Secure multiparty quantum computation with few qubits

Lipinska, Victoria; Ribeiro, Jérémy; Wehner, Stephanie

DOI
10.1103/PhysRevA.102.022405

Publication date
2020

Document Version
Final published version

Published in
Physical Review A

Secure multiparty quantum computation with few qubits

Victoria Lipinska,* Jérémy Ribeiro, and Stephanie Wehner
QuTech, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands
and Kavli Institute of Nanoscience, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

(Received 28 April 2020; accepted 14 July 2020; published 7 August 2020)

We consider the task of secure multiparty distributed quantum computation on a quantum network. We propose a protocol based on quantum error correction which reduces the number of necessary qubits. That is, each of the n nodes in our protocol requires an operational workspace of $n^2 + 4n$ qubits, as opposed to the previously shown $\Omega((n^3 + n^2 s^2)\log n)$ qubits, where s is a security parameter. Additionally, we reduce the communication complexity by a factor of $O(n^3 \log(n))$ qubits per node compared to existing protocols. To achieve universal computation, we develop a distributed procedure for verifying magic states, which allows us to apply distributed gate teleportation and which may be of independent interest. We showcase our protocol in a small example for a seven-node network.

DOI: 10.1103/PhysRevA.102.022405

I. INTRODUCTION

Secure multiparty computation is a task which allows n nodes of a network to jointly compute a function on their inputs [1]. The inputs are private, meaning that they are only known to the nodes who supplied them. What is more, the only information that can be inferred about the private inputs is whatever can be inferred from the outputs of the computation and the computation itself. Multiparty computation allows for distributed evaluation of any function, and hence it is a powerful cryptographic primitive with many practical (e.g., clearing a commodity derivative market) and theoretical (e.g., zero-knowledge proofs) applications [2].

In the domain of quantum computation the problem of multiparty quantum computation (MPQC) on quantum data was first introduced in [3]. It can be defined as follows: each node $i = 1, \ldots, n$ gets one, possibly unknown, input quantum state ρ_i. The nodes jointly perform an n-input arbitrary quantum circuit $\mathfrak{R}$ on their inputs $\rho_1, \ldots, \rho_n$. The output of the circuit is divided into n parts and each node i gets the ith part of the output state (see Fig. 1). In MPQC there can be nodes who do not follow the protocol (cheaters). We then require that an MPQC protocol satisfies the following informal requirements:

(a) Correctness: If there are no cheaters, then the protocol implements the intended circuit $\mathfrak{R}$ on the inputs of the nodes.

(b) Soundness: Cheaters cannot affect the outcome of the computation of the other nodes, beyond their ability to choose their own inputs.

(c) Privacy: Cheaters do not learn anything about private inputs and outputs of the other nodes.

Throughout this paper we consider that an input ρ_i of each node is a single-qubit state.

The approach taken in the original work [3] is based on a subroutine called verifiable quantum secret sharing (VQSS) and is a generalization of a classical multiparty computation [4]. The security achieved by the protocol is information theoretical, meaning that the cheaters are not constrained by computational assumptions. However, the number of cheaters has to be strictly smaller than $\frac{n}{6}$. This bound was later lifted to $\frac{n}{2}$ in [5], where authentication schemes and approximate error correction were used. However, this solution requires significantly more qubits to be realized. At the same time, there exist parallel approaches that tolerate a cheating majority and whose security relies on computational assumptions, for example, for the case of $n = 2$ [6] or its recent generalization to $n > 2$ [7]. Note that a protocol tolerating more than $\frac{n}{2}$ cheaters is not possible without additional computational assumptions, since that would imply the existence of unconditionally secure bit commitment [8,9].

In this work we are interested in the former approach to MPQC, namely, the one based on VQSS from [3]. Our objective is to perform MPQC on a quantum network with n nodes using as few qubits as possible. The approach we take is based on [3] and extensively relies on techniques from fault-tolerant quantum error correction. It can be intuitively understood as follows. Nodes use a chosen quantum error correcting code and create a global logical state $\bar{\Psi}$ by encoding each of the single-qubit input states. Each node holds a part of this logical state; we call this part a *share*. They verify the encoding of each state using a verifiable secret sharing protocol, perform local operations to evaluate a logical version of the circuit $\mathfrak{R}$, and then locally reconstruct their outputs.

To be able to apply any circuit $\mathfrak{R}$ this way, we need two properties. First, $\mathfrak{R}$ must be composed of gates which form a universal set, i.e., any circuit can be decomposed into gates from that set. Second, if the nodes apply only local operations Λ from the universal set, it should yield a meaningful logical operation $\bar{\Lambda}$ for the global state $\bar{\Psi}$. This property is called *transversality*. However, for any error correcting code, it is impossible to perform universal quantum computation using only transversal gates [10]. For this reason, it is common to extend a transversal set of gates (for example, Clifford

*v.lipinska@tudelft.nl

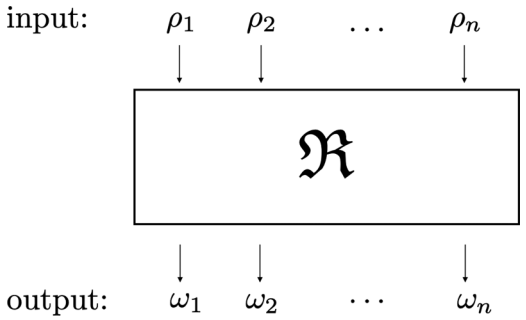

FIG. 1. Each of the nodes $1, \ldots, n$ provides a single-qubit input $\rho_1, \ldots, \rho_n$. The goal of the multiparty quantum computation (MPQC) protocol is to implement circuit $\mathfrak{R}$ such that each node gets an output $\omega_1, \ldots, \omega_n$ without gaining any knowledge of the other inputs or outputs beyond their ability to choose their own inputs. Note that the inputs (and outputs) can be entangled.

gates) with a nontransversal gate (for example, the T gate or the Toffoli gate). Note that there exist methods to realize single nontransversal gates in a distributed way, for example, by using ancilla states [11] or locally modifying the error correcting code [12].

In particular, Ref. [3] considers quantum polynomial codes and a universal set of gates with the Toffoli gate [12]. This solution is very expensive in qubits. First, the polynomial codes require local shares whose dimension scales with the number of nodes and, therefore, require $\Omega(\log n)$ qubits per share. Moreover, the nodes need to perform a distributed encoding of the shares in order to apply the Toffoli gate. This means that each input state must be encoded three times using the polynomial code. Performing the three-level encoding serves one more purpose, namely, it localizes all of the errors in the encoding to the positions of the cheaters. As a result, the cheaters cannot force the protocol to abort, since any error they introduce will always be corrected by the underlying polynomial code. All in all, each node needs an operational workspace of $\Omega((n^3 + n^2 s^2)\log n)$ qubits, where s is the security parameter of the protocol (see Table I). We remark that in schemes based on exact error correcting codes, the number of cheaters t is intrinsically constrained by the distance d of the underlying code as $t \leqslant \lfloor \frac{d-1}{2} \rfloor$, which in principle can reach $\frac{n}{4}$ [13,14]. However, the technique for applying the Toffoli gate in [3] puts a constraint on the number of cheaters to $\frac{n}{6}$.

Since near-term quantum networks will be able to support only a small number of qubits, it would be preferable to implement an MPQC protocol with as few qubits as possible. So far, reducing quantum resources has received a lot of attention in the domain of nondistributed quantum computation and simulation (see, for example, [15–19]). Recently, in [20] we considered a problem of reducing quantum resources for a distributed protocol, namely, verifiable secret sharing of a quantum state. Here we address a similar issue of whether distributed multiparty quantum computation can be performed on a quantum network with fewer quantum resources. We answer this question positively by proposing a scheme for universal distributed computation which uses fewer qubits compared to the existing approach from [3] outlined above.

This paper is organized as follows. In Sec. II we summarize our contributions; in Sec. II A we discuss the implications of our protocol for resource reduction and in Sec. II B we give an explicit example of the protocol in a seven-node network. In Sec. III we zoom in on the technical aspects of our work. There we present the protocol in detail and provide formal security statements. We leave our technical proofs for the Appendix.

II. RESULTS

We propose a protocol for secure multiparty quantum computation where each node holds single-qubit shares. Our approach is based on quantum error correcting codes, similar to the idea in [3,5,21]. The key to our results is using error correcting codes which encode a single qubit into n single qubits. Since our interest lies in reducing the quantum resources necessary to realize the protocol, we abandon the original idea of three-level encoding at the cost of allowing the protocol to abort if the initial encoding of the shares is incorrect. Thanks to this, we are able to execute the protocol with fewer qubits per node in the workspace and a lower communication complexity (see Table I). Moreover, we develop a procedure for the distributed verification of any logical state which is stabilized by a Clifford gate. This allows us to perform distributed gate teleportation and implement a universal set of gates without creating three levels of encoding. What is more, we follow the approach outlined in [20], which allows for a sequential execution of the verification of the inputs. This solution reduces the operational workspace to $n^2 + 4n$ qubits per node. We elaborate on these techniques in the next section, Sec. II A. We show that our protocol is secure in the presence

TABLE I. Summary of qubit savings presented in this paper. s denotes the security parameter of the protocol, #ancillas denotes the number of ancillas in circuit $\mathfrak{R}$, $\#T$ denotes the number of T gates, and #Toff is the number of Toffoli gates. The size of the workspace in our protocol does not depend on the security parameter, because of the sequential execution of the verification phase (see Sec. III B 1). Note that here we do not list the work in [7], since the protocol there does not use techniques based on error correction and achieves computational security guarantees.

	Our protocol	Crepeau *et al.* [3]
Size of the input, in qubits per node	1	$\Omega(\log n)$
Size of an individual share during the computation, in qubits per node	1	$\Omega(\log n)$
Number of qubits in workspace per node	$n^2 + 4n$	$\Omega((n^3 + n^2 s^2)\log n)$
Number of qubits sent per node	$O((n + \#\text{ancillas} + \#T)ns^2)$	$O((n^2 + \#\text{ancillas} + \#\text{Toff})n^3 s^2 \log(n))$

of active nonadaptive cheaters (see paragraph "Adversary model," below), where the number of cheaters is constrained by the distance d of the underlying error correcting code, i.e., $t \leqslant \lfloor \frac{d-1}{2} \rfloor$. Finally, we showcase our protocol in a small example for seven nodes using Steane's seven-qubit code [22].

Outline 1 (multiparty quantum computation).
Input: single-qubit state ρ_i from each node, CSS code $\hat{\mathcal{C}}$ with transversal Cliffords, circuit $\mathfrak{R}$.

1. *Sharing and verification*
   Each node $i = 1, \ldots, n$ encodes her input ρ_i using code $\hat{\mathcal{C}}$ into an n-qubit logical state and sends one qubit (i.e., one single-qubit share) of the logical state to every other node, while keeping one for herself. The nodes jointly verify the encoding done by node i using a verifiable quantum secret sharing protocol (see Protocol 1; Sec. III B 1).
2. *Computation*
   (a) For every Clifford gate in circuit $\mathfrak{R}$:
   The nodes apply transversal Clifford gates locally to qubits specified by circuit $\mathfrak{R}$.
   (b) For every T gate in circuit $\mathfrak{R}$ applied to qubit i:
   Node i prepares the magic state $|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle)$. The nodes verify it using the Verification of Clifford-Stabilized States protocol (see Protocol 3; Sec. III B 3). If the verification is successful, the nodes perform Distributed Gate Teleportation (see Protocol 2; Sec. III B 2).
   Every $|0\rangle$ ancilla state required for circuit $\mathfrak{R}$, which is prepared by node i, is jointly verified by the nodes using verifiable quantum secret sharing, Protocol 1 (Sec. III B 1).
   If the verification of any step fails, the nodes substitute their shares for $|0\rangle$ and abort the protocol at the end of the computation.
3. *Reconstruction*
   Each node i collects all shares of her part of the output. She corrects errors using code $\hat{\mathcal{C}}$ and reconstructs her output.

Network model. We consider a quantum network with n nodes. Each node can locally process $O(n^2)$ qubits and can perfectly process and store classical information. Each pair of nodes is connected via private and authenticated classical [23] and quantum [24] channels. Additionally, we assume that the nodes have access to an authenticated classical broadcast channel [25] and a public source of randomness. Note that a source of randomness can be created, for example, by running a classical multiparty computation protocol [26].

Adversary model. We say that t out of n nodes are *active* cheaters during the protocol. This means that they can act maliciously throughout the entire execution of the multiparty computation and perform arbitrary joint quantum operations on their shares, possibly with quantum side information. Therefore, the security of our protocol does not rely on computational assumptions. We assume that the active cheaters are *nonadaptive*, meaning that they are determined prior to the beginning of the protocol and stay fixed throughout its execution. On the other hand, the nodes which follow the protocol exactly are *honest*. A protocol *tolerates* the presence of t active cheaters if they cannot influence the output of the protocol beyond choosing their own inputs.

A. Techniques

Thanks to using single-qubit error correcting codes, distributed verification of magic states, and the possibility of aborting the protocol and sequential verification of the inputs, our MPQC protocol lowers the number of qubits that each node needs to control and send. Here we discuss in detail all the reductions made by our protocol. Then we give an explicit example of a protocol based on Steane's seven-qubit code.

(1) *Single-qubit Calderbank-Shor-Steane (CSS) codes.* We consider a class of CSS error correcting codes [22,27], which encode a single logical qubit into n physical qubits and for which applying Clifford gates is transversal (see Sec. III A for details). In particular, this means that each input state and each encoded ancilla are encoded and distributed using single-qubit shares. For comparison, the protocol in [3] uses a class of polynomial codes, called Reed-Solomon codes [12], where the size of individual shares increases with the number of nodes n in the network as $\Omega(\log n)$ qubits.

(2) *MPQC with abort.* We introduce an "abort" event in the MPQC protocol. That is, the protocol aborts if there are more than t errors introduced by the cheaters, accumulated over all inputs. This condition is necessary, since applying a transversal gate between different logical inputs can still propagate errors between them. As a result, we are able to perform the MPQC protocol on the two-level encoding created by the VQSS subroutine (see Sec. III B). This allows us to achieve a lower communication complexity: in our protocol each node sends $O((n + \#\text{ancillas} + \#T)ns^2)$ qubits, as opposed to $O((n^2 + \#\text{ancillas} + \#\text{Toff})n^3 s^2 \log(n))$ qubits in [3], where s denotes the security parameter of the protocol, #ancillas denotes the number of ancillas in circuit $\mathfrak{R}$, $\#T$ denotes the number of T gates, and #Toff is the number of Toffoli gates. Note that in our protocol we can avoid the abort event by creating the third level of encoding, following the idea from [3]. This approach confines the errors of all inputs only to the positions of t cheaters (see Sec, IV for discussion). However, this solution significantly increases the quantum communication complexity. Since our objective is to reduce the number of qubits, we do not consider this approach here.

(3) *Verification of Clifford-stabilized states.* We develop a distributed method for verifying states stabilized by Clifford gates, which, in particular, can be applied to verify magic states. This solution allows us to perform distributed gate teleportation and apply the T gate in a distributed way. Recall that for our MPQC protocol we choose CSS codes with transversal Clifford gates. This, together with distributed gate teleportation and transversal measurements, provides a way to apply a universal set of gates in a distributed way. Thanks to using magic-state ancillas, we can perform the computation on a two-level encoding created during the verification phase (see Protocol 4; Sec. III C). This means that each node controls n^2 single-qubit shares of all inputs. In contrast, in the approach in [3] the nodes need to apply a nonlinear Toffoli gate to achieve universality of computation. This, in turn, requires a workspace of $\Omega((n^3 + n^2 s^2)\log n)$ qubits per node.

(4) *Sequential verification.* We use the verifiable quantum secret sharing protocol from [3] to verify that the encoding was carried out correctly and that at the end of the computation there will be a state to reconstruct. The verification procedure requires ancillary states. However, following the idea developed in [20], we perform the verification in a sequential way. That is, to verify each input we use the ancillas one by one instead of all at once as in [3]. In particular, the nodes use at most $2n$ single-qubit ancillas at a time to verify the input states (or ancillas in $\mathfrak{R}$) and at most $4n$ single-qubit ancillas to apply the T gate.

All in all, this amounts to an operational workspace of at most $n^2 + 4n$ single-qubit shares for our protocol. Of those, n^2 shares correspond to the input states on which the distributed computation is performed. For comparison, the protocol in [3] requires simultaneous control over $\Omega((n^3 + n^2 s^2)\log n)$ qubits per node, where s is the security parameter of the protocol. Moreover, due to the possibility of aborting the protocol, our MPQC scheme lowers the communication complexity. That is, our protocol reduces the number of qubits that each node has to send by a factor of $O(n^3 \log(n))$ compared to the protocol in [3].

Finally, when the number of cheaters t is restricted by the distance d of the CSS code, i.e., when $t \leqslant \lfloor \frac{d-1}{2} \rfloor$, we prove that our protocol satisfies the usual security requirements (soundness, completeness, and privacy; see Sec. I). Our statements follow from the fact that any error correcting code has the ability to correct at most $\lfloor \frac{d-1}{2} \rfloor$ arbitrary errors, and therefore, any errors introduced by the cheaters can be corrected by the honest nodes. What is more, the inputs and outputs of honest nodes will also be private, since if they recover the outputs exactly, then the cheaters get no information about inputs or outputs [28]. Our statements hold with a probability exponentially close to 1 in the security parameter s.

B. Example for seven nodes

Let us consider a network of $n = 7$ nodes and assume that the nodes want to perform a CNOT between input ρ_1 and input ρ_2 of nodes "1" and "2" of the network. For the execution of this protocol we need a workspace of 28 qubits per node. For the sake of the example, we also assume that the inputs are pure single-qubit states, $\rho_1 = |\psi_1\rangle\langle\psi_1|$ and $\rho_2 = |\psi_2\rangle\langle\psi_2|$, and that the protocol does not abort. The seven-qubit Steane's code [22] is the smallest example of a qubit CSS code with transversal Cliffords. This code has distance $d = 3$, meaning that it can correct $\lfloor \frac{d-1}{2} \rfloor = 1$ arbitrary error. This also means that in an MPQC protocol built on the seven-qubit code, we can tolerate $t = 1$ cheater.

Sharing and verification. Node 1 encodes her single-qubit pure input $|\psi_1\rangle$ into seven physical qubits using Steane's code encoding map $\mathcal{E}$. She sends one qubit to each of the remaining six nodes, while keeping one qubit to herself. Each node again encodes the received qubit using Steane's code and shares six qubits of that encoding with other nodes. At this point the input state $|\psi_1\rangle$ has been encoded twice, i.e.,

$$\bar{\bar{\Psi}}_1 = \mathcal{E}^{\otimes 7} \circ \mathcal{E}(|\psi_1\rangle\langle\psi_1|), \tag{1}$$

Each node holds seven qubits in total (Fig. 2).

The nodes run the verification procedure according to [20], verifying that the encoding of each node i was done correctly. The encoding of each input state can be verified one at a time. In one round of verification of a single input, each node uses at most 14 local ancilla qubits. The ancilla shares are encoded twice with the seven-qubit code and distributed in the same way as the input states. The nodes randomly perform the CNOT gate between $\bar{\bar{\Psi}}_1$ and an ancilla, to identify errors possibly introduced by cheating nodes. These ancillas are then measured and the outcome of the measurement allows the nodes to jointly conclude whether verification of the encoding was correct, i.e., whether the distributed input states have at most $t = 1$ error on the same position. If so, then the errors are correctable by the seven-qubit code, and the nodes hold a valid logical state of the code. This procedure is repeated $s^2 + 2s$ times in total, where s is the security parameter.

The same sharing and verification procedure is carried out for node 2 and her single-qubit pure input $|\psi_2\rangle$: it is first shared, as the logical state

$$\bar{\bar{\Psi}}_2 = \mathcal{E}^{\otimes 7} \circ \mathcal{E}(|\psi_2\rangle\langle\psi_2|), \tag{2}$$

and then verified. As before, the verification requires at most 14 local ancilla qubits at a time. After the second verification each node holds 14 verified data qubits corresponding to the logical inputs $\bar{\bar{\Psi}}_1 \otimes \bar{\bar{\Psi}}_2$. Note that the input states are never measured.

Computation. Each node applies the CNOT gate locally to shares coming from nodes 1 and 2. The CNOT gate is a Clifford gate. Therefore, since the inputs are verified to be logical states of the seven-qubit code, applying the CNOT locally is well defined and yields a logical operation between logical inputs $\bar{\bar{\Psi}}_1 \otimes \bar{\bar{\Psi}}_2$. Let us define the output of the computation $\bar{\bar{\omega}}$,

$$\bar{\bar{\omega}} = \overline{\overline{\text{CNOT}}}(\bar{\bar{\Psi}}_1 \otimes \bar{\bar{\Psi}}_2). \tag{3}$$

Reconstruction. Nodes 1 and 2 get all of the shares corresponding to their own outputs, i.e.,

$$\bar{\bar{\omega}}_1 = \text{tr}_2(\bar{\bar{\omega}}), \quad \bar{\bar{\omega}}_2 = \text{tr}_1(\bar{\bar{\omega}}). \tag{4}$$

They separately run the local error correcting circuit of the seven-qubit code on $\bar{\bar{\omega}}_1$ and $\bar{\bar{\omega}}_2$, respectively. They identify errors [see Reconstruction in Protocol 4 (Sec. III C) for details]. This is necessary, since the cheater might have introduced

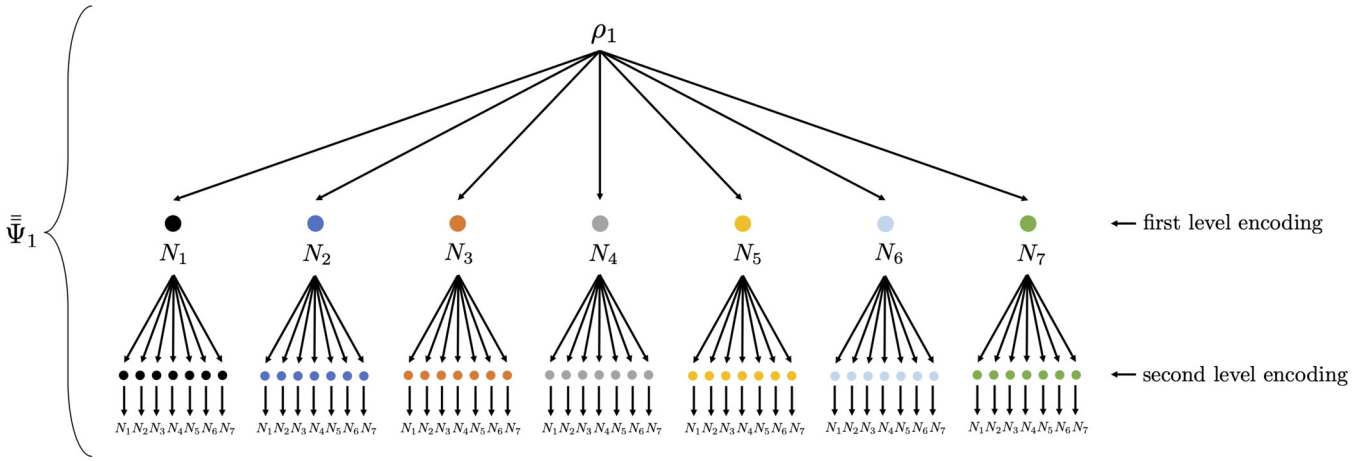

FIG. 2. Two-level encoding of the input qubit state ρ_1 of node 1. The double-encoded distributed state is denoted $\bar{\bar{\Psi}}_1$. Each circle represents a single-qubit share.

errors during or after the computation, and right before the reconstruction. Each of nodes 1 and 2 corrects errors and reconstructs her output, ω_1 and ω_2, respectively. The outputs are single-qubit states and are such that

$$\omega_1 = \mathrm{tr}_2(\mathrm{CNOT}(|\psi_1\rangle\langle\psi_1| \otimes |\psi_2\rangle\langle\psi_2|)), \tag{5}$$

$$\omega_2 = \mathrm{tr}_1(\mathrm{CNOT}(|\psi_1\rangle\langle\psi_1| \otimes |\psi_2\rangle\langle\psi_2|)). \tag{6}$$

We remark that to tolerate a larger numbers of cheaters t one can use CSS error correcting codes $\hat{\mathcal{C}}$ with a larger distance for which implementation of Clifford gates is transversal. For example, using the so-called color codes [29], one can construct MPQC with the total number of nodes expressed in terms of the number of cheaters t as $n = 2t^2 + 4t + 1$, $n = 3t^2 + 3t + 1$, and $n = 6t^2 + 1$.

III. METHODS

In this section we discuss our MPQC protocol in detail. We lay down the framework by first discussing properties of CSS codes which will be useful for the distributed computation in Sec. III A. Then we introduce a few important subroutines, namely, verifiable secret sharing (Sec. III B 1), distributed gate teleportation (Sec. III B 2), and verification of Clifford-stabilized states (Sec. III B 3). Finally, we discuss our multiparty quantum computation protocol in Sec. III C and state its security in Sec. III D.

A. CSS codes

In our considerations we focus on a class of error correcting codes called Calderbank-Shor-Steane codes [22,27]. A CSS code $\mathcal{C}$ is defined through two binary classical linear codes, V and W, satisfying $V^* \subseteq W$, where V^* is the dual code of V. Then $\mathcal{C} := V \cap \mathcal{F}W$ is a set of states of n qubits which yield a code word in V when measured in the standard basis and a code word in W when measured in the Fourier basis. A code encoding one logical qubit into n physical qubits is commonly denoted by double square brackets, $[[n, 1, d]]$. Here d is the distance of the code, which relates to the maximum number of arbitrary errors t which the code can correct as $t \leqslant \lfloor \frac{d-1}{2} \rfloor$.

In distributed computation each node can only apply local operations. Therefore, we want logical operations $\bar{\Lambda}$ to be implemented by applying local operations Λ on the individual qubits held by the nodes and encoded with $\mathcal{C}$, i.e., $\bar{\Lambda} = \Lambda^{\otimes n}$. This property is called transversality. For our construction of the MPQC protocol we choose specific CSS codes $\hat{\mathcal{C}}$ with transversal operations, which satisfy the following.

(1) $\hat{\mathcal{C}}$ uses the same classical code to correct X and Z errors, i.e., $V = W$.

(2) The weight of the stabilizer generators of $\hat{\mathcal{C}}$ is a multiple of 4, and the logical Pauli operators X and Z have weight 1 mod 4 or 3 mod 4.

Property 1 guarantees that the Hadamard gate H can be applied transversally, while property 2 guarantees that the phase gate $P = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ can be applied transversally. Additionally, note that the CNOT gate is transversal for any CSS code. Since H, P, and CNOT generate the Clifford set, one can apply any Clifford gate on the code $\hat{\mathcal{C}}$ transversally [30]. Finally, any CSS code has the property that measurements can be performed qubitwise, but the measurement outcome of every qubit must be communicated classically to obtain the result of the logical measurement.

B. Subroutines

Here we list and describe the subroutines we use later as building blocks in our MPQC protocol. We start by reviewing an existing construction of verifiable quantum secret sharing used for verifying inputs in MPQC. Next we discuss two of our contributions: distributed gate teleportation and verification of states stabilized by Clifford gates. The latter two subroutines are essential for implementing universal circuits in MPQC.

1. Verifiable quantum secret sharing

One of the first ingredients of our MPQC protocol is verifiable quantum secret sharing, first introduced in [3] (see Protocol 1, above). Here we use a modified version of the

scheme, which we introduced in [20] to reduce the qubit workspace required for each node. A VQSS scheme is one which shares a quantum state among n nodes in a verifiable way using quantum shares. The scheme we use is based on a CSS code $\mathcal{C}$ with distance d and tolerates at most $t \leqslant \lfloor \frac{d-1}{2} \rfloor$ nonadaptive active cheaters. We remark that the scheme works for any CSS code $\mathcal{C}$.

Let us describe the task in detail. In VQSS the dealer D encodes her input state ρ using the code $\mathcal{C}$. The encoding produces an n-qubit entangled state. D shares this state among the nodes by sending one qubit to each node. Each node then encodes the received one-qubit share again, with the same error correcting code, into n qubits and sends one qubit to each of the n nodes. This way each node holds n single-qubit shares. We denote a double-encoded logical global state of the nodes with a double bar, $\bar{\bar{\Psi}}$. Henceforth throughout the paper we use the index $i = 1, \ldots, n$ to denote the encoding performed by node i and $\ell = 1, \ldots, n$ to denote the share held by node ℓ. The share held by node ℓ and coming from encoding performed by node i is denoted $\bar{\bar{\Psi}}_{i_\ell}$.

The nodes run a verification procedure to verify that $\bar{\bar{\Psi}}$ is a valid code word of the code $\mathcal{C}$. The verification is a generalization of Steane's error correction method to the distributed setting [31]. More specifically, the nodes publicly check that there are at most $t \leqslant \lfloor \frac{d-1}{2} \rfloor$ errors at the first level of encoding, i.e., the encoding done by the dealer. To do so, they use ancilla qubits encoded twice with the same code $\mathcal{C}$. These ancillas are measured during the verification. Since $\mathcal{C}$ is a CSS code, the measurement outcomes yield a code word from a classical code V (code W) when measured in the standard (Fourier) basis. Using an error correcting procedure for the classical linear codes allows the nodes to identify shares of the first-level encoding which carry errors. The positions of these shares are collected in a public set B of *apparent* cheaters (indeed, there is no way to distinguish errors introduced by the dealer from errors introduced by the cheaters in the first-level encoding). If there are at most t first-level errors (i.e., $|B| \leqslant t$), the dealer passes the verification. Moreover, since the protocol assumes the existence of at most t cheaters, there can be at most t errors in each second-level encoding. Therefore, if the dealer passes the verification, at the end of the protocol there will always be a state to reconstruct, since errors in both first- and second-level encoding can be corrected by the code $\mathcal{C}$. Following the idea introduced in [20], this verification procedure can be performed by encoding and measuring one ancilla qubit at a time. There are $s^2 + 2s$ iterations of the verification procedure, where s is the security parameter. Additionally, similarly to [20], we use CSS codes which encode a single qubit into n single qubits. The sequential VQSS protocol requires a $3n$-qubit workspace per node to verify one single-qubit input state (see [20] for details). Each node needs to send $O(n^2 s^2)$ qubits.

Verification of logical 0 [VQSS(0)]. In the following sections we make use of a handy property of the VQSS protocol in [3]. Namely, the protocol can verify that the state shared by the nodes is exactly the logical $|\bar{\bar{0}}\rangle$ of code $\mathcal{C}$ (see [3,20,21]). The verification phase is almost the same as in the VQSS protocol in [3], except now the nodes check whether the classical measurement outcomes interpolate to 0 after decoding them twice with a classical decoder (see [3,21] for details). We refer to this verification procedure as VQSS(0).

Protocol 1 (verifiable quantum secret sharing (VQSS) [3,20]).
Input: Single-qubit state ρ of dealer D to share, CSS error correcting code $\mathcal{C}$.

1. *Sharing*
   The dealer D encodes her input ρ into a logical state using code $\mathcal{C}$ and sends each qubit of the logical state to every other node, while keeping one for herself. Each node encodes the share received from D again using $\mathcal{C}$ and shares among the nodes, keeping one qubit for herself. Therefore, the nodes create a two-level encoding of ρ. At this point each node holds n single-qubit shares coming from every other node.
2. *Verification*
   The nodes verify whether D is honest, i.e., that the shares held by the nodes are consistent with a code word of $\mathcal{C}$ and at the end of the protocol a state will be reconstructed. The nodes construct a public set B which records the positions of nodes with inconsistent shares in the first level of encoding.
   Each node uses at most additional $2n$ ancilla qubits for one iteration of the verification procedure. There are $s^2 + 2s$ iterations of verification, where s is the security parameter. If $|B| \leqslant t$, the dealer passes the verification phase.

2. Distributed gate teleportation

To perform universal computation, we need a universal set of gates. However, Clifford gates by themselves are not a universal set. An example of a set that *is* universal is the set generated by the Clifford gates extended with the $T = \sqrt{P}$ gate [32], denoted Cliff $+ T$.[1] On the other hand, for any error correcting code, it is impossible to perform universal quantum computation using only transversal gates [10]. In particular, for the class of CSS codes under consideration, $\hat{\mathcal{C}}$, the Clifford gates can be applied transversally (see Sec. III A), but the T gate cannot.

To remedy this problem in the domain of quantum (nondistributed) computing, one can use a technique called gate teleportation [11]. In particular, for the T gate, the idea is to use a specially created ancilla state, measure, and apply a correction depending on the measurement outcome (see Fig. 3). Importantly, this correction is done with $XP^\dagger$, and since $XP^\dagger$

[1] One can efficiently approximate any gate G within distance ϵ using polylog$(1/\epsilon)$ gates from the set Cliff $+ T$ [33].

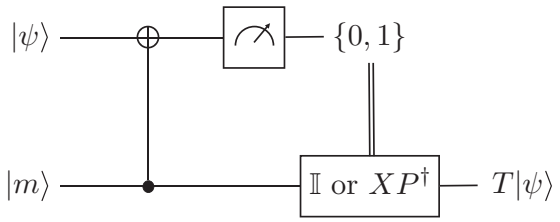

FIG. 3. Gate teleportation of the T gate. The circuit applies the T gate to an arbitrary single-qubit state ρ. Each state may be logical and each operation may be applied transversally.

is a Clifford gate, it can be applied transversally. The cost of this procedure is to create the special ancilla state, which is commonly referred to as a magic state. In the case of the T gate it is $|m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$.

We generalize this procedure to a distributed setting (see Protocol 2, below). Our protocol takes two states as an input, logical $\bar{\bar{\Psi}}$ and logical $|\bar{\bar{m}}\rangle$, each encoded twice (two-level encoding) with code $\hat{\mathcal{C}}$. We assume at this point that both states are verified with respect to the same dealer D. The verification of $\bar{\bar{\Psi}}$ can be performed with VQSS. However, verifying that $|\bar{\bar{m}}\rangle$ is *exactly* the magic state is nontrivial and we introduce it in the next section.

To apply a logical T gate to $\bar{\bar{\Psi}}$ the nodes first perform a logical transversal CNOT operation on their shares, taking shares of $|\bar{\bar{m}}\rangle$ as a control and shares of $\bar{\bar{\Psi}}$ as a target. Then each node $i = 1, \ldots, n$ measures the target qubit in the standard basis and announces the measurement outcome. The nodes publicly check whether the measurement collapsed the target state onto a classical string corresponding to a logical $|\bar{\bar{0}}\rangle$ or a logical $|\bar{\bar{1}}\rangle$. To do so, they check whether the resulting string of measurement outcomes $\mathbf{v}_i$ interpolates to 0 or to 1 using the classical decoder twice. At the same time the nodes update the set of errors B. If the interpolated value is 0, then no correction is necessary. If the interpolated value is 1, then the nodes apply the correction $XP^\dagger$ transversally.

Protocol 2 [distributed gate teleportation (GTele)].
Input: $\bar{\bar{\Psi}}$, $|\bar{\bar{m}}\rangle$ distributed by D to the nodes and verified by the nodes using VQSS (Protocol 1; Sec. III B 1), set of apparent cheaters B from verification of $\bar{\bar{\Psi}}$ and $|\bar{\bar{m}}\rangle$.
Output: Logical T gate applied to the input logical state, $\bar{\bar{T}}(\bar{\bar{\Psi}})$.
1. Each node ℓ, for a share coming from node i:
   (a) applies CNOT with $|\bar{\bar{m}}\rangle_{i_\ell}$ as the control qubit and $\bar{\bar{\Psi}}_{i_\ell}$ as the target qubit;
   (b) measures the target qubit in the Z basis and broadcasts the result using the secure broadcast channel (see paragraph "Network model"; Sec. II).
2. Broadcasted values yield words $\mathbf{v}_i$. Nodes publicly check at which positions the errors occurred using the classical decoder and update set B with the positions of errors. They decode the classical value a:
   (a) If $a = 0$, the nodes do not apply any correction.
   (b) If $a = 1$, the nodes apply $XP^\dagger$ to their shares.

3. Verification of Clifford-stabilized states

One last ingredient we need to perform distributed computation is to verify that the logical magic state $|\bar{\bar{m}}\rangle$ is indeed the logical magic state. This is necessary since we want to be sure that when we apply the T gate in a distributed way, the result will be the T gate on the shares of honest nodes.

Here we present a protocol, Protocol 3, to verify the magic state in a distributed way. In fact, our protocol works for any qubit state $|g\rangle$ stabilized by a single-qubit Clifford gate G. Our idea is inspired by the so-called stabilizer measurement in quantum error correction (see Fig. 4). Consider a single-qubit gate $XP^\dagger$ with a $+1$ eigenstate $|m\rangle$. Then it holds that state $|+\rangle|m\rangle$ is stabilized by the controlled $XP^\dagger$ gate, $C\text{-}XP^\dagger$, where $|+\rangle$ is used as a control and $|m\rangle$ is used as a target. That is,

$$C\text{-}XP^\dagger(|+\rangle|m\rangle) = |+\rangle|m\rangle. \tag{7}$$

This gives us an insight into how the verification of $|m\rangle$ should work: if the target state was the magic state, then after performing $C\text{-}XP^\dagger$ we will always measure the control in $|+\rangle$ (or, equivalently, first apply H and measure 0). Additionally, if the target was not in the magic state and we measure the control in $|+\rangle$, we will project the target onto $|m\rangle$. For this to work, we need to make sure that the control qubit was in $|+\rangle$ before applying the controlled gate.

We adapt this procedure to run on the logical level in a distributed way as follows. Using VQSS(0), the nodes first verify a logical $|\bar{\bar{0}}\rangle$ encoded and shared by D. They also share $|\bar{\bar{m}}\rangle$ and verify that it is a valid code word of $\hat{\mathcal{C}}$ using the VQSS protocol (Protocol 1; Sec. III B 1). This step is

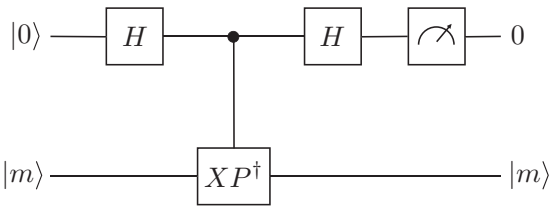

FIG. 4. Verification of the magic state using stabilizer measurement. The circuit verifies that the target input is the magic state using the fact that state $|+\rangle|m\rangle$ is stabilized by the controlled $C\text{-}XP^\dagger$ gate.

necessary since we want the transversal operations which the nodes will perform in the next steps to be well defined. Each of the nodes now applies the Hadamard gate to her share of $|\bar{\bar{0}}\rangle$ to turn it into a logical $|\bar{\bar{+}}\rangle$ and subsequently performs $C\text{-}XP^\dagger$ between her shares of $|\bar{\bar{+}}\rangle$ and $|\bar{\bar{m}}\rangle$. Then the nodes apply the Hadamard gate to the control qubits one more time and measure in the standard basis. They announce their measurement results and use the classical decoder to get the value a, just as in VQSS(0) and GTele. Note that the protocol works as long as the gate $C\text{-}XP^\dagger$ can be applied transversally with respect to the code used to encode $|\bar{\bar{0}}\rangle$ and $|\bar{\bar{m}}\rangle$.

Protocol 3 requires an operational workspace of $4n$ qubits per node. First, the verification of $|\bar{\bar{m}}\rangle$ requires a $3n$-qubit workspace per node. After this verification step, each node needs to store n qubits of $|\bar{\bar{m}}\rangle$ and uses an extra $3n$-qubit workspace to verify $|\bar{\bar{0}}\rangle$. This amounts to a $4n$-qubit workspace per node. The communication complexity is the same as in the sequential VQSS protocol, that is, $O(n^2s^2)$ qubits per node, where s is the security parameter.

Protocol 3 [verification of Clifford-stabilized states (VMagic)].
Input: $|0\rangle$ and $|g\rangle$ prepared by D, single-qubit Clifford gate G stabilizing $|g\rangle$, error correcting code $\hat{\mathcal{C}}$, set of apparent cheaters B.
Output: verified logical states $|\bar{\bar{0}}\rangle$ and $|\bar{\bar{g}}\rangle$.
1. The nodes run VQSS(0) with $|0\rangle$ as an input and VQSS with $|g\rangle$ as an input with dealer D. They update set B with apparent cheaters B_0 revealed in verifying $|0\rangle$ and apparent cheaters B_g revealed in verifying $|g\rangle$.
2. Each node ℓ, for all shares coming from node i:
   (a) applies H to $|\bar{\bar{0}}\rangle_{i_\ell}$;
   (b) applies $C\text{-}G$ with $|\bar{\bar{0}}\rangle_{i_\ell}$ as the control qubit and $|\bar{\bar{g}}\rangle_{i_\ell}$ as the target qubit;
   (c) applies H to control qubit;
   (d) measures the control qubit in the Z basis and broadcasts the result using the secure broadcast channel (see paragraph "Network model"; Sec. II).
3. Broadcasted values yield words $\mathbf{v}_i$. Nodes publicly check at which positions the errors occurred using the classical decoder and update set B with the positions of errors. They decode the classical value a:
   (a) If $a = 0$, continue.
   (b) If $a = 1$, set $B = [n]$ (this will cause the MPQC protocol to abort after the computation phase).

C. Multiparty quantum computation

We are now ready to perform a distributed computation using the ingredients from the previous sections. Recall that the goal of the protocol is to perform a circuit $\mathfrak{R}$ in a distributed way on n single-qubit private inputs $\rho_1, \ldots, \rho_n$, each coming from one node $1, \ldots, n$. Note that the inputs can possibly be entangled. In universal MPQC we compute an arbitrary circuit $\mathfrak{R}$. We choose Clifford gates supplemented with a T gate to be our universal set of gates.

Sharing and verification. During this phase the nodes jointly verify whether dealer D_i is honest, i.e., whether there are fewer than $t \leqslant \lfloor \frac{d-1}{2} \rfloor$ errors in the first-level encoding performed by D_i. They publicly record the positions at which the errors occurred in the set of apparent cheaters B_i corresponding to dealer D_i. After all of the dealers are verified, they publicly construct a global set of apparent cheaters B (see step 2 of Protocol 4, below). If $|B| \leqslant t$, the protocol continues. Note that $|B| \leqslant t$ implies that each of the honest nodes holds shares with at most t errors at the same positions of the first level of encoding. Otherwise, when $|B| > t$, the honest nodes know they will abort the protocol after the computation and replace their shares with $|0\rangle$. This step is necessary to complete the security proof.

In this phase each node requires a workspace of $n^2 + 2n$ qubits to verify all of the inputs in a sequential way and sends $(n+1)ns^2$ qubits, where s is the security parameter. The size of the workspace for our MPQC protocol does not depend on s since the verification phase of VQSS is performed in a sequential way.

Computation. In the computation phase, the goal is to compute circuit $\mathfrak{R}$ on the twice-encoded (see Fig. 2) and verified inputs. Note that the set B of apparent cheaters created during the verification is public and cumulative throughout the protocol. This means that it accumulates errors from executions of VMagic, VQSS(0), and GTele in the computation phase. If at any point $|B| > t$ during these protocols, the honest nodes proceed in the same way as in the verification phase: they replace their shares with $|0\rangle$. At the end of the computation phase the nodes look at set B. If $|B| > t$, the protocol aborts. Otherwise, the nodes proceed to the reconstruction phase.

The inputs require a workspace of n^2 qubits per node. For application of the T gate, each node needs a workspace of an additional $4n$ qubits (see Protocol 3; Sec. III B 3). Additionally, the verification of every ancilla in $\mathfrak{R}$ requires a workspace of $3n$ qubits per node. This means that each node requires a workspace of at most $n^2 + 4n$ qubits in total. In this phase, each node sends $O((\#\text{ancillas} + \#T)ns^2)$ qubits.

At this point the nodes hold a global state $\bar{\bar{\omega}}$. Let $\bar{\bar{\omega}}_k = \text{tr}_{[n]\setminus i}(\bar{\bar{\omega}})$ be the outcome of each node i.

Reconstruction. After the computation phase the cheating nodes can still introduce errors to the shares they hold before sending them back to the corresponding dealers. Therefore, each of the dealers, after receiving her original shares back,

runs an error correcting circuit for the code $\hat{\mathcal{C}}$ and identifies further errors. If there are no more than t errors, she reconstructs her output state ω_i. In this phase, the nodes just exchange the existing qubits, therefore the operational workspace does not increase from $n^2 + 4n$. Each node sends n^2 qubits.

Protocol 4 (multiparty quantum computation (MPQC)).
Input: private input ρ_i for every node i, circuit $\mathfrak{R}$, error correcting code $\hat{\mathcal{C}}$.

Sharing and verification

1. Each node $i = 1, \ldots, n$ runs sequential verifiable quantum secret sharing (VQSS; Protocol 1; Sec. III B 1) with single-qubit input ρ_i and code $\hat{\mathcal{C}}$, acting as dealer D_i. This way nodes create logical $\bar{\bar{\Psi}}_i$ encoded twice with $\hat{\mathcal{C}}$ (see Fig. 2).
2. The nodes publicly create sets $B_{i,\ell}$ containing all second-level errors from all n executions of sequential VQSS (see [3] and [20] for details). For each node ℓ, if $|B_{i,\ell}| > t$, then they add node ℓ to the set of apparent cheaters B_i for dealer D_i. After all n executions of VQSS, they create a global set of apparent cheaters $B = \bigcup_i B_i$. If $|B| > t$, the nodes know they will abort after the computation. They replace all the shares they hold with $|0\rangle$.

Computation

3. For every Clifford gate C of circuit $\mathfrak{R}$ the nodes apply C transversally to their local qubits. For every T gate in $\mathfrak{R}$ applied to the input of D_i:
   (a) D_i creates $|0\rangle$ and $|m\rangle$. The nodes run verification of Clifford-stabilized states (VMagic; Protocol 3; Sec. III B 3). The nodes update set B with apparent cheaters from execution of VMagic. If $|B| > t$, the nodes replace all the shares they hold with $|0\rangle$.
   (b) The nodes apply distributed gate teleportation (GTele; Protocol 2; Sec. III B 2) to their shares of $\bar{\bar{\Psi}}_i$ and verified $|\bar{\bar{m}}\rangle$. The nodes update set B with apparent cheaters from execution of GTele. If $|B| > t$, the nodes replace all the shares they hold with $|0\rangle$ and do not apply a correction in GTele (treating the measurement outcome as 0).
4. For every $|0\rangle$ ancilla necessary to perform circuit $\mathfrak{R}$, a node $i \notin B$ chosen at random using the public source of randomness runs VQSS(0) acting as a dealer. She updates B with the set of apparent cheaters from the execution of VQSS(0). The nodes use the verified $|\bar{\bar{0}}\rangle$ to perform $\mathfrak{R}$. If $|B| > t$, the nodes replace all the shares they hold with $|0\rangle$.
5. If $|B| > t$, the protocol aborts. Otherwise, continue.

Let the logical global outcome of the computation be $\bar{\bar{\omega}}$, with $\bar{\bar{\omega}}_i = \mathrm{tr}_{[n]\setminus i}(\bar{\bar{\omega}})$ corresponding to the outcome of each node i.

Reconstruction

6. Each node sends all of the shares of $\bar{\bar{\omega}}_i$ to D_i.
7. Each D_i:
   (a) For each share coming from node $j \notin B$, D_i runs an error correcting circuit for code $\hat{\mathcal{C}}$. She creates a set of errors $\tilde{B}_{i,j}$ such that it contains $B_{i,j}$, i.e., $B_{i,j} \subseteq \tilde{B}_{i,j}$. If $|\tilde{B}_{i,i}| \leqslant t$, then errors are correctable, and D_i corrects them and decodes the ith share obtaining $\bar{\omega}_i$. Otherwise, D_i adds j to the global set B.
   (b) For all $j \notin B$, D_i randomly chooses $n - 2t$ shares of $\bar{\omega}_i$ and applies an erasure-recovery circuit to them. She obtains ω_i.

Altogether, each node requires an operational workspace of $n^2 + 4n$ qubits and sends $O((n + \#\text{ancillas} + \#T)ns^2)$ qubits throughout the execution of the MPQC protocol, Protocol 4 (above).

D. Security statements

In this section we prove the security of our MPQC protocol. To do so, we first state the security framework and definition following the work of [34–37]. We employ the simulator-based security definition (see Definition 1, below). It implies that the three properties—correctness, soundness, and privacy—defined in Sec. I are automatically satisfied. Our security definition uses two models of the protocol: "real" and "ideal." The real model corresponds to the execution of the actual MPQC protocol. In the ideal model the nodes interact with an oracle that perfectly realizes the MPQC task and is incorruptible. The general idea is that the protocol is secure if one cannot distinguish a real execution of MPQC from the ideal one.

In the ideal model the honest nodes can only interact with the oracle. What is more, they do so in a so-called "dummy" way, i.e., they simply forward their input to the oracle and output whatever they receive from the oracle. The cheating nodes can collude and perform any joint operation on their inputs before sending it to the oracle. Similarly, they can perform any joint operation on whatever they receive from the oracle before they output their state. Recall that we do not make any assumption about the computational power of the cheaters. For the purpose of the proof we will say that

the cheaters can be corrupted by an adversary $\mathcal{A}$ who can corrupt at most t nodes but otherwise is arbitrarily powerful. Moreover, by $\mathcal{A}_{\text{real}}$ we denote the adversary in the "real" protocol, and by $\mathcal{A}_{\text{ideal}}$ the adversary in the "ideal" protocol.

Definition 1 (ϵ-security). We say that an MPQC protocol Π is ϵ secure if, for any input state ρ and any real adversary $\mathcal{A}_{\text{real}}$, there exists an ideal adversary $\mathcal{A}_{\text{ideal}}$, such that the output state $\omega_{\text{real}} := \Pi_{\text{real}}(\rho)$ of the real protocol is ϵ close to the output state $\omega_{\text{ideal}} := \Pi_{\text{ideal}}(\rho)$ of the ideal protocol, that is,

$$\frac{1}{2} \| \omega_{\text{real}} - \omega_{\text{ideal}} \|_1 \leqslant \epsilon. \tag{8}$$

To prove the security of the MPQC protocol, Protocol 4 (Sec. III C), we first restate the soundness of the VQSS protocol [3,20,21].

Lemma 1 (soundness of VQSS). In the verifiable quantum secret sharing protocol, Protocol 1 (Sec. III B 1), either the honest parties hold a consistently encoded secret or the dealer is caught with a probability of at least $1 - 2^{-\Omega(s)}$.

Theorem 1. The multiparty quantum computation protocol, Protocol 4 (Sec. III C), is $\kappa 2^{-\Omega(s)}$ secure, where $\kappa = n + \#T$ gates + #ancillas in $\mathfrak{R}$.

Idea of the proof. Our proof is inspired by the approach taken in [3] and [21], on which we expand and explicitly show that the outputs of the real and ideal protocol are ϵ close (see the Appendix). We construct an ideal protocol using a common simulation technique, where $\mathcal{A}_{\text{ideal}}$ locally simulates the MPQC protocol, Protocol 4 (Sec. III C), with honest nodes interacting with the cheaters. This means that for any real adversary $\mathcal{A}_{\text{real}}$ we construct an ideal adversary $\mathcal{A}_{\text{ideal}}$ by saying that it internally simulates the execution of a real protocol with the real adversary $\mathcal{A}_{\text{real}}$. Then we formally write the execution of the real protocol. We show that the outputs of both protocols are equal in the case where the encoding in the sharing phase of Protocol 4 is done correctly. We also prove that the ϵ error in the security comes from the fact that the verification of inputs and any ancillas needed for MPQC can fail with a probability defined by Lemma 1 (above).

We remark that our security definition follows the paradigm of sequential composability, formalized by the real-vs-ideal security definition, Definition1 (above). The extendability of our security definition to the more general framework of universal composability [36,37] is left as an open problem.

IV. DISCUSSION

In our protocol we allow an abort event when there are too many errors introduced by the cheaters (see Protocol 4; Sec. III C). However, this condition can be removed following the approach outlined in [3] and [21] (there called *top-level sharing*), at the cost of more rounds of quantum communication. Given that our objective is to save resources, we did not pursue this path in this paper. However, we can introduce a step before computation, in which the nodes perform a distributed encoding (creating the third level of encoding) of the verified inputs. In the following we expand on how the distributed encoding works and how it modifies the subsequent steps of our MPQC protocol.

The nodes run the VQSS verification procedure for every input state ρ_i but do not create a global set of cheaters. Instead, they create a set B_i recording first-level errors in input state ρ_i. To perform the distributed encoding of input ρ_i the nodes use ancilla states prepared and encoded by the corresponding dealer D_i. The nodes also verify the ancillas using VQSS and add the errors that occurred in the first level of encoding of ancillas to B_i. If $|B_i| \leqslant t$, the nodes perform the distributed encoding with the verified ancillas. The encoding can be done transversally, since for any stabilizer error correcting code the encoding procedure is a Clifford operation [38].

If a dealer is caught cheating, $|B_i| > t$, the protocol does not abort. Instead, a node which has not been caught cheating yet prepares an encoding of $|0\rangle$ and the nodes proceed to verify it in the same way as before. Note that there will be at most t failed tries in preparing a valid encoding of $|0\rangle$ since there are at most t cheaters. Otherwise, upon a successful verification of the encoded $|0\rangle$, the nodes proceed to the distributed encoding. This step replaces the invalid input from the cheater with a valid encoding of $|0\rangle$. The same procedure, "try until you succeed," can be adapted to verify magic states and $|0\rangle$ ancillas needed to perform circuit $\mathfrak{R}$. The nodes simply try until the verification of an ancilla has at most t errors.

Performing the distributed encoding of the inputs adds the third level of encoding before the computation phase. To perform the computation, the shares initially dealt by dealer D_i are then sent back to D_i, who reconstructs them and corrects the errors using the reconstruction step from VQSS (as in Reconstruction of the MPQC protocol; Sec. III C). Note that this reconstruction procedure removes two levels of encoding. As a result, each node holds a single qubit corresponding to a correctly encoded input state ρ_i, with at most t errors confined to the cheaters' positions. The protocol proceeds with the distributed computation, but now the circuit is performed on a single level of encoding. Since the errors are only on the shares held by the cheaters, the errors will not propagate to the honest shares during the computation. Therefore, after the computation it will be possible to reconstruct outputs for the honest nodes.

We remark that implementing the distributed encoding in our MPQC protocol, Protocol 4, can be done in a sequential way, similar to the execution of VQSS we present in Protocol 1 (Sec. III B 1). In fact, this does not increase the qubit workspace per node; no node will exceed the workspace of $n^2 + 4n$. However, this approach has a significantly higher quantum communication complexity. Specifically, in this version of the protocol, each node needs to send $O(n^5 s^2)$ qubits.

Finally, an implementation of our MPQC protocol in a quantum network can suffer from noise, for example, in the communication channels or in local gates and storage. Therefore, a careful analysis of the influence of noise on the MPQC protocol and the security statements is still required. Some steps towards analyzing the influence of noise in the sending channels have already been taken in the context of the verifiable quantum secret sharing protocol [39]. The work analyzes the correctness and soundness of the VQSS protocol in the presence of depolarizing and erasure noise in the communication channels. It also proposes ways to improve the performance of the protocol under these conditions. We expect that these results should generalize to the MPQC setting.

ACKNOWLEDGMENTS

We thank B. Dirkse for useful discussions and K. Chakraborty and M. Skrzypczyk for detailed feedback on the manuscript. This work was supported by an NWO VIDI Grant, an ERC Starting Grant, and NWO Zwaartekracht QSC. This project (QIA) has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 820445.

APPENDIX: SECURITY PROOF

Here we provide the security proof of our protocol based on the simulator definition (see Definition 1; Sec. III D). We first construct the ideal protocol step by step and model each operation performed in this protocol by general maps and, finally, express the output of this protocol ω_{ideal} in terms of these maps. Then we analyze the real protocol and similarly express its output ω_{real} in terms of the maps modeling the real protocol. Finally, we compare the two outputs, ω_{ideal} and ω_{real}, and show that they are exponentially close in the security parameter s.

To prove security of the MPQC protocol, Theorem 1 (Sec. III D), we first state the following useful lemma. Intuitively, it says that sharing and verifying the input, performing the distributed circuit, and decoding are equivalent to applying the circuit to the inputs directly. Note that we consider the decoding to be "hypothetical": after the computation phase in MPQC the nodes send all of the shares coming from input of node i to node i, and node i reconstructs it.

Lemma 2. Let B be a set of apparent cheaters at the end of the computation phase, such that $|B| \leqslant t$, and A be a set of cheaters. Let $\mathcal{D}$ denote the decoding procedure for code $\hat{\mathcal{C}}$ and $\hat{\mathcal{D}}$ denote the erasure recovery circuit for code $\hat{\mathcal{C}}$. If the state $\bar{\bar{\rho}}$ encoded twice with the code $\hat{\mathcal{C}}$ is decodable, i.e.

$$\rho = \bigotimes_{i\in[n]} \left(\hat{\mathcal{D}}_{\overline{B\cup A}} \circ \bigotimes_{\ell\in\overline{B\cup A}} \mathcal{D}_\ell \right)(\bar{\bar{\rho}}), \tag{A1}$$

then applying a logical gate $\bar{\bar{G}}$ ($G \in \text{Cliff} + T$) on $\bar{\bar{\rho}}$ is also decodable, i.e.,

$$G(\rho) = \bigotimes_{i\in[n]} \left(\hat{\mathcal{D}}_{\overline{B\cup A}} \circ \bigotimes_{\ell\in\overline{B\cup A}} \mathcal{D}_\ell \right)(\bar{\bar{G}}(\bar{\bar{\rho}})), \tag{A2}$$

where $\bar{\bar{G}}$ is gate G applied transversally on the CSS code $\hat{\mathcal{C}}$ if $G \in$ Cliff, or $\bar{\bar{G}}$ is the implementation of the T gate described in Protocol 2 (Sec. III B 2) if $G = T$. The same property holds when replacing G with the projective measurement in the Z basis denoted P and where $\bar{\bar{P}}$ corresponds to measuring each qubit of the double-encoded state in the Z basis followed by broadcasting the outcome classically.

Proof. Lemma 2 follows from the fact that to realize a logical gate $\bar{\bar{G}}$ it is sufficient to apply G honestly on shares in $\overline{B \cup A}$. Indeed, applying a Clifford gate transversally on shares in $\overline{B \cup A}$ realizes a logical Clifford gate [30]. For a CSS code $\hat{\mathcal{C}}$ measuring each qubit in the Z basis and broadcasting the measurement result realizes the logical transversal measurement. Additionally, we implement the T gate by composing an ancilla state, the Z measurement, and a Clifford operation. Therefore, the transversal properties of Cliffords and Z measurement can be transferred to this implementation of the T gate.

Property 1. Let $\mathfrak{R}$ be a circuit implementing a completely positive trace preserving (CPTP) map. Lemma 2 holds when replacing G with any circuit $\mathfrak{R}$,

$$\mathfrak{R}(\rho) = \bigotimes_{i\in[n]} \left(\hat{\mathcal{D}}_{\overline{B\cup\tilde{H}}} \circ \bigotimes_{\ell\in\overline{B\cup\tilde{H}}} \mathcal{D}_\ell \right)(\bar{\bar{\mathfrak{R}}}(\bar{\bar{\rho}})). \tag{A3}$$

This follows from the fact that any circuit $\mathfrak{R}$ can be represented as $\mathfrak{R} = P \circ \mathcal{U}$, where $\mathcal{U}$ can be decomposed into gates from the set Cliff $+ T$ and P is a measurement.

As a reminder, let us restate the security of our MPQC protocol.

The multiparty quantum computation protocol, Protocol 4 (Sec. III C), is $\kappa 2^{-\Omega(s)}$ secure, where $\kappa = n + \#T$ gates $+$ #ancillas in $\mathfrak{R}$.

Proof of Theorem 1. This proof is inspired by the approach taken in [3] and [2]. In the following we construct a proof aiming to show that the outputs of the real and ideal protocols are ϵ close. We first construct an ideal protocol using a simulator approach and formally state every step of the simulation. We then formally write the execution of the real protocol.

Box 1. Registers used in the security proof.

Ideal protocol:

H_S: registers of "dummy" inputs of the honest nodes in the simulation.
A_S: registers of the cheaters' inputs.
H_0: registers of the simulated honest nodes.
A_0: registers of the simulated cheaters.

Real protocol:

H_R: registers of honest nodes.
A_R: registers of cheaters.

1. Ideal protocol

$\mathcal{A}_{\text{ideal}}$ will locally simulate the MPQC protocol, Protocol 4 (Sec. III C), with honest nodes interacting with the cheaters. The cheaters are controlled by $\mathcal{A}_{\text{real}}$, and $\mathcal{A}_{\text{real}}$ is simulated within $\mathcal{A}_{\text{ideal}}$ (see Fig. 5). In the ideal model $\mathcal{A}_{\text{ideal}}$ and the honest nodes interact with an oracle that perfectly realizes the MPQC task and is incorruptible. The oracle requires two types of inputs: first, the input registers H_S, A_0 on which the computation of the circuit will occur; second, a flag input indicates whether the oracle should abort or continue. If the flag input is "abort," the oracle outputs $|\perp\rangle\langle\perp|$. If the flag input is "continue," the oracle outputs the evaluation of circuit $\mathfrak{R}$ on the inputs $H_S A_0$. At any moment in this simulated execution, the ideal adversary has access to all the simulated registers, in particular, the set B of apparent cheaters. Let the input to the simulation be

$$\rho_{H_S A_S} \otimes |0\rangle\langle 0|_{H_0 A_0}, \tag{A4}$$

where $\rho_{H_S A_S}$ denotes the input state of all nodes, such that $\text{tr}_{[n]\setminus i}(\rho_{H_S A_S}) = \rho_i$.

(1) $\mathcal{A}_{\text{ideal}}$ locally simulates sharing and verification with simulated honest nodes using $|0\rangle$ as their input. The input

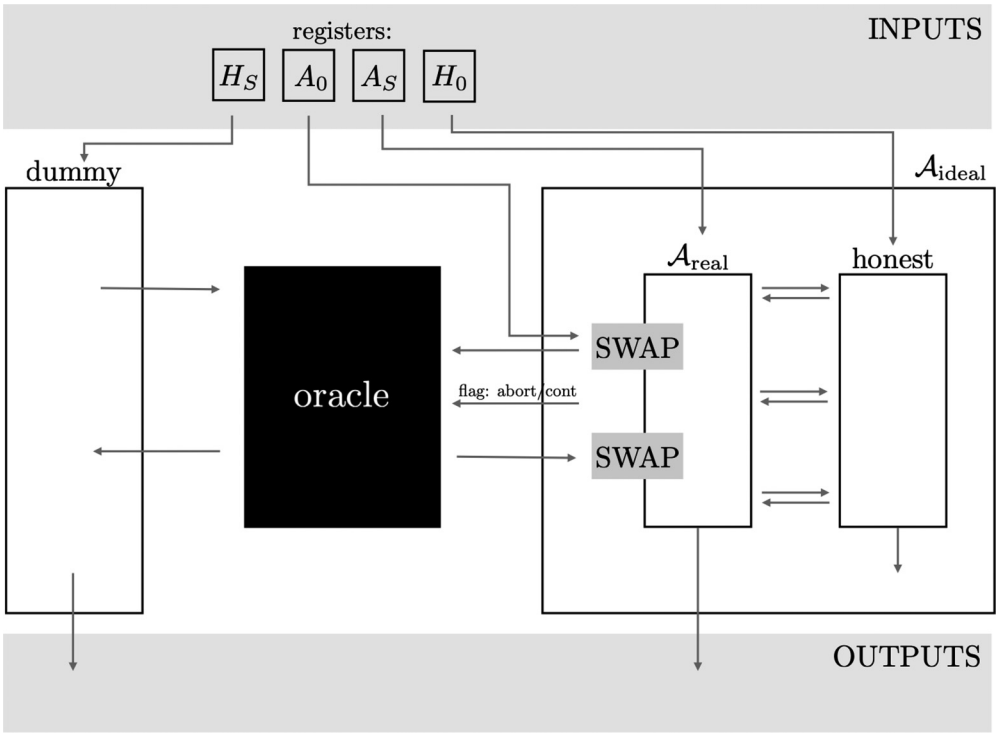

FIG. 5. Schematic of our simulator-based security proof of the MPQC protocol, Protocol 4 (Sec. III C).

registers H_0A_S given to $\mathcal{A}_{\text{ideal}}$ are forwarded to the simulated $\mathcal{A}_{\text{real}}$, i.e.,

$$\sigma^{(1)}_{H_0A_0H_SA_S} = \mathcal{SV}_{H_0A_S}\left(\rho_{H_SA_S} \otimes |0\rangle\langle 0|_{H_0A_0}\right), \tag{A5}$$

where $\mathcal{SV}_{H_0A_S}$ denotes the sharing and verification (see Protocol 4; Sec. III C) performed on registers H_0 and A_S. We assume that the identity operation is applied on all the registers that are not on the map $\mathcal{SV}$, i.e., $\mathbb{1}_{H_SA_0}$.

(2) Before $\mathcal{A}_{\text{ideal}}$ proceeds with the simulation of the computation phase, for each input of the cheaters $\mathcal{A}_{\text{ideal}}$ creates an encoding of $|0\rangle$ in register A_0. Then $\mathcal{A}_{\text{ideal}}$ performs a swap gate between A_0 and cheaters' input A_S.

(a) In the case where set $|B| \leqslant t$, there are sufficiently few errors on both levels of encoding. Then $\mathcal{A}_{\text{ideal}}$ can apply an erasure-recovery circuit twice (for the double encoding), denoted $\tilde{\mathcal{D}}_{A_0}$, to the input of nodes not in B and pass it to the oracle. Applying decoding $\tilde{\mathcal{D}}_{A_0}$ is necessary, since the oracle accepts only single-qubit inputs.

(b) Otherwise, when $|B| > t$, $\mathcal{A}_{\text{ideal}}$ simply passes previously prepared $|0\rangle$ states as inputs of the cheaters to the oracle and the simulated honest nodes H_S replace their shares with $|0\rangle$. The simulated cheaters apply an arbitrary map $\mathcal{M}_{A_S}$ to their shares.

We therefore describe this step as

$$\sigma^{(2)}_{H_0A_0H_SA_S} = \begin{cases} \tilde{\mathcal{D}}_{A_0} \circ \text{Swap}_{A_0A_S} \circ \mathcal{E}_{A_0}\left(\sigma^{(1)}_{H_0A_0H_SA_S}\right) & \text{if} \quad |B| \leqslant t, \\ \mathcal{M}_{A_S} \otimes \text{tr}_{H_0}\left[\sigma^{(1)}_{H_0A_0H_SA_S}\right] \otimes |0\rangle\langle 0|_{H_0} & \text{if} \quad |B| > t. \end{cases} \tag{A6}$$

(3) $\mathcal{A}_{\text{ideal}}$ proceeds with the simulation of the computation phase on registers H_0 and A_S. At the same time, the oracle computes the ideal circuit $\mathfrak{R}^{\text{ideal}}_{H_SA_0}$ on the simulated honest shares H_S and register A_0 of the cheaters. The state after this step is, therefore,

$$\sigma^{(3)}_{H_0A_0H_SA_S} = \begin{cases} \left(\mathfrak{R}^{\text{ideal}}_{H_SA_0} \otimes \bar{\bar{\mathfrak{R}}}_{H_0A_S}\right)\left(\sigma^{(2)}_{H_0A_0H_SA_S}\right) & \text{if} \quad |B| \leqslant t, \\ \left(\mathfrak{R}^{\text{ideal}}_{H_SA_0} \otimes \bar{\bar{\mathfrak{R}}}_{H_0A_S}\right)\left(\sigma^{(2)}_{H_0A_0H_SA_S}\right) & \text{if} \quad |B| > t. \end{cases} \tag{A7}$$

(4) If $|B| > t$, $\mathcal{A}_{\text{ideal}}$ sends the flag "abort" to the oracle, and otherwise sends "continue."

(a) If the oracle receives "abort," it outputs a flag $|\perp\rangle\langle\perp|$ to all nodes.

(b) Otherwise, it outputs the computation of the ideal circuit on the inputs.

(5) The nodes in H_S output whatever they received from the oracle. Upon receiving the oracle's output, $\mathcal{A}_{\text{ideal}}$ does the following:

(a) If "abort" was sent in the previous step, then it must be that $|B| > t$. The simulated protocol aborts. Therefore, $\mathcal{A}_{\text{ideal}}$ outputs the output of the $\mathcal{A}_{\text{real}}$. Note that the simulated cheaters could have applied an arbitrary map $\mathcal{M}'_{A_S}$ on their register.

(b) If "continue" was sent in the previous step, then $\mathcal{A}_{\text{ideal}}$ applies double encoding $\mathcal{E}_{A_0}$ to all shares of the cheating nodes A_0. Then $\mathcal{A}_{\text{ideal}}$ applies the swap gate between the simulated registers of cheaters A_S and A_0 and proceeds to the next step.

$$\begin{aligned} &\text{Swap}_{A_0A_S}\circ\mathcal{E}_{A_0}\circ\big(\mathfrak{R}^{\text{ideal}}_{H_SA_0}\otimes\bar{\bar{\mathfrak{R}}}_{H_0A_S}\big)\big(\sigma^{(2)}_{H_0A_0H_SA_S}\big)\otimes|\text{cont}\rangle\langle\text{cont}| \quad \text{if} \quad |B|\leqslant t,\\ &|\perp\rangle\langle\perp|_{H_SA_0}\otimes\text{tr}_{H_SA_0}\big[\mathcal{M}'_{A_S}\big(\sigma^{(3)}_{H_0A_0H_SA_S}\big)\big]\otimes|\text{abort}\rangle\langle\text{abort}| \quad \text{if} \quad |B|>t. \end{aligned} \tag{A8}$$

Let us denote the following expression by $\sigma^{(5)}_{H_0A_0H_SA_S}$, where we use the explicit form of $\sigma^{(2)}_{H_0A_0H_SA_S}$ for $|B|\leqslant t$, Eq. (A7):

$$\sigma^{(5)}_{H_0A_0H_SA_S}=\text{Swap}_{A_0A_S}\circ\mathcal{E}_{A_0}\circ\big(\mathfrak{R}^{\text{ideal}}_{H_SA_0}\otimes\bar{\bar{\mathfrak{R}}}_{H_0A_S}\big)\circ\tilde{\mathcal{D}}_{A_0}\circ\text{Swap}_{A_0A_S}\circ\mathcal{E}_{A_0}\big(\sigma^{(1)}_{H_0A_0H_SA_S}\big). \tag{A9}$$

We now simplify the above expression. For this we first state the following useful property.

Property 2. For any operation O_{ABCD} on registers $ABCD$, the following identity holds:

$$\text{Swap}_{BC}\circ O_{ABCD}\circ\text{Swap}_{BC}=O_{ACBD}. \tag{A10}$$

Using this property for $\sigma^{(5)}_{H_0A_0H_SA_S}$ we get that

$$\text{Swap}_{A_0A_S}\circ\mathcal{E}_{A_0}\circ\big(\mathfrak{R}^{\text{ideal}}_{H_SA_0}\otimes\bar{\bar{\mathfrak{R}}}_{H_0A_S}\big)\circ\tilde{\mathcal{D}}_{A_0}\circ\text{Swap}_{A_0A_S}\circ\mathcal{E}_{A_0}=\mathcal{E}_{A_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}\circ\tilde{\mathcal{D}}_{A_S}\otimes\bar{\bar{\mathfrak{R}}}_{H_0A_0}\circ\mathcal{E}_{A_0}. \tag{A11}$$

This means that that the composition of the swaps with the ideal circuit performed by the oracle is equivalent to applying the ideal circuit to registers H_SA_S by the oracle. Therefore, we can simplify $\sigma^{(5)}_{H_0A_0H_SA_S}$ to

$$\sigma^{(5)}_{H_0A_0H_SA_S}=\big(\mathcal{E}_{A_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}\circ\tilde{\mathcal{D}}_{A_S}\big)\otimes\big(\bar{\bar{\mathfrak{R}}}_{H_0A_0}\circ\mathcal{E}_{A_0}\big)\big(\sigma^{(1)}_{H_0A_0H_SA_S}\big), \tag{A12}$$

and using Eq. (A5) we obtain

$$\sigma^{(5)}_{H_0A_0H_SA_S}=\big(\mathcal{E}_{A_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}\circ\tilde{\mathcal{D}}_{A_S}\big)\otimes\big(\bar{\bar{\mathfrak{R}}}_{H_0A_0}\circ\mathcal{E}_{A_0}\circ\mathcal{SV}_{H_0A_S}\big)\big(\rho_{H_SA_S}\otimes|0\rangle\langle0|_{H_0A_0}\big) \tag{A13}$$

$$=\big(\mathcal{E}_{A_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}\circ\tilde{\mathcal{D}}_{A_S}\circ\mathcal{SV}_{A_S}(\rho_{H_SA_S})\big)\otimes\big(\bar{\bar{\mathfrak{R}}}_{H_0A_0}\circ\mathcal{E}_{A_0}\circ\mathcal{SV}_{H_0}\big(|0\rangle\langle0|_{H_0A_0}\big)\big). \tag{A14}$$

(6) If the protocol did not abort, $\mathcal{A}_{\text{ideal}}$ proceeds to the reconstruction phase, in which the simulated honest nodes H_0 first use the decoding procedure for code $\hat{\mathcal{C}}$ and then apply an erasure recovery circuit, as in the Reconstruction phase of Protocol 4 (Sec. III C). We denote this procedure $\tilde{\mathcal{D}}_{H_0}$. On the other hand, the simulated cheaters A_S apply an arbitrary map $\mathcal{W}_{A_S}$. $\mathcal{A}_{\text{ideal}}$ outputs whatever is the output of the simulated $\mathcal{A}_{\text{real}}$. Therefore, the output of the ideal protocol is

$$\omega_{\text{ideal}}=\text{tr}_{H_0A_0}\big[\tilde{\mathcal{D}}_{H_0}\otimes\mathcal{W}_{A_S}\big(\sigma^{(5)}_{H_0A_0H_SA_S}\big)\big]. \tag{A15}$$

Using Eq. (A14) and the fact that the sharing and verification followed by the double decoding, $\tilde{\mathcal{D}}_{A_S}\circ\mathcal{SV}_{A_S}$, is equivalent to $\mathbb{1}_{A_S}$, we obtain

$$\omega_{\text{ideal}}=\mathcal{W}_{A_S}\circ\mathcal{E}_{A_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}(\rho_{H_SA_S}). \tag{A16}$$

Similarly, for later comparison with the real protocol, we write the identity map on H_S as $\mathbb{1}_{A_S}=\tilde{\mathcal{D}}_{H_S}\circ\mathcal{E}_{H_S}$ and get

$$\omega_{\text{ideal}}=\big(\tilde{\mathcal{D}}_{H_S}\otimes\mathcal{W}_{A_S}\big)\circ\mathcal{E}_{H_SA_S}\circ\mathfrak{R}^{\text{ideal}}_{H_SA_S}\big(\rho_{H_SA_S}\big). \tag{A17}$$

2. Real protocol

In the real protocol whenever the honest nodes observe $|B|>t$ they replace all of their shares with $|0\rangle$. This is necessary because in the ideal protocol the oracle receives "abort" at the end of the computation phase. Therefore, in the real protocol we also abort at the end of the computation phase. However, it could happen that in the case where $|B|>t$, continuing the computation leaks some information about the honest nodes' inputs. To avoid this situation, we make the honest nodes substitute their shares with $|0\rangle$.

(1) The protocol starts with the sharing and verification phase, which we describe by the map $\mathcal{SV}$ acting on inputs of all the nodes $\rho_{H_RA_R}$. The state after this step is

$$\mathcal{SV}_{H_RA_R}(\rho_{H_RA_R}). \tag{A18}$$

(2) The protocol continues:

(a) In the case where $|B|\leqslant t$, the nodes apply the distributed circuit $\bar{\bar{\mathfrak{R}}}_{H_RA_R}$.

(b) In the case where $|B|>t$, the honest nodes replace their shares with $|0\rangle$ and the cheaters apply an arbitrary map $\mathcal{M}_{A_R}$.

At the end of the computation phase the state is, therefore,

$$\sigma^{(2)}_{H_RA_R}=\begin{cases}\bar{\bar{\mathfrak{R}}}_{H_RA_R}\circ\mathcal{SV}_{H_RA_R}(\rho_{H_RA_R}) & \text{if} \quad |B|\leqslant t,\\ \mathcal{M}_{A_R}(\text{tr}_{H_R}[\mathcal{SV}_{H_RA_R}(\rho_{H_RA_R})])\otimes|0\rangle\langle0|_{H_R} & \text{if} \quad |B|>t.\end{cases} \tag{A19}$$

(3) The nodes check the size of set B.

(a) If $|B| \leqslant t$, then the protocol continues to the reconstruction phase, where the honest nodes apply first a decoding operator for code $\hat{\mathcal{C}}$ and then an interpolation circuit, denoted $\mathcal{D}_{H_R}$. At the same time, the cheaters can apply an arbitrary map on their registers, which we denote $\mathcal{W}_{A_R}$.

(b) In the case where $|B| > t$, the nodes output the abort flag $|\perp\rangle\langle\perp|$ and the cheaters output their part of $\sigma^{(2)}_{H_R A_R}$, possibly with an arbitrary map $\mathcal{M}'_{A_R}$. The protocol aborts.

We can describe this step as

$$\begin{aligned} &(\mathcal{D}_{H_R} \otimes \mathcal{W}_{A_R}) \circ \bar{\bar{\mathfrak{R}}}_{H_R A_R} \circ \mathcal{SV}_{H_R A_R}(\rho_{H_R A_R}) \otimes |\text{cont}\rangle\langle\text{cont}| \quad \text{if} \quad |B| \leqslant t, \\ &|\perp\rangle\langle\perp|_{H_R} \otimes \mathcal{M}'_{A_R}\left(\text{tr}_{H_R}\left[\sigma^{(2)}_{H_R A_R}\right]\right) \otimes |\text{abort}\rangle\langle\text{abort}| \quad \text{if} \quad |B| > t. \end{aligned} \tag{A20}$$

We introduce the identity map as encoding followed by double encoding on both registers, i.e., $\mathbb{1}_{H_R A_R} = \tilde{\mathcal{D}}_{H_R A_R} \circ \mathcal{E}_{H_R A_R}$. Then, plugging in this $\mathbb{1}_{H_R A_R}$ between $\bar{\bar{\mathfrak{R}}}_{H_R A_R}$ and $(\mathcal{D}_{H_R} \otimes \mathcal{W}_{A_R})$, the first case can be rewritten as

$$\omega_{\text{real}} = \left(\mathcal{D}_{H_R} \otimes \mathcal{W}_{A_R}\right) \circ \mathcal{E}_{H_R A_R} \circ \tilde{\mathcal{D}}_{H_R A_R} \circ \bar{\bar{\mathfrak{R}}}_{H_R A_R} \circ \mathcal{SV}_{H_R A_R}\left(\rho_{H_R A_R}\right), \tag{A21}$$

which defines the output of the real protocol when it does not abort.

Now we aim to simplify ω_{real} to compare it to the output of the ideal protocol. Our goal is to show that sharing and verifying the input, performing the distributed circuit, and decoding are equivalent to applying the circuit to the inputs directly:

$$\tilde{\mathcal{D}}_{H_R A_R} \circ \bar{\bar{\mathfrak{R}}}_{H_R A_R} \circ \mathcal{SV}_{H_R A_R}(\rho_{H_R A_R}) = \mathfrak{R}_{H_R A_R}(\rho_{H_R A_R}). \tag{A22}$$

Indeed, this follows from Lemma 2 and Property 1. By security of the VQSS [3,20,21], if the protocol does not abort, there exists a unique double-encoded state after the verification phase, i.e., $\mathcal{SV}_{H_R A_R}(\rho_{H_R A_R})$. By definition the decoding $\tilde{\mathcal{D}}_{H_R A_R}$ is exactly the one performed in Lemma 2. Therefore, we have that

$$\omega_{\text{real}} = \left(\mathcal{D}_{H_R} \otimes \mathcal{W}_{A_R}\right) \circ \mathcal{E}_{H_R A_R} \circ \tilde{\mathcal{D}}_{H_R A_R} \circ \bar{\bar{\mathfrak{R}}}_{H_R A_R} \circ \mathcal{E}_{H_R A_R}\left(\rho_{H_R A_R}\right) \tag{A23}$$

$$= \left(\mathcal{D}_{H_R} \otimes \mathcal{W}_{A_R}\right) \circ \mathcal{E}_{H_R A_R} \circ \mathfrak{R}_{H_R A_R}\left(\rho_{H_R A_R}\right). \tag{A24}$$

This, together with Eq. (A17), gives us that the outputs of the ideal and real protocols are equal for $|B| \leqslant t$,

$$\omega_{\text{ideal}} = \omega_{\text{real}}. \tag{A25}$$

Similarly, when $|B| > t$, one can compare (A7) with (A8) and obtain that the states are the same for the real and ideal protocols.

What we have described so far considers that the encoding in the sharing phase was performed correctly in the real protocol. However, this does not have to be the case. Every verification performed during the MPQC has a probability of error inherited from the VQSS. Recall that from Lemma 2 (Sec. III D) the probability of unsuccessful verification in VQSS is lower-bounded by $2^{-\Omega(s)}$. In MPQC we verify:

(a) each of the n inputs,

(b) each $|\bar{\bar{0}}\rangle$ and $|\bar{\bar{m}}\rangle$ necessary to perform the T gate, and

(c) each $|\bar{\bar{0}}\rangle$ necessary for circuit $\mathfrak{R}$.

Let $\kappa = n + \#T\text{gates} + \#\text{ancillas for } \mathfrak{R}$. Then the total probability of error in MPQC is $\kappa 2^{-\Omega(s)}$. ■

[1] A. C. Yao, in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, *SFCS '82* (IEEE Computer Society, New York, 1982), pp. 160–164.

[2] R. Cramer, I. B. Damgard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed. (Cambridge University Press, New York, 2015).

[3] C. Crépeau, D. Gottesman, and A. Smith, in *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, *STOC '02* (ACM Press, New York, 2002), pp. 643–652.

[4] D. Chaum, C. Crépeau, and I. Damgard, in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, *STOC '88* (ACM Press, New York, 1988), pp. 11–19.

[5] C. Crépeau, D. Gottesman, and A. Smith, in *Advances in Cryptology—EUROCRYPT 2005*, edited by R. Cramer (Springer, Berlin, 2005), pp. 285–301.

[6] F. Dupuis, J. B. Nielsen, and L. Salvail, in *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2012* (Springer-Verlag, Berlin, 2012), Vol. 7417, pp. 794–811.

[7] Y. Dulek, A. B. Grilo, S. Jeffery, C. Majenz, and C. Schaffner, in *Advances in Cryptology—EUROCRYPT 2020*, edited by A. Canteaut and Y. Ishai, Lecture Notes in Computer Science Vol. 12107 (Springer, Cham, 2020).

[8] D. Mayers, Phys. Rev. Lett. **78**, 3414 (1997).

[9] H.-K. Lo and H. F. Chau, Physica D: Nonlin. Phenom. **120**, 177 (1998).

[10] B. Eastin and E. Knill, Phys. Rev. Lett. **102**, 110502 (2009).

[11] D. Gottesman and I. L. Chuang, Nature **402**, 390 (1999).

[12] D. Aharonov and M. Ben-Or, in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, *STOC '97* (ACM Press, New York, 1997), pp. 176–188.

[13] E. M. Rains, IEEE Trans. Inf. Theor. **45**, 1827 (1999).

[14] M. Grassl, T. Beth, and M. Rotteler, Int. J. Quantum Inf. **02**, 55 (2004).

[15] S. Bravyi, G. Smith, and J. A. Smolin, Phys. Rev. X **6**, 021043 (2016).
[16] M. Steudtner and S. Wehner, New J. Phys. **20**, 063010 (2018).
[17] N. Moll, A. Fuhrer, P. Staar, and I. Tavernelli, J. Phys. A **49**, 295301 (2016).
[18] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme, arXiv:1701.08213.
[19] T. Peng, A. Harrow, M. Ozols, and X. Wu, arXiv:1904.00102.
[20] V. Lipinska, G. Murta, J. Ribeiro, and S. Wehner, Phys. Rev. A **101**, 032332 (2020).
[21] A. Smith, arXiv:quant-ph/0111030.
[22] A. Steane, Proc. R. Soc. London A **452**, 2551 (1996).
[23] R. Canetti, in *Proceedings of the 17th IEEE Computer Security Foundations Workshop* (IEEE, New York, 2004), pp. 219–233.
[24] H. Barnum, C. Crepeau, D. Gottesman, A. Smith, and A. Tapp, in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (IEEE, New York, 2002), pp. 449–458.
[25] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies: IEEE INFOCOM 99, The Future Is Now* (IEEE, New York, 1999), Vol. 2, pp. 708–716.
[26] T. Rabin and M. Ben-Or, in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, *STOC '89* (ACM Press, New York, 1989), pp. 73–85.
[27] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).
[28] D. Gottesman, Phys. Rev. A **61**, 042311 (2000).
[29] A. J. Landahl, J. T. Anderson, and P. R. Rice, arXiv:1108.5738.
[30] D. Gottesman, Phys. Rev. A **57**, 127 (1998).
[31] A. M. Steane, Phys. Rev. Lett. **78**, 2252 (1997).
[32] G. Nebe, E. M. Rains, and N. J. A. Sloane, Designs Codes Cryptogr. **24**, 99 (2001).
[33] A. Y. Kitaev, Russ. Math. Surv. **52**, 1191 (1997).
[34] D. Beaver, in *Advances in Cryptology—CRYPTO '91*, edited by J. Feigenbaum (Springer, Berlin, 1992), pp. 377–391.
[35] S. Micali and P. Rogaway, in *Advances in Cryptology—CRYPTO '91*, edited by J. Feigenbaum (Springer, Berlin, 1992), pp. 392–404.
[36] R. Canetti, in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science* (IEEE, New York, 2001), pp. 136–145.
[37] D. Unruh, in *Advances in Cryptology—EUROCRYPT 2010*, edited by H. Gilbert (Springer, Berlin, 2010), pp. 486–505.
[38] D. Gottesman, arXiv:0904.2557.
[39] A. G. Iñesta, Master's thesis, Delft University of Technology (2020).