal{T}_{\mathrm{GL}(tn,\mathbb{F}_2)} \left(E_{i-1}^{\dagger} \sigma^* E_{i-1} \right) \right) E_i^{\dagger} \right) \\
&= E_i \left(\mathcal{L}^{P_{\mathsf{part},i}} \circ \mathcal{T}_{\mathrm{GL}(tn,\mathbb{F}_2)} \left(E_{i-1}^{\dagger} \sigma^* E_{i-1} \right) \right) E_i^{\dagger} \\
&\approx_{\mathrm{neg}(\kappa)} E_i \left(\mathcal{L}^{P_{\mathsf{full},i}} \left(E_{i-1}^{\dagger} \sigma^* E_{i-1} \right) \right) E_i^{\dagger},
\end{aligned}$$

where the third equality follows from the fact that, E_i and $P_{\mathsf{part},i}$ act on different registers, and $E_i |\bot\rangle\langle\bot| E_i^{\dagger} = |\bot\rangle\langle\bot|$, and the forth transition is due to Lemma 5.2. From the definition of $\mathcal{L}^{P_{\mathsf{full},i}}$, it follows that

$$\begin{aligned}
E_i \left(\mathcal{L}^{P_{\mathsf{full},i}} \left(E_{i-1}^{\dagger} \sigma^* E_{i-1} \right) \right) E_i^{\dagger} &= E_i \left(|0^{t_i}\rangle\langle 0^{t_i}|^{\overline{T}_i T_i} E_{i-1}^{\dagger} \sigma^* E_{i-1} |0^{t_i}\rangle\langle 0^{t_i}|^{\overline{T}_i T_i} + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right) \right) E_i^{\dagger} \\
&= E_i \left(|0^{t_i}\rangle^{\overline{T}_i T_i} (q_{\mathsf{Acc}} \cdot \rho^{*MR}) \langle 0^{t_i}|^{\overline{T}_i T_i} + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right) \right) E_i^{\dagger} \\
&= E_i \left(q_{\mathsf{Acc}} \left(\rho^* \otimes |0^{t_i}\rangle\langle 0^{t_i}| \right) + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right) \right) E_i^{\dagger} \\
&= E_i \left(q_{\mathsf{Acc}} \left(\rho^* \otimes |0^{t_i}\rangle\langle 0^{t_i}| \right) \right) E_i^{\dagger} + E_i \left(q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right) \right) E_i^{\dagger} \\
&= q_{\mathsf{Acc}} \left(\mathsf{CAuth.Enc}_{E_i}(\rho^*) \otimes |0^t\rangle\langle 0^t| \right) + q_{\mathsf{Rej}} \left(|\bot\rangle\langle\bot| \otimes \sigma_{\mathsf{Rej}}^{R} \right).
\end{aligned}$$

We conclude that, up to a negligible difference, Equation (10) states that both worlds have the same reject probability q_{Rej} at step i_F. Moreover, on reject, in both the real and ideal worlds, the adversary holds only $\sigma_{\mathsf{Rej}}^{R}$, and honest parties output $(\mathtt{abort}, \mathsf{Q}_{i-1}, \mathsf{Q}_i)$, hence they are indistinguishable. Let us now condition on the event where the protocol does not abort on the first part. In this case, the adversary holds ρ^{*R} in both worlds. In the real world Q_{i_F} holds $\mathsf{CAuth.Enc}_{E_i} \left(\rho^{*M} \right)$, where E_i is held by cMPC, while in the ideal world the simulator (which emulates the honest relay Q_{i_F}) generates $\mathsf{CAuth.Enc}_{E_i} \left(|0\rangle\langle 0| \right)$, and send ρ^{*M} to the trusted party.

Analysis of the second part. We now analyze the simulation of the second part of the protocol. Denote by τ^{MR} as the joint message-reference state by the end of the previous part. That is, if $\mathsf{Q}_1 \in \mathcal{I}$ is corrupted, then τ is the ρ^* given by the discussion above, together with the adversary's reference system at the end of the first part. Otherwise if $\mathsf{Q}_1 \notin \mathcal{I}$ is honest, then τ is the joint input-reference state given by the input of the sender.

For $i \in \{i_F + 1, \ldots, \ell\}$, let $\mathrm{v}_{\mathrm{R},i}$ denote the contents of registers $M\overline{T}_i R$ right *after* measuring T_i (and before sending the state). Moreover, these are defined to be contents conditioned on Q_i sending Acc, and are thus are subnormalized. Similarly, let $\mathrm{v}_{\mathrm{I},i}$ denote the contents in the corresponding

registers in the ideal world.

The next claim asserts that in the second part of the protocol, the view in both worlds can be expressed as an operator applied to the ith ciphertext and $\mathcal{A}$'s auxiliary reference register. In the following, we say that the protocol aborts at Q_i if it sent Rej to cMPC.

Claim 5.3. *For all $i \in \{i_F, \ldots, i_L\}$ and for all possible sequences $(V_j)_{j \leq i, j \in \mathcal{I}}$, the following holds.*

- *The probability of aborting at Q_i is the same in both real and ideal worlds. Moreover, if the protocol aborts, then the real and ideal worlds are indistinguishable.*

- *If the protocol does not abort at Q_i, then there exists a CP map W_i, acting on registers M, R, and $\overline{T}_i$, such that*

$$
V_{R,i} \approx_{\mathrm{neg}(\kappa)} W_i \left(\mathsf{CAuth.Enc}_{E_i}^{M \to M\overline{T}_i} \left(\tau^{MR} \right) \right)
$$

where E_i is the random Clifford sampled by cMPC, and

$$
V_{I,i} \approx_{\mathrm{neg}(\kappa)} W_i \left(\mathsf{CAuth.Enc}_{E_i}^{M \to M\overline{T}_i} \left(|0^m\rangle\langle 0^m| \right) \otimes \tau^R \right).
$$

where E_i is the random sampled by $\mathsf{Sim}_{\mathcal{A}}$. Moreover, if $Q_i \notin \mathcal{I}$, then W_i acts on register R alone.

The proof of the claim is done using a simple inductive argument, and is proven below. Let us first use it to argue for the success of simulation of the second part of the protocol. First note that the distributions of V_j are the same in real and ideal worlds. Therefore, we may fix $(V_j)_{j \leq i, j \in \mathcal{I}}$. Moreover, by Claim 5.3 we may assume that the protocol did not abort at the second part.

Let us now consider the case where the receiver Q_ℓ is honest, i.e., $i_L = \ell$. By Claim 5.3, the adversary $\mathcal{A}$ holds the state $W_\ell^R \left(\tau^R \right)$ in both worlds. Thus,

$$
\mathrm{REAL}_{\pi_{\mathsf{SA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} W_\ell \left(\mathsf{CAuth.Enc}_{E_\ell}^M \left(\tau^{MR} \right) \right) \otimes |E_\ell\rangle\langle E_\ell|
$$

where E_ℓ is the output of cMPC. On the other hand, in the ideal world, $\mathsf{CAuth.Enc}_{E_\ell}(|0^m\rangle)$ and E_ℓ are discarded. Instead, cMPC outputs a uniform Clifford key E and the honest receiver Q_ℓ outputs $\mathsf{CAuth.Enc}_E(\tau^M)$ as given to them by the trusted party. Therefore

$$
\mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} W_\ell \left(\mathsf{CAuth.Enc}_E^M \left(\tau^{MR} \right) \right) \otimes |E\rangle\langle E|.
$$

So we conclude that

$$
\mathrm{REAL}_{\pi_{\mathsf{SA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} \mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \mathtt{PATH}, t, \rho).
$$

Now, we consider the case where $i_L \neq \ell$.

Analyzing the third part. Similarly to Claim 5.3, we can express the view in the third part in both worlds using an appropriate operator.

Claim 5.4. *For all $i \in \{i_L + 1, \ldots, \ell\}$ and for all possible sequences $(V_j)_{j \leq i, j \in \mathcal{I}}$, the following holds.*

- *The probability of aborting at Q_i is the same in real and ideal worlds. Moreover, if the protocol aborts, then the real and ideal worlds are indistinguishable.*

- *If the protocol does not abort at Q_i, then there exists a CP map W_i, acting on registers M, R, and $\overline{T}_i$, so that*

$$\mathrm{v}_{\mathrm{R},i} \approx_{\mathrm{neg}(\kappa)} W_i \left(\mathsf{CAuth.Enc}_{E_i}^{M \to M\overline{T}_i} \left(\tau^{MR} \right) \right)$$

and

$$\mathrm{v}_{\mathrm{I},i} \approx_{\mathrm{neg}(\kappa)} W_i \left(U_i^\dagger \left(\tilde{\sigma}^{M\overline{T}_\ell R} \otimes |0^{(\ell-i)t}\rangle\langle 0^{(\ell-i)t}| \right) U_i \right)$$

where $\tilde{\sigma}$ is the joint state of output σ and where τ^R, $U_i = V_\ell \ldots V_{i+1}$ for $i < \ell$ and $U_\ell = I$.

Similarly to Claim 5.3, the proof of the above claim is done using a inductive argument. The analysis is almost identical to that of Claim 5.3 and is therefore omitted. Observe that the distributions of the V_j's are uniformly random in both the real and ideal worlds, so we can fix a particular sequence of $(V_j)_{j \leq i, j \in \mathcal{I}}$. Now, assuming the protocol terminates successfully, by Claim 5.4, it follows that

$$\mathrm{REAL}_{\pi_{\mathsf{SA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} W_\ell \left(\mathsf{CAuth.Enc}_{E_\ell}^{M} \left(\tau^{MR} \right) \right) \otimes |E_\ell\rangle\langle E_\ell| .$$

On the other hand, in the ideal world cMPC outputs a uniformly random Clifford E and the output sent to the simulator is $\sigma^{M\overline{T}_\ell} = \mathsf{CAuth.Enc}_E^{M}(\tau^M)$. Therefore, $\tilde{\sigma} = \mathsf{CAuth.Enc}_E^{M}(\tau^{MR})$, hence

$$\mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} W_\ell \left(\mathsf{CAuth.Enc}_E^{M} \left(\tau^{MR} \right) \right) \otimes |E\rangle\langle E|$$

We conclude that

$$\mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \mathtt{PATH}, t, \rho) \approx_{\mathrm{neg}(\kappa)} \mathrm{REAL}_{\pi_{\mathsf{SA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) .$$

$\square$

We now work towards proving the claims used earlier Claims 5.3 and 5.4. First, we show that the adversary has no information on the Clifford key E_i sampled by cMPC, when it chooses its attack.

Claim 5.5. *Let $n \in \mathbb{N}$ and let $(t_i)_{i=1}^n$ be a decreasing sequence of positive integers. For $i \in [n]$, let $\mathcal{G}_i$ be a random variables over $\mathcal{C}_{t_i}$, and let $\mathcal{E}_i$ be uniform random variable over $\mathcal{C}_{t_i}$. For $2 \leq i \leq n$, define the random variable $\mathcal{V}_i = (\mathcal{E}_i \otimes I^{\otimes(t_{i-1}-t_i)})(I^{\otimes(t_{i-1}-t_i)} \otimes \mathcal{G}_i)\mathcal{E}_{i-1}^\dagger$ over $\mathcal{C}_{t_i}$. Then, assuming $\{\mathcal{G}_i, \mathcal{E}_i\}_{i=1}^n$ are independent, for all $V_i \in \mathcal{C}_{t_{i-1}}$ and $E_n \in \mathcal{C}_{l_n}$, it holds that*

$$\Pr \left(\mathcal{E}_n = E_n \, \middle| \, \bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i) \right) = \Pr(\mathcal{E}_n = E_n).$$

Proof. Equivalently, we may prove that for all $V_i \in \mathcal{C}_{t_{i-1}}, E_n \in \mathcal{C}_{l_n}$, it holds that

$$\Pr \left(\bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i) \, \middle| \, \mathcal{E}_n = E_n \right) = \Pr \left(\bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i) \right). \tag{11}$$

First, consider the left-hand side of Equation (11). By the chain rule of probabilities it holds that

$$\Pr \left(\bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i) \, \middle| \, \mathcal{E}_n = E_n \right) = \prod_{2 \leq i \leq n} \Pr \left(\mathcal{V}_i = V_i \, \middle| \, \left(\bigwedge_{i+1 \leq j \leq n} \mathcal{V}_j = V_j \right) \wedge \mathcal{E}_n = E_n \right). \tag{12}$$

To simplify notation, we denote by B_i the event that

$$\left(\left(\bigwedge_{i+1 \leq j \leq n} \mathcal{V}_j = V_j\right) \wedge \mathcal{E}_n = E_n\right).$$

Then it suffices to show that for all $2 \leq i \leq n$, it holds that

$$\Pr\left(\mathcal{V}_i = V_i \mid B_i\right) = \frac{1}{|\mathcal{C}_{t_{i-1}}|}. \tag{13}$$

Indeed, by the definition of $\mathcal{V}_i$, it holds that

$$\Pr\left(\mathcal{E}_i \mathcal{G}_i \mathcal{E}_{i-1}^\dagger = V_i \mid B_i\right) = \Pr\left(\mathcal{E}_{i-1} = V_i^\dagger \mathcal{E}_i \mathcal{G}_i \mid B_i\right). \tag{14}$$

Therefore, by the law of total probabilities it follows that

$$\Pr\left(\mathcal{E}_{i-1} = V_i^\dagger \mathcal{E}_i \mathcal{G}_i \mid B_i\right) = \sum_{E \in \mathcal{C}_{t_{i-1}}} \Pr\left(\mathcal{E}_{i-1} = E \mid V_i^\dagger \mathcal{E}_i \mathcal{U}_i = E \wedge B_i\right) \Pr\left(V_i^\dagger \mathcal{E}_i \mathcal{U}_i = E \mid B_i\right)$$

$$= \sum_{E \in \mathcal{C}_{t_{i-1}}} \frac{1}{|\mathcal{C}_{t_{i-1}}|} \Pr\left(V_i^\dagger \mathcal{E}_i \mathcal{U}_i = E \mid B_i\right)$$

$$= \frac{1}{|\mathcal{C}_{t_{i-1}}|},$$

where the third equality follows from the assumption that $\{\mathcal{E}_i, \mathcal{U}_i\}_{i=1}^n$ are independent, hence the events $\mathcal{E}_{i-1} = E$ and $V_i^\dagger \mathcal{E}_i \mathcal{U}_i = E \wedge B_i$ are independent.

Now consider the right-hand side of Equation (11). By the law of total probabilities it follows that

$$\Pr\left(\bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i)\right) = \sum_{E \in \mathcal{C}_{l_n}} \Pr\left(\bigwedge_{2 \leq i \leq n} (\mathcal{V}_i = V_i) \mid \mathcal{E}_n = E\right) \cdot \Pr\left(\mathcal{E}_n = E\right)$$

$$= \sum_{E \in \mathcal{C}_{l_n}} \prod_{2 \leq i \leq n} \frac{1}{|\mathcal{C}_{t_{i-1}}|} \cdot \Pr(\mathcal{E}_n = E)$$

$$= \prod_{2 \leq i \leq n} \frac{1}{|\mathcal{C}_{t_{i-1}}|}.$$

where the second equality follows also from Equations 12 and 13 above. $\qquad\square$

We now show the invariant in the second part of π_{SA}.

Proof of Claim 5.3. We prove this by induction over i. The base case $i = i_F$ follows from Equation (10) and taking $W_i = I$.

Now, let us assume the claim is true for i and let us prove it for $i + 1$. Let us first consider the real world. We assume that $\mathrm{v}_{\mathrm{R},i} \approx_{\mathrm{neg}(\kappa)} W_i^{M\overline{T}_i}\left(\mathsf{CAuth.Enc}_{E_i}\left(\tau^{MR}\right)\right)$ and $\mathrm{v}_{\mathrm{I},i} \approx_{\mathrm{neg}(\kappa)}$ $W_i^{M\overline{T}_i}\left(\mathsf{CAuth.Enc}_{E_i}\left((|0^m\rangle\langle 0^m|) \otimes \tau^R\right)\right)$. Suppose Q_{i+1} is honest. Following the same analysis of Equation (10), it follows that the probability that Q_{i+1} sends Rej to cMPC is negligibly close to the

same probability that $\mathsf{Sim}_{\mathcal{A}}$ aborts. Therefore, by the security of Clifford codes, it follows that the output in the real world is

$$
\begin{aligned}
\mathrm{v}_{\mathrm{R},i+1} &\approx_{\mathrm{neg}(\kappa)} \mathsf{CAuth.Enc}_{E_{i+1}}(\mathsf{CAuth.Dec}_{E_i}(\mathrm{v}_{\mathrm{R},i}) \otimes |0^t\rangle\langle 0^t|) \\
&\approx_{\mathrm{neg}(\kappa)} q_{\mathsf{Acc},i+1} \cdot \mathsf{CAuth.Enc}_{E_{i+1}}(\mathscr{U}_{\mathsf{Acc},i+1}^R(\tau^{MR})) + q_{\mathsf{Rej},i+1} \cdot |\bot\rangle\langle\bot| \otimes \mathrm{tr}_M(\mathscr{U}_{\mathsf{Rej},i+1}^R(\tau^{MR})) \\
&= q_{\mathsf{Acc},i+1} \cdot \mathsf{CAuth.Enc}_{E_{i+1}}(\mathscr{U}_{\mathsf{Acc},i+1}^R(\tau^{MR})) + q_{\mathsf{Rej},i+1} \cdot |\bot\rangle\langle\bot| \otimes \mathscr{U}_{\mathsf{Rej},i+1}^R(\tau^{R}),
\end{aligned}
$$

where $\mathscr{U}_{\mathsf{Acc},i+1}$ and $\mathscr{U}_{\mathsf{Rej},i+1}$ are as in Definition 2.10, and where $q_{\mathsf{Acc},i+1} = 1 - q_{\mathsf{Rej},i+1}$ is the probability that Q_{i+1} sent Acc to cMPC. On the other hand, in the ideal world it holds that

$$
\begin{aligned}
\mathrm{v}_{\mathrm{I},i+1} &\approx_{\mathrm{neg}(\kappa)} \mathsf{CAuth.Enc}_{E_{i+1}}(\mathsf{CAuth.Dec}_{E_i}(\mathrm{v}_{\mathrm{I},i}) \otimes |0^t\rangle\langle 0^t|) \\
&\approx_{\mathrm{neg}(\kappa)} q_{\mathsf{Acc},i+1} \cdot \mathsf{CAuth.Enc}_{E_{i+1}}(\mathscr{U}_{\mathsf{Acc},i+1}^R(|0\rangle\langle 0|^M \otimes \tau^{R})) + q_{\mathsf{Rej},i+1} \cdot |\bot\rangle\langle\bot| \otimes \mathrm{tr}_M(\mathscr{U}_{\mathsf{Rej},i+1}^R(\tau^{MR})) \\
&= q_{\mathsf{Acc},i+1} \cdot \mathsf{CAuth.Enc}_{E_{i+1}}(\mathscr{U}_{\mathsf{Acc},i+1}^R(|0\rangle\langle 0|^M \otimes \tau^{R})) + q_{\mathsf{Rej},i+1} \cdot |\bot\rangle\langle\bot| \otimes \mathscr{U}_{\mathsf{Rej},i+1}^R(\tau^{R}).
\end{aligned}
$$

Thus, if the protocol aborts, then the ideal world simulator will abort as well and output negligibly close states. Otherwise, if the protocol continues, then we take $W_{i+1} = \mathscr{U}_{\mathsf{Acc},i+1}$.

Now suppose $\mathsf{Q}_{i+1} \in \mathcal{I}$ is corrupted. Let $A_{i+1}{}^{M\overline{T}_{i+1}T_{i+1}F}$ be the attack $\mathcal{A}$ instructs Q_{i+1} to apply after V_{i+1}. Upon applying these operations, in the real world $\mathcal{A}$ obtains

$$
A_{i+1}V_{i+1}\mathrm{v}_{\mathrm{R},i}(A_{i+1}V_{i+1})^\dagger \approx_{\mathrm{neg}(\kappa)} A_{i+1}V_{i+1}\left(W_i\left(\mathsf{CAuth.Enc}_{E_i}\left(\tau^{MR}\right)\right)\right)(A_{i+1}V_{i+1})^\dagger.
$$

Let $\mathcal{F}_{i+1}{}^{M\overline{T}_{i+1}T_{i+1}R}(\cdot) = A_{i+1}V_{i+1}\left(W_i\left(V_{i+1}^\dagger(\cdot)V_{i+1}\right)\right)V_{i+1}^\dagger A_{i+1}^\dagger$. Therefore, Q_{i+1} now holds

$$
\mathcal{F}_{i+1}\left(V_{i+1}\left(\mathsf{CAuth.Enc}_{E_i}\left(\tau^{MR}\right)\right)V_{i+1}^\dagger\right) = \mathcal{F}_{i+1}\left(\mathsf{CAuth.Enc}_{E_{i+1}}\left(\tau^{MR}\right) \otimes |0^t\rangle\langle 0^t|^{T_{i+1}}\right)
$$

Therefore, the contents of register T_{i+1} is negligibly close to $|0^t\rangle\langle 0^t|$ before $\mathcal{F}_{i+1}$ is applied. Moreover, as it is never used again, we can assume without loss of generality that $\mathcal{F}_{i+1}$ does not act on T_i. We can then therefore write $W_{i+1}(\cdot) = \langle\mathsf{Acc}^F|\mathcal{F}_{i+1}{}^{M\overline{T}_{i+1}R}(\cdot)|\mathsf{Acc}\rangle$ as the projection of $\mathcal{F}_{i+1}$ to acceptance. Similarly, we can do the same in the ideal world (by replacing the contents of register M with $|0\rangle\langle 0|$).

Clearly, by the security of Clifford code, the probability that $\mathcal{A}$ aborts is the same in both worlds. Therefore, as the Clifford key is erased, it follows that $\mathrm{v}_{\mathrm{R},i+1} \approx_{\mathrm{neg}(\kappa)} \mathrm{v}_{\mathrm{I},i+1}$ in case of abort. $\qquad\square$

5.2 Input-Ciphertext Sequential Authentication

We also make use of a variant of SA, where the input – instead of an arbitrary state – is encrypted under Clifford encryption, with the key being held by cMPC. We call this variant *input-ciphertext* SA (CTSA). The protocol follows the same lines as π_{SA}, with the exception that cMPC does not send to Q_1 the first Clifford. We stress that unlike π_{SA}, this protocol will *not* be secure.

Protocol 3 Input-Ciphertext Sequential Authentication π_{CTSA}

Inputs: The sender Q_1 holds an input $\sigma_1^{M\overline{T}_1}$ and cMPC holds a Clifford $E_1 \in \mathcal{C}_{m+(\ell+1)t}$.
Common input: All parties hold the security parameter 1^κ, a path $\mathtt{PATH}$ that goes through every

party, and the trap-size t.

1. For $i = 2$ to ℓ:

   (a) cMPC samples a Clifford $E_i \leftarrow \mathcal{C}_{m+t_i-t}$ and an invertible linear function $G_i \leftarrow \mathrm{GL}_{t_i}(\mathbb{F}_2)$ independently and uniformly at random. It sets

   $$V_i := \left(E_i^{M\overline{T}_i} \otimes I^{\otimes t^{T_i}} \right) \left(I^{\otimes m^M} \otimes G_i^{\overline{T}_i T_i} \right) {E_{i-1}^\dagger}^{M\overline{T}_i T_i}$$

   and sends it to Q_i.

   (b) Q_i evaluates $\tau_i^{M\overline{T}_i T_i} = V_i \sigma_{i-1} V_i^\dagger$ and measures register T_i in the computational basis. If the measurement result is 0^t then Q_i sends Acc to cMPC, otherwise it sends Rej to cMPC. Let $\sigma_i^{M\overline{T}_i}$ denote the post-measurement state.

   (c) If cMPC received Rej, then sends $(\mathtt{abort}, \mathsf{Q}_{i-1}, \mathsf{Q}_i)$ to all parties. They then output this message and halt.

   (d) If $i < \ell$, Q_i sends σ_i to the next relay Q_{i+1}.

2. The receiver Q_ℓ outputs σ_ℓ, cMPC outputs E_ℓ, and the other parties output nothing.

Although we cannot argue for the security of the protocol, we can translate an attacker of π_{CTSA} to an attacker of π_{SA}.

Claim 5.6. *For every non-uniform* QPT *adversary* $\mathcal{A}$*, controlling a set* $\mathcal{I} \subset \mathcal{P}$ *in protocol* π_{CTSA}*, there exists a non-uniform* QPT *adversary* $\mathcal{A}_{\mathsf{SA}}$*, controlling* $\mathcal{I}$ *in the protocol* π_{SA}*, such that*

$$\left\{ \mathrm{HYBRID}_{\pi_{\mathsf{SA}}, \mathcal{A}_{\mathsf{SA}}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \right\}_{\kappa, t, \mathtt{PATH}, \rho, \rho_{\mathsf{aux}}} = \left\{ \mathrm{HYBRID}_{\pi_{\mathsf{CTSA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \right\}_{\kappa, t, \mathtt{PATH}, \rho, \rho_{\mathsf{aux}}}.$$

In particular, taking $\mathcal{I} = \emptyset$ *results in the receiver* Q_ℓ *outputting* $\mathsf{CAuth.Enc}_{E_\ell}(\rho)$*, and* cMPC *outputting* E_ℓ*,* $\rho = \mathsf{CAuth.Dec}_{E_1}(\sigma_1)$ *and* $E_\ell \in \mathcal{C}_m$

Proof. Fix an attacker $\mathcal{A}$ for π_{CTSA}. If $\mathsf{Q}_1 \in \mathcal{I}$, then the adversary $\mathcal{A}_{\mathsf{SA}}$ will simply interact in π_{SA} like $\mathcal{A}$ would on input $\mathsf{CAuth.Enc}_{E_1}(\rho)$, where E_1 is the Clifford received from cMPC at the start of the protocol. Otherwise, if Q_1 is honest, then $\mathcal{A}_{\mathsf{SA}}$ will act like $\mathcal{A}$. Finally, $\mathcal{A}_{\mathsf{SA}}$ outputs whatever $\mathcal{A}$ outputs. Clearly, the joint state held by $\mathcal{A}$ and the honest parties in π_{CTSA} is the same as the joint state held by $\mathcal{A}_{\mathsf{SA}}$ and the honest parties in π_{SA}. $\square$

We thus have the following simple corollary asserting that any adversary attacking π_{CTSA} can be simulated in the SA ideal world. Using the construction of our simulation of SA, it further holds that it does not change its input it sends to the trusted party.

Corollary 5.7. *For every non-uniform* QPT *adversary* $\mathcal{A}$*, controlling a set* $\mathcal{I} \subset \mathcal{P}$ *in protocol* π_{CTSA}*, there exists a non-uniform* QPT *adversary* $\mathsf{Sim}_{\mathcal{A}}$*, controlling* $\mathcal{I}$ *in the ideal-world for* SA*, such that*

$$\left\{ \mathsf{SA}_{\mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})} (\kappa, \mathtt{PATH}, t, \rho) \right\}_{\kappa, t, \mathtt{PATH}, \rho, \rho_{\mathsf{aux}}} \approx_{\mathrm{neg}(\kappa)} \left\{ \mathrm{HYBRID}_{\pi_{\mathsf{CTSA}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}} (\kappa, \mathtt{PATH}, t, \rho) \right\}_{\kappa, t, \mathtt{PATH}, \rho, \rho_{\mathsf{aux}}}.$$

Moreover, $\mathsf{Sim}_{\mathcal{A}}$ *does not change the input it sends to its trusted party.*

6 Authenticated Routing

In this section, we present a protocol, called *authenticated routing* (AR), that allows the parties to securely send an encryption of their inputs to a designated party. The security achieved by this protocol is *IDPD-security* (i.e., security-with-identifiable-abort-and-packet-drops), as was defined in Section 3.1. We extensively use AR as a building block in order to construct a secure-with-identifiable-abort protocol for a general circuit.

We next define the AR functionality. For a polynomial $t = t(\kappa) \geq \kappa$, representing *trap-size*, denote by $\mathsf{AR} = \mathsf{AR}_t$ the following mapping. Each party P_i holds ℓ_{in} packets $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, where each $\rho_{ij} \in \mathcal{D}^m$. An output is given only to P_1 – called the receiver – and to cMPC. Specifically, cMPC receives a collection of Cliffords $(E_{ij})_{i \in [n], j \in [\ell_{\mathsf{in}}]}$, where $E_{ij} \leftarrow \mathcal{C}_{m+t}$ are sampled independently and uniformly at random, and the receiver P_1 receives the Clifford encoding of each packet ρ_{ij} under E_{ij}; that is, P_1 receives $(\mathsf{CAuth}.\mathsf{Enc}_{E_{ij}}(\rho_{ij}))_{i \in [n], j \in [\ell_{\mathsf{in}}]}$. In the following section we present a protocol that computes AR with $(n^2, 0)$-IDPD-security in the cMPC-hybrid model, i.e., at most n^2 input-packets can be dropped by the adversary while no output-packets can be dropped.

6.1 The Authenticated Routing Protocol

In this section, we present our protocol for authenticated routing. Roughly, the idea is as follows. Throughout the entire interaction, we let cMPC maintain a graph G that represents *trustfulness*. In more details, each vertex in G corresponds to a party and the graph is initialized as the complete graph. Then, whenever a party accuses another party, the corresponding edge will be removed from G.[9] Following the initialization, each party P_i tries to send its packets (ρ_{ij}), one by one, to P_1 along a path that goes through all parties. Such a path can be computed, and thus agreed upon, by having cMPC repeatedly applying BFS/DFS to find a path from P_i to P_2, then to P_3 until it reaches the last party P_n, from which it finds a path to P_1. The parties will send the packets along the path using the SA functionality.[10] If SA aborted, then the parties now hold two identities P_a and P_b given to them by SA, one of which is guaranteed to be corrupted. cMPC will then remove the edge ab from the graph G. Now, party P_i will try to send the rest of its packets using a *different* route that does not pass through the edge ab. The parties continue in this fashion until either most qubits were sent successfully, or until G becomes disconnected, in which case all honest parties are in the same connected component. Therefore, they can agree to identify a party *not* connected to them as malicious.

Let us now consider the case where a call SA ended in abort. Here, a single packet had been dropped, and so by the ideal-world definition of IDPD-security, the parties must agree to replace this packet with the 0 state. To do this, the parties will call SA again with the *empty state* $\varepsilon \in \mathcal{D}^0$ and $m + \kappa$ traps. To see why this work, notice that the Clifford encoding of the empty state with $m + \kappa$ traps, is equivalent to a Clifford encoding of $|0^m\rangle$ with κ traps. Moreover, by the security of Clifford encoding, if the adversary changes the traps from 0 then SA will abort again, which will remove another edge from the graph. Thus, this can be done repeatedly until either G becomes disconnected or the parties successfully encode the 0 state.

The IDPD-security of the protocol described so far can still be breached by a malicious adversary, due to the following difficulty one would encounter while constructing a simulator. Suppose that the adversary corrupted the receiver P_1, and consider a call to SA, for a packet $(1, j)$ for some $j \in [\ell_{\mathsf{in}}]$,

---

[9]We note that this technique, of using the graph to allow honest parties to unanimously agree on the identity of a corrupted party, was independently used in another recent paper by [BMMMQ20].

[10]Recall that SA requires the path to be of length n^2. Note that the way cMPC computes the path always generates a path of length at most n^2. If the path is shorter, then cMPC can just add the last party repeatedly.

i.e., the receiver sends to itself an encoding of the packet. To generate the corresponding transcript, the simulator in the ideal-world must query the adversary for its input ρ to the SA functionality and must send to $\mathcal{A}$ an output in return. Observe that $\mathcal{A}$ expects to receive $\sigma = \mathsf{CAuth.Enc}_E(\rho)$ where the key E is held by the cMPC. Since the simulator does not know the key E, it cannot generate a-priori a value σ that is consistent with the output of cMPC. On the other hand, the simulator may not be ready to send inputs to the trusted party either, as it must hold *all* input packets from the adversary. Since rewinding the adversary can help the environment to distinguish between the real world and ideal world, it seems to be the case where there are no good ways to generate this output of SA. One possible solution is to modify the AR ideal functionality so it will immediately encode and output each received packet before receiving the next packet. Unfortunately, the resulting ideal functionality would be too weak for our purposes later.

It is possible to overcome this challenge by tweaking the protocol. A simple solution to this issue would be to have the receiver, after it has received all of the packets, to send them to itself again using CTSA (that is, the ciphertext-input version of SAfrom Section 5.2). With this modification, the simulator can send authentication of 0s to the receiver as the outputs of SA. To see why this works, recall that Clifford authentication ciphertexts are identical to maximally mixed states due to Lemma 2.3 (Clifford twirling). Thus, the simulator can then collect all input packets and interact with the trusted party to receive the correct outputs. When the receiver sends the dummy ciphertexts to itself, the simulator collects and verifies them, before replacing them with the correct outputs.

Although this approach works, as CTSA is not secure as a standalone protocol, it hard to formally argue the security of the above protocol. Instead, we slightly modify the protocol and also construct a slightly different simulator. The idea is to have the simulator send halves of EPR pairs as the output of SA, instead of authentications of 0s. Since halves of EPR pairs are indistinguishable from Clifford authentication ciphertexts, the adversary will reply with the same messages in both the real and the ideal world. After the simulator collects all packets and interact with the trusted party, it then replaces these halves of EPR pairs with the correct output via quantum teleportation (see Figure 4). To complete the teleportation, the simulator must send a Pauli operator for the adversary to apply. Thus, we correspondingly add a key-update step at the end of the protocol to provide an opportunity for this.

Protocol 4 Authenticated Routing protocol π_{AR}

Inputs: Each party P_i holds private input $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$ where $\rho_{ij} \in \mathcal{D}^m$.
Common input: The parties hold thesecurity parameter 1^κ and the packet-size m.

1. cMPC initializes G as the complete graph with n vertices where each vertex represents a party, and initializes a set $\mathcal{R}_{\mathsf{in}} = \emptyset$, which will keep track of all packets that were sent successfully.

2. For each packet $(i, j) \in [n] \times [\ell_{\mathsf{in}}]$:

   (a) cMPC computes a path $\mathtt{PATH}_{ij}$ in G from P_i to P_1 that goes through all parties of size exactly n^2, and send it to all parties.

   (b) The parties call SA with P_i's input being ρ_{ij}, with κ as the common trap-size, and $\mathtt{PATH}$ as the common path.

      - If SA outputs $(\mathtt{abort}, \mathrm{P}_a, \mathrm{P}_b)$ for some $a, b \in [n]$, then cMPC removes the edge ab from the graph G. If G becomes disconnected, then cMPC sends ab to all parties. Each party then outputs $(\mathtt{abort}, \mathrm{P})$, where $\mathrm{P} \in \{\mathrm{P}_a, \mathrm{P}_b\}$ is the party on the edge *not* connected to it on the graph, and halts.

- Otherwise SA terminates successfully, sending a uniform random Clifford $F_{ij} \leftarrow \mathcal{C}_{m+\kappa}$ to cMPC, sending σ_{ij} to P_1, where $\sigma_{ij} = \mathsf{CAuth.Enc}_{F_{ij}}(\rho_{ij})$, and sending continue to all other parties. In this case, cMPC adds (i,j) to $\mathcal{R}_{\mathsf{in}}$.

3. For each dropped packet $(i,j) \in ([n] \times [\ell_{\mathsf{in}}]) \setminus \mathcal{R}_{\mathsf{in}}$:

   (a) cMPC finds a path $\mathtt{PATH}'_{ij}$ in G from P_1 to itself that passes through every party and send it to all parties.

   (b) The parties call SA, trap-size $m+\kappa$, no private inputs, and $\mathtt{PATH}'_{ij}$ as the common input.
      - If SA outputs $(\mathtt{abort}, P_a, P_b)$ for some $a, b \in [n]$, then, similarly to Step 2b, cMPC removes the edge ab from the graph G. If G becomes disconnected, then cMPC sends ab to all parties. Each party P then outputs $(\mathtt{abort}, P')$, where $P' \in \{P_a, P_b\}$ is the party on the edge *not* connected to P, and halt. Otherwise, if the graph is still connected, then the parties go back to Step 3a.
      - Otherwise SA terminates successfully, sending a uniform random Clifford $F_{ij} \leftarrow \mathcal{C}_{m+\kappa}$ to cMPC, sending σ_{ij} to P_1, where $\sigma_{ij} \leftarrow \mathsf{CAuth.Enc}_{F_{ij}}(\varepsilon)$, and sending continue to all other parties.

4. For all packets $(i,j) \in [n] \times [\ell_{\mathsf{in}}]$:

   (a) cMPC samples a Pauli independently and uniformly at random $P_{ij} \leftarrow \mathcal{P}_{m+\kappa}$ and sends it to P_1.

   (b) P_1 applies P_{ij} to σ_{ij}, obtaining τ_{ij}.

   (c) cMPC updates its Clifford key to be $E_{ij} = P_{ij} F_{ij}$.

5. cMPC sends $\mathcal{R}_{\mathsf{in}}$ to all parties.

6. Each party outputs $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}})$, cMPC additionally outputs the Cliffords $\{E_{ij}\}_{ij \in S_{\mathsf{input}}}$, and P_1 additionally outputs $\{\tau_{ij}\}_{ij \in S_{\mathsf{input}}}$.

We next state and prove the security of the protocol.

Lemma 6.1. *Protocol π_{AR} computes the functionality* AR *with perfect $(n^2, 0)$-IDPD-security in the* $\{\mathsf{cMPC}, \mathsf{SA}\}$*-hybrid model.*

Proof. Fix an adversary $\mathcal{A}$ corrupting a subset $\mathcal{I}$ of the parties holding auxiliary input ρ_{aux}. We next construct a simulator $\mathsf{Sim}_{\mathcal{A}}$ for the adversary in the $(n^2, 0)$-IDPD ideal world.
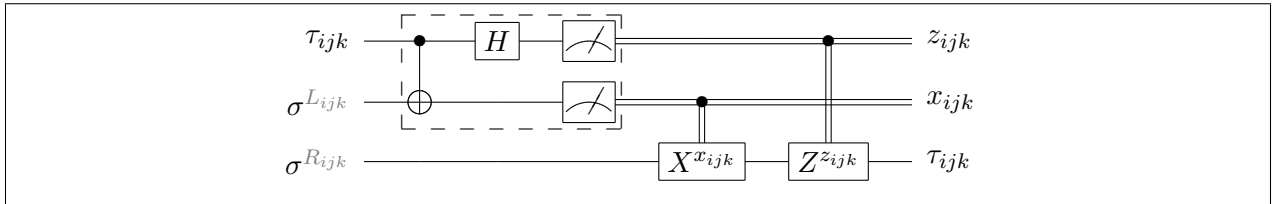
Figure 4: Teleportation-inspired circuit for output equivocation

---

Simulator 3 Simulator $\mathsf{Sim}_{\mathcal{A}}$ for authenticated routing

1. Initialize G as the complete graph and $\mathcal{R}_{\mathsf{in}} = \emptyset$ as an empty set.

2. For all $(i, j) \in S_{\mathsf{input}}$:

   (a) Find the path $\mathtt{PATH}_{ij}$ in G from P_i to P_1 that passes through every party, computed the same as cMPC computed and send it to $\mathcal{A}$.

   (b) If $\mathrm{P}_i \in \mathcal{I}$ is corrupted, then query $\mathcal{A}$ for its input ρ_{ij} to SA.

   (c) If $\mathrm{P}_1 \in \mathcal{I}$ is corrupted, then prepare $m + \kappa$ EPR pairs, $\left(\sigma^{L_{ijk} R_{ijk}}\right)_{k=1}^{m+\kappa} = |\Phi^+\rangle^{\otimes m+\kappa}$, and send $\left(\sigma^{R_{ijk}}\right)_{k=1}^{m+\kappa}$ (that is, half of each EPR pair) to $\mathcal{A}$ as the output of SA.

   (d) The adversary sends back either $\mathtt{continue}$ or $(\mathtt{abort}, (\mathrm{P}_a, \mathrm{P}_b))$. If it sent $\mathtt{continue}$, then add (i, j) to $\mathcal{R}_{\mathsf{in}}$. Otherwise, if it sent $(\mathtt{abort}, (\mathrm{P}_a, \mathrm{P}_b))$, then remove the edge $e = ab$ from the graph G. If G becomes disconnected, send $(\mathtt{abort}, \mathrm{P}')$ to T where $\mathrm{P}' \in \{\mathrm{P}_a, \mathrm{P}_b\}$ is the party on e not connected to any of the honest parties,[11] output whatever $\mathcal{A}$ outputs, and halt.

3. For all dropped packets $(i, j) \in S_{\mathsf{input}} \setminus \mathcal{R}_{\mathsf{in}}$:

   (a) Find a path $\mathtt{PATH}'_{ij}$ in G from the receiver P_1 to itself that passes through all parties, and send it and the trap-size $m + \kappa$ to the adversary as a message from SA.

   (b) If $\mathrm{P}_1 \in \mathcal{I}$ is corrupted, then update $\left(\sigma^{L_{ijk} R_{ijk}}\right)_{k=1}^{m+\kappa} = |\Phi^+\rangle^{\otimes m+\kappa}$ as fresh EPR pairs, and send $\left(\sigma^{R_{ijk}}\right)_{k=1}^{m+\kappa}$ to $\mathcal{A}$ as the output of SA.

   (c) The adversary sends back either $\mathtt{continue}$ or $(\mathtt{abort}, (\mathrm{P}_a, \mathrm{P}_b))$ where either $\mathrm{P}_a \in \mathcal{I}$ or $\mathrm{P}_b \in \mathcal{I}$. If it sent $\mathtt{continue}$, then add (i, j) to $\mathcal{R}_{\mathsf{in}}$. Otherwise, if it sent $(\mathtt{abort}, (\mathrm{P}_a, \mathrm{P}_b))$, then remove the edge $e = ab$ from the graph G. If G becomes disconnected, send $(\mathtt{abort}, \mathrm{P}')$ to T where $\mathrm{P}' \in \{\mathrm{P}_a, \mathrm{P}_b\}$ is the party on e not connected to any of the honest parties, outputs whatever $\mathcal{A}$ outputs, and halt. Otherwise, if G is still connected, go back to Step 3a with the same packet (i, j).

4. Send $\left((\rho_{ij})_{i \in \mathcal{I}, j \in [\ell_i]}, \mathcal{R}_{\mathsf{in}}\right)$ to the trusted party T.

5. If P_1 is corrupted, then T sends back to the simulator the ciphertexts $(\tau_{ij})_{(i,j) \in S_{\mathsf{input}}}$. Rewrite each τ_{ij} qubit-by-qubit, i.e., as $(\tau_{ijk})_{(i,j,k) \in S_{\mathsf{input}} \times [m+\kappa]}$. For all $(i, j) \in S_{\mathsf{input}}$ and $k \in [m + \kappa]$, run the portion of Figure 4 in the dashed lines, generating classical bits x_{ijk} and z_{ijk}. Send $P_{ij} = \bigotimes_{k=1}^{m+\kappa} Z^{z_{ijk}} X^{x_{ijk}}$ for all $(i, j) \in S_{\mathsf{input}}$ to $\mathcal{A}$.

6. Send $\mathtt{continue}$ to T.

7. Send $\mathcal{R}_{\mathsf{in}}$ to $\mathcal{A}$, output whatever it outputs, and halt.

We now analyze the above simulator, showing that it perfectly simulate the real world. That is, we show that

$$\mathrm{IDEAL}^{(n^2, 0)\text{-}\mathsf{IDPD}}_{\mathrm{AR}, \mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})}\left(\kappa, (\rho_i)_{i=1}^n\right) = \mathrm{HYBRID}^{\mathsf{cMPC}, \mathsf{SA}}_{\pi_{\mathrm{AR}}, \mathcal{A}(\rho_{\mathsf{aux}})}\left(\kappa, (\rho_i)_{i=1}^n\right).$$

[11]Formally, the simulator should first send inputs to the trusted party. For simplicity, however, we ignore such technicalities, as the simulator can first send arbitrary inputs and then abort.

First, we show that for all $(i,j) \in S_{\mathsf{input}}$ in Step 2 in the real world and Step 2 in the ideal world, the view of $\mathcal{A}$ are identical. Clearly, $\mathtt{PATH}_{ij}$ is the same in both worlds. Moreover, if P_1 is corrupted, the adversary receives a Clifford authentication ciphertext in the real world, and halves of EPR pairs in the ideal world, both of which are maximally mixed state (see Lemma 2.3). Thus, the adversary's answer to SA is identical in both worlds, hence the graph G and the set $\mathcal{R}_{\mathsf{in}}$ of remaining packets are updated the same as well. In particular, if the protocol aborts in this step, then the adversary has the same view in both worlds. Furthermore, the honest parties output the same identity of a corrupted party due to them forming a clique in G. Similarly, assuming the adversary did not abort during Step 2 of the protocol, the view of the adversary during Step 3 of the protocol, is identical to the view generated by the simulator.

Now we assume the protocol has not aborted in the previous steps. Observe that the protocol, by construction, does not have any more chances to abort. Let us first consider the case where the receiver P_1 is honest. Here, the adversary does not receive anymore messages. In the ideal world, the honest receiver receives ciphertexts $\mathsf{CAuth.Enc}_{E_{ij}}(\rho_{ij})$ from the trusted party and cMPC receives the corresponding keys E_{ij}. In the real world, $\mathsf{CAuth.Enc}_{F_{ij}}(\rho_{ij})$ and F_{ij} which has the same distribution is given to them by SA, then F_{ij} is shifted by a Pauli P_{ij} to generate E_{ij}. Since this uniformly random Pauli shift has no effect on the distribution, the real and ideal worlds are identical.

Now, let us consider the case where the receiver $\mathrm{P}_1 \in \mathcal{I}$ is corrupted. Here, the dropped packets are replaced with Clifford encoding of 0's in both worlds. Furthermore, in both worlds, each survived packets ρ_{ij} can be written as $\sigma_{ij} = P_{ij}^{\dagger} \tau_{ij} P_{ij}$, where P_{ij} is the Pauli sent to $\mathcal{A}$, and $\tau_{ij} = \mathsf{CAuth.Enc}_{F_{ij}}(\rho_{ij})$ and F_{ij} is the encoded packet. Furthermore, by the properties of quantum teleportation, each P_{ij} is also uniformly random in the ideal world. Thus, joint output of the adversary and cMPC in the real-world is identical to their output in the ideal-world.

Finally, as the trustfulness graph G will become disconnected after at most n^2 edge removal, it follows that the number of remaining input-qubits is $|\mathcal{R}_{\mathsf{in}}| \geq n \cdot \ell_{\mathsf{in}} - n^2$, hence

$$\mathrm{IDEAL}_{\mathsf{AR},\mathsf{Sim}_{\mathcal{A}(\rho_{\mathsf{aux}})}}^{(n^2,0)\text{-}\mathsf{IDPD}} \left(\kappa, (\rho_i)_{i=1}^n\right) = \mathrm{HYBRID}_{\pi_{\mathsf{AR}},\mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC},\mathsf{SA}} \left(\kappa, (\rho_i)_{i=1}^n\right).$$

$\square$

6.2 Authenticated Routing With Input-Ciphertext

We also make use of the *input-ciphertext* variant of AR, denoted CTAR. Similarly to CTSA, this is also insecure in general, so we only describe the protocol. Here, unlike in AR, there is only a single sender, denoted P_1, and multiple receivers. The goal variant is to send each *plaintext* from the P_1 to its designated receiver, encrypted under a new key.

We now describe the protocol. It is similar to π_{AR}, with the following exceptions. First, the dropped packets are not replaced with 0's. Second, the calls to SA are replaced with executions of to π_{CTSA}. Third, there is no need to update the keys held by cMPC at the end. Fourth, the parties hold the size of the plaintexts, denoted m, and the size of traps used in the Cliffords encryption, denoted t, as common input. Note that these might depend on the security parameter κ. Finally, since the ciphertexts may be transmitted through n^2 relays, the initial ciphertexts should have the traps of size tn^2.

Protocol 5 Protocol π_{CTAR} for input-ciphertext authenticated routing

Inputs: The sender P_1 has a set of Clifford ciphertexts $(\sigma_{ij})_{i\in[n],j\in[\ell_{in}]}$, where cMPC holds the corresponding Clifford keys $(E_{ij})_{i\in[n],j\in[\ell_{in}]}$, where each $E_{ij} \in \mathcal{C}_{m+tn^2}$.
Common input: The parties hold the security parameter 1^κ, the size of the plaintexts m, and the size of the traps t.

1. cMPC initializes G as the complete graph with n vertices where each vertex represents a party, and initializes a set $\mathcal{R}_{in} = \emptyset$, which will keep track of all packets that were sent successfully.

2. For each packet $(i,j) \in [n] \times [\ell_{in}]$:

   (a) cMPC computes a path $\texttt{PATH}_{ij}$ of the length n^2 in G from P_1 to P_i that goes through all parties, and send it to all parties.

   (b) The parties execute π_{CTSA} with P_1's input being the set of Clifford ciphertext σ_{ij} and cMPC's input being its corresponding Clifford key E_{ij}, and $\texttt{PATH}_{ij}$ and trap size t as the common input.

      - If the output of π_{CTSA} is $(\texttt{abort}, P_a, P_b)$ for some $a, b \in [n]$,[12] then cMPC removes the edge ab from G. If G becomes disconnected, then cMPC sends ab to all parties. Each party then outputs $(\texttt{abort}, P)$, where $P \in \{P_a, P_b\}$ is the party on the edge not connected to it.

      - Otherwise, π_{CTSA} terminates successfully, sending to P_i the ciphertext $\hat{\sigma}_{ij} \leftarrow \mathsf{CAuth.Enc}_{F_{ij}}(\rho_{ij})$, and sending to cMPC the corresponding key $F_{ij} \in \mathcal{C}_{m+t}$. In this case, cMPC adds (i,j) to $\mathcal{R}_{in}$.

3. Each party P_i outputs $(\texttt{continue}, \mathcal{R}_{in})$ and $(\hat{\sigma}_{ij})_{j:(i,j)\in\mathcal{R}_{in}}$, and cMPC additionally outputs the Cliffords $(F_{ij})_{(i,j)\in\mathcal{R}_{in}}$.

By Corollary 5.7, in an honest execution of the above protocol outputs the correct output. That is, we have the following claim.

Claim 6.2. *For all $i \in [n]$ and $j \in [\ell_{in}]$ let $\rho_{ij} = \mathsf{CAuth.Dec}_{E_{ij}}(\sigma_{ij})$. Then in an honest execution of π_{CTAR}, for all $i \in [n]$ the output of P_i is $(\mathsf{CAuth.Enc}_{F_{ij}}(\rho_{ij}))_{j\in[\ell_{in}]}$, and the output of cMPC is $(F_{ij})_{i\in[n],j\in[\ell_{in}]}$.*

7 Magic State Preparation

Recall that we aim to have a single designated party to homomorphically evaluate a universal circuit over encrypted values. Towards achieving this, the parties require five kinds of magic states. These include ancilla zero states, P magic states, T magic states, H magic state and gadgets γ. In this section, we show how the parties can prepare these states using a secure-with-identifiable-abort protocol. In all magic state preparation protocols, an output will be given to only two parties: the server P_1 and cMPC. In Section 7.1 we show how to prepare ancilla zero state and P magic state, in Section 7.2 we show how prepare H magic state, in Section 7.3 we show how to prepare gadget states preparation, and in Section 7.4 we show how to prepare T magic states. To simplify the presentation, all protocols will prepare a single magic state. These, however, can be easily generalized to create more magic states using the same number of rounds.

---

[12]Recall that by Corollary 5.7 all parties will agree on the identities in case of abort, one of which is guaranteed to be corrupted.

7.1 Preparing Ancilla and P Magic States

In this section we show how to prepare ancilla zero states and P magic states. Let us first define the mappings. We denote by MSP_Z the following functionality. It is a no-input functionality that outputs to P_1 the Clifford encoding $\mathsf{CAuth.Enc}_E(|0\rangle\langle 0|)$ and outputs to cMPC the corresponding key E, where $E \leftarrow \mathcal{C}_{3\kappa}$ is uniform random Clifford. Then the mapping outputs to P_1 an encryption of the zero state $\mathsf{CAuth.Enc}_E(|0\rangle\langle 0|)$ and outputs to cMPC the corresponding key E. Similarly, We denote by MSP_P the mapping that outputs to P_1 a Clifford encoding of $|P\rangle = P|+\rangle$.

We next present the protocols for computing MSP_Z with security-with-identifiable-abort. The idea is as follows. First, the parties call the functionality AR *without* any input, with trap size 3κ, and with the P_1 as the receiver. Syntatically, AR takes as input a list of "packets," where n^2 of them may be dropped. Thus, the parties actually call AR with $n^2 + 1$ packets, where each packets consists of no input state with trap size 3κ. Then, the server and cMPC keep the lexicographically first remaining packet. This result in P_1 holding $E(\varepsilon \otimes |0^{3\kappa}\rangle\langle 0^{3\kappa}|)E^\dagger$, and cMPC holding the random Clifford E. The correctness immediately follows from the fact that

$$E(\varepsilon \otimes |0^{3\kappa}\rangle\langle 0^{3\kappa}|)E^\dagger = E(|0\rangle\langle 0| \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|)E^\dagger.$$

To generate multiple magic states at once, the parties can call AR with more packets initially. In the following, we omit the number of initial packets and only focus on preparing single magic state each time for simplicity.

To see why the protocol admits security-with-identifiable-abort, observe that there is no interaction, and moreover, there are no inputs to AR hence it either aborts or send to P_1 and cMPC their outputs.

Now, to see how the parties can compute MSP_P, observe that if P_1 decrypts with the key EPH instead of E, where PH is applied only to the first bit, then the result would be $|P\rangle$. Indeed,

$$(EPH)(|0\rangle\langle 0| \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|)(EPH)^\dagger = E(|P\rangle\langle P| \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|)E^\dagger.$$

Thus, we let cMPC output EPH instead of the key given to it by AR. We next formally describe both protocols. We denote by π_{ms}, where $\mathsf{ms} \in \{Z, P\}$, the protocol that computes $\mathsf{MSP}_{\mathsf{ms}}$.

Protocol 6 Protocol π_{ms} for preparing ancilla and P magic states

Common input: The parties hold the security parameter 1^κ.

1. The parties call $(\mathsf{AR}, (n^2, 0))$-IDPD without inputs, P_1 as the receiver, and trap-size 3κ. If AR aborts, then the parties have the identity of a corrupted party, they output it, and halt.

2. Otherwise, cMPC receives from AR a Clifford $E \leftarrow \mathcal{C}_{3\kappa}$ sampled uniformly at random, and the server P_1 receives $\sigma := \mathsf{CAuth.Enc}_E(\varepsilon)$.

3. The server P_1 outputs σ. If $\mathsf{ms} = Z$, cMPC outputs E; if $\mathsf{ms} = P$, it outputs EPH. The other parties output nothing.

The following claim asserts the security the protocol

Claim 7.1. *For all* $\mathsf{ms} \in \{Z, P\}$ *it holds that protocol* π_{ms} *computes the functionality* $\mathsf{MSP}_{\mathsf{ms}}$ *with perfect security-with-identifiable-abort in* $\{\mathsf{cMPC}, \mathsf{AR}\}$*-hybrid model.*

7.2 Preparing H Magic States

We next show how to prepare H magic states. Recall that $|H\rangle = (H \otimes I)\mathsf{CNOT}(|+\rangle \otimes |0\rangle)$ is a two-qubit system. Define the functionality MSP_H as follows. This is a no-input functionality whose that outputs to P_1 a Clifford encoding for qubit in H, i.e., $(\mathsf{CAuth.Enc}_{E_1} \otimes \mathsf{CAuth.Enc}_{E_2})(|H\rangle\langle H|)$, and outputs to cMPC the corresponding keys E_1 and E_2, where $E_1, E_2 \leftarrow \mathcal{C}_{3\kappa}$ are sampled independently and uniformly at random.

We now present our protocol for computing MSP_H. Roughly, the protocol starts the same as π_Z and π_P; that is, the parties call AR with no inputs. Unlike in the previous protocol, as H is a two-state qubit, it is not enough to have cMPC simply update the Clifford it received from AR. Instead, we let cMPC send to the server P_1 a carefully chosen Clifford for it to apply over the state it received from AR.

Protocol 7 Protocol π_H for preparing H magic states

Common input: The parties hold the security parameter 1^κ.

1. The parties call $(\mathsf{AR}, (n^2, 0))$-IDPD without inputs, P_1 as the receiver, and trap-size 6κ. If AR aborts, the parties have the identity of a corrupted party. They output it and halt.

2. Otherwise, cMPC receives from AR a Clifford $E \leftarrow \mathcal{C}_{6\kappa}$ sampled uniformly at random, and the server P_1 receives

$$
\begin{aligned}
\sigma &= \mathsf{CAuth.Enc}_E\left(\varepsilon\right) \\
&= E(\varepsilon \otimes |0^{6\kappa}\rangle\langle 0^{6\kappa}|)E^\dagger \\
&= E(|0\rangle\langle 0|^{M_1} \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|^{T_1} \otimes |0\rangle\langle 0|^{M_2} \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|^{T_2})E^\dagger.
\end{aligned}
$$

3. cMPC sets $U^{M_1 M_2} = (H \otimes I)\mathsf{CNOT}(H \otimes I)$. Additionally, cMPC samples two random Cliffords $E_1, E_2 \leftarrow \mathcal{C}_{3\kappa}$ independently and uniformly at random.

4. Then, cMPC sends to P_1 the Clifford $V = (E_1^{M_1 T_1} \otimes E_2^{M_2 T_2})UE^\dagger$.

5. The server outputs $V\sigma V^\dagger$, cMPC outputs (E_1, E_2), and the other parties outputs nothing.

We next state and prove the security of the protocol.

Claim 7.2. *Protocol π_H computes the functionality MSP_H with perfect security-with-identifiable-abort in $\{\mathsf{cMPC}, \mathsf{AR}\}$-hybrid model.*

Proof. To see correctness, observe that $U(|0\rangle \otimes |0\rangle) = |H\rangle$, so we have

$$
\begin{aligned}
&V\sigma V^\dagger \\
&= \left((E_1^{M_1 T_1} \otimes E_2^{M_2 T_2})U^{M_1 M_2}E^\dagger\right)\left(E\left(|0\rangle\langle 0|^{M_1} \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|^{T_1} \otimes |0\rangle\langle 0|^{M_2} \otimes |0^{3\kappa-1}\rangle\langle 0^{3\kappa-1}|^{T_2}\right)E^\dagger\right) \\
&\qquad ((E_1 \otimes E_2)UE^\dagger)^\dagger \\
&= (\mathsf{CAuth.Enc}_{E_1} \otimes \mathsf{CAuth.Enc}_{E_2})(|H\rangle\langle H|).
\end{aligned}
$$

As for security, notice that the only message sent in the protocol is the Clifford V sent from cMPC to P_1. As V is uniform random Clifford, security follows. $\qquad\square$

7.3 Preparing Gadget Magic States

We now show how to prepare a gadget magic state γ, which is a multi-qubit state defined below.

Notations. Before formally introducing our protocol, we first present some notations. Let $\ell_{\sf gad}$ be the number of qubits in a single gadget state. Recall that in VQFHE scheme TrapTP, the key generation algorithm TrapTP.KeyGen outputs a classical string sk viewed as a secret key. This secret key includes the secret key of a classical message authentication code MAC, the public key and secret key of classical homomorphic encryption scheme HE, a global permutation Π_0 for the trap-code encryptions, and local permutations $(\Pi_d)_{d=1}^L$ for each gadget, where L will represent a bound on the circuit size. Toward creating the gadget states, a classical algorithm $M^{\sf Gad}({\sf sk}, d)$ that generates all the classical description of the garden-hose gadget is executed. It is given an secret key sk and a permutation index d, and outputs the classical description g_d of $d^{\rm th}$ gadget and a Clifford operator U_d used for creating the gadget, defined to be $|\gamma_d\rangle\langle\gamma_d| := U_d |0^{\ell_{\sf gad}}\rangle\langle 0^{\ell_{\sf gad}}| U_d^\dagger$. We refer the reader to [ADSS17] for a detailed construction of each classical algorithm.

Our construction. Let us now present the definition of the functionality MSP_γ for preparing gadget states. The parties P_i hold no input, and cMPC is given the secret key sk and the permutation index d. The output given to the server is Clifford encryptions of each qubit in $d^{\rm th}$ gadget; that is, it receives $(\otimes_{j=1}^{\ell_{\sf gad}} \mathsf{CAuth.Enc}_{E_j})(|\gamma_d\rangle\langle\gamma_d|)$ where $E_j \leftarrow \mathcal{C}_{3\kappa}$ for all $j \in [\ell_{\sf gad}]$, and cMPC is given the Clifford keys $(E_j)_{j=1}^{\ell_{\sf gad}}$ and classical description of the gadget $g_d \in \{0,1\}^*$.

We now present our protocol for computing MSP_γ. It is similar to π_H for preparing H magic states.

Protocol 8 Protocol π_γ for preparing gadget magic states

Inputs: cMPC holds ${\sf sk} \in \{0,1\}^*$ and the permutation index d.
Common inputs: The parties hold the security parameter 1^κ.

1. The parties call $(\mathsf{AR}, (n^2, 0))$-IDPD without inputs, P_1 as the receiver, and trap-size $3\kappa \cdot \ell_{\sf gad}$. If the AR aborts, the parties have the identity of a corrupted party. They output it and halt.

2. Otherwise, cMPC receives from AR a Clifford $E \leftarrow \mathcal{C}_{3\kappa \cdot \ell_{\sf gad}}$, sampled uniformly at random, and the P_1 receives

$$\sigma = \mathsf{CAuth.Enc}_E(\varepsilon) = E(|0^{3\kappa \cdot \ell_{\sf gad}}\rangle\langle 0^{3\kappa \cdot \ell_{\sf gad}}|)E^\dagger = E(|0^{\ell_{\sf gad}}\rangle\langle 0^{\ell_{\sf gad}}|^M \otimes |0^{(3\kappa-1)\cdot\ell_{\sf gad}}\rangle\langle 0^{(3\kappa-1)\cdot\ell_{\sf gad}}|^T)E^\dagger.$$

3. cMPC computes $(U_d^M, g_d) \leftarrow M^{\sf Gad}({\sf sk}, d)$.

4. Partition the registers M and T into $\ell_{\sf gad}$ registers $M_1, \ldots, M_{\ell_{\sf gad}}$ and $T_1, \ldots, T_{\ell_{\sf gad}}$, respectively, where M_i is a single qubit register and T_i is a κ-qubit register for all $i \in [\ell_{\sf gad}]$. For each $i \in [\ell_{\sf gad}]$, cMPC samples Cliffords $E_i^{M_i T_i} \leftarrow \mathcal{C}_{3\kappa}$ independently and uniformly at random.

5. cMPC sets $V = (\bigotimes_{i=1}^{\ell_{\sf gad}} E_i^{M_i T_i})(EU_d^\dagger)^\dagger$ and sends it to P_1.

6. Party P_1 outputs $V\sigma V^\dagger$, cMPC outputs the gadget descriptions g_d and the Clifford keys $(E_i)_{i=1}^{\ell_{\sf gad}}$, and the other parties output nothing.

We next state and prove the security of the protocol.

Claim 7.3. *Protocol π_γ computes MSP_γ with perfect security-with-identifiable-abort in the $\{\mathsf{cMPC}\}$-hybrid model.*

Proof. For correctness, observe that

$$
V\sigma V^\dagger = \left(\bigotimes_{i=1}^{\ell_{\mathsf{gad}}} E_i^{M_i T_i}\right)\left(E(U_\gamma^\dagger)^M\right)^\dagger E \left(|0^{\ell_{\mathsf{gad}}}\rangle\langle 0^{\ell_{\mathsf{gad}}}|^M \otimes \bigotimes_{i=1}^{\ell_{\mathsf{gad}}} |0^\kappa\rangle\langle 0^\kappa|^{T_i}\right) E^\dagger \left(EU_\gamma^\dagger\right)\left(\bigotimes_{i=1}^{\ell_{\mathsf{gad}}} E_i\right)^\dagger
$$

$$
= \left(\bigotimes_{i=1}^{\ell_{\mathsf{gad}}} E_i^{M_i T_i}\right)\left(|\gamma\rangle\langle\gamma|^M \otimes \bigotimes_{i=1}^{\ell_{\mathsf{gad}}} |0^\kappa\rangle\langle 0^\kappa|^{T_i}\right)\left(\bigotimes_{i=1}^{\ell_{\mathsf{gad}}} E_i\right)^\dagger
$$

$$
= \left(\bigotimes_{i=1}^{\ell_{\mathsf{gad}}} \mathsf{CAuth.Enc}_{E_i}\right)(|\gamma\rangle\langle\gamma|).
$$

As for security, similarly to π_H, the only message sent in the protocol is the Clifford V that is distributed uniformly. $\qquad\square$

7.4 Preparing T Magic States

In this section we show how to prepare T magic states, $|T\rangle := T|+\rangle$. We define MSP_T to be a no-input functionality whose output is defined as follows. Let $E \leftarrow \mathcal{C}_{3\kappa}$ be a uniform random Clifford. Then the mapping outputs to P_1 a Clifford encryption of the T magic state $\mathsf{CAuth.Enc}_E(|T\rangle\langle T|)$ and outputs to cMPC the corresponding key E.

Since T gates are not Clifford, preparing T magic states requires a different approach from the other magic states. Toward constructing such a protocol, we first show how to homomorphically evaluate a classically controlled Clifford circuit over a Clifford ciphertext. We present such a protocol in Section 7.4.1. Then, in Section 7.4.2 we present a secure protocol for preparing $|T\rangle$ states.

7.4.1 Homomorphic Evaluation of Classically Controlled Clifford Circuit Over Clifford Codes

In this section we present a way for the parties to homomorphically evaluate classically controlled Cliffords over a Clifford authentication code. This is a two-party protocol executed between the server P_1 and cMPC. We stress that, similarly to π_{CTSA} and π_{CTAR}, the protocol we present is not secure on its own, and will be used as a subroutine during the T magic state preparation protocol in Section 7.4.2.

The input of the server P_1 is a Clifford ciphertext $\sigma_0 = E(\rho \otimes |0^t\rangle\langle 0^t|)E^\dagger$ of an r-qubit state ρ under the key $E \in \mathcal{C}_{r+t}$ with t traps, and the input cMPC is the corresponding Clifford key E. The number $t = t(\kappa)$ is a polynomial in the security parameter κ. Roughly, the idea is the following. Let C be the circuit to be evaluated and let d denote the number of gates in C. First, we represent the circuit C to be evaluated as $C = G_d \cdots G_2 G_1$, where each G_i is either a function for generating a Clifford gate or a measurement on a specific qubit. The server and cMPC will compute the G_i's in order. If G_i represents a function, cMPC computes $G_i(m)$ where $m \in \{0,1\}^*$ represents all previous measurement outcomes, and obtain a Clifford E_i. Similarly to the protocols for preparing P magic states, to emulate P_1 applying E_i we can simply have cMPC update its key.

Otherwise, G_i represents a measurement. Here we aim to have the server measuring the indicated qubits, and send the outcome to cMPC. To perform this task homomorphically over the Clifford ciphertexts, we let the parties *re-arrange* the ciphertext so that the qubit that should be measured will be re-encrypted to a trap authentication code, while the remaining qubits will be encrypted by Clifford code under a new key.

Formally, given an integer N, let S be a subsequence over $[N]$ without repetitions. We define $\mathcal{W}_N(S)$ to be the permutation over N qubits that moves the qubits indicated by S to the end, while keeping the order. For example, suppose we have eight qubits labelled by 1 to 8 in an ascending order. The permutation $\mathcal{W}_8((1,3,8))$ will move the sequence 12345678 to 24567138. Additionally, given two integers $a < b$ we denote by $[a : b]$ to be the sequence $(a, a+1, \cdots, b-1, b)$. Now, suppose we have a ciphertext in the form of $\sigma = E(\rho \otimes |0^t\rangle\langle 0^t|)E^\dagger$, where ρ is an r-qubit state, and we wish the measure the jth qubit of ρ. Then cMPC will send to P_1 the following Clifford. Let $W = \mathcal{W}_{r+t}((j, [r+1 : r+\kappa]))$ be the permutation that moves the jth qubits and κ of the traps to the end. It send to P_1 the operator

$$V = (E' \otimes \mathsf{TAuth.PEnc}_{\Pi,x,z})WE^\dagger,$$

where E' is the new key of the Clifford code, and Π, x, and z are the new keys of the trap authentication code. After P_1 applies V to σ, the last $\kappa + 1$ qubits are encrypted by trap code. Thus, the measurement outcome of the underlying message can be got by measuring the last $\kappa + 1$ qubits transversally in the computational basis and running $\mathsf{TAuth.Dec}$ over the encrypted outcome.

We next present the formal description of the protocol.

Protocol 9 Protocol π_{CCC} of Classically Controlled Clifford Circuit and Measurement under Clifford Code

---

Inputs: The server P_1 holds a Clifford ciphertext $\sigma_0 = E_0(\rho \otimes |0^t\rangle\langle 0^t|)E_0^\dagger$ of an r-qubit state ρ under the key $E_0 \in \mathcal{C}_{r+t}$, and cMPC holds the corresponding Clifford key E_0.
Common inputs: The parties hold the security parameter 1^κ.

1. For $i = 1$ to d:

   (a) cMPC initializes $\mathsf{count} = 0$ and an empty string m^*, which will include all previous measurement outcomes.

   (b) If G_i is Clifford gate, then cMPC computes $G_i(m^*)$ and updates its key $E_i := E_{i-1}G_i^\dagger$, and P_1 sets $\sigma_i = \sigma_{i-1}$.

   (c) If G_i is a measurement on j^{th} qubit:

      i. cMPC Does the following. Sample a Clifford $E_i \leftarrow \mathcal{C}_{r+t-(\mathsf{count}+1)(\kappa+1)}$, a permutation $\Pi_i \leftarrow \mathrm{Sym}_{\kappa+1}$, and strings $x_i, z_i \leftarrow \{0,1\}^{\kappa+1}$ uniformly at random and independently. Additionally, cMPC sets $W_i := \mathcal{W}_{r+t-\mathsf{count}(\kappa+1)}(j, [r - \mathsf{count} + 1 : r - \mathsf{count} + \kappa])$.

      ii. cMPC sends $V_i := (E_i \otimes \mathsf{TAuth.PEnc}_{\Pi_i,x_i,z_i})W_iE_{i-1}^\dagger$ to P_1.

      iii. P_1 applies V_i to σ_{i-1}, measures the last $\kappa + 1$ qubits in the computational basis, and sends the measurement outcome $\tilde{m}_i \in \{0,1\}^{\kappa+1}$ to cMPC. Let σ_i be the remaining unmeasured $r + t - (\mathsf{count} + 1)(\kappa + 1)$ qubits.

      iv. cMPC evaluates $(\mathtt{flag}, m_i) = \mathsf{TAuth.Dec}_{\Pi_i,x_i,z_i}(\tilde{m}_i)$. If $\mathtt{flag} = \mathsf{Rej}$, then cMPC output $\mathtt{abort}$ and halt. Otherwise, if $\mathtt{flag} = \mathsf{Acc}$, then cMPC appends m_i to m^*.

v. cMPC sets $\text{count} = \text{count} + 1$.

2. P_1 outputs the state σ_d, and cMPC outputs the corresponding Clifford key E_d.

Notice that, similarly to π_{CTSA} in Section 5.2, the protocol is insecure since the inputs include secret keys. However, in Section 7.4.2 below, we use π_{CCC} as a subroutine where the keys will be generated honestly, and thus the security will not be compromised. The following claim asserts the correctness of the protocol.

Claim 7.4. *In an honest execution of π_{CCC}, the output of P_1 is $\text{CAuth.Enc}_{E_d}(C(\rho))$ and the output of cMPC is E_d.*

Proof. Fix $i \in [d]$. First, if G_i is a Clifford gate, then the correctness follows from the same argument as in the previous magic state preparation protocols. If G_i is measurement, then by Lemma 2.18, the authenticity of all measurement outcomes are guaranteed. In other words, if the protocol is not aborted, all measurements are applied honestly, so the Clifford operations can be applied correctly by cMPC via key updating. □

7.4.2 T Magic State Preparation Protocol

In this section we finally present a secure-with-identifiable-abort protocol for computing MSP_T. Our protocol is a modification of the protocol of [DGJ+20] for preparing T magic states, which achieves only security-with-abort. Let us first give an overview of the protocol of [DGJ+20]. First, recall from Section 2.4, that given $\text{poly}(\log(1/\delta_0))$ copies of noisy T magic states with a constant fraction error, using the T distillation circuit we can obtain δ_0-close $|T\rangle$ state. Now, in [DGJ+20], the server P_1 prepares κn copies of $|T\rangle$. Then, the parties execute the *secure-with-abort* input-encoding protocol of [DGJ+20], which generates a Clifford encoding of each $|T\rangle$ state, i.e., $(\text{CAuth.Enc}_{E_j}(|T\rangle\langle T|))_{j=1}^{\kappa n}$, and outputs to cMPC the corresponding Clifford keys. Following this, cMPC samples disjoint sets $S_1, \cdots, S_n \subseteq [\kappa n]$, each of size κ, uniformly at random, and sends them to all parties. For each subset S_i, the server P_1 sends $\{\text{CAuth.Enc}_{E_j}(|T\rangle\langle T|)\}_{j \in S_i}$ to P_i and cMPC sends $\{E_j\}_{j \in S_i}$ to P_i. Then, party P_i decrypts and measures the received states in the $\{|T\rangle, |T^\perp\rangle\}$-basis, and broadcast an $\texttt{abort}$ if it gets $|T^\perp\rangle$ in any of the measurements. We refer to this step as the *random sampling test.* Upon receiving an $\texttt{abort}$ from a party, all parties abort and halt. Otherwise, [DGJ+20] showed that the remaining copies the server holds differ by a constant fraction from $\{\text{CAuth.Enc}_{E_j}(|T\rangle\langle T|)\}_{j \in S_1}$ with respect to the trace distance. The server can then get a state of negligible trace distance from $\text{CAuth.Enc}_E(|T\rangle\langle T|)$ with respect to a new key E, by running the T distillation circuit guaranteed to exist by Theorem 2.8.

Clearly, the above protocol does not admits security-with-identifiable-abort. Indeed, not only is the input-encoding protocol of [DGJ+20] does not admit security-with-identifiable-abort, it further holds that a corrupted party may accuse an honest server by lying about the states that it measured. To overcome this two issues, we perform the following modifications to the protocol of [DGJ+20].

First, we make the following observations. Regardless of what the server prepares, assuming all states were successfully sent during the random sampling test, with overwhelming probability all honest parties will have roughly the same number of $|T^\perp\rangle$ states upon measurement. Therefore, instead of broadcasting $\texttt{abort}$, we consider each party's *error rate*, defined to be number of $|T^\perp\rangle$ it holds divided by κ. These will be sent to cMPC, who compares them. If there are two parties

whose error rates are significantly far apart, then cMPC can publish them, and the honest parties can agree on which is corrupted.

Second, note that the previous observation holds only if the states were faithfully distributed by during the random sampling test. Indeed, the corrupted parties can bias the error rates by dropping the packets. Consider the following example. Suppose that P_1 is corrupted and $P_2, \ldots, P_n$ are honest. The server P_1 prepares each state to be $|T\rangle$ with probability $1/2$ and $|T^\perp\rangle$ with probability $1/2$. Then, after cMPC sends to P_1 the set S_2, the adversary drops the all $|T\rangle$ states that belong to S_2. As for the the states that belong to S_3, the adversary will drop the $|T^\perp\rangle$ states. Consequently, the error rate ϵ_2 will be much higher than $1/2$ while ϵ_3 will be much lower than $1/2$. To solve this issue, we would like that for any state that was dropped, the error rate will not include it. We achieve this as follows. The server first prepares $N_0 := 3n^2 + 1$ copies of $|T\rangle\langle T|^{\otimes \kappa n}$. Then, for each copy of $|T\rangle\langle T|^{\otimes \kappa n}$ a random sampling test is applied, where each party receives κ of the qubits among $|T\rangle\langle T|^{\otimes \kappa n}$. Moreover, the qubits will be transmitted to their destination using the AR functionality, to ensure that the adversary cannot drop too many qubits. If there is a state that was lost during the transmission, all parties start the random sampling test for another copy of $|T\rangle\langle T|^{\otimes \kappa n}$.

Third, the parties need to be able to sample a subset of the qubits held by P_1, even if $|T\rangle\langle T|^{\otimes \kappa n}$ are encrypted under a Clifford code. The idea is to have cMPC update the encryption keys so that they will randomly permute the qubits and re-encrypt using a new key of the form $(E_1 \otimes \cdots \otimes E_n)$, where each $E_i \leftarrow \mathcal{C}_{\kappa+t}$, where t is the number of traps. Additionally, the permutation must ensure that each κ of the T states have t trap states $|0\rangle$ appended to. Observe that decrypting using such key would result in n pieces of T states, each of size κ and encrypted under a different key with t traps. The parties can then use π_{CTAR} (Protocol 5) to distribute the states.

Finally, we encounter the following difficulty when constructing the simulator. The T distillation circuit is not deterministic in the sense that which gates should be applied depend on previous measurement outcomes. To simplify the security proof, after performing the T distillation, the parties execute π_{CTAR} to have the server P_1 route all ciphertexts generated by the T distillation circuit *to itself*. Now, similarly to π_{AR}, we let cMPC updating its key using a random Pauli. Then, similarly to the simulation for AR, the simulator can send halves of EPR pairs as the output during the execution of π_{CTAR}. Among the remaining packets after π_{CTAR}, the simulator can teleport the output it got from the trusted party to P_1.

We now formally describe our protocol in the $\{\mathsf{cMPC}, \mathsf{AR}\}$-hybrid model. In the following, we let $\delta = \frac{0.04}{n}$. The rationale of choosing the number of traps is explained in the security proof.

Protocol 10 Protocol π_T for preparing T magic states

Common input: The parties hold the security parameter 1^κ.

1. P_1 prepares N_0 copies of $(|T\rangle\langle T|)^{\otimes \kappa n}$. Let $\rho_j \in \mathcal{D}^{\kappa n}$ be the jth copy.

2. The parties call $(\mathsf{AR}, (n^2, 0))$-IDPD with trap-size $n^3 \cdot (n^2(3\kappa - 1) + \kappa^2)$, with P_1 as the receiver, and with only P_1 having an input, being a list of packets $(\rho_j)_{j=1}^{N_0}$.

   - If the output is $(\mathtt{abort}, P)$, then the parties output $(\mathtt{abort}, P)$ and halt.
   - Otherwise, if AR terminated successfully, each party receives $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}})$, and P_1 additionally gets a list of Clifford ciphertexts $(\sigma_j)_{j=1}^{N_0}$, and cMPC gets the corresponding Clifford keys $(E'_j)_{j=1}^{N_0}$, where each $E'_j \in \mathcal{C}_{\kappa n + n^3 \cdot (n^2(3\kappa - 1) + \kappa^2)}$. The server P_1 and cMPC discard the packets where $(1, j) \notin \mathcal{R}_{\mathsf{in}}$ and relabel the remaining packets as $(\sigma_j)_{j=1}^{N_1}$ and $(E'_j)_{j=1}^{N_1}$, where $N_1 = |\mathcal{R}_{\mathsf{in}}| \geq 2n^2 + 1$.

3. For all $j \in [N_1]$:

   (a) For all $i = 1$ to n cMPC does the following:

      i. Sample a subset $S_{ij} \subseteq [i\kappa]$ of size $|S_{ij}| = \kappa$ uniformly at random.

      ii. Set $W_{ij} := \mathcal{W}_{\kappa n + n^3 \cdot (n^2(3\kappa-1)+\kappa^2)}(S_{ij}, [i\kappa + 1 : i\kappa + \kappa])$.

      iii. Sample $E_{ij} \leftarrow \mathcal{C}_{\kappa + n^2 \cdot (n^2(3\kappa-1)+\kappa^2)}$.

   (b) cMPC sets
   $$V_j = (E_{1j} \otimes \cdots \otimes E_{nj})(W_{nj} \cdots W_{1j})E_j'^{\dagger},$$
   and sends V_j to P_1.

   (c) P_1 applies V_j to σ_j to obatin
   $$\tau_{1j} \otimes \tau_{2j} \otimes \cdots \otimes \tau_{nj} = V_j \sigma_j V_j^{\dagger},$$
   where $\tau_{ij} = E_{ij}(|T\rangle\langle T|^{\otimes\kappa} \otimes |0\rangle\langle 0|^{\otimes n^2 \cdot (n^2(3\kappa-1)+\kappa^2)})E_{ij}^{\dagger}$ for all $i \in [n]$.

4. The parties execute π_{CTAR} (Protocol 5). Here, P_1 is the sender with its input being $(\tau_{ij})_{(i,j)\in[n]\times[N_1]}$, cMPC's input are the corresponding Cliffords $(E_{ij})_{(i,j)\in[n]\times[N_1]}$, and P_i is the receiver of $(\tau_{ij})_{j\in[N_1]}$ for all $i \in [n]$. The common inputs are κ as the size of the plaintext, and $n^2 \cdot (n^2(3\kappa - 1) + \kappa^2)$ as the trap-size.

   - If π_{CTAR} outputs $(\mathtt{abort}, P)$, then the parties output $(\mathtt{abort}, P)$ and halt.

   - Otherwise, if π_{CTAR} terminated successfully, each party receives $(\mathtt{continue}, \mathcal{R}'_{\mathsf{in}})$, and for all i, party P_i additionally gets a list of Clifford ciphertexts $(\tilde{\tau}_{ij})_{ij\in\mathcal{R}'_{\mathsf{in}}}$, and cMPC gets the corresponding Clifford keys $(\tilde{E}_{ij})_{ij\in\mathcal{R}'_{\mathsf{in}}}$.

5. cMPC updates $\mathcal{R}'_{\mathsf{in}}$ to ensure each party holds the same list of packets:

   (a) For each packet $(i, j) \in \mathcal{R}'_{\mathsf{in}}$, if there exists $i^* \in [n]$ such that $(i^*, j) \notin \mathcal{R}'_{\mathsf{in}}$, then remove (i, j) from $\mathcal{R}'_{\mathsf{in}}$ (note that at most n^2 packets can be dropped during the execution of π_{CTAR}, hence $\mathcal{R}'_{\mathsf{in}}$ is not empty).

   (b) Send the updated $\mathcal{R}'_{\mathsf{in}}$ to all parties. Let $\mathcal{R}_J$ be the set of all remaining indices j, i.e., $\mathcal{R}_J = \{j \in [N_1] : (i, j) \in \mathcal{R}'_{\mathsf{in}}\}$.

6. For each packet $j \in \mathcal{R}_J$:

   (a) For each non-server party $i \in \{2, 3, \ldots, n\}$:

      i. cMPC sends $\tilde{E}_{ij}$ to P_i.

      ii. P_i runs $\mathsf{CAuth.Dec}_{\tilde{E}_{ij}}(\tilde{\tau}_{ij})$ and measures the decrypted message in the $\{|T\rangle, |T^{\perp}\rangle\}$-basis. Let ϵ_{ij} be the error-rate, defined to be the number of $|T^{\perp}\rangle$ divided by κ.

      iii. P_i sends ϵ_{ij} to cMPC

   (b) cMPC check the error-rates for packet j:

      i. Set $\epsilon_{1j} = 0$.

      ii. Sort the error-rates $\epsilon_{1j}, \epsilon_{2j}, \cdots, \epsilon_{nj}$ of all parties. Let $\mu_{1j} \leq \cdots \leq \mu_{nj}$ be the sorted result.

iii. For $k = 1$ to $n - 1$, if $\mu_{(k+1)j} - \mu_{kj} > \delta$, then do the following. First, cMPC sends (μ_{kj}, P_a, P_b) to all parties, where P_a is the party who sent the error-rate μ_{kj}, and where P_b is the party who sent the error-rate $\mu_{(k+1)j}$. Then, each party P_i then outputs $(\mathtt{abort}, P_a)$ if $\epsilon_{ij} > \mu_{kj}$, and outputs $(\mathtt{abort}, P_b)$ if $\epsilon_{ij} \leq \mu_{kj}$, and halt.

7. For each packet $j \in \mathcal{R}_J$, cMPC and P_1 execute π_{CCC} (Protocol 9) for the magic state distillation circuit guaranteed to exist by Theorem 2.8, with P_1's input being $\tilde{\tau}_{1j}$, and cMPC's input being $\tilde{E}_{1j}$

   - If the output of cMPC is $\mathtt{abort}$, then it sends $(\mathtt{abort}, P_1)$ to all parties, they output it, and halt.

   - Otherwise, if π_{CCC} terminated successfully, P_1 holds η_j which is the output of distillation under Clifford encoding, and cMPC holds the corresponding Clifford key F_j.

8. The parties execute π_{CTAR}. Here, P_1 is the sender and the only receiver, with its input being the list of packets $(\eta_j)_{j \in \mathcal{R}_J}$, and cMPC holds the corresponding Clifford keys $(F_j)_{j \in \mathcal{R}_J}$. The common inputs are 1 as the size of the plaintext, and $n^2(3\kappa - 1)$ as the trap-size.

   - If π_{CTAR} outputs $(\mathtt{abort}, P)$, the parties then outputs $(\mathtt{abort}, P)$ and halt.

   - Otherwise, if π_{CTAR} terminated successfully, each party receives $(\mathtt{continue}, \mathcal{R}''_{\mathsf{in}})$, and P_1 additionally gets a list of Clifford ciphertexts $(\tilde{\eta}_j)_{(1,j) \in \mathcal{R}''_{\mathsf{in}}}$, and cMPC gets the corresponding Clifford keys $(\tilde{F}_j)_{(1,j) \in \mathcal{R}''_{\mathsf{in}}}$.

9. Let $(1, j^*) = \operatorname{argmin}_j \{(1, j) \in \mathcal{R}''_{\mathsf{in}}\}$ with respect to to lexicographic ordering. cMPC and P_1 do the following.

   (a) cMPC samples a Pauli $Q \leftarrow \mathcal{P}_{3\kappa}$ and sends it to P_1.
   (b) P_1 applies Q to $\tilde{\eta}_{j^*}$, to obtain $\hat{\eta}$.
   (c) cMPC updates its Clifford key $E = Q\tilde{F}_{j^*}$.

10. P_1 outputs $\hat{\eta}$ and cMPC outputs E.

We now state and prove the security of π_T.

Lemma 7.5. *Protocol π_T computes the functionality MSP_T with statistical security-with-identifiable-abort in $\{\mathsf{cMPC}, \mathsf{AR}\}$-hybrid model.*

Proof. Let us first prove correctness. Observe that in an honest execution, all copies that P_1 prepares at the beginning are $(|T\rangle\langle T|)^{\otimes \kappa n}$. Observe that at the first execution of π_{CTAR}, the state τ_{ij} that suppose to be routed to P_i is $E_{ij}(|T\rangle\langle T|^{\otimes \kappa} \otimes |0\rangle\langle 0|^{\otimes (n^2(3\kappa-1)+\kappa^2)})E_{ij}^\dagger$, because $(n^2 - 1)/n^2$ portion of traps are consumed in π_{CTAR}. Then, by Claim 6.2, each party gets the Clifford encryption of $(|T\rangle\langle T|)^{\otimes \kappa}$ at the end of Step 4, and each party will report error rate equals to zero at Step 6. Then, for each packet $j \in \mathcal{R}_J$, by Claim 7.4 and Theorem 2.8, the state that P_1 holds at Step 7 is a state negligibly close to $|T\rangle$ with respect to trace distance, under Clifford encryption. Here, the number of traps reduces to $n^2(3\kappa - 1)$ since κ^2 traps are used in π_{CCC}. Finally, by Claim 6.2 the output of P_1 at Step 8 are Clifford encryptions of nearly perfect $|T\rangle$ states, where the number of traps reduces to $3\kappa - 1$ due to π_{CTAR}. Because $|T\rangle$ is a single-qubit state, the total size of the output is 3κ, concluding correctness.

We now prove security. Fix an adversary $\mathcal{A}$ corrupting a subset $\mathcal{I}$ of the parties holding auxiliary input ρ_{aux}. We next show how to construct a simulator $\mathsf{Sim}_{\mathcal{A}}$. Roughly, the idea is to emulate the honest parties and cMPC until the second execution of π_{CTAR}. There, we use the same trick used to simulate π_{AR}, and send half of EPR pairs as the output, and use quantum teleportation to correct the output of the server (see Figure 4). Additionally, to simulate an execution of π_{CTAR}, the simulator must also simulate many executions of π_{CTSA}. For this, we can use the simulator operating in the SA ideal world, denoted $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$, guaranteed to exists by Corollary 5.7. We now formalize the above intuition.

Simulator 4 Simulator $\mathsf{Sim}_{\mathcal{A}}$ for T magic state preparation

---

1. Simulate the call to AR at Step 2:

   (a) If $P_1 \in \mathcal{I}$ is corrupted, query $\mathcal{A}$ for its input $(\rho_j)_{j=1}^{N_0}$ to AR. Otherwise, if P_1 is honest, sets $\rho_j = (|T\rangle\langle T|)^{\otimes \kappa n}$ for all $j \in [N_0]$.

   (b) Sample $E'_j \leftarrow \mathcal{C}_{\kappa n + n^3 \cdot (n^2(3\kappa-1)+\kappa^2)}$ for all $j \in [N_0]$.

   (c) Send $(\mathsf{CAuth.Enc}_{E'_j}(|0\rangle\langle 0|))_{j=1}^{N_0}$ to $\mathcal{A}$ as the output of AR. If $\mathcal{A}$ replies with $(\mathtt{abort}, P)$, forward it to T, output whatever $\mathcal{A}$ outputs, and halt. Otherwise, it sent $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}})$. Let $N_1 = |\mathcal{R}_{\mathsf{in}}|$.

2. For each packet $j \in [N_1]$:

   (a) For all $i \in [n]$, sample the subset $S_{ij} \subseteq [i\kappa]$ and Clifford $E_{ij} \in \mathcal{C}_{\kappa + n^2 \cdot (n^2(3\kappa-1)+\kappa^2)}$, and set V_j as cMPC does at Step 3.

   (b) If $P_1 \in \mathcal{I}$ is corrupted, send V_j to $\mathcal{A}$.

   (c) Divide ρ_j according to S_{ij} for all i. Let ρ_{ij} be the states sampled from ρ_j by S_{ij}.

3. Simulate the execution of π_{CTAR} (Protocol 5) at Step 4:

   (a) For all $i \in [n]$ and $j \in [N_1]$ sample $\tilde{E}_{ij} \leftarrow \mathcal{C}_{\kappa + (n^2(3\kappa-1)+\kappa^2)}$.

   (b) Initialize G as the complete graph and $\mathcal{R}'_{\mathsf{in}} = \emptyset$ as an empty set.

   (c) For all $(i, j) \in [n] \times [N_1]$:

       i. Find the path $\mathtt{PATH}_{ij}$ in G from P_1 to P_i that passes through every party, computed the same as cMPC computed, and sends it to $\mathcal{A}$.

       ii. Run $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ with the following as input. If $P_1 \notin \mathcal{I}$ then $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ has no inputs. Otherwise, the input is ρ_{ij}. If $P_i \in \mathcal{I}$, then $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ sends ρ_{ij} back as input. Send it back $\mathsf{CAuth.Enc}_{\tilde{E}_{ij}}(\rho_{ij})$ as the output $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ receives from the trusted party.

       iii. If $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ replied with $\mathtt{continue}$, then add (i, j) to $\mathcal{R}'_{\mathsf{in}}$.

       iv. Otherwise, if it replied with $(\mathtt{abort}, P_a, P_b)$, then remove the edge ab from G. If G becomes disconnected, send $(\mathtt{abort}, P)$ to T where $P \in \{P_a, P_b\}$ is the party not connected to the honest parties, output whatever $\mathcal{A}$ outputs, and halt.

4. Update $\mathcal{R}'_{\mathsf{in}}$ as cMPC does at Step 5, and set $\mathcal{R}_{\mathsf{J}} = \{j \in [N_1] : (i, j) \in \mathcal{R}'_{\mathsf{in}}\}$.

5. For each packet $j \in \mathcal{R}_{\mathsf{J}}$:

   (a) For each malicious non-server party $P_i \in \mathcal{I} \setminus \{P_1\}$, send $\tilde{E}_{ij}$ to $\mathcal{A}$, and receive back ϵ_{ij} as the error-rate of P_i.

(b) For each honest non-server party $P_i \notin \mathcal{I} \setminus \{P_1\}$, measure ρ_{ij} in the $\{|T\rangle, |T^\perp\rangle\}$-basis and set the error rate ϵ_{ij} to be the number of $|T^\perp\rangle$ divided by κ.

(c) Set $\epsilon_{1j} = 0$.

(d) Sort the error rates $\epsilon_{1j}, \epsilon_{2j}, \cdots, \epsilon_{nj}$ of all parties, and let $\mu_{1j} \leq \cdots \leq \mu_{nj}$ be the sorted result.

(e) For $k = 1$ to $n - 1$:

    i. If $\mu_{(k+1)j} - \mu_{kj} > \delta$ and there exists an error rate $\epsilon_{ij} \leq \mu_{kj}$ for some $P_i \notin \mathcal{I}$ is honest, send $(\texttt{abort}, P)$ to T, where P is the corrupted party with the error rate $\mu_{(k+1)j}$.

    ii. If $\mu_{(k+1)j} - \mu_{kj} > \delta$ and there exists an error rate $\epsilon_{ij} \geq \mu_{(k+1)j}$ for some $P_i \notin \mathcal{I}$ is honest, send $(\texttt{abort}, P)$ to T, where P is the corrupted party with the error rate μ_{kj}.

6. If $P_1 \in \mathcal{I}$ is corrupted, for every $j \in \mathcal{R}_{\mathsf{J}}$, simulate the execution of π_{CCC} at Step 7. This is done by emulating cMPC on input $\tilde{E}_{1j}$. Let F_j be the Clifford keys, being the output of cMPC has computed by $\mathsf{Sim}_{\mathcal{A}}$ in the jth execution.

7. Simulate the execution of π_{CTAR} at Step 8:

(a) Initialize G as the complete graph and $\mathcal{R}''_{\mathsf{in}} = \emptyset$ as an empty set.

(b) For all $(i, j) \in \mathcal{R}'_{\mathsf{in}}$:

    i. Find the path $\texttt{PATH}_{ij}$ in G from P_1 to P_i that passes through every party, computed the same as cMPC computed, and sends it to $\mathcal{A}$.

    ii. If $P_1 \in \mathcal{I}$ is corrupted:

        A. Put 3κ EPR pairs $|\Phi^+\rangle^{3\kappa}$ into a 6κ-qubit register S_{ij}^{LR}, where S_{ij}^L and S_{ij}^R represents the 3κ-qubit register storing the first and the second qubit of each EPR pair, respectively.

        B. Run $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ on the input that P_1 holds after the execution of π_{CCC}. When $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ requests for an output from its trusted party, send it the content of register S_j^R.

    iii. If $P_1 \notin \mathcal{I}$ is honest, run $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ on input $\mathsf{CAuth.Enc}_F(|0\rangle\langle 0|)$, where $F \leftarrow \mathcal{C}_{3\kappa n^2}$ (note that $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ never requests for an output if P_1 is honest).

    iv. If $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ replied with $\texttt{continue}$, then add (i, j) to $\mathcal{R}''_{\mathsf{in}}$.

    v. Otherwise, if it replied with $(\texttt{abort}, P_a, P_b)$, then remove the edge ab from G. If G becomes disconnected, send $(\texttt{abort}, P)$ to T where $P' \in \{P_a, P_b\}$ is the party not connected to the honest parties, output whatever $\mathcal{A}$ outputs, and halt.

8. If $P_1 \in \mathcal{I}$ is corrupted, receive the ciphertext $\hat{\eta}$ from the trusted party T, and do the following.

(a) Let j^* be the lexicographically first packet, i.e., $(1, j^*) = \mathrm{argmin}_j\{(1, j) \in \mathcal{R}''_{\mathsf{in}}\}$.

(b) Put the state $\hat{\eta}$ into register S'. Run the circuit in the dashed lines of Figure 4 over register S' and $S_{j^*}^L$, generating classical strings x and z. This step teleports $\hat{\eta}$ to register $S_{j^*}^R$.

(c) Send $P = Z^z X^x$ to $\mathcal{A}$.

9. Send $\texttt{continue}$ to T, output whatever $\mathcal{A}$ outputs, and halt.

We now analyze the above simulator. There are five steps where the protocol can abort. a) During the call to AR at Step 2; b) during the executions of π_{CTAR} at Steps 4 and 8; c) during error estimation at Step 6; d) and during T distillation at Step 7.

Let us first consider the case where $P_1 \notin \mathcal{I}$ is honest. Observe that in this case, the only messages $\mathcal{A}$ receives, are during the executions of π_{CTAR}. By Corollary 5.7, the states of $\mathcal{A}$ generated by $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ is indistinguishable from the real world execution of π_{CTSA}. Moreover, in the first execution of π_{CTSA}, the output sent to $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ are Clifford encryptions of $|T\rangle\langle T|^{\otimes \kappa}$, which is the same as in the real world. Thus, the adversary replies with the same messages (up to a negligible difference).

Let us now consider the case where $P_1 \in \mathcal{I}$. Because at Step 3, the V_js that the adversary receives are random Cliffords from the perspective of the adversary, we can assume without loss of generality that the adversary always applies V_j honestly, and attacks the ciphertexts then. By a similar argument to the previous case, we may condition on the event that there was no abort during the first execution of π_{CTAR}. Then the error rates $\mathcal{A}$ sent to $\mathsf{Sim}_{\mathcal{A}}$ in the ideal world, are statistically close to the error rates it sends to cMPC in the real world. Moreover, the error rates of the honest parties computed by $\mathsf{Sim}_{\mathcal{A}}$ are also statistically close their real world counter parts. By Hoeffding's inequality[13] (Lemma 2.5) and the union bound, it follows that except with negligible probability, the error rates of the honest parties are at most δ apart. That is,

$$\Pr[\exists i, i' \in [n] \setminus \mathcal{I}, j \in \mathcal{R}_\mathsf{J} : |\epsilon_{ij} - \epsilon_{i'j}| > \delta] < 4n^2 N_2 \cdot e^{-2(\delta/2)^2 \kappa},$$

where $N_2 = |\mathcal{R}_\mathsf{J}|$.

Next, if the error rates are all at most δ apart (i.e., there was no abort during Step 6), then in particular the error rates are at most $n\delta = 0.04$. Therefore, by Lemma 2.7, for all $j \in \mathcal{R}_\mathsf{J}$, up to trace distance $2e^{-0.0005^2 \kappa}$, the message of the ciphertext $\tilde{\tau}_{1j}$ in the real world, lies in span $(\{|s\rangle_T : s \in B\})$ where $|0\rangle_T := |T\rangle, |1\rangle_T := |T^\perp\rangle$ and $B = \{s \in \{0,1\}^\kappa : |w(s) - n\delta| \leq 0.001\}$. Therefore, by Theorem 2.8, as long as the distillation protocol terminates successfully, the output state η_j satisfies

$$\Delta\left(\mathsf{CAuth.Dec}_{E_j}(\eta_j), |T\rangle\langle T|\right) \leq \mathrm{neg}(\kappa)$$

for all j.

Finally, by the similar argument as the previous execution of π_{CTAR}, the views at the end of Step 9 are identical in both worlds. To be more precise, if the protocol does not abort, then the output in the ideal world is teleported to the qubits corresponding to the packet $(1, j^*)$. Also, all other discarded packets are maximally mixed states from the adversary's view, which do not provide any distinguishing power. Therefore, we conclude that

$$\left\{\mathrm{IDEAL}_{\mathsf{MSP}_T, \mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})}(\kappa)\right\}_{\kappa \in \mathbb{N}, \rho_{\mathsf{aux}} \in \mathcal{D}^*} \stackrel{\mathrm{C}}{\equiv} \left\{\mathrm{HYBRID}_{\pi_T, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}, \mathsf{AR}}(\kappa)\right\}_{\kappa \in \mathbb{N}, \rho_{\mathsf{aux}} \in \mathcal{D}^*}.$$

$\square$

8 Secure Delegation of The Computation – Preparation

In this section, we present a protocol, which we call *pre-computation*, that allows the parties to securely delegate the computation to a designated party, called the server. More concretely, at

---

[13]Notice that to compute the error rates, the parties first partition all states and then measure. Clearly, this is equivalent to first measuring and then partitioning.

the end of the protocol, the server will hold an encryption of each of the parties' inputs, while cMPC will hold the keys used for the encryption. Later, in Section 9, we show how the server and cMPC can homomorphically evaluate a quantum circuit over the encrypted inputs. Thus, we require that the encryption used by the parties to be the TrapTP VQFHE scheme of [ADSS17] (see Section 2.7.1). Recall that a VQFHE is four-tuple of QPT algorithms (KeyGen, Enc, Eval, Dec), that generate keys, encrypt plaintext, evaluate a circuit of an encrypted message, and decrypt a ciphertext while verifying the evaluation was performed honestly. Our pre-computation protocol can thus be viewed as a way to implement TrapTP.KeyGen and TrapTP.Enc – the key generation and encryption algorithms of TrapTP, respectively – in a distributed manner. We next present a formal definition of the functionality we wish to compute.

Let $L = L(\kappa)$ be a polynomial (this will later represent an upper-bound on the size of a circuit to be evaluated and the number of ancilla 0 states it requires). Denote by $\mathsf{PreComp} = \mathsf{PreComp}_L$ the following mapping. Party P_i holds an ℓ_{in}-qubit input $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, where $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\kappa) \in \mathbb{N}$ is some polynomial. Only the server P_1 and cMPC are given outputs, defined as follows. Let $(\mathsf{sk}, \rho_{\mathsf{evk}}) \leftarrow \mathsf{TrapTP.KeyGen}(1^\kappa, 1^L)$ (recall that $\mathsf{sk} \in \{0,1\}^*$ is a classical string and ρ_{evk} is a quantum state). Then cMPC receives sk and P_1 receives the encryptions of each input-qubit, encryptions of 0 states, and the evaluation key ρ_{evk}, i.e., it receives $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ where $\hat{\rho} = (\mathsf{TrapTP.Enc}_{\mathsf{sk}}(\rho_{ij}))_{(i,j) \in [n] \times [\ell_{\mathsf{in}}]}$, and $\hat{\rho}_0 = (\mathsf{TrapTP.Enc}_{\mathsf{sk}}(|0\rangle\langle0|))_{i=1}^L$. In the following section we present a protocol computing $\mathsf{PreComp}$ with $(n^2, 0)$-IDPD-security that outputs ciphertexts of size 3κ.

8.1 The Pre-Computation Protocol

In this section, we present our protocol for pre-computation. Conceptually, the protocol consists of three main steps. First, the private inputs of each party are routed to P_1 by a call to AR. This results in P_1 holding the Clifford ciphertexts of all private inputs, and cMPC holding the keys. Second, the parties call the magic state preparation functionalities. This include MSP_Z that prepares ancilla 0 states, MSP_P that prepares magic P states, MSP_H that prepares magic H states, MSP_T that prepares magic T states, and MSP_γ that prepares gadget states. At the end of the call, the server P_1 holds the Clifford ciphertexts of all these magic states, while cMPC holds the Clifford keys. Finally, as we use the TrapTP scheme, the homomorphic evaluation can only be applied to trap-code ciphertexts. Thus, we show how the server and cMPC can *re-encrypt* all Clifford ciphertexts to trap-code ciphertext.

Notations. Before formally introducing our protocol, we first present some notations. First, it will be convenient to denote by $\mathcal{M} = \{Z, P, H, T, \gamma\}$ and let $\mathsf{MSP} = \{\mathsf{MSP}_{\mathsf{ms}}\}_{\mathsf{ms} \in \mathcal{M}}$ be the set of corresponding ideal functionalities. Now recall that in the VQFHE scheme TrapTP, both algorithms TrapTP.KeyGen and TrapTP.Enc, are executed by the client, who is an honest party with quantum power. In our case, however, only the hybrid cMPC, which can perform only classical computation, may be viewed as an honest party. Thus, we divide TrapTP.KeyGen and TrapTP.Enc into their *classical part* and its *quantum part* Then, we let cMPC perform the classical part and let the server P_1 perform the quantum part.

Formally, the classical part of TrapTP.KeyGen consists of two algorithms. The first algorithm, denoted $M_1^{\mathsf{KG}}(1^\kappa, 1^L)$, is given the security parameter and outputs a secret key $\mathsf{sk} \in \{0,1\}^*$. The secret key sk includes the secret key of a classical message authentication code MAC, the public key and secret key of classical homomorphic encryption scheme HE, a global permutation Π_0 for the trap-code encodings, and local permutations $(\Pi_i)_{i=1}^L$ for each gadget. The second algorithm, denoted $M_2^{\mathsf{KG}}(sk, (g_i)_{i=1}^L)$, is given the secret key sk and a list of gadget descriptions $(g_i)_{i=1}^L$. It outputs a classical string m_{evk} that m_{evk} contains all classical information that the server would later

need for the evaluation. Similarly, we also divide TrapTP.Enc into the classical and the quantum parts. Here, there is only a single classical part denoted M^E. The algorithm $M^E(\mathsf{sk}, P, i)$ takes as inputs the secret key sk, a Pauli operator P and a permutation index i, and outputs a classical string, which includes the encryption $\mathsf{HE.Enc}(P)$ of the Pauli, signed by MAC. We refer the reader to [ADSS17] for a detailed construction of each classical algorithm.

Re-encrypting Clifford Codes to trap code. We now present a simple *two-party* protocol, between the server P_1 and cMPC, that allows the server to re-encrypt a Clifford ciphertext into a trap-code ciphertext.

Protocol 11 Re-Encryption π_{ReEnc}

Inputs: P_1 holds a Clifford ciphertext $\sigma = \mathsf{CAuth.Enc}_E(\rho)$, and cMPC holds the corresponding key E, a secret key sk and a permutation index i.
Common input: Both parties hold the security parameter 1^κ.

1. cMPC does the following.

   (a) Sample a random Pauli $P \leftarrow \mathcal{P}_{3\kappa}$.

   (b) Compute $m \leftarrow M^E(\mathsf{sk}, P, i)$.[14]

   (c) Set $V = \mathsf{TAuth.PEnc}_{\Pi_i, P} E^\dagger$, where Π_i is the local permutation for the ith gadget in sk if $i > 0$, and Π_0 is the global permutation if $i = 0$.

   (d) Send m and V to P_1.

2. P_1 applies V to σ, obtaining τ.

3. P_1 outputs (m, τ) and cMPC outputs sk.

Clearly, if P_1 faithfully applies V to its input, then the output τ is a trap-code of ρ.

At the first glance, it seems that the operator V that P_1 received contains the information about Clifford ciphertext and trapcode ciphertext, so it may potentially harm the security. Here, we sketch the intuition that why it is still secure, and the formal statement is given in Theorem 8.1. Initially, the key E of the Clifford ciphertext is uniformly random to P_1. In this case, E acts like the encryption key of one-time pad, where V is like the ciphertext of one-time pad and $\mathsf{TAuth.PEnc}_{\Pi_i, P}$ is like the plaintext of one-time pad. As long as E is uniformly random to begin with, V reveals nothing about $\mathsf{TAuth.PEnc}_{\Pi_i, P}$ by the security of one-time pad. It is the reason why the simulator can simulate cMPC by choosing V uniformly at random. Then, without loss of generality, we assume P_1 always applies V honestly at Step 2. After this step, P_1 holds a quantum state $\mathsf{TAuth.PEnc}_{\Pi_i, P}(\rho)$ and a classical information V. Because V is independent to Π_i, P, the security holds by the security of trap code.

---

[14]Recall that m includes an encryption of the Pauli P under a key included in sk held by cMPC.

Protocol's construction. We are now ready to introduce our protocol π_{PreComp} for performing the pre-computation task. The protocol has an additional common input $L = L(\kappa)$ and is described in the $\{\mathsf{cMPC}, \mathsf{AR}, \mathsf{MSP}\}$-hybrid model.

Protocol 12 Pre-Computation π_{PreComp}

Inputs: Each party P_i holds private inputs $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, where each $\rho_{ij} \in \mathcal{D}^1$ is a single qubit.
Common input: The parties hold the security parameter 1^κ and the level parameter L.

1. The parties call $(\mathsf{AR}, (n^2, 0)\text{-}\mathsf{IDPD})$ with trap-size $3\kappa - 1$, with P_i's input being $(\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, and with P_1 as the receiver.

   - If AR outputs $(\mathtt{abort}, \mathrm{P})$, then the parties output $(\mathtt{abort}, \mathrm{P})$ and halt.
   - Otherwise, if AR terminated successfully, each party receives $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}})$, P_1 additionally gets Clifford ciphertext $(\sigma_{ij})_{(i,j) \in [n] \times [\ell_{\mathsf{in}}]}$, and cMPC additionally gets the corresponding Clifford keys $(E_{ij})_{(i,j) \in [n] \times [\ell_{\mathsf{in}}]}$, where each $E_{ij} \in \mathcal{C}_{3\kappa}$.

2. The parties prepare magic state as follows.

   (a) cMPC generates a secret key $\mathsf{sk} \leftarrow M_1^{\mathsf{KG}}(1^\kappa, 1^L)$.

   (b) For all $\mathsf{ms} \in \mathcal{M} \setminus \gamma$, the parties call $\mathsf{MSP}_{\mathsf{ms}}$ L times.

   - If at any of the calls, the parties receive the output $(\mathtt{abort}, \mathrm{P})$, then they output $(\mathtt{abort}, \mathrm{P})$ and halt.
   - Otherwise, from the ith call to $\mathsf{MSP}_{\mathsf{ms}}$ the server P_1 gets σ_i^{ms} and cMPC gets F_i^{ms}. Here, we have that if $\mathsf{ms} \neq H$ then σ_i^{ms} is a Clifford encryption of a single qubit where F_i^{ms} is the Clifford key used. Otherwise, σ_i^H is the encryption of $|H\rangle$ where each qubit is encrypted independently and $F_i^H = (F_{i1}^H, F_{i2}^H)$ is the pair of Clifford keys used.

   (c) For all $i \in [L]$:

   i. The parties prepare gadget states by calling MSP_γ with cMPC's input being sk and the permutation index i.

   - If the protocol outputs $(\mathtt{abort}, \mathrm{P})$, then the parties output $(\mathtt{abort}, \mathrm{P})$ and halt.
   - Otherwise, P_1 receives σ_i^γ and cMPC receives $(g_i, (F_{ij}^\gamma)_{j=1}^{\ell_{\mathsf{gad}}})$.

   (d) cMPC computes $m_{\mathsf{evk}} \leftarrow M_2^{\mathsf{KG}}(\mathsf{sk}, (g_i)_{i=1}^L)$ and sends m_{evk} to P_1.

3. The server P_1 and cMPC re-encrypt all Clifford ciphertext as follows.

   (a) For each private inputs $(i, j) \in [n] \times [\ell_{\mathsf{in}}]$, cMPC and P_1 execute π_{ReEnc} with P_1's input being the state σ_{ij}, and cMPC's input being E_{ij}, sk and gadget index 0. Let (m_{ij}, τ_{ij}) be the output of P_1.

   (b) For all $\mathsf{ms} \in \{Z, P, T\}$ and $i \in [L]$, cMPC and P_1 execute π_{ReEnc} with P_1's input being the state σ_i^{ms}, and cMPC's input being F_i^{ms}, sk and gadget index 0. Let $(m_i^{\mathsf{ms}}, \tau_i^{\mathsf{ms}})$ be the output of P_1.

   (c) For all H magic states $i \in [L]$:

   i. Put the state σ_i^H in register R_i^H of 6κ-qubit system. Partition R_i^H into two registers $R_{i,1}^H$ and $R_{i,2}^H$, each of size 3κ.

ii. For $j \in \{1, 2\}$, cMPC and P_1 execute π_{ReEnc} with P_1's input being the qubits of $R_{i,j}^H$, and cMPC's input being F_{ij}^H, sk and permutation index 0.

iii. Denote m_i^H to be the two classical outputs of the executions of π_{ReEnc}, and let τ_i^H be the state held by P_1 in register R_i^H.

(d) For each gadget $i \in [L]$:

i. Put the state σ_i^γ in register R_i^γ of $(3\kappa \cdot \ell_{\mathsf{gad}})$-qubit system. Partition R_i^γ into ℓ_{gad} registers $R_{i,1}^\gamma, \ldots, R_{i,\ell_{\mathsf{gad}}}^\gamma$, each of size 3κ.

ii. For $j \in \{1, 2\}$, cMPC and P_1 execute π_{ReEnc} with P_1's input being the jth register R_{ij}^γ, and cMPC's input being F_{ij}^γ, sk and the permutation index that equals to 0 if $j \in \{1, 2\}$ and equal to i otherwise.[15]

iii. Denote m_i^γ to be the ℓ_{gad} classical outputs of the executions of π_{ReEnc}, and let τ_i^γ be the state held by P_1 in register R_i^γ.

4. Each party outputs $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}})$. Additionally, cMPC outputs sk, and P_1 outputs $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$, where

$$\hat{\rho} = (m_{ij}, \tau_{ij})_{i \in [n], j \in [\ell_{\mathsf{in}}]},$$
$$\hat{\rho}_0 = (m_i^Z, \tau_i^Z)_{i=1}^L,$$
$$\rho_{\mathsf{evk}} = \left(m_{\mathsf{evk}}, (m_i^{\mathsf{ms}}, \tau_i^{\mathsf{ms}})_{i \in [L], \mathsf{ms} \in \mathcal{M} \backslash \{Z\}} \right). \tag{15}$$

Theorem 8.1. *Protocol π_{PreComp} computes the functionality* PreComp *with perfect $(n^2, 0)$-IDPD security in the $\{\mathsf{cMPC}, \mathsf{AR}, \mathsf{MSP}\}$-hybrid model.*

Proof. Let us first prove correctness. Observe that in an honest execution, P_1 holds m_{evk}, and all encrypted private inputs and encrypted magic states under Clifford code at the end of Step 2. Furthermore, it is clear that executions of π_{ReEnc} transform transforms all Clifford ciphertexts into trap-code ciphertexts with respect to a new random Pauli one-time pad and a proper permutations (depending on whether it re-encrypts a gadget or not) as the keys. Moreover, P_1 gets all classical information for each re-encrypted ciphertexts from cMPC during the execution of π_{ReEnc}, and cMPC gets sk, which includes all the keys to the trap-code ciphertext, by running $M_1^{\mathsf{KG}}(1^\kappa)$. Since the keys generated by the functionality are distributed the same, we conclude that the output of P_1 and cMPC in the real world is identical.

Let $\mathcal{A}$ be an adversary corrupting a subset $\mathcal{I}$ of the parties, holding auxiliary input ρ_{aux}. Observe that if $P_1 \notin \mathcal{I}$ is honest, then $\mathcal{A}$ receive no messages. Thus simulation in this case is trivial. Now, suppose $P_1 \in \mathcal{I}$ is corrupted. We next construct the simulator $\mathsf{Sim}_{\mathcal{A}}$.

Simulator 5 Simulator $\mathsf{Sim}_{\mathcal{A}}$

---

1. Query the adversary for the inputs $(\rho_i)_{i \in \mathcal{I}}$ for AR and forwards it to T.

2. Let $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ be the output received from T.

---

[15]Recall that the input and output qubit (labelled by 1st and 2nd respectively) of the gadget are encrypted under the global permutation Π_0, while the other qubits in the gadget are encrypted under the local permutation Π_i for i^{th} gadget.

3. Parse each entry in $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ into its classical part and its quantum part, that is, write

$$\hat{\rho} = (m_{ij}, \tau_{ij})_{i \in [n], j \in [\ell_{\mathsf{in}}]},$$

$$\hat{\rho}_0 = \left(m_i^Z, \tau_i^Z\right)_{i=1}^L,$$

$$\rho_{\mathsf{evk}} = \left(m_{\mathsf{evk}}, (m_i^{\mathsf{ms}}, \tau_i^{\mathsf{ms}})_{i \in [L], \mathsf{ms} \in \mathcal{M} \setminus \{Z\}}\right).$$

4. For each quantum part τ_{ij} of $\hat{\rho}$, where $i \in [n]$ and $j \in [\ell_{\mathsf{in}}]$, sample $V_{ij} \leftarrow \mathcal{C}_{3\kappa}$ and compute $\sigma_{ij} = V_{ij}^\dagger \tau_{ij} V_{ij}$.

5. Similarly, for each quantum part τ_i^{ms}, where $i \in [L]$ and $\mathsf{ms} \in \mathcal{M}$, sample $V_i^{\mathsf{ms}} \leftarrow \mathcal{C}_{3\kappa}$ and compute $\sigma_i^{\mathsf{ms}} = (V_i^{\mathsf{ms}})^\dagger \tau_i^{\mathsf{ms}} V_i^{\mathsf{ms}}$.

6. Send to $\mathcal{A}$ the values $(\sigma_{ij})_{i, \in [n], j \in [\ell_{\mathsf{in}}]}$ as the output of AR. If $\mathcal{A}$ replied with $(\texttt{abort}, \mathrm{P})$, send it to T, output whatever $\mathcal{A}$ outputs, and halt.

7. Otherwise, let $\mathcal{R}_{\mathsf{in}}$ be the set $\mathcal{A}$ sends to the AR ideal model.

8. For all $\mathsf{ms} \in \mathcal{M}$ and the ith call to $\mathsf{MSP}_{\mathsf{ms}}$, send σ_i^{ms} to $\mathcal{A}$ as the output of $\mathsf{MSP}_{\mathsf{ms}}$. If at any of the called, $\mathcal{A}$ replied with $(\texttt{abort}, \mathrm{P})$, send it to T, output whatever $\mathcal{A}$ outputs, and halt.

9. Send m_{evk} to $\mathcal{A}$.

10. Simulate each execution of π_{ReEnc}. This is done by simply sending the V_{ij}'s and V_i^{ms}.

11. Send $(\texttt{continue}, \mathcal{R}_{\mathsf{in}})$ to T, output whatever $\mathcal{A}$ outputs, and halt.

Clearly, each message $\mathcal{A}$ receive from $\mathsf{Sim}_{\mathcal{A}}$ is identical to whatever it received in the real world. Additionally, if no aborts occurred, then in the ideal world, by applying the Cliffords V's sampled by the simulator, the Clifford ciphertexts sent to $\mathcal{A}$ can be transformed into the correct output, which includes encryptions whose keys are sent to cMPC by T. Moreover, as the only packets that were dropped are those indicated by $\mathcal{R}_{\mathsf{in}}$, it follows that

$$\mathrm{IDEAL}_{\mathsf{PreComp}, \mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})}^{(n^2, 0)\text{-IDPD}}\left(\kappa, (\rho_i)_{i=1}^n\right) = \mathrm{HYBRID}_{\pi_{\mathsf{PreComp}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}, \mathsf{AR}, \mathsf{MSP}}\left(\kappa, (\rho_i)_{i=1}^n\right).$$

$\square$

9 Secure Delegation of The Computation – Computation

In the previous section we introduced the $\mathsf{PreComp}$ protocol for generating a key of TrapTP and outputs to the server P_1 encryptions of all inputs. In this section, we present a protocol that securely implements evaluation and verified decryption. This in turn, allows P_1 to perform the circuit evaluation, while ensuring to the other parties that the evaluation was done correctly. Looking ahead, as the server would need to distribute the outputs, unlike in $\mathsf{PreComp}$, the resulting encrypted value held by the server would be under Clifford code. In Section 10 below, we will show how to securely distribute these Clifford ciphertexts.

Formally, let C be an n-ary circuit. Define the functionality Comp as follows. Let ℓ_{in} and ℓ_{out} be the number of input-qubits and output-qubits, respectively, of each party P_i. Denote the input of

P_i as ρ_i. An output is given only to the server P_1 and cMPC, as follows. cMPC receives a uniform random Clifford for each output-qubit, namely it receives $(E_{ij})_{(i,j)\in[n]\times[\ell_{\mathsf{out}}]}$ where $E_{ij} \leftarrow \mathcal{C}_{1+n^2\kappa}$ are sampled independently and uniformly at random. The server P_1 receives the Clifford encryptions of each of the output-qubits encrypted with the Cliffords given to cMPC. That is, P_1 receives $\mathsf{CAuth.Enc}_{E_{ij}}(\sigma_{ij})$, where for all $i \in [n]$ and $j \in [\ell_{\mathsf{out}}]$ it holds that $\sigma_{ij} \in \mathcal{D}^1$ is a single qubit, and these are defined as $(\sigma_{ij})_{(i,j)\in[n]\times[\ell_{\mathsf{out}}]} = C(\rho_1,\ldots,\rho_n)$.

Roughly, our protocol works as follows. First, the parties prepare the values required to run TrapTP.Eval. That is, they generate keys and ciphertexts of their inputs under TrapTP using PreComp. The server P_1 can now run TrapTP.Eval on the ciphertexts to homomorphically evaluate the circuit, obtaining the outcome $\hat{\sigma}_{ij}$ for every party i and qubit j (possibly after some qubits where dropped). As $\hat{\sigma}_{ij}$ is encrypted using TrapTP, the parties now need to re-encrypt it to a Clifford ciphertext. To do this, the parties call AR, to have the server route the encrypted results *to itself*. This results in a Clifford ciphertext τ_{ij} of a trap code ciphertext of the outputs.

We now explain how cMPC can verify the computation. Let us first recall two important properties of TrapTP.Eval and TrapTP.VerDec. Recall that TrapTP.Eval, in addition to performing a computation over trap codes, if further produces a log of the computation, that includes all the classical messages including randomness, computation steps, and all intermediate results during evaluation. Next recall, that although TrapTP.VerDec is a quantum procedure, it includes a classical subroutine that verifies these logs. We denote this subroutine by CheckLogs. It is given a secret key sk (generated from TrapTP.KeyGen used in PreComp) and a log to be checked, and outputs updated Pauli keys and a flag to indicate whether the computation was performed faithfully. We refer the reader to [ADSS17] for a detailed construction of the classical algorithm.

Now, P_1 sends log to cMPC who applies CheckLogs to check validity. Then, the server split the trap registers into Z_1 and Z_2 for each ciphertext τ_{ij}, and cMPC sends it the Clifford

$$V_{ij} = (E_{ij}^{MZ_2} \otimes R_{ij}^S \otimes \mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{TZ_1})\mathsf{TAuth.PDec}_{\Pi_0,P_{ij}}^{MST}F_{ij}^\dagger.$$

Here, the first term $F_{ij}^\dagger$ would remove the Clifford encryption added by AR, resulting in a trap code ciphertext. Then the partial decryption of the trap code TAuth.PDec is applied (see Definition 2.15), using the global permutation Π_0 (written in sk), and the Pauli P_{ij} it got from checking the logs using CheckLogs. This converts the trap code ciphertext into a plaintext in register M, traps to be verified in register T, and the syndrome in register S (recall that trap codes use QECC in their construction). The term E_{ij} is a new Clifford that re-encrypts the plaintexts under a Clifford code, the term R_{ij} is a random Pauli that overwrites the syndromes to prevent leak of information, and $\mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}$ perform partial encryption of the trap code (see Definition 2.16), to re-encrypt the traps in register T under trap code with a newly sampled key. The server is then asked to homomorphically measure these traps and send the measurement results to cMPC to verify, who aborts if the verification failed. Specifically, the verification compares the measured traps to 0's. If the protocol does not abort, the server outputs the computation results that is now under a Clifford code, and cMPC outputs the corresponding keys.

We now formally present the protocol π_{Comp} in the $\{\mathsf{cMPC}, \mathsf{PreComp}, \mathsf{AR}\}$-hybrid model that computes Comp with (n^2, n^2)-IDPD security.

Protocol 13 Protocol π_{Comp} for computing Comp

Inputs: Each party P_i holds private inputs $\rho_i = (\rho_{ij})_{j=1}^{\ell_{\mathsf{in}}}$, where each $\rho_{ij} \in \mathcal{D}^1$.
Common input: The parties hold the security parameter 1^κ.

1. The parties call $(\mathsf{PreComp}_L, (n^2, 0)\text{-IDPD})$ with level parameter L taken to be the maximum

between the size of C and number of ancilla $|0\rangle$ states it uses, and with ρ_i as the input of P_i.

- If the output is $(\texttt{abort}, P)$, then the parties output $(\texttt{abort}, P)$ and halt.
- Otherwise, all parties receive $(\texttt{continue}, \mathcal{R}_{\textsf{in}})$, where $|\mathcal{R}_{\textsf{in}}| \geq n\ell_{\textsf{in}} - n^2$, P_1 additionally receives encryptions and evaluation key $(\hat{\rho}, \hat{\rho}_0, \rho_{\textsf{evk}})$, and $\textsf{cMPC}$ receives the secret key $\textsf{sk}$.

2. The server P_1 compute $\textsf{TrapTP.Eval}(C, \rho_{\textsf{evk}}, \hat{\rho} \otimes \hat{\rho}_0)$ and obtains $\left((\hat{\sigma}_{ij})_{(i,j)\in[n]\times[\ell_{\textsf{out}}]}, \textsf{log}\right)$. It then sends $\textsf{log}$ to $\textsf{cMPC}$.

3. The parties call $(\textsf{AR}, (n^2, 0)\text{-IDPD})$ with trap-size $t = 6\kappa^2 - 2\kappa + n^2\kappa$, with P_1 as the receiver and on input $(\hat{\sigma}_{ij})_{(i,j)\in[n]\times[\ell_{\textsf{out}}]}$.

- If the output is $(\texttt{abort}, P)$, then the parties output $(\texttt{abort}, P)$ and halt.
- Otherwise, all parties receive $(\texttt{continue}, \mathcal{R}_{\textsf{in}}^{\textsf{AR}})$. Additionally, $\textsf{cMPC}$ receives Cliffords $(F_{ij})_{(i,j)\in\mathcal{R}_{\textsf{in}}^{\textsf{AR}}}$, where $F_{ij} \in \mathcal{C}_{6\kappa^2+(n^2+1)\kappa}$[16] for all $(i,j) \in \mathcal{R}_{\textsf{in}}^{\textsf{AR}}$, and P_1 receives the Clifford ciphertexts of the trap-code ciphertexts $(\hat{\sigma}_{ij})_{(i,j)\in\mathcal{R}_{\textsf{in}}^{\textsf{AR}}}$, denoted $(\tau_{ij})_{(i,j)\in\mathcal{R}_{\textsf{in}}^{\textsf{AR}}}$. Note that $\mathcal{R}_{\textsf{in}}^{\textsf{AR}}$ and will be used to indicated dropped *output-qubits*.

4. P_1 partitions the trap register of each τ_{ij} into two registers $Z_{i,j,1}$ and $Z_{i,j,2}$ of sizes $6\kappa^2 - 2\kappa$ and $n^2\kappa$ qubits, respectively. When clear from context we remove the i's and j's from the subscripts to alleviate the notation.

5. Define C' as the circuit that first applies C and then discards the qubits outside of $\mathcal{R}_{\textsf{in}}^{\textsf{AR}}$.

6. $\textsf{cMPC}$ creates $\textsf{log}'$ for the homomorphic evaluation of C' as follows. Recall that $\textsf{TrapTP.Eval}$ is done gate-by-gate, generating $\textsf{log}$ at each step. $\textsf{log}'$ is then defined to be the parts in $\textsf{log}$ that corresponds to the gates whose output is not discarded by C'. $\textsf{cMPC}$ then computes $\left((P_{ij})_{(i,j)\in\mathcal{R}_{\textsf{in}}^{\textsf{AR}}}, \textsf{flag}_{\textsf{log}}\right) = \textsf{CheckLogs}(\textsf{sk}, \textsf{log}')$.

7. For every $(i,j) \in \mathcal{R}_{\textsf{in}}^{\textsf{AR}}$, $\textsf{cMPC}$ does the following.

   (a) Sample a permutation $\Pi_{ij} \leftarrow \textsf{Sym}_{6\kappa^2}$, Paulis $Q_{ij} \leftarrow \mathcal{P}_{6\kappa^2}$ and $R_{ij} \leftarrow \mathcal{P}_{\kappa-1}$, and a Clifford $E_{ij} \leftarrow \mathcal{C}_{1+n^2\kappa}$, independently and uniformly at random.
   (b) Send to P_1 the Clifford

   $$V_{ij} = (E_{ij}^{MZ_2} \otimes R_{ij}^{S} \otimes \textsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{TZ_1})\textsf{TAuth.PDec}_{\Pi_0,P_{ij}}^{MST} F_{ij}^{\dagger},$$

   where Π_0 is the global permutation of evaluation written in $\textsf{sk}$.

8. For all $(i,j) \in \mathcal{R}_{\textsf{in}}^{\textsf{AR}}$ the server does the following.

   (a) Compute $\tilde{\sigma}_{ij}^{MZ_2STZ_1} = V_{ij}\tau_{ij}V_{ij}^{\dagger}$
   (b) Transversally measure $\tilde{\sigma}_{ij}^{TZ_1}$ to obtain $\tilde{z}_{ij}$ and send it to $\textsf{cMPC}$.

9. $\textsf{cMPC}$ computes $(\textsf{flag}_{ij}, z_{ij}) = \textsf{TAuth.Dec}_{\Pi_{ij},Q_{ij}}(\tilde{z}_{ij})$ for each $(i,j) \in \mathcal{R}_{\textsf{in}}^{\textsf{AR}}$. If $\textsf{flag}_{ij} = \textsf{Rej}$ or $z_{ij} \neq 0^{2\kappa}$ for some $(i,j) \in \mathcal{R}_{\textsf{in}}^{\textsf{AR}}$, or $\textsf{flag}_{\textsf{log}} = \textsf{Rej}$, then it sends $(\texttt{abort}, P_1)$ to all parties, they output it, and halt.

---

[16] Recall that each $\hat{\sigma}_{ij} \in \mathcal{D}^{3\kappa}$, and that given an m-size input and trap-size t, $\textsf{AR}$ outputs Cliffords operating over $m + t$ qubits.

10. Otherwise, all parties and cMPC outputs $\mathcal{R}_{\mathsf{in}}$ and $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$. The server P_1 additionally outputs $\left(\tilde{\sigma}_{ij}^{MZ_2}\right)_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$, and cMPC additionally outputs the Clifford keys $(E_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$.

Lemma 9.1. *Protocol π_{Comp} computes Comp with computational (n^2, n^2)-IDPD security in $\{\mathsf{cMPC}, \mathsf{PreComp}, \mathsf{AR}\}$-hybrid model.*

Proof. We first show that the protocol is correct. Observe that after running $\mathsf{PreComp}$, the server P_1 receives $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ and cMPC receives sk. P_1 then evaluates

$$\left(\hat{\sigma}_{ij}^{MST}\right)_{i\in[n],j\in[\ell_{\mathsf{out}}]} = \mathsf{TrapTP.Eval}(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}}).$$

By the correctness of $\mathsf{TrapTP.Eval}$ it follows that

$$\left(\left(\sigma_{ij}^{MST}\right)_{i\in[n],j\in[\ell_{\mathsf{out}}]}, \mathsf{Acc}\right) \approx_{\mathrm{neg}(\kappa)} \mathsf{TrapTP.VerDec}_{\mathsf{sk}}\left(\left(\hat{\sigma}_{ij}\right)_{i\in n,j\in\ell_{\mathsf{out}}}\right),$$

where $(\sigma_{ij})_{i\in[n],j\in[\ell_{\mathsf{out}}]} = C(\rho_1,\ldots,\rho_n)$ are the outputs of the circuit. Moreover, as TrapTP is trap-code based, there exists a permutation Π and Paulis $(P_{ij})_{i\in[n],j\in[\ell_{\mathsf{out}}]}$ such that for every $(i,j) \in [n] \times [\ell_{\mathsf{out}}]$, it holds that

$$\hat{\sigma}_{ij} \approx_{\mathrm{neg}(\kappa)} \mathsf{TAuth.Enc}_{\Pi,P_{ij}}(\sigma_{ij}).$$

Recall that by the definition of $\mathsf{PreComp}$, Π is a part of sk held by cMPC. Furthermore, cMPC can compute the P_{ij}'s by checking the logs, i.e.,

$$\left((P_{ij})_{(i,j)\in[n]\times[\ell_{\mathsf{out}}]}, \mathsf{Acc}\right) = \mathsf{CheckLogs}(sk, \log').$$

Following this, P_1 sends $(\hat{\sigma}_{ij})_{(i,j)\in[n]\times[\ell_{\mathsf{out}}]}$ to AR, and receives

$$\mathsf{CAuth.Enc}_{F_{ij}}(\hat{\sigma}_{ij}) = F_{ij}\left(\hat{\sigma}_{ij} \otimes |0^{6\kappa^2-2\kappa+n^2\kappa}\rangle\langle 0^{6\kappa^2-2\kappa+n^2\kappa}|^{Z_1 Z_2}\right) F_{ij}^{\dagger}$$

for every $(i,j) \in [n] \times [\ell_{\mathsf{out}}]$. Then, at Step 7b, cMPC sends to P_1 the Cliffords

$$V_{ij} = W_{ij}\mathsf{TAuth.PDec}_{\Pi_0,P_{ij}}^{MST}F_{ij}^{\dagger},$$

for every $(i,j) \in [n] \times [\ell_{\mathsf{out}}]$, where

$$W_{ij} = E_{ij}^{MZ_2} \otimes R_{ij}^S \otimes \mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{TZ_1}.$$

Observe that by applying V_{ij} to the $(i,j)^{\mathrm{th}}$ output of AR results in the following state.

$$\begin{aligned}
\tilde{\sigma}_{ij}^{MSTZ_1Z_2} &= V_{ij}F_{ij}\left(\sigma_{ij}^{MST} \otimes |0^{6\kappa^2-2\kappa+n^2\kappa}\rangle\langle 0^{6\kappa^2-2\kappa+n^2\kappa}|^{Z_1 Z_2}\right) F_{ij}^{\dagger}V_{ij}^{\dagger} \\
&= W_{ij}\mathsf{TAuth.PDec}_{\Pi_0,P_{ij}}^{MST}\left(\sigma_{ij}^{MST} \otimes |0^{6\kappa^2-2\kappa+n^2\kappa}\rangle\langle 0^{6\kappa^2-2\kappa+n^2\kappa}|^{Z_1 Z_2}\right) \mathsf{TAuth.PDec}_{\Pi_0,P_{ij}}^{\dagger}W_{ij}^{\dagger} \\
&= W_{ij}\left(\sigma_{ij}^M \otimes |0^{\kappa-1}\rangle\langle 0^{\kappa-1}|^S \otimes |0^{2\kappa}\rangle\langle 0^{2\kappa}|^T \otimes |0^{6\kappa^2-2\kappa+n^2\kappa}\rangle\langle 0^{6\kappa^2-2\kappa+n^2\kappa}|^{Z_1 Z_2}\right) W_{ij}^{\dagger} \\
&= E_{ij}^{MZ_2}\left(\sigma_{ij}^M \otimes |0^{n^2\kappa}\rangle\langle 0^{n^2\kappa}|^{Z_2}\right) E_{ij}^{\dagger} \otimes \left(\chi^{\otimes\kappa-1}\right)^S \otimes \mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{TZ_1}\left(|0^{6\kappa^2}\rangle\langle 0^{6\kappa^2}|\right) \mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{\dagger} \\
&= \mathsf{CAuth.Enc}_{E_{ij}}(\sigma_{ij}) \otimes \chi^{\otimes\kappa-1} \otimes \mathsf{TAuth.Enc}_{\Pi_{ij},Q_{ij}}\left(|0^{2\kappa^2}\rangle\langle 0^{2\kappa^2}|\right),
\end{aligned}$$

68

where the penultimate equality follows from that fact that $R_{ij} |0^{\kappa-1}\rangle\langle 0^{\kappa-1}| R_{ij}^\dagger = \chi^{\otimes\kappa-1}$ due to Lemma 2.2.

Then, P_1 measures $\tilde{\sigma}^{TZ_1} = \mathsf{TAuth.Enc}_{\Pi_{ij},Q_{ij}}\left(|0^{2\kappa^2}\rangle\langle 0^{2\kappa^2}|\right)$ to obtain a classical string $\tilde{z}_{ij}$ and sends it to cMPC. By the correctness of homomorphic measurement over trap code given in Theorem 2.18, cMPC will obtain

$$(0^{\kappa-1}, \mathsf{Acc}) = \mathsf{TAuth.Dec}_{\Pi_{ij},Q_{ij}}(\tilde{z}_{ij})$$

at Step 9. Thus, cMPC outputs the Clifford keys of the encryptions that P_1 outputs, concluding correctness.

We now prove the security. Let $\mathcal{A}$ be an adversary corrupting a subset $\mathcal{I}$ of the parties, holding auxiliary input ρ_{aux}. Observe that if the server is honest, then the adversary receives no messages throughout the whole interaction. By correctness, it follows that simulating this case is trivial. Thus, we focus on the case where $P_1 \in \mathcal{I}$. We next construct the simulator $\mathsf{Sim}_\mathcal{A}$. It roughly does as follows. It begins by collecting the corrupted parties' inputs to $\mathsf{PreComp}$, sending them to the trusted party T as inputs, and sending encryptions of $|0\rangle$'s to P_1 as the output of $\mathsf{PreComp}$. The adversary is then asked to apply the circuit C homomorphically and send its evaluation outcomes to AR. Now, $\mathsf{Sim}_\mathcal{A}$ can collect and verify the outcome and abort if needed. The adversary then expects to receive packets τ_{ij} from AR and Cliffords V_{ij} from cMPC, so that $V_{ij}\tau_{ij}V_{ij}^\dagger$ is a tensor product of its output $\tilde{\sigma}_{ij}$ together with a trap-code ciphertexts $\tilde{\tau}_{ij}$ that it is suppose to measure, as well as used qubits $\chi^{\otimes\kappa-1}$ that should be discarded. Thus, we let the simulator sample the Cliffords V_{ij} and send them to $\mathcal{A}$ alongside the states

$$V_{ij}^\dagger \left(\tilde{\sigma}_{ij} \otimes \chi^{\otimes\kappa-1} \otimes \mathsf{TAuth.Enc}_{\Pi_{ij},P_{ij}}(|0^{2\kappa}\rangle\langle 0^{2\kappa}|)\right) V_{ij},$$

as the output of AR. Finally, $\mathsf{Sim}_\mathcal{A}$ receives $\tilde{z}_{ij}$ from the server and verifies it similarly to cMPC, and aborts if needed. We now formally state our construction.

Simulator 6 $\mathsf{Sim}_\mathcal{A}$

---

1. Query $\mathcal{A}$ for its inputs $(\rho_i)_{i\in\mathcal{I}}$ to $\mathsf{PreComp}$, and the set $\mathcal{R}_{\mathsf{in}}$ of remaining input-qubits, and forward them to T.

2. Let $(\tilde{\sigma}_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}}$ be the output received.

3. Generate keys $(\mathsf{sk}, \rho_{\mathsf{evk}}) \leftarrow \mathsf{TrapTP.KeyGen}(1^\kappa, 1^L)$, dummy ciphertexts $\hat{\rho} \leftarrow \mathsf{TrapTP.Enc}_{\mathsf{sk}}^{\otimes n\ell_{\mathsf{in}}}(|0^{n\ell_{\mathsf{in}}}\rangle\langle 0^{n\ell_{\mathsf{in}}}|)$, and ciphertexts of 0's $\hat{\rho}_0 \leftarrow \mathsf{TrapTP.Enc}_{\mathsf{sk}}^{\otimes L}(|0^L\rangle\langle 0^L|)$.

4. Send $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ to $\mathcal{A}$ as the output of $\mathsf{PreComp}$. If $\mathcal{A}$ replies with $(\mathtt{abort}, P)$, send it to T, output whatever $\mathcal{A}$ outputs, and halt.

5. Otherwise, $\mathcal{A}$ sends $\mathtt{continue}$, and additionally sends log that was purportedly generated by $\mathsf{TrapTP.Eval}$.

6. Query the adversary to receive its input $(\hat{\sigma}_{ij})_{(i,j)\in[n]\times[\ell_{\mathsf{out}}]}$ for AR, and the $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ of remaining *input-qubits* for AR.

7. Evaluate $\mathsf{TrapTP.VerDec}((\hat{\sigma}_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}})$. This returns a flag, denoted $\mathtt{flag}$, and additional states to be discarded.

8. For all $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$, sample a Clifford $V_{ij} \leftarrow \mathcal{C}_{6\kappa^2+(n^2+1)\kappa}$, a permutation $\Pi_{ij} \leftarrow \mathrm{Sym}_{6\kappa^2}$, and a Pauli $P_{ij} \leftarrow \mathcal{P}_{6\kappa^2}$.

9. Send to $\mathcal{A}$ the Clifford encryptions

$$\tau_{ij} = V_{ij}^\dagger \left(\tilde{\sigma}_{ij} \otimes \chi^{\otimes \kappa - 1} \otimes \mathsf{TAuth.Enc}_{\Pi_{ij}, P_{ij}}(|0^{2\kappa}\rangle \langle 0^{2\kappa}|) \right) V_{ij}$$

as the output of AR. If $\mathcal{A}$ replies with $(\mathtt{abort}, \mathrm{P})$, send it to T, output whatever $\mathcal{A}$ outputs, and halt.

10. Otherwise, $\mathcal{A}$ replies with set $\mathtt{continue}$.

11. The adversary replies with purported measurement results $(\tilde{z}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$.

12. For each $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$, compute $\mathsf{TAuth.Dec}_{\Pi_{ij}, P_{ij}}(\tilde{z}_{ij})$. This returns a flag, denoted $\mathtt{flag}_{ij}$, and additional states to be discarded.

13. If $\mathtt{flag} = \mathsf{Rej}$ or any $\mathtt{flag}_{ij} = \mathsf{Rej}$, then send $(\mathtt{abort}, \mathrm{P}_1)$ to T. Otherwise send $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}})$ to T.

14. Output whatever $\mathcal{A}$ outputs and halt.

We now prove that

$$\left\{ \mathrm{IDEAL}_{C, \mathsf{Sim}_{\mathcal{A}}(\rho_{\mathsf{aux}})}^{(n^2, n^2)\text{-}\mathsf{IDPD}} \left(\kappa, (\rho_i)_{i=1}^n \right) \right\}_{\kappa, \rho_1, \ldots, \rho_n, \rho_{\mathsf{aux}}} \stackrel{\mathrm{C}}{\equiv} \left\{ \mathrm{HYBRID}_{\pi_{\mathsf{Comp}}, \mathcal{A}(\rho_{\mathsf{aux}})}^{\mathsf{cMPC}, \mathsf{PreComp}, \mathsf{AR}} \left(\kappa, (\rho_i)_{i=1}^n \right) \right\}_{\kappa, \rho_1, \ldots, \rho_n, \rho_{\mathsf{aux}}}. \tag{16}$$

Fix a non-uniform QPT distinguisher D and inputs $\rho_1, \ldots, \rho_n$ for π_{Comp}. We prove Equation (16) by reducing to the security of the VQFHE scheme. Recall that the security of VQFHE is defined using the $\mathsf{VerGame}$. In the following, we denote by $\mathsf{D_H}$ the probability D outputs 1 when given the output of the protocol, and denote by $\mathsf{D_I}$ the probability D outputs 1 when given the output of the ideal world. The next claim (proven below) asserts the existence of an attacker for TrapTP with certain properties.

Claim 9.2. *There exists an adversary $\mathcal{A}^*$ for TrapTP satisfying*

$$\mathsf{D_H} \approx_{\mathrm{neg}(\kappa)} \mathrm{VERGAME}_{\mathcal{A}^*}\big|_{r=0}$$

and

$$\mathsf{D_I} \approx_{\mathrm{neg}(\kappa)} \mathrm{VERGAME}_{\mathcal{A}^*}\big|_{r=1},$$

where $\mathrm{VERGAME}_{\mathcal{A}^*}\big|_{r=r'}$ is the probability $\mathcal{A}^*$ outputs r in $\mathsf{VerGame}_{\mathsf{TrapTP}, \mathcal{A}^*}$ conditioned on the challenger $\mathfrak{C}$ using $r = r'$.

Clearly, Equation (16) follows from Claim 9.2 and the security of TrapTP stating that[17]

$$\left\{ \mathrm{VERGAME}_{\mathcal{A}^*}\big|_{r=0} \right\}_{\kappa \in \mathbb{N}, \rho_1, \ldots, \rho_n, \rho_{\mathsf{aux}} \in \mathcal{D}^*} \stackrel{\mathrm{C}}{\equiv} \left\{ \mathrm{VERGAME}_{\mathcal{A}^*}\big|_{r=1} \right\}_{\kappa \in \mathbb{N}, \rho_1, \ldots, \rho_n, \rho_{\mathsf{aux}} \in \mathcal{D}^*}.$$

$\square$

We now prove the claim.

---

[17][ADSS17] considered only uniform adversaries. We note that their construction is secure against non-uniform adversaries if there exists a post-quantum classical homomorphic encryption scheme.

Proof of Claim 9.2. Roughly, $\mathcal{A}^*$ collects the inputs $(\rho_i^*)_{i \in \mathcal{I}}$ that the protocol's adversary $\mathcal{A}$ sends to PreComp as well as the set $\mathcal{R}_{\mathsf{in}}$, and replaces the dropped inputs with $|0\rangle$. It then sends the inputs to the challenger $\mathfrak{C}$, who replies with encryptions of each input sent to it. Then, to apply the homomorphic evaluation, $\mathcal{A}^*$ forwards these ciphertexts to $\mathcal{A}$ who sends back its inputs to AR. These are then forwarded to $\mathfrak{C}$ to learn whether the outcomes can be decrypted successfully. Finally, $\mathcal{A}^*$ emulates the rest of π_{Comp} in a way that is similar to $\mathsf{Sim}_{\mathcal{A}}$, and runs the distinguisher D on the outcomes. Formally, $\mathcal{A}^*$ is described as follows.

Algorithm 13 Adversary $\mathcal{A}^*$ for $\mathsf{VerGame}_{\mathsf{TrapTP}}$

---

1. Upon receiving ρ_{evk} from the challenger, query $\mathcal{A}$ for its inputs $(\rho_i^*)_{i \in \mathcal{I}}$ to PreComp, and the set $\mathcal{R}_{\mathsf{in}}$ of remaining input qubits. For each $(i, j) \in [n] \times [\ell_{\mathsf{in}}]$, send

$$
\rho_{ij}' = \begin{cases} |0\rangle & \text{if } (i, j) \notin \mathcal{R}_{\mathsf{in}} \\ \rho_{ij} & \text{if } (i, j) \in \mathcal{R}_{\mathsf{in}} \text{ and } i \notin \mathcal{I} \\ \rho_{ij}^* & \text{if } (i, j) \in \mathcal{R}_{\mathsf{in}} \text{ and } i \in \mathcal{I} \end{cases}
$$

and $|0^L\rangle$ to the challenger.

2. Receive $(\hat{\rho}, \hat{\rho}_0)$ from the challenger and do the following.

   (a) Forward $(\hat{\rho}, \hat{\rho}_0, \rho_{\mathsf{evk}})$ to $\mathcal{A}$ as its outputs of PreComp.

   (b) The adversary $\mathcal{A}$ then replies with either $(\mathtt{abort}, \mathrm{P})$ or $\mathtt{continue}$. If $\mathcal{A}$ sent $(\mathtt{abort}, \mathrm{P})$, then query it for its output, forward $(\mathtt{abort}, \mathrm{P})$ and the output of $\mathcal{A}$ to D, output whatever D outputs, and halt.

   (c) Otherwise, if $\mathcal{A}$ sent $\mathtt{continue}$, query it for $\log$ that was purportedly generated by $\mathsf{TrapTP.Eval}$.

   (d) Query $\mathcal{A}$ for its inputs $(\hat{\sigma}_{ij})_{(i,j) \in [n] \times [\ell_{\mathsf{out}}]}$ to AR, and the set $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ of remaining input-qubits for the functionality.

   (e) Define C' and $\log'$ similarly to the cMPC and $\mathsf{Sim}_{\mathcal{A}}$, and send $((\hat{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}, C', \log')$ to the challenger.

3. Receive $(\sigma, \mathtt{flag})$ from the challenger and do the following.

   (a) If $\mathtt{flag} = \mathsf{Rej}$ then set $\sigma = |0^{|\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}|}\rangle\langle 0^{|\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}|}|$.

   (b) Write $\sigma = (\sigma_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$, where each $\sigma_{ij} \in \mathcal{D}^1$.

   (c) For all $(i, j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$, sample $E_{ij} \leftarrow \mathcal{C}_{1+\kappa n^2}$, $\Pi_{ij} \leftarrow \mathsf{Sym}_{6\kappa^2}$, $Q_{ij} \leftarrow \mathcal{P}_{6\kappa^2}$, and $V_{ij} \leftarrow \mathcal{C}_{(n^2+1)\kappa + 6\kappa^2}$.

   (d) Send

$$
\tau_{ij} = V_{ij}^\dagger \left(\mathsf{CAuth.Enc}_{E_{ij}}(\sigma_{ij}) \otimes \chi^{\otimes \kappa - 1} \otimes \mathsf{TAuth.Enc}_{\Pi_{ij}, Q_{ij}}(|0^{2\kappa}\rangle\langle 0^{2\kappa}|) \right) V_{ij}
$$

   to $\mathcal{A}$ as outputs of AR, where $\chi^{\otimes \kappa - 1}$ is a $(\kappa - 1)$-qubit maximally mixed state.

   (e) Query $\mathcal{A}$ for either $(\mathtt{abort}, \mathrm{P})$ or $\mathtt{continue}$ to AR. If it sends $(\mathtt{abort}, \mathrm{P})$, then query it for its outputs, then forward its outputs and $(\mathtt{abort}, \mathrm{P})$ to D and output whatever D outputs.

(f) Otherwise, if $\mathcal{A}$ sends `continue`, for each $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$, send $(V_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$ to $\mathcal{A}$

(g) $\mathcal{A}$ replies with purported measurements $\tilde{z}_{ij}$. Compute $(\mathtt{flag}_{ij}', z_{ij}) \leftarrow \mathsf{TAuth.Dec}_{\Pi_{ij},Q_{ij}}(\tilde{z}_{ij})$. If any $\mathtt{flag}_{ij}' = \mathsf{Rej}$ or $\mathtt{flag} = \mathsf{Rej}$ then send to D the output $(\mathtt{abort}, \mathrm{P}_1)$, output whatever D outputs, and halt.

(h) Otherwise, query $\mathcal{A}$'s output and forward it and $(\mathcal{R}_{\mathsf{in}}, \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}, (E_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}})$ to D, output whatever D outputs and halt.

Recall that if $r = 1$, then the challenger sends to $\mathcal{A}^*$ dummy encryptions of 0's. Thus, the output that $\mathcal{A}^*$ submits to D is the same as in the ideal world, hence $\mathsf{D}_\mathrm{I} = \textsc{VerGame}_{\mathcal{A}^*}\big|_{r=1}$. It is left to show that $\mathsf{D}_\mathrm{H} \approx_{\mathrm{neg}(\kappa)} \textsc{VerGame}_{\mathcal{A}^*}\big|_{r=0}$. First, recall that the adversary $\mathcal{A}$ for the protocol, can only do the following. It can change and drop some of the inputs when calling $\mathsf{PreComp}$, instruct P_1 to apply some arbitrary operation over the output it receives from $\mathsf{PreComp}$ (the honest operation would be applying $\mathsf{TrapTP.Eval}$), change and drop some of the inputs of P_1 to AR, and finally, send to cMPC different measurement results.

Notice that the messages $\mathcal{A}$ see in the protocol, are indistinguishable from the messages it receives from $\mathcal{A}^*$ in $\mathsf{VerGame}$. Thus, for the sake of simplifying the proof, we assume that $\mathcal{A}$ replies with the same messages. Moreover, if $\mathcal{A}$ aborted during either the call to $\mathsf{PreComp}$ or to AR, the joint output of it and the honest parties are the same as what $\mathcal{A}^*$ sent to D. Therefore we condition on the case where $\mathcal{A}$ did not abort.

Let us now analyze the real world. Recall that $(\tau_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$ are the outputs that $\mathcal{A}$ received from AR. Furthermore, each of these values are encrypted under a Clifford F_{ij}. We assume without loss of generality that $\mathcal{A}$ applied the V_{ij}'s honestly, followed by a unitary of the adversary's choice applied to the register and its auxiliary reference system. Define

$$\tau_{ij}' = \left(\mathsf{TAuth.PDec}_{\Pi_0,P_{ij}} F_{ij}^\dagger \tau_{ij} (\mathsf{TAuth.PDec}_{\Pi_0,P_{ij}} F_{ij}^\dagger)^\dagger \right)^T$$

to be the result in register T after applying the partial decoding to inputs sent to AR. Then for all $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ it holds that

$$\mathsf{TAuth.Enc}_{\Pi_{ij},Q_{ij}}^T \tau_{ij}' \left(\mathsf{TAuth.Enc}_{\Pi_{ij},Q_{ij}} \right)^\dagger$$
$$= \mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}}^{TZ_1} \mathsf{TAuth.PDec}_{\Pi_{ij},P_{ij}}^{MST} F_{ij}^\dagger \tau_{ij} \left(\mathsf{TAuth.PEnc}_{\Pi_{ij},Q_{ij}} \mathsf{TAuth.PDec}_{\Pi_0,P_{ij}} F_{ij}^\dagger \right)^\dagger .$$

We can now apply Lemma 2.18 on $(\tau_{ij}')_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$ with the reference system being MSZ_2R, where R is $\mathcal{A}$'s auxiliary reference system. Therefore, for all $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ there exists $q \in [0,1]$ and two CPTPs A_{Acc} and A_{Rej} acting on register MZ_2SR, such that the joint state held by cMPC and $\mathcal{A}$ is negligibly close to

$$q \cdot \mathop{\mathbb{E}}_{z_{ij}\leftarrow\Lambda(\tau_{ij}')} \left[(|\mathtt{flag}_{\mathsf{log}}\rangle\langle\mathtt{flag}_{\mathsf{log}}| \otimes z_{ij}) \otimes A_{\mathsf{Acc}} \left(V_{ij}\tau_{ij}V_{ij}^\dagger \right)^{MZ_2S} \otimes \langle z_{ij}|\tau_{ij}'|z_{ij}\rangle \right]$$
$$+ (1-q) \left[|\mathtt{flag}_{\mathsf{log}}\rangle\langle\mathtt{flag}_{\mathsf{log}}| \otimes |\bot_\mathrm{H}\rangle\langle\bot_\mathrm{H}| \otimes A_{\mathsf{Rej}} \left(V_{ij}\tau_{ij}V_{ij}^\dagger \right)^{MZ_2S} \right],$$

where $|\bot_\mathrm{H}\rangle\langle\bot_\mathrm{H}|$ represents the state of cMPC if there exists $(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ such that $\mathtt{flag}_{ij} = \mathsf{Rej}$.

Observe that we can write the expectation as follows.

$$\underset{z_{ij}\leftarrow\Lambda(\tau'_{ij})}{\mathbb{E}}\left[(\mathtt{flag_{log}}\otimes z_{ij})\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}\otimes\langle z_{ij}|\tau'_{ij}|z_{ij}\rangle\right]$$

$$=\Pr\left[\mathtt{flag_{log}}=\mathsf{Acc}\wedge z_{ij}=0\right]|\mathsf{Acc}\rangle\langle\mathsf{Acc}|\otimes|0\rangle\langle 0|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}$$

$$+\Pr\left[\mathtt{flag_{log}}=\mathsf{Rej}\wedge z_{ij}=0\right]|\mathsf{Rej}\rangle\langle\mathsf{Rej}|\otimes|0\rangle\langle 0|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}$$

$$+\sum_{z_{ij}\neq 0}\left[\Pr\left[\mathtt{flag_{log}}\wedge z_{log}\right]|\mathtt{flag_{log}}\rangle\langle\mathtt{flag_{log}}|\otimes|z_{ij}\rangle\langle z_{ij}|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}\right]. \qquad (17)$$

We now show that in $\mathsf{VerGame_{TrapTP,\mathcal{A}^*}}$, the state that $\mathcal{A}^*$ generates for cMPC and $\mathcal{A}$ (before sending to D) are the same. Indeed, if τ_{ij} is the state that $\mathcal{A}^*$ sent to $\mathcal{A}$ as the output of AR, then Lemma 2.18 the joint state prepared for $\mathcal{A}$ and cMPC is

$$\left(q\cdot\left[(|\mathtt{flag}\rangle\langle\mathtt{flag}|\otimes|0\rangle\langle 0|)\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}\right]\right.$$

$$\left.+(1-q)\left[(|\mathtt{flag}\rangle\langle\mathtt{flag}|\otimes|\perp_{\mathrm{V}}\rangle\langle\perp_{\mathrm{V}}|)\otimes A_{\mathsf{Rej}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}\right]\right)_{ij},$$

where $|\perp_{\mathrm{V}}\rangle\langle\perp_{\mathrm{V}}|$ represent the state generated for cMPC by $\mathcal{A}^*$ if there exists $(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$ such that $\mathtt{flag}'_{ij}=\mathsf{Rej}$. Moreover, q, A_{Acc} and A_{Rej} are also the same as in the real world, since $\mathcal{A}$ cannot distinguish the real world output of AR from what $\mathcal{A}^*$ generated.

Now, recall that the $(\sigma,\mathtt{flag})$ that the challenger $\mathfrak{C}$ sends to $\mathcal{A}^*$ at the beginning of Step 3 is computed as follows. $\mathfrak{C}$ first decodes the encryptions that $\mathcal{A}^*$ sent to it, i.e., it computes $\mathsf{TrapTP.VerDec}((\hat{\sigma}_{ij})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}},C',\mathsf{log}')$. This is done first checking the logs like cMPC does in the real world. Then it computes $\mathsf{TAuth.PDec}(\hat{\sigma}_{ij})$, measures the traps, decodes them using $\mathsf{TAuth.Dec}$, and compares them to 0, for all $(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$. If any of the verifications failed, it outputs $\mathtt{flag}=\mathsf{Rej}$. Let $\mathtt{flag_{log}^{\mathfrak{C}}}$ and $(z_{ij}^{\mathfrak{C}})_{(i,j)\in\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$ be the output of $\mathsf{CheckLogs}$ and $\mathsf{TAuth.Dec}$, respectively, as computed by $\mathfrak{C}$. Then $\mathtt{flag}=\mathsf{Acc}$ if and only if $\mathtt{flag_{log}^{\mathfrak{C}}}=\mathsf{Acc}$ and $z_{ij}^{\mathfrak{C}}=0$ for all i,j. Therefore

$$(|\mathtt{flag}\rangle\langle\mathtt{flag}|\otimes|0\rangle\langle 0|)\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}$$

$$=\Pr\left[\mathtt{flag_{log}^{\mathfrak{C}}}=\mathsf{Acc}\wedge z_{ij}^{\mathfrak{C}}=0\right]|\mathsf{Acc}\rangle\langle\mathsf{Acc}|\otimes|0\rangle\langle 0|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}$$

$$+\Pr\left[\mathtt{flag_{log}^{\mathfrak{C}}}=\mathsf{Rej}\wedge z_{ij}^{\mathfrak{C}}=0\right]|\mathsf{Rej}\rangle\langle\mathsf{Rej}|\otimes|0\rangle\langle 0|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}$$

$$+\sum_{z_{ij}^{\mathfrak{C}}\neq 0}\left[\Pr\left[\mathtt{flag_{log}^{\mathfrak{C}}}\wedge z_{log}^{\mathfrak{C}}\right]|\mathtt{flag_{log}}\rangle\langle\mathtt{flag_{log}}|\otimes|z_{ij}^{\mathfrak{C}}\rangle\langle z_{ij}^{\mathfrak{C}}|\otimes A_{\mathsf{Acc}}\left(V_{ij}\tau_{ij}V_{ij}^{\dagger}\right)^{MZ_2S}\right],$$

which equals to the expression computed in Equation (17). In particular, the subnormalized accept state is the same in both the real world and the $\mathsf{VerGame}$. Now, as given that state $|\perp_{\mathrm{V}}\rangle\langle\perp_{\mathrm{V}}|$ would cause $\mathcal{A}^*$ to abort, it follows that the subnormalized abort state is also the same. Therefore, the state given to D in the real world is negligibly close the the state it gets from $\mathcal{A}^*$.

$\square$

10 Secure Computation of a Quantum Circuit With Packet Drops

In this section, we are finally ready to present our protocol for computing an arbitrary quantum circuit C with IDPD-security. That is, we prove the following lemma.

Lemma 10.1 (Restatement of Lemma 4.2). *Assume the existence of a classical quantum-resistant fully homomorphic encryption scheme with decryption circuit of logarithmic depth. Let C be an n-ary quantum circuit. Then C can be computed with computational $(n^2, 2n^2)$-IDPD security in the* cMPC-*hybrid model. Moreover, the round complexity of the protocol is independent of the circuit depth.*

We next construct a protocol in the $\{\mathsf{cMPC}, \mathsf{Comp}\}$-hybrid model. Given the functionality Comp from the previous section, the protocol is rather simple. The parties first call Comp, which ensures that P_1 will hold a Clifford encoding of the output of each party, and cMPC will hold the keys. The parties then execute π_{CTAR} to route each output held by P_1 to the correct party. Finally, if the protocol did not yet abort, then cMPC will send the keys to each output to the corresponding party.

We now formally state our protocol in the $\{\mathsf{cMPC}, \mathsf{Comp}\}$-hybrid model.

Protocol 14 Protocol π for computing a circuit C

Inputs: Each party P_i holds input $\rho_i \in \mathcal{D}^{\ell_{\mathsf{in}}}$.
Common input: The parties hold the security parameter 1^κ.

1. The parties call $(\mathsf{Comp}, (n^2, n^2)\text{-}\mathsf{IDPD})$, with the input of P_i being ρ_i.

    - If the output is $(\mathtt{abort}, P)$, then the parties output $(\mathtt{abort}, P)$ and halt.
    - Otherwise, all parties receive $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}}, \mathcal{R}_{\mathsf{out}})$. Additionally, P_1 receives Cliffords encoding of the outputs $(\hat{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{out}}}$ and cMPC receives the corresponding Clifford keys $(E_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{out}}}$,

2. The parties execute π_{CTAR} (Protocol 5). Here, P_1 is the sender with its input being $(\hat{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{out}}}$, cMPC's input are the Clifford $(E_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{out}}}$, and P_i is the receiver for the content in the registers $(\hat{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{out}}}$ for all $i \in [n]$. The common inputs are 1 as the size of the plaintext, and κ as the trap-size.

    - If the output is $(\mathtt{abort}, P)$, then the parties output $(\mathtt{abort}, P)$ and halt.
    - Otherwise, all parties receive $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}})$, and additionally, for all $i \in [n]$ party P_i receive the Clifford ciphertexts $(\tilde{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$, and cMPC receives the corresponding Clifford keys $\left(F'_{ij}\right)_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$. Note that $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}} \supseteq \mathcal{R}_{\mathsf{out}}$ and will be used to indicate dropped *output-qubits.*

3. For all $i \in [n]$, cMPC sends to P_i the Cliffords $(F_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$, which reconstructs the output $\sigma_i = (\mathsf{CAuth.Dec}_{F_{ij}}(\tilde{\sigma}_{ij}))_{j:(i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$.

4. Each party P_i outputs $(\sigma_i, \mathcal{R}_{\mathsf{in}}, \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}})$.

We now state and prove the security of the protocol

Lemma 10.2. *Protocol π computes C with perfect $(n^2, 2n^2)$-IDPD security in the $\{\mathsf{cMPC}, \mathsf{Comp}\}$-hybrid model.*

Proof. First, observe that correctness directly follows from the definition of Comp, the correctness of π_{CTAR} as given by Claim 6.2, and the correctness of the Clifford decryption.

We next prove security. Fix an adversary $\mathcal{A}$ holding auxiliary input ρ_{aux}. The construction of the simulator $\mathsf{Sim}_{\mathcal{A}}$ is roughly as follows. It will forward $\mathcal{A}$'s input to Comp and send them to the trusted party T. Following this, it can prepare Clifford encryptions of the outputs of the corrupted parties that where given to it by T, and prepare dummy ciphertext as the output for the honest parties. The simulator then send these ciphertexts to $\mathcal{A}$ as the output of Comp. Then, to simulate π_{CTAR}, the simulator will emulate cMPC, and use the simulator in the SA ideal world, guaranteed to exists by Corollary 5.7, to simulate the adversary's view during the executions of π_{CTSA} inside π_{CTAR}. Formally, let $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ be the simulator in ideal world of SA, that simulates the adversary's view in π_{CTSA} as guaranteed by Corollary 5.7. Then $\mathsf{Sim}_{\mathcal{A}}$ works works as follows.

Simulator 7 $\mathsf{Sim}_{\mathcal{A}}$

---

1. Query the adversary $\mathcal{A}$ for its inputs $(\rho_i)_{i \in \mathcal{I}}$ and the set $\mathcal{R}_{\mathsf{in}}$ for Comp, and forward these to the trusted party T. Let $(\sigma_{ij})_{i \in \mathcal{I}, (i,j) \in \mathcal{R}_{\mathsf{in}}}$ be the outputs received.

2. For all $(i,j) \in \mathcal{R}_{\mathsf{in}}$, sample Clifford $E_{ij} \in \mathcal{C}_{1+n^2\kappa}$ independently and uniformly at random.

3. For all $(i,j) \in \mathcal{R}_{\mathsf{in}}$ encrypt dummy 0 states $\hat{\sigma}_{ij} = \mathsf{CAuth.Enc}_{E_{ij}}(|0\rangle\langle 0|)$.

4. Send $(\hat{\sigma}_{ij})_{(i,j) \in \mathcal{R}_{\mathsf{in}}}$ to $\mathcal{A}$ as the output of Comp. If it replies with $(\mathtt{abort}, \mathrm{P})$, then send it to T, output whatever $\mathcal{A}$ outputs, and halt.

5. Otherwise, $\mathcal{A}$ sends $(\mathtt{continue}, \mathcal{R}_{\mathsf{out}})$.

6. Simulate the execution of π_{CTAR}:

   (a) Initialize G_T as the complete graph and $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}} = \emptyset$

   (b) For all $(i,j) \in \mathcal{R}_{\mathsf{out}}$:

      i. Find a path $\mathtt{PATH}_{ij}$ in G_T from P_1 to P_i that passes through every party, computed the same as cMPC computed, and send it to $\mathcal{A}$.

      ii. Run $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$, where the input is σ_{ij} if $\mathrm{P}_1 \in \mathcal{I}$, and it has no input otherwise. If $\mathrm{P}_i \in \mathcal{I}$, then it sends back the same σ_{ij} as input to the trusted party. Send it back $\mathsf{CAuth.Enc}_{F_{ij}}(\sigma_{ij})$ where $F_{ij} \leftarrow \mathcal{C}_{1+\kappa}$, as the output.

      iii. If $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ replied with $\mathtt{continue}$, then add (i,j) to $\mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}$.

      iv. Otherwise, if it replied with $(\mathtt{abort}, \mathrm{P}_a, \mathrm{P}_b)$, then remove the edge ab from G_T. If G_T becomes disconnected, send $(\mathtt{abort}, \mathrm{P})$ to T where $\mathrm{P}' \in \{\mathrm{P}_a, \mathrm{P}_b\}$ is the party not connected to the honest parties, output whatever $\mathcal{A}$ outputs, and halt.

7. Send $(\mathtt{continue}, \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}})$ to T.

8. Send to $\mathcal{A}$ the Cliffords $(F_{ij})_{i \in \mathcal{I}, (i,j) \in \mathcal{R}_{\mathsf{in}}^{\mathsf{AR}}}$ and all views that $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ produced in each iteration, output whatever it outputs, and halt.

Clearly, in the ideal the state given to $\mathcal{A}$ as the output of Comp are maximally mixed state as in the real world. Thus, it replies with the same answer. Next, consider a single execution of π_{CTAR}. Since the paths are computed the same in both worlds, by Corollary 5.7, it follows that the state of $\mathcal{A}$ generated by $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ is indistinguishable from the state generated in the real world execution of π_{CTSA}. As the output given to $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{SA}}$ (when $\mathrm{P}_i \in \mathcal{I}$) is the Clifford encryption of the output given by T, it follows that upon receiving the Cliffords F_{ij}'s, the joint state of the adversary and the honest parties are the indistinguishable from the real world. $\square$

Remark 2. *Interestingly, assuming that* cMPC *sends all Clifford keys to the parties* simultaneously, *it follows that identifiable fair*[18] *classical MPC is sufficient for identifiable fair quantum MPC (see Appendix A for a formal definition). Indeed, observe that until Step 3 of the above protocol, all quantum states held by each party is encrypted. Therefore, an adversary that causes the protocol to abort gains no information on the output. Thus, if the last call to* cMPC *is fair, then all parties will receive the keys to their respective encrypted output simultaneously. This results in a fair MPQC protocol in the* cMPC-*hybrid model.*

Acknowledgements

References

[ABDR04] Andris Ambainis, Harry Buhrman, Yevgeniy Dodis, and Hein Rohrig. Multiparty quantum coin flipping. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 250–259. IEEE, 2004.

[ABOE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 453–469. Tsinghua University Press, 01 2010.

[ADSS17] G. Alagic, Y. Dulek, C. Schaffner, and F. Speelman. Quantum fully homomorphic encryption with verification. 2017.

[BCG+02] H. Barnum, C. Crepeau, D. Gottesman, A. Smith, and A. Tapp. Authentication of quantum messages. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE Comput. Soc, 2002.

[BF10] Niek J. Bouman and Serge Fehr. Sampling in a quantum population, and applications. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, page 724–741, Berlin, Heidelberg, 2010. Springer-Verlag.

---

[18]In addition to fairness, identifiable fair computation have the added property that in case the protocol aborts, the honest parties agree on the identity of at least one corrupted party.

[BGS13]   Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Advances in Cryptology – CRYPTO 2013*, pages 344–360. Springer Berlin Heidelberg, 2013.

[BK05]   Sergei Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 2005.

[BMMMQ20]   Nicholas-Philip Brandt, Sven Maier, Tobias Müller, and Jörn Müller-Quade. Constructing secure multi-party computation with identifiable abort. *IACR Cryptol. ePrint Arch.*, 2020:153, 2020.

[BMP+99]   P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for shor's basis. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, page 486, USA, 1999. IEEE Computer Society.

[BMP+00]   P.Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. A new universal and fault-tolerant quantum basis. *Information Processing Letters*, 75(3):101 – 107, 2000.

[BOCG+06]   Michael Ben-Or, Claude Crepeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006.

[BV11]   Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.

[CGS02]   Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*. ACM Press, 2002.

[Chi05]   Andrew M. Childs. Secure assisted quantum computation. *Quantum Info. Comput.*, 5(6):456–466, September 2005.

[Cle86]   Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369, 1986.

[DGJ+20]   Yfke Dulek, Alex B Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. *Advances in Cryptology - EUROCRYPT 2020.*, 2020.

[DLT02]   D. P. DiVincenzo, D. W. Leung, and B. M. Terhal. Quantum data hiding. *IEEE Transactions on Information Theory*, 48(3):580–598, 2002.

[DNS12]   Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In *Annual Cryptology Conference*, pages 794–811. Springer, 2012.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*. ACM Press, 1987.

[Got98] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57:127–137, Jan 1998.

[IOZ14] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2014*, pages 369–386. Springer Berlin Heidelberg, 2014.

[Kit] Alexei Kitaev. Quantum coin-flipping. Talk at QIP 2003 (slides and video at MSRI), December 2002.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85, 1989.

[Unr10] Dominique Unruh. Universally composable quantum multi-party computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–505. Springer, 2010.

A Definition of Fairness

The following ideal model specifies the requirements for identifiable fair computation of a quantum circuit C. Roughly, it states that either all parties learn the output or non of them do. Moreover, in case of abort, the honest parties agree on the identity of at least one corrupted party.

Security with identifiable fairness

Inputs: Each honest party P_i holds the security parameter 1^κ where $\kappa \in \mathbb{N}$ and an input ρ_i. The adversary is given 1^κ, input ρ_i of every corrupted party $P_i \in \mathcal{I}$, and an auxiliary input ρ_{aux}. Finally, the trusted party T is given the security parameter 1^κ.

Parties send inputs: Each honest P_i sends ρ_i to T. The adversary either sends to the trusted party $(\mathtt{abort}, P_i)$, for some corrupted party $P_i \in \mathcal{I}$, or for every corrupted party P_i, the adversary sends a quantum state ρ_i^*.

The trusted party performs the computation: If the adversary sent $(\mathtt{abort}, P_i)$, then T forwards it to all honest parties and halt. Otherwise, if the adversary sent $\mathtt{continue}$, set $\rho' = (\rho'_1, \ldots, \rho'_n)$ to be the tuple of inputs received by T. The trusted party T prepares ancilla zero states ρ_0 and computes the mapping

$$(\sigma_1, \ldots, \sigma_n, \sigma_{\mathsf{discard}}) = C(\rho', \rho_0)$$

where σ_i is the output associated with party P_i. For every honest party $P_i \notin \mathcal{I}$, the

---

trusted party sends it σ_i. For every corrupted party $\mathrm{P}_i \in \mathcal{I}$, the trusted party sends σ_i to the adversary $\mathcal{A}$.

Outputs: Each honest party outputs whatever it received from the trusted party, the parties in $\mathcal{I}$ output nothing, and the adversary outputs some function of its view.