

Efficient self-testing system for quantum computations based on permutations

Shuquan Ma(马树泉), Changhua Zhu(朱畅华), Min Nie(聂敏), and Dongxiao Quan(权东晓)
Citation: Chin. Phys. B, 2021, 30 (**4**): 040305. DOI: 10.1088/1674-1056/abe29a
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Taking tomographic measurements for photonic qubits 88 ns before they are created

Zhibo Hou(侯志博), Qi Yin(殷琪), Chao Zhang(张超), Han-Sen Zhong(钟翰森), Guo-Yong Xiang(项国勇)†, Chuan-Feng Li(李传锋), Guang-Can Guo(郭光灿), Geoff J. Pryde, and Anthony Laing‡
Chin. Phys. B, 2021, 30 (**4**): 040304. DOI: 10.1088/1674-1056/abe29c

Realization of arbitrary two-qubit quantum gates based on chiral Majorana fermions

Qing Yan(闫青) and Qing-Feng Sun(孙庆丰)
Chin. Phys. B, 2021, 30 (**4**): 040303. DOI: 10.1088/1674-1056/abe296

Nonlocal advantage of quantum coherence in a dephasing channel with memory

Ming-Liang Hu(胡明亮), Yu-Han Zhang(张宇晗), and Heng Fan(范桁)
Chin. Phys. B, 2021, 30 (**3**): 030308. DOI: 10.1088/1674-1056/abcf4a

A proposal for preparation of cluster states with linear optics

Le Ju(鞠乐), Ming Yang(杨名), and Peng Xue(薛鹏)
Chin. Phys. B, 2021, 30 (**3**): 030306. DOI: 10.1088/1674-1056/abd74b

Quantum algorithm for a set of quantum 2SAT problems

Yanglin Hu(胡杨林), Zhelun Zhang(张哲伦), and Biao Wu(吴飙)
Chin. Phys. B, 2021, 30 (**2**): 020308. DOI: 10.1088/1674-1056/abd741

SPECIAL TOPIC — Quantum computation and quantum simulation

Efficient self-testing system for quantum computations based on permutations*

Shuquan Ma(马树泉)[1], Changhua Zhu(朱畅华)[1,2,†], Min Nie(聂敏)[2,3], and Dongxiao Quan(权东晓)[1]

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[2] Shaanxi Key Laboratory of Information Communication Network and Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China
[3] School of Communications and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

(Received 29 October 2020; revised manuscript received 9 January 2021; accepted manuscript online 3 February 2021)

Verification in quantum computations is crucial since quantum systems are extremely vulnerable to the environment. However, verifying directly the output of a quantum computation is difficult since we know that efficiently simulating a large-scale quantum computation on a classical computer is usually thought to be impossible. To overcome this difficulty, we propose a self-testing system for quantum computations, which can be used to verify if a quantum computation is performed correctly by itself. Our basic idea is using some extra ancilla qubits to test the output of the computation. We design two kinds of permutation circuits into the original quantum circuit: one is applied on the ancilla qubits whose output indicates the testing information, the other is applied on all qubits (including ancilla qubits) which is aiming to uniformly permute the positions of all qubits. We show that both permutation circuits are easy to achieve. By this way, we prove that any quantum computation has an efficient self-testing system. In the end, we also discuss the relation between our self-testing system and interactive proof systems, and show that the two systems are equivalent if the verifier is allowed to have some quantum capacity.

Keywords: quantum computation, verification, self-testing systems, complexity theory

PACS: 03.67.−a, 03.67.Ac, 03.67.Lx **DOI:** 10.1088/1674-1056/abe29a

1. Introduction

It is generally believed that quantum computers can accelerate the solving of some difficult problems that are thought to be intractable on classical computers.[1,2] Nowadays, many experimental implementations of quantum computation have been demonstrated.[3–7] In late 2019, Google Inc. shocked the quantum computing community by announcing that their quantum computer achieved the quantum supremacy.[8] This is the watershed where the quantum computer surpasses the classical computer in processing some specially practical issues. In spite of this declaration aroused a continuing controversy, it brings us an inevitable question: If a problem can not be efficiently solved by a classical computer, how can one believe that the solution obtained by a quantum computer is exactly correct? Indeed, this question has been investigated in complexity theory for many years.[9] However, as far as we know, almost all existing works only concern interactive proof (IP) systems[10] (see Ref. [9]). Informally speaking, an interactive proof system traditionally is modeled as a two-party interactive game where two players, an all-powerful prover and a capacity-limited verifier, interact with each other in a back-and-forth way. At the end of the game, the verifier accepts or rejects the output of the prover, i.e., the computation outcome. In this paper, we take a different perspective to analyze this question for the first time by introducing the concept of self-testing systems into the complexity class. Simply speaking, we say a problem has a self-testing system if the computing machine can verify the correctness of the computation by itself according to a part of the computation output. In another word, a self-testing system is a computing machine such that it takes a problem as the input, then after finite steps, outputs the solution of the problem together with a message which can be used to verify if the computation is performed correctly. One may argue that whether it is necessary to define such a model. We give a few reasons to make this point more substantial.

First, the IP system only suits the scenario where a client and a server, probably in distance, interact with each other, e.g., one-way quantum computation or delegated quantum computation,[11–15] in which a client remotely instructs a server to perform his private computation. Clearly, this model does not capture a special situation where the computation and the verification are performed by the same party. For instance, imagine that a material company purchases a quantum computer from a quantum computing company, they want to use it to simulate some complicated quantum systems. How can they convince themselves that the result of simulation is cor-

*Project supported by the National Natural Science Foundation of China (Grant Nos. 61372076, 61971348, and 62001351), Foundation of Shaanxi Key Laboratory of Information Communication Network and Security (Grant No. ICNS201802), Natural Science Basic Research Program of Shaanxi, China (Grant No. 2021JM-142), and Key Research and Development Program of Shaanxi Province, China (Grant No. 2019ZDLGY09-02).

†Corresponding author. E-mail: chhzhu@xidian.edu.cn

© 2021 Chinese Physical Society and IOP Publishing Ltd

rect? Surely, an IP system may still work in this case if we allow workers to interact with the quantum computer back and forth. Nevertheless, they may prefer a way of more direct verification, i.e., they wish they could check the validity of computations according to a part of the output. Second, recall that in delegated quantum computations, most references related to this topic model an untrusted server as an honest-but-curious or malicious server. From the viewpoint of the protocol security, this is a reasonable hypothesis since the data privacy is an important factor in nowadays cloud computing. But, from a business perspective, to maintain a good reputation a sensible server should offer clients a correct computation experience as clients expect. For instance, imagine that two quantum computing companies both promote their quantum cloud services to you, will you choose the one who has a bad reputation due to offering terrible services? So, in this respect, a quantum server should perform a self-testing procedure to make itself sure that the computation task delegated by clients has been accomplished correctly. Even if a malicious server who wants to steal clients' information may plug some backdoor programs in clients' computation, in this case we hope a self-testing system can make the server give up this intention. Finally, self-testing is actually not a new idea in quantum information processing. In 2004, Mayers and Yao introduced the concept of self-testing in quantum apparatus.[16] They showed that a self-testing method can be used to check the unknown quantum apparatus. Then a lot of related works arose (see Ref. [17] for a lately comprehensive review). However, the idea of self-testing in those works is mainly related to device-independence in which the quantum apparatuses are treated as the untrusted devices and the self-testing is mainly used to check if those devices work as we desire. Besides that, we first note that most existing works only consider some special quantum processes, e.g., generating a certain quantum state, performing a certain quantum measurement, and so on. To generalize those to a general case, we should consider this question from an abstract perspective, which means we should turn back to complexity theory and ask if there is a self-testing system such that given any quantum process it can verify the correctness of executions by itself. Although in Ref. [18] the authors had already put forward the concept of self-testing in quantum circuits. However, they did not promote this idea into the level of complexity classes. Note that Mckague *et al.* also introduced the concept of self-testing into interactive proofs for BQP problems[19] (see Subsection 2.2 for further explanation on BQP problems), but in their work they still considered an interactive proof system with non-communicating quantum provers and a classical verifier.

Many different verification schemes for quantum computations have been proposed (a lately comprehensive review can be found in Ref. [9]), almost all of them use the language of interactive proof systems to describe the verification. Nevertheless, the basic idea behind these works is somewhat similar. Essentially, to verify whether a computation task f is honestly performed, the method used in the previous work is that the verifier hides some easy question g in the original task f, then the prover is required to compute this mixed task. When appropriate, the verifier retrieves information on the question g and checks if it is the same with his expectation. If so, the computation task f is thought to be performed correctly with a high probability, otherwise the verifier rejects. For example, in Ref. [20], to make sure that the quantum computation is performed correctly, each input qubit is attached by an extra quantum state then they are encrypted by a random Clifford operator, when a basic gate is required to apply on this qubit, the verifier first decrypts the whole state then checks that if the extra state is intact. If all qubits are authenticated, then the verifier accepts the outcome. Similarly, in Ref. [21], to guarantee that the sever performs the computation honestly, the client randomly sets some known qubits called traps in the universal graph state. Once any trap is triggered, the client checks its outcome. If the outcome does not accord with his expectation, the client rejects. From the previous works, we can see that one requirement must be met in this method: the simple question g must be blind for the prover, so that he can not bypass the question g to cheat the verifier. In this paper, we follow the same basic idea. Our main construction technique is to introduce the permutation in the computation. However, since we are considering a self-testing system, we do not need to hide the questions f and g. But, we do hope that the questions f and g are mixed up mutually in a way so that any deviation on f can be projected onto the output of g with a high probability. To do this, we use some extra ancilla qubits as a flag system, then randomly interpolate them into an extended circuit where the ancilla qubits are used to compute the function g while the rest qubits are used to compute the function f. This extended circuit is generated by a sequence of permutations operators and the additional function g is also generated by as a sequence of permutations operators, meanwhile this function g can be efficiently simulated by a classical computer. As a result, we will see that any problem in BQP class can naturally be equipped with a self-testing system if we allow some extra ancilla qubits to be added into the input.

The rest of this paper is organized as follows. We start by introducing some necessary preliminaries in Section 2. In Section 3, we define the self-testing system and state our main conclusions. Then in Section 4, we describe the construction of the self-testing system. Its completeness and soundness are shown in Section 5. In the final section, we discuss the link between the self-testing system and the interactive proof system.

2. Preliminaries

We assume that readers are familiar with the fundamental of quantum computation theory.[22] The n-qubit Pauli group $\mathbb{P}_n$ and the n-qubit Clifford group $\mathbb{C}_n$ are defined as follows:

$$\mathbb{P}_n \triangleq \{\alpha P_1 \otimes P_2 \otimes \ldots \otimes P_n | \alpha \in \{\pm 1, \pm i\}, P_i \in \{I, X, Y, Z\}\},$$
$$\mathbb{C}_n \triangleq \{C \in U(2^n) | \forall P \in \mathbb{P}_n \to CPC^\dagger \in \mathbb{P}_n\}, \quad (1)$$

where $U(2^n)$ denotes the set of all n-qubit unitary operators. A well-known fact is that the n-qubit Clifford group can be treated as a subset of $\mathbb{U}(2^n)$ generated by the Hadamard gate H, the phase gate S, and the controlled-NOT gate CX. Besides that, according to the Gottesman–Knill theorem[23] any polynomial size quantum circuit which is composed of H, S, and CX can be efficiently simulated by a classical computer.

In this paper, we consider the n-qubit quantum circuit in the form of $U = U_T U_{T-1} \ldots U_2 U_1$ where each U_i is an n-qubit unitary operator consisting of a tensor product of at most n basic unitary gates from some universal gate set, and each basic gate in U_i acts on different qubits. We call such an operator U_i a basic tensor gate. To further explain the meaning of tensor gates, we give a concrete example in Fig. 1.

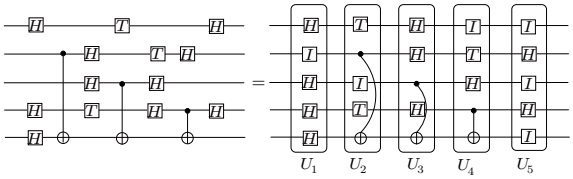


Fig. 1. Given a quantum circuit U (the left) generated from the gate set $\{H, T, \text{CX}\}$, one can always normalize this circuit into a sequence of basic tensor gates. Some identity gates I may be needed in the normalized circuit. We can see that $U = U_5 U_4 U_3 U_2 U_1$ where $U_1 = H \otimes I \otimes H \otimes H \otimes H$, and so on.

A special basic gate needed in our protocol is the SWAP gate, the effect of a SWAP gate is to swap two related qubits, that is, $\text{SWAP}|i, j\rangle = |j, i\rangle$. Let $|\psi\rangle$ be an n-qubit state, to identify each qubit we label each one with a unique number from the set $[n] \triangleq \{1, 2, \ldots, n\}$. Specifically, we use $|\psi\rangle_{1,2,\ldots,n}$ to denote the n-qubit state $|\psi\rangle$ where the first qubit in $|\psi\rangle$ is labeled by 1, the second is labeled by 2, and so on. Correspondingly, each basic gate in a tensor gate U_i will be attached a (or two) unique qubit label(s) to indicate which qubit(s) it will act. For example, X_p denotes a Pauli X gate on qubit p, $\text{SWAP}_{p,q}$ denotes a SWAP gate on qubits p and q, while C_pX_q denotes a CX gate on qubits p and q where p is the control qubit and q is the target qubit. For ease of writing, in rest of this paper we will use $|\psi\rangle$ to denote $|\psi\rangle_{1,2,\ldots,n}$.

2.1. Value and position permutations

Informally, a permutation can be viewed as a one-to-one function that its domain and range are identical. Following this way, we can define a value permutation on $\{0,1\}^n$ as follows.

Definition 1 (value permutations) A value permutation V on n-qubit strings is a unitary operator such that for any n-qubit strings $|x\rangle$ and $|x'\rangle$, where $x, x' \in \{0,1\}^n$ and $x \neq x'$, applying the operator V on $|x\rangle$ and $|x'\rangle$ will obtain two different qubit strings respectively, that is, $\langle x|V^\dagger V|x'\rangle = 0$.

Obviously, the value permutation V is not unique, and given a value permutation V there also exist different methods to construct it. We will discuss the value permutations further in Appendix A. It is easy to check that a CX gate on $|x\rangle, x \in \{0,1\}^2$, can be treated as a 2-qubit value permutation. Indeed, a similar result can be generalized into any k-qubit controlled-NOT gate denoted by C^kX. More generally, we can prove that any n-qubit value permutation V can be constructed using the controlled-NOT gates from the gate set $\{\text{C}^k\text{X}\}_{k=0}^n$, where we set $\text{C}^1\text{X} = \text{CX}$ and $\text{C}^0\text{X} = X$. In order to keep the content consistent, the concrete proof of this statement is shown in Appendix A. On the other hand, a permutation operation can be also used to describe a rearrangement on some ordered set. For instance, we know that given any state $|\psi\rangle$, mathematically we can rearrange the order of qubits and obtain a different but equivalent expression denoted by $|\overline{\psi}\rangle_s$, e.g., $|010\rangle_{1,2,3} \equiv |100\rangle_{2,1,3}$, where $s = (2,1,3)$ indicates the rearranged order of subscripts. Clearly, this operation from $|\psi\rangle$ to $|\overline{\psi}\rangle$ can be seen as a permutation from $(1,2,\ldots,n)$ to s. Since this permutation operation is related to the order of subscripts, we call it a position permutation. Now let $\mathbb{S}_n$ be the set of all possible orders of subscripts $\{1,2,\ldots,n\}$ except $(1,2,\ldots,n)$, we formally define a position permutation as follows.

Definition 2 (position permutations) Given two different n-qubit states $|\psi\rangle$ and $|\overline{\psi}\rangle$, a unitary operator P is said to be a position permutation from $|\psi\rangle$ to $|\overline{\psi}\rangle$ if it satisfies: (1) there exists an $s \in \mathbb{S}_n$ such that $|\psi\rangle \equiv |\overline{\psi}\rangle_s$; (2) $P|\psi\rangle = |\overline{\psi}\rangle$. More specifically, we use P_s to denote this operation.

It is not hard to show that any position permutation P_s can be decomposed into a sequence of SWAP gates. This is because that mathematically given any two different elements $s, s' \in \mathbb{S}_n$, one can always obtain s' from s via a sequence of swap operations. Besides that, suppose that $|\psi\rangle \equiv |\overline{\psi}\rangle_s$ where $s \in \mathbb{S}_n$, and let $|\varphi\rangle = U|\psi\rangle, |\overline{\varphi}\rangle = \overline{U}|\overline{\psi}\rangle$, if $|\varphi\rangle \equiv |\overline{\varphi}\rangle_s$, then given the operator U and the ordered subscripts s, one can easily construct the operator $\overline{U}$ by exchanging the positions of the basic gates in U. For example, if $U = H \otimes H \otimes I \otimes X$ and $s = (1,3,4,2)$, then $\overline{U} = H \otimes I \otimes X \otimes H$. In Section 4, we will use such circuits $\overline{U}$ to construct a self-testing system for the quantum computation circuit U.

2.2. The BQP class

The abbreviation BQP denotes the bounded-error quantum polynomial time, whose classical counterpart is the bounded-error probabilistic polynomial time, abbreviated as BPP. Simply speaking, the BQP class contains all problems

that can be solved by a quantum computer in polynomial time. Similarly, the BPP class is the problem set that can be efficiently solved by a classical computer. Since we are only interested in problems in BQP class, in the last part of this section we formally introduce the definition of the BQP class.[9]

Definition 3 A language or decision problem $L \in \{0,1\}^*$ belongs to BQP if there exists a polynomial $p(\cdot)$ and a uniform quantum circuit family $Q = \{Q_n, n \in \mathbb{N}\}$, where each circuit has at most $p(n)$ basic gates, such that for any $x \in \{0,1\}^n$ the following properties hold true:

- if $x \in L$, then $Q_n(x)$ accepts with probability at least c,
- if $x \notin L$, then $Q_n(x)$ accepts with probability at most s,

where c and s are known as the completeness and soundness parameters, respectively. Traditionally, we can set $c = 2/3$ and $s = 1/3$. However, in full of generality, the rigorous requirement is that $c - s \geq 1/p(n)$ and $p(n)$ is some polynomial function.[9]

Generally speaking, the output state of the n-qubit quantum circuit Q_n is obviously an n-qubit state. However, in quantum complexity theory, we can define the last qubit of the output state as the output of the computation. In this paper, the acceptance is defined as obtaining the measurement outcome 1 when measuring the last qubit in basis $\{|0\rangle, |1\rangle\}$.

3. Main results

In this work, our main conclusion is that for any problem in BQP class there exists an efficient quantum self-testing system, which can test if the computation result is reliable. Follow the same technicality used in Refs. [20,24], in the reminder of this paper we will consider promise problems,[25] instead of decision problems. A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is a pair of disjoint sets of strings where the strings in the sets Π_{YES} and Π_{NO} are called YES and NO instances of the problem, and have answers YES and NO, respectively. Using promise problems can simplify our analysis, we only need to concern one special promise problem named Q-CIRCUIT problem.[20]

Definition 4 The promise problem Q-CIRCUIT = {Q-CIRCUIT$_{\text{YES}}$, Q-CIRCUIT$_{\text{NO}}$} consists of two disjoint quantum circuit sets such that

$$U \in \text{Q-CIRCUIT}_{\text{YES}} : ||(\mathbb{I}_{n-1} \otimes |1\rangle\langle 1|) U |0\rangle^{\otimes n}||^2 \geq 2/3,$$
$$U \in \text{Q-CIRCUIT}_{\text{NO}} : ||(\mathbb{I}_{n-1} \otimes |1\rangle\langle 1|) U |0\rangle^{\otimes n}||^2 \leq 1/3, \quad (2)$$

where $U = U_T, \ldots, U_1$ is a quantum circuit made of a sequence of gates acting on n input qubits.

Roughly speaking, the quantum circuit U can be seen as an input to the problem Q-CIRCUIT. If $U \in \text{Q-CIRCUIT}_{\text{YES}}$, then after applying U on the initial state $|0\rangle^{\otimes n}$, the probability of obtaining the n^{th} output qubit in state $|1\rangle$ is $\geq 2/3$. Otherwise, the probability is $\leq 1/3$. As known, the problem Q-CIRCUIT is BQP-complete,[20,26] thus any problem in BQP class can be reduced into the Q-CIRCUIT problem.

According to the definition of the BQP class, given any problem in BQP class there exists a uniform quantum circuits family $Q = \{Q_n : n \in \mathbb{N}\}$ such that each circuit Q_n can take n qubits as an input then output a result of the problem. Intuitively, in order to verify the output by the circuit Q_n itself, one can introduce some ancilla qubits in the circuit Q_n and use the output state of those ancilla qubits to indicate if the computation has been performed correctly.

Definition 5 A promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ in BQP class is said to have a self-testing system if and only if there exists a polynomial-time generated family of uniform quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$, where each circuit Q_n takes $n+d$ input qubits and produces one output qubit, the first n qubits ρ_{in} is the input of the problem while the additional d qubits ρ_{self} is used for self-testing. The following properties hold true:

- (completeness) if $\rho_{\text{in}} \in \Pi_{\text{YES}}$, then $\Pr[Q \text{ accepts } (\rho_{\text{in}}, \rho_{\text{self}})] \geq c$;
- (soundness) if $\rho_{\text{in}} \in \Pi_{\text{NO}}$, then $\Pr[Q \text{ accepts } (\rho_{\text{in}}, \rho_{\text{self}})] \leq s$,

where $c - s \geq 1/p(n)$. We say the circuit Q_n accepts the input if the measurement outcome of the output qubit is 1 (measuring in basis $\{|0\rangle, |1\rangle\}$), otherwise it rejects. The integer number d is a security parameter about self-testing. In order to guarantee the efficiency of the self-testing system, the number d should not be chosen too big. From the perspective of complexity theory, we choose it by some polynomial-bounded function on n, i.e. $d \leq \text{poly}(n)$.

The class of promise problems having a self-testing system is denoted by BQP$^{\text{ST}}$. In brief, in this work we show the following theorem.

Theorem 1 BQP = BQP$^{\text{ST}}$.

The main proof of the above theorem is shown in Section 5. In the next section, we first describe the construction of our self-testing system.

4. Self-testing systems for BQP problems

As mentioned, the basic idea of self-testing is that given an n-qubit quantum circuit $U = U_T U_{T-1} \ldots U_2 U_1$ we introduce a d-qubit value permutation circuit $V = V_T V_{T-1} \ldots V_2 V_1$. Then the positions of qubits in these two circuits are randomly permuted by an m-qubit position permutation circuit $P = P_{s_T} P_{s_{T-1}} \ldots P_{s_2} P_{s_1}$ where each $s_i \in \mathbb{S}_m$ and $m = n + d$. Finally, we use the output of the circuit V to verify if the circuit U is performed correctly. In the following content, we first introduce two algorithms $\mathcal{V}$ and $\mathcal{P}$ which can be used to generate the random permutation circuits V and P, respectively.

4.1. Generating algorithms

We have known that any d-qubit value permutation circuit can be constructed from the gate set $\{\mathrm{C}^k\mathrm{X}\}_{k=0}^{d}$, it is clear that the d-qubit circuits formed from the subset $\{\mathrm{C}^k\mathrm{X}\}_{k=0}^{d'}$ ($d' < d$) are also value permutation circuits. Since the circuit V is only used for self-testing, thus it is sufficient to generate those circuits from the gate set $\{X, \mathrm{CX}\}$ (see Appendix B for further discussions). Moreover, if V and V' are two value permutations, then the product VV' is also a value permutation. This is because mathematically, for a given parameter d, all d-qubit value permutations $\{V\}$ form a permutation group, which means if V, V' are two elements in this permutation group, then the product VV' is also in the group. So, in order to satisfy the condition that $V = V_T V_{T-1} \ldots V_2 V_1$, the algorithm $\mathcal{V}$ is supposed to generate T simple value permutation $\{V_i\}_{i=1}^{T}$, where each V_i is a basic tensor gate consisting of some X and CX gates randomly. The detailed procedure is shown as follows.

Algorithm $\mathcal{V}$: Generating a random value permutation circuit V

1. Input d and T;
2. For i from 1 to T:
3. Choose an integer $k_i \in_R \{1,2,\ldots,d\}$;
4. Choose a finite string $\boldsymbol{a}_i = (a_{i,1}, a_{i,2}, \ldots)$ from $\{0,1\}^*$ such that $\sum_j (a_{i,j}+1) = k_i$;
5. For each $a_{i,j} \in \{0,1\}$:
6. If $a_{i,j} = 1$, then
7. Choose a unique number $b_{i,j}$ from $\{1,2,\ldots,d\}$;
8. Otherwise
9. Simply make $b_{i,j} = \emptyset$;
10. Choose another unique number $c_{i,j}$ from $\{1,2,\ldots,d\}$;
11. Construct a circuit V_i according to the parameters $\{a_{i,j}, b_{i,j}, c_{i,j}\}_j$ as follows

$$V_i = C_{\boldsymbol{b}_i}^{\boldsymbol{a}_i} X_{\boldsymbol{c}_i} = C_{b_{i,1}}^{a_{i,1}} X_{c_{i,1}} \otimes C_{b_{i,2}}^{a_{i,2}} X_{c_{i,2}} \otimes \cdots \tag{3}$$

where we define that $C_{\emptyset}^{0} X_{c_i} = X_{c_i}$ and $C_{b_i}^{1} X_{c_i} = C_{b_i} X_{c_i}$.

12. Output the circuits $\{V_i\}_{i=1}^{T}$.

Similarly, we have known that any position permutation P can be decomposed into a sequence of SWAP gates. And for any position permutations P and P', the product PP' is also a position permutation. Thus in essence the algorithm $\mathcal{P}$ generates T simple position permutation circuits $\{P_{s_i}\}_{i=1}^{T}$, where each P_{s_i} is a basic tensor gate consisting of some SWAP gates randomly. The detailed procedure is described as follows.

Algorithm $\mathcal{P}$: Generating a random position permutation circuit P

1. Input m and T;
2. For i from 1 to T:
3. Choose an integer $k_i \in_R \{1,2,\ldots,[\frac{m}{2}]\}$; // $[\cdot]$ indicates rounding down
4. For j from 1 to k_i:
5. Choose two unique numbers from $\{1,2,\ldots,m\}$ denoted by $p_{i,j}$ and $q_{i,j}$;
6. Construct a circuit P_{s_i} according to $\{p_{i,j}, q_{i,j}\}_j$ as follows

$$P_{s_i} = \mathrm{SWAP}_{\boldsymbol{p}_i, \boldsymbol{q}_i} = \mathrm{SWAP}_{p_{i,1}, q_{i,1}} \otimes \mathrm{SWAP}_{p_{i,2}, q_{i,2}} \otimes \cdots \tag{4}$$

7. Compute the corresponding subscript order $s_i \in \mathbb{S}_m$ according to $\{p_{i,j}, q_{i,j}\}_j$;
8. Output the circuits $\{P_{s_i}\}_{i=1}^{T}$.

We use two concrete examples to explain the algorithms $\mathcal{V}$ and $\mathcal{P}$, respectively. For $\mathcal{V}$, suppose one of inputs $d = 8$, let $k_i = 7, \boldsymbol{a}_i = (0,1,1,0,0)$, $\boldsymbol{b}_i = (\emptyset, 7, 1, \emptyset, \emptyset)$, and $\boldsymbol{c}_i = (4,5,6,8,2)$. According to the algorithm $\mathcal{V}$, the parameters $\{k_i, \boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i\}$ can be used to generate the following value permutation circuit:

$$V_i = C_{\emptyset}^{0} X_4 \otimes C_7^1 X_5 \otimes C_1^1 X_6 \otimes C_{\emptyset}^{0} X_8 \otimes C_{\emptyset}^{0} X_2. \tag{5}$$

For $\mathcal{P}$, suppose one of inputs $m = 8$, then let $k_i = 3$, $\boldsymbol{p}_i = (3,1,5)$, and $\boldsymbol{p}_i = (2,6,8)$. According to the algorithm $\mathcal{P}$, the parameters $\{k_i, \boldsymbol{p}_i, \boldsymbol{q}_i\}$ can be used to generate the following position permutation circuit:

$$P_{s_i} = \mathrm{SWAP}_{3,2} \otimes \mathrm{SWAP}_{1,6} \otimes \mathrm{SWAP}_{5,8} \tag{6}$$

and the corresponding subscript order s_i can be obtained from the following procedures:

$$\begin{aligned} &(1,2,3,4,5,6,7,8) \xrightarrow{3\leftrightarrow 2} (1,3,2,4,5,6,7,8) \\ &\xrightarrow{1\leftrightarrow 6} (6,3,2,4,5,1,7,8) \xrightarrow{5\leftrightarrow 8} (6,3,2,4,8,1,7,5), \end{aligned} \tag{7}$$

where the symbol $i \leftrightarrow j$ means exchanging the positions of numbers i and j. Obviously, the generated circuits V_i and P_{s_i} both satisfy the form of tensor gates. Besides, for each P_{s_i} output by $\mathcal{P}$ it satisfies that $P_{s_i} = P_{s_i}^{\dagger}$. The following lemma will

be useful when we talk about our protocol in detail.

Lemma 1 Both $\mathcal{V}$ and $\mathcal{P}$ are efficient generating algorithms, thus they can be used to efficiently generate random permutation circuits V and P, respectively. Furthermore, let V be a random output from the algorithm $\mathcal{V}(d,T)$, then given any d-qubit string $|x\rangle$, where $x \in \{0,1\}^d$, the quantum computation $V|x\rangle$ can be efficiently simulated by a classical computer. In another word, let $|y\rangle = V|x\rangle$, then the value y can be efficiently computed by a classical computer.

Proof Apparently, according to the descriptions of algorithms $\mathcal{V}$ and $\mathcal{P}$, we know that both of them can be implemented in polynomial time on (d,T) and (m,T), respectively. Second, we can see that the circuits V and P output by $\mathcal{V}$ and $\mathcal{P}$ are described by the parameters $\{\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i\}_{i=1}^T$ and $\{\boldsymbol{p}_i, \boldsymbol{q}_i\}_{i=1}^T$, respectively, which are clearly bounded in polynomial sizes. Thus, these two algorithms $\mathcal{V}$ and $\mathcal{P}$ are both efficient generating algorithms.

Then, we know that the Gottesman–Knill theorem asserts any polynomial size Clifford circuit (which is composed only of H, S, and CX gates) can be efficiently simulated by a classical computer, which means that if the input state for a Clifford circuit can be also generated by a Clifford circuit acting on some qubits $|0\rangle$, then the output can be computed by a classical computer (we assume that the output is measured in computational basis). Note that the Pauli X gate can be decomposed as $X = HSSH$, thus the circuit V generated by the algorithm $\mathcal{V}$ is clearly a Clifford circuit. Furthermore, the input state $|x\rangle$ where $x \in \{0,1\}^d$ can be easily obtained by applying a sequence of Pauli X gates on a fixed state $|0\rangle^{\otimes d}$, so the classical value $y \in \{0,1\}^d$ that satisfies $|y\rangle = V|x\rangle$ can be efficiently computed by a classical computer.

4.2. Output of self-testing system

Note that the mentioned construction for self-testing systems requires more than one qubit output: one qubit is the output of the quantum circuit U, while the rest d qubits are the output of the circuit V. However, for the input of the Q-CIRCUIT problem, we only need to measure the last qubit as the output. To deal with it, we can introduce another one qubit as the output qubit and use a special $(d+1)$-qubit controlled-NOT gate to project the output of the circuits U and V on this ancilla qubit.

For example, if the output of the circuit V is $|y\rangle = V|x\rangle$, then we define a $(d+1)$-qubit controlled-NOT gate such that if the last qubit of U is $|1\rangle$ meanwhile the output of V is $|y\rangle$, then apply a Pauli X gate on the ancilla qubit, otherwise apply the identity operator I on it. For ease of writing, we use $\mathrm{C}_{\boldsymbol{p}}^{d+1}\mathrm{X}$ to denote this special gate, where the subscript $\boldsymbol{p} = (n, n+1, \ldots, n+d)$ denotes the control qubits. Obviously, by this way we can make the self-testing system output one qubit as the final result.

4.3. Self-testing system

The main algorithm of a self-testing system for a quantum computation can be divided into three stages. The specific procedure is described as follows.

Main protocol: Self-testing system for quantum computation

Stage 1: Initialization

1. Let $U = U_T U_{T-1} \ldots U_2 U_1$ be the n-qubit quantum circuit and $|\psi\rangle = |0\rangle^{\otimes n}$ be the corresponding n-qubit input, where each U_i is a basic tensor gate.
2. Choose an integer number d, which can be treated as a polynomial-bounded function on n.

Stage 2: Construction

1. Given the parameters n, T, and d, run the algorithms $\mathcal{V}(d,T)$ and $\mathcal{P}(m,T)$, where $m = n+d$, obtaining two kinds of permutation circuits as follows
$$\{V_i, P_{s_i}\}_{i=1}^T. \tag{8}$$
2. Then given the circuits $\{U_i, V_i\}_{i=1}^T$ and the subscript orders $\{s_i\}_{i=1}^T$, construct the following circuits
$$\{\overline{U_i \otimes V_i}\}_{i=1}^T \tag{9}$$
where for any $|\psi\rangle \equiv |\overline{\psi}\rangle_{s_i}$ each $\overline{U_i \otimes V_i}$ satisfies that $(U_i \otimes V_i)|\psi\rangle \equiv (\overline{U_i \otimes V_i}|\overline{\psi}\rangle)_{s_i}$.
3. Finally, given the circuits $\{P_{s_i}, \overline{U_i \otimes V_i}\}_{i=1}^T$, construct the following circuit
$$\widetilde{U \otimes V} \triangleq \left(P_{s_T}\overline{U_T \otimes V_T}P_{s_T}\right)\left(P_{s_{T-1}}\overline{U_{T-1} \otimes V_{T-1}}P_{s_{T-1}}\right)\ldots\left(P_{s_2}\overline{U_2 \otimes V_2}P_{s_2}\right)\left(P_{s_1}\overline{U_1 \otimes V_1}P_{s_1}\right). \tag{10}$$

Stage 3: Computation

1. Choose a random input $|x\rangle$ where $x \in \{0,1\}^d$ and let $|y\rangle$ be the output of the circuit V, where $V = V_T V_{T-1} \ldots V_2 V_1$. Compute the value of y on a classical computer.
2. Perform the extended quantum circuit $\widetilde{U \otimes V}$ on qubits $|0\rangle^{\otimes n} \otimes |x\rangle$, then project the last $d+1$ qubits on an extra ancilla qubit $|0\rangle$ by the $(d+1)$-qubit controlled-NOT gate $\mathrm{C}_{\boldsymbol{p}}^{d+1}\mathrm{X}$ (see Fig. 2).
3. Measure the extra ancilla qubit in basis $\{|0\rangle, |1\rangle\}$, obtaining the measurement outcome $c \in \{0,1\}$. If $c = 1$, then the self-testing system accepts. Otherwise, it rejects.

From Fig. 2 we can see that if a promise problem can be solved by a quantum circuit U using at most nT gates, then there exists a self-testing system for the circuit U such that one can also solve the same problem using at most $3(n+d)T$ gates, where the number d can be seen as a security parameter for testing the computation. Furthermore, since the number d is defined as some polynomial-bounded function on n, we can take d as a function such that $d \leq n$, for example $d = [c\log n]$ or $d = [c\sqrt{n}]$, in this case we can see that $3(n+d)T = O(nT)$, thus the self-testing system constructed in this section is an efficient quantum computation for any BQP problems. To complete the proof of the main theorem, it must be shown that both parameters completeness and soundness meet the requirements in Definition 5. In the following sections, we will prove the completeness and soundness, respectively.

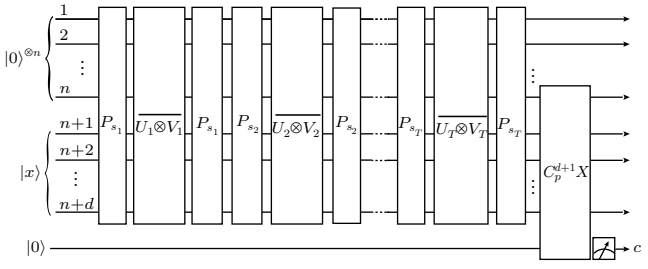


Fig. 2. The basic circuit framework of a self-testing system for a quantum circuit U.

5. Completeness and soundness

5.1. Completeness

We first consider what happens if the protocol is performed correctly. For each component $P_{s_i}\overline{U_i\otimes V_i}P_{s_i}$, and an arbitrary $(n+d)$-qubit state $|\psi\rangle$, we know that

$$P_{s_i}\overline{U_i\otimes V_i}P_{s_i}|\psi\rangle = P_{s_i}\overline{U_i\otimes V_i}|\overline{\psi}\rangle, \tag{11}$$

where $|\psi\rangle \equiv |\overline{\psi}\rangle_{s_i}$. Let $|\varphi\rangle = (U_i\otimes V_i)|\psi\rangle$ and $|\overline{\varphi}\rangle = \overline{U_i\otimes V_i}|\overline{\psi}\rangle$, then according to the step 2 in Stage 2 we can see that the states $|\varphi\rangle$ and $|\overline{\varphi}\rangle$ satisfy the relation $|\varphi\rangle \equiv |\overline{\varphi}\rangle_{s_i}$. Furthermore, since $P_{s_i} = P_{s_i}^{\dagger}$ and $P_{s_i}^{\dagger}|\overline{\varphi}\rangle = |\varphi\rangle$, thus we can infer that

$$P_{s_i}\overline{U_i\otimes V_i}|\overline{\psi}\rangle = P_{s_i}^{\dagger}|\overline{\varphi}\rangle = |\varphi\rangle = (U_i\otimes V_i)|\psi\rangle. \tag{12}$$

Thus, the above two equations tell us that

$$P_{s_i}\overline{U_i\otimes V_i}P_{s_i} = U_i\otimes V_i, \tag{13}$$

from which it follows that the extended circuit $\widetilde{U\otimes V}$ is actually equivalent to the circuit $U\otimes V$

$$\begin{aligned}\widetilde{U\otimes V} &= (P_{s_T}\overline{U_T\otimes V_T}P_{s_T})\ldots(P_{s_2}\overline{U_2\otimes V_2}P_{s_2})(P_{s_1}\overline{U_1\otimes V_1}P_{s_1})\\ &= (U_T\otimes V_T)\ldots(U_2\otimes V_2)(U_1\otimes V_1)\\ &= (U_T\ldots U_2U_1)\otimes(V_T\ldots V_2V_1)\\ &= U\otimes V,\end{aligned} \tag{14}$$

where $U = U_T\ldots U_2U_1$ and $V = V_T\ldots V_2V_1$. So the computation $\widetilde{U\otimes V}|0\rangle^{\otimes n}|x\rangle$ is exactly equivalent to the computation $(U\otimes V)|0\rangle^{\otimes n}|x\rangle$.

In essence, for each tensor gate $\overline{U_i\otimes V_i}$ in the computation circuit $\widetilde{U\otimes V}$, it can be thought as a permuted version of the tensor gate $U_i\otimes V_i$ in the corresponding circuit $U\otimes V$. Although it seems to be only an uninteresting trick, it surprisingly offers a good way to verify the computation that we actually want to perform. Simply speaking, the effect of each position permutation circuit P_{s_i} is to randomly permute the circuit lines in $U\otimes V$, if we suppose that the distribution of any deviation during the computation is uniform, then we can determine if, with a high probability, the desired computation $U|0\rangle^{\otimes n}$ is performed correctly by checking the result of $V|x\rangle$.

Now suppose $U\otimes V$ is a YES-instance of the promise problem Q-CIRCUIT, where the circuit V can be viewed as a redundant input for U, then we can infer that if the protocol is performed correctly, then the self-testing system accepts with probability at least $2/3$.

5.2. Soundness

In this section we study the soundness, that is, we estimate the upper bound of the probability of accepting a wrong decision. We begin this discussion from the input of the circuit $\widetilde{U\otimes V}$ as follows:

$$\rho_{\text{in}}^{x} = (|0\rangle\langle 0|)^{\otimes n}\otimes(|x\rangle\langle x|), \tag{15}$$

where the first n-qubit state $|0\rangle^{\otimes n}$ is the initialized input of the circuit U while the rest d-qubit state $|x\rangle$ is the random input of the circuit V. Furthermore, we define the output of the circuit $\widetilde{U\otimes V}$ by ρ_{out}^{x}. Since we have already known that the quantum circuit $\widetilde{U\otimes V}$ is equivalent to the circuit $U\otimes V$ in the end, so the output state ρ_{out}^{x} is trivially $\rho_c\otimes\rho_s^x$, where $\rho_c = U(|0\rangle\langle 0|)^{\otimes n}U^{\dagger}$ is the output state of U and $\rho_s^x = |y\rangle\langle y| = V(|x\rangle\langle x|)V^{\dagger}$ is the output state of V.

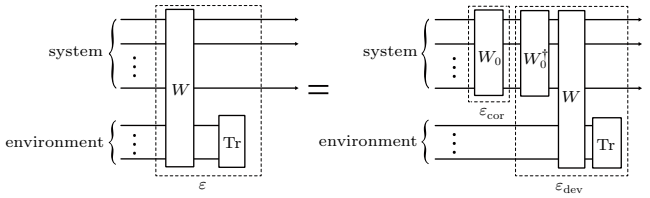


Fig. 3. Any quantum operation $\mathcal{E}$ on a system can be modeled as a unitary operation W on the system and an extra environment,[22] let W_0 be the correct quantum computation circuit then we can obtain $W = WW_0^{\dagger}W_0$.

Then, let $\mathcal{E}$ be the whole quantum operation during the protocol (including any possible deviation), it is easy to see that any such an operation $\mathcal{E}$ can be decomposed as a correct computation $\mathcal{E}_{\text{cor}}$ followed by another deviation $\mathcal{E}_{\text{dev}}$ (see Fig. 3), that is,

$$\sigma_x = \mathcal{E}(\rho_{\text{in}}^{x}) = \mathcal{E}_{\text{dev}}\cdot\mathcal{E}_{\text{cor}}(\rho_{\text{in}}^{x}) = \mathcal{E}_{\text{dev}}(\rho_{\text{out}}^{x}). \tag{16}$$

Finally, we define the projection operator as follows:

$$P^x_{\text{incor}} = (\mathcal{I}_n - \rho_c) \otimes \rho^x_s, \tag{17}$$

where the operator $\mathcal{I}_n$ denotes the identity operator in Pauli group $\mathbb{P}_n$. This projection operator is a tensor product of two state spaces: the former is orthogonal to the space spanned by $U|0\rangle^{\otimes n}$ while the latter is exactly the space spanned by $V|x\rangle$. In this paper, we will show that for any output state σ_x the probability of projecting it onto the space P^x_{incor} is bounded by some small positive number. Specifically, we will show that the following inequality holds true:

$$\frac{1}{2^d}\sum_x \text{Tr}\,(P^x_{\text{incor}}\sigma_x) \leq 2^{-d}. \tag{18}$$

Our basic argument is based mainly on the proof techniques in Refs. [20,24]. The complete proof is shown in Appendix B. Now according to the conclusion in Eq. (18), we immediately obtain that if $U \otimes V$ is a NO-instance of Q-CIRCUIT, then no matter what deviation $\mathcal{E}_{\text{dev}}$ happens during the protocol, the soundness parameter s always satisfies that

$$s = \frac{1}{2^d}\sum_x \text{Tr}(P^x_{\text{incor}}\sigma_x) \leq 2^{-d}, \tag{19}$$

where we can see that $c - s \geq 2/3 - 2^{-d} \geq 1/\text{poly}(n)$, so it completes the proof of the main theorem.

6. Links between self-testing systems and interactive proof systems

We showed that BQP = BQP^{ST} in Section 5. Since we have already known that BQP = QPIP[20,24] (the abbreviation QPIP denotes the quantum-prover interactive proofs), as mentioned most previous works considered verifiable quantum computation using the language of the QPIP class, it follows immediately that BQP^{ST} = QPIP, which means for any BQP problem there exists at least a pair of verification schemes: a self-testing system and an interactive proof system. A natural question is whether we can construct a system from another one. Clearly, an interactive proof system can be treated as a special kind of self-testing systems if we treat the prover and the verifier as one single party. So, in this paper we are only interested in one direction: constructing an interactive proof system from a self-testing system. In the following content, we will show that given a self-testing system one can easily construct an equivalent interactive proof system. The equivalence means that if a self-testing system is c-completeness and s-soundness, then the same completeness and soundness parameters hold in the constructed interactive proof system. Clearly, to construct an interactive proof system from the self-testing system, a simple way is to hide the complete self-testing system from the prover, which can be easily done by blind quantum computation (BQC).

6.1. Constructions via blind quantum computation

We list two implement methods in the following contents, the first one is based on universal blind quantum computation protocol,[12] and the second one is based on secure assisted quantum computation protocol.[27]

Method 1 The first method is considerably direct, it uses the universal blind quantum computation (UBQC) protocol as a subroutine. Specifically, let U be an n-qubit quantum computation circuit such that $U = U_T U_{T-1} \ldots U_2 U_1$, the verifier first calls the generating algorithms $\mathcal{V}$ and $\mathcal{P}$, obtaining the descriptions of two permutation circuits V and P such that $V = V_T V_{T-1} \ldots V_2 V_1$ and $P = P_{s_T} P_{s_{T-1}} \ldots P_{s_2} P_{s_1}$. Then using the tensor gates $\{U_i, V_i, P_{s_i}\}_{i=1}^T$ the verifier constructs the quantum circuit $\widetilde{U \otimes V}$. After that, the verifier and the prover call the UBQC protocol to compute the circuit $\widetilde{U \otimes V}$. According to the blindness of UBQC protocol, a verifier can delegate any quantum computation to an untrusted prover in a way such that the prover can learn nothing except the computation size. Therefore, the verifier can easily hide the self-testing system from the prover.

Clearly, the above construction is efficient, this is because all subroutines called by the verifier and the prover can be done in polynomial time. Moreover, the new interactive proof system has the same completeness and soundness parameters, this is because if both verifier and prover are honest, the UBQC protocol will always output the correct outcome. Besides that, in Ref. [12], the UBQC protocol can be implemented in a way such that any interfering prover either fails to alter the computation or escapes from been detected with an exponentially small probability.

Method 2 Secure assisted quantum computation can make a verifier delegate his computation $U|\psi\rangle$ to an untrusted prover, where the prover can only know the computation circuit U but has no information on the input and output. According to the computation theory, there exists an efficient universal quantum circuit $\mathcal{U}$ such that on inputs $|\psi\rangle$ and $|c(U)\rangle$ it can output $U|\psi\rangle$ and $|c(U)\rangle$, that is,

$$\mathcal{U}(|\psi\rangle \otimes |c(U)\rangle) = (U|\psi\rangle) \otimes |c(U)\rangle, \tag{20}$$

where $c(U)$ can be seen as the canonical (classical) description of the circuit U. One can immediately see that if the verifier and the prover call secure assisted quantum computation protocol to compute $\mathcal{U}|\psi\rangle \otimes |c(U)\rangle$, then it is guaranteed that the prover cannot obtain any information about the input $|\psi\rangle \otimes |c(U)\rangle$ and the output $U|\psi\rangle$.

The specific procedures are described as follows. First, as Method 1, the verifier calls the generating algorithms $\mathcal{V}$ and $\mathcal{P}$, obtaining the descriptions of two permutation circuits V and P. Then using the circuits U, V, and P, the verifier constructs the quantum circuit $\widetilde{U \otimes V}$. After that, the verifier encodes the circuit $\widetilde{U \otimes V}$ into a qubit string, denoted by $|c(\widetilde{U \otimes V})\rangle$. Finally,

the verifier and the prover call a secure assisted quantum computation protocol to compute the circuit $\mathcal{U}|\psi\rangle \otimes |c(\widetilde{U \otimes V})\rangle$. As mentioned before, the verifier can hide the information about the input $|\psi\rangle \otimes |c(\widetilde{U \otimes V})\rangle$ from the prover. In another word, the prover cannot learn any information about the state $|\psi\rangle$ and the computation circuit $\widetilde{U \otimes V}$.

It is worth noting that the above construction requires the verifier to have some quantum capacity. Typically, in the universal quantum computation protocol, the verifier is required to have an ability to prepare some random single qubits,[12] while in the secure assisted quantum computation protocol, the verifier is required to have an ability not only to prepare some random single qubits but also to apply some quantum operations[27] (in Ref. [24], the requirement is reduced to prepare some random single qubits).

7. Conclusions

At final, we briefly discuss our work. First, we showed that any BQP problem admits a self-testing system which can be used to verify if the computation circuit is performed correctly. The main technique using in our protocol is introducing permutations in the computation circuit. By the position-permutation operation, the positions of all qubits in the circuit including ancilla qubits are rearranged randomly. By the state-permutation operation, an additional quantum computation which can be efficiently simulated by a classical computer is introduced, the computation result of this additional circuit is used as a witness that checks if the original computation circuit is performed correctly. Second, we showed that given a self-testing system one can easily transform it into an equivalent interactive proof system. There are two different but equivalent implement methods, and they both require the BPP verifier to have some quantum capacity, that is, the verifier can prepare some single qubits. Finally, it is worthy to mention that although our protocol is designed for detecting any possible deviations during the execution by a self-testing way, it does not include the innocent noise during the computation, for example the quantum gate errors. Nevertheless, it is totally possible that we can use some quantum error correction codes to overcome this shortcoming. Consequently, we will consider a fault-tolerant self-testing system in our further works.

It is natural that most of verifiable quantum computation protocols require the verifier to have some quantum capacity except for considering the post-quantum security[28,29] or the multiple entangled and non-communicating provers.[30,31] Indeed, many researchers believe that there is no interactive proof system such that the prover is restricted to a fully BQP machine while the verifier is restricted to a BPP machine (for example, see Ref. [32]). At first glance, this is an evident conclusion because when a verifier delegates a computation to a prover, if all information sent to the prover is classical, then the BQP prover can generate plenty of duplicates and run those computations locally. As a result, the prover can definitely obtain the computation result, then from the perspective of blind quantum computation, we cannot accomplish the verification.

However, we notice that it is possible to limit the verifier to a complete BPP machine by adding the permutations in computation circuits. The basic idea starts from the two independent quantum circuits: an n-qubit quantum circuit U that we want it to be performed correctly and an d-qubit quantum circuit V that is used to check if U is performed correctly. We require that the circuit V consists only of gates from Clifford group. Next, to make the whole quantum circuit $U \otimes V$ blind to the prover, we permute each line of the circuit randomly. After that the verifier delegates this permuted circuit, denoted by $\overline{U \otimes V}$, to the prover. Suppose that this permutation is known only to the verifier, then to obtain the right tensor form $U \otimes V$ from the permuted circuit $\overline{U \otimes V}$, the best strategy of the prover is randomly guessing, which means the worst case is that the prover has to guess $(n+d)!$ times. Here, our main hypothesis is that given a quantum circuit $\overline{U \otimes V}$ it is a hard problem that one constructs the correct circuit $U \otimes V$ from $\overline{U \otimes V}$ even for a BQP machine. This remains to be proved in a rigorous way. While, this is not a high priority problem for a self-testing system. Our work proposes an efficient self-testing method to quantum computations and takes an important step towards the implementation of verifiable quantum computation.

Appendix A: The value permutation circuits

In this section, we concretely describe how to use a sequence of multi-qubit controlled-NOT gates $\{\mathrm{C^kX}\}_{k=0}^{d}$ to construct any given d-qubit value permutation V. The construction may not be the optimal one, nonetheless it suffices for demonstrating the statement.

First, from the perspective of bit strings we know that a d-qubit value permutation V actually implements a classical permutation $p : \{0,1\}^d \to \{0,1\}^d$. For example, as we mentioned, a CX gate on qubits $|x\rangle, x \in \{0,1\}^2$ can be viewed as a 2-qubit value permutation, specifically it implements a classical permutation $p : \{0,1\}^2 \to \{0,1\}^2$ whose permutation rules are specified as follows:

$$p(00) = 00, \quad p(01) = 01, \quad p(10) = 11, \quad p(11) = 10. \tag{A1}$$

Then, given any two different strings $x, y \in \{0,1\}^d$, we can define a special d-qubit unitary operator $V_{x,y}$ such that

$$\text{for any } \ v \in \{0,1\}^d, \ \ V_{x,y}|v\rangle = \begin{cases} |x\rangle & \text{if } \ v = y, \\ |y\rangle & \text{if } \ v = x, \\ |v\rangle & \text{otherwise.} \end{cases} \tag{A2}$$

In another words, the unitary operator $V_{x,y}$ is a special value permutation which only swaps the values $|x\rangle$ and $|y\rangle$. We now show that this unitary operator $V_{x,y}$ can be accomplished by a sequence of $(d-1)$-qubit controlled-NOT gates. To do this, we make use of Gray codes. Given two distinct strings $x, y \in \{0,1\}^d$, a Gray code connecting x and y is a sequence of binary strings, starting with x and concluding with y, such that adjacent members of the list differ in exactly one bit. For instance, if $x = 0000$ and $y = 1111$, then the corresponding Gray code is $\{0000, 0001, 0011, 0111, 1111\}$. Obviously, according to this Gray code we can construct the circuit $V_{0000,1111}$ using a sequence of 3-qubit controlled-NOT gates (see Fig. A1). Following the same method, we can see that a sequence of $(d-1)$-qubit controlled-NOT gates can accomplish any d-qubit unitary operator $V_{x,y}$.

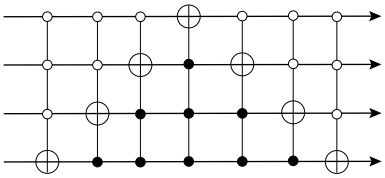


Fig. A1. The quantum circuit $V_{0000,1111}$ consists of a sequence of 3-qubit controlled-NOT gates. Here we use the open circle notation to indicate conditioning on the qubit being set to zero, while a closed circle indicates conditioning on the qubit being set to one, see Ref. [22] for further explanations.

Next, we modify slightly the unitary operator $V_{x,y}$ by defining a $(d+1)$-qubit value permutation $V_{0||x,1||y}$, where the notation $||$ denotes the concatenation of two strings. For example, $00||111$ denotes the string 00111. Finally, we construct the following $(d+1)$-qubit circuit $\widehat{V}$:

$$\widehat{V} = \prod_{x\in\{0,1\}^d} V_{0||x,1||p(x)}. \tag{A3}$$

We can show that each $V_{0||x,1||y}$ commutes with the other component in Eq. (A3), thus the product order in the right side of Eq. (A3) is negligible. Let $V_{x,y}$ and $V_{x',y'}$ be two different d-qubit unitary operators, where $x, x', y, y' \in \{0,1\}^d$ are four distinct strings. For any quantum state $|\psi\rangle = \sum_{v\in\{0,1\}^d} c_v |v\rangle$, one can see that

$$\begin{aligned}
&V_{x,y}V_{x',y'}|\psi\rangle \\
&= V_{x,y}V_{x',y'}\sum_{v\in\{0,1\}^d} c_v|v\rangle \\
&= V_{x,y}V_{x',y'}\left[c_{x'}|x'\rangle + c_{y'}|y'\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x',y'}} c_v|v\rangle\right] \\
&= V_{x,y}\left[c_{x'}|y'\rangle + c_{y'}|x'\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x',y'}} c_v|v\rangle\right] \\
&= V_{x,y}\left[c_x|x\rangle + c_y|y\rangle + c_{x'}|y'\rangle + c_{y'}|x'\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y,x',y'}} c_v|v\rangle\right] \\
&= c_x|y\rangle + c_y|x\rangle + c_{x'}|y'\rangle + c_{y'}|x'\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y,x',y'}} c_v|v\rangle,
\end{aligned} \tag{A4}$$

similarly,

$$\begin{aligned}
&V_{x',y'}V_{x,y}|\psi\rangle \\
&= V_{x',y'}V_{x,y}\sum_{v\in\{0,1\}^d} c_v|v\rangle \\
&= V_{x',y'}V_{x,y}\left[c_x|x\rangle + c_y|y\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y}} c_v|v\rangle\right] \\
&= V_{x',y'}\left[c_x|y\rangle + c_y|x\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y}} c_v|v\rangle\right] \\
&= V_{x',y'}\left[c_{x'}|x'\rangle + c_{y'}|y'\rangle + c_x|y\rangle + c_y|x\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y,x',y'}} c_v|v\rangle\right] \\
&= c_{x'}|y'\rangle + c_{y'}|x'\rangle + c_x|y\rangle + c_y|x\rangle + \sum_{\substack{v\in\{0,1\}^d \\ v\neq x,y,x',y'}} c_v|v\rangle.
\end{aligned} \tag{A5}$$

From the above two equations, one can immediately obtain that for any distinct x, y, x', y', the unitary operators $V_{x,y}$ and $V_{x',y'}$ satisfy the following relation:

$$V_{x,y}V_{x',y'} - V_{x',y'}V_{x,y} = 0. \tag{A6}$$

Now using the above conclusion, one can easily check that for any $x \in \{0,1\}^d$, the following equation holds true:

$$\begin{aligned}
\widehat{V}|0,x\rangle &= \prod_{x'\in\{0,1\}^d} V_{0||x',1||p(x')}|0,x\rangle \\
&= V_{0||x,1||p(x)}\prod_{\substack{x'\in\{0,1\}^d \\ x'\neq x}} V_{0||x',1||p(x')}|0,x\rangle \\
&= V_{0||x,1||p(x)}|0,x\rangle \\
&= |1,p(x)\rangle.
\end{aligned} \tag{A7}$$

From the above equation we can see that if we fix the first qubit of the input of the circuit $\widehat{V}$ as $|0\rangle$, then the first qubit of the output of $\widehat{V}$ is always $|1\rangle$. Since the input and output of the first qubit are fixed, we can treat them as two intrinsic parameters of the circuit $\widehat{V}$, then this circuit accomplishes the permutation $p : \{0,1\}^d \to \{0,1\}^d$. According to the aforementioned conclusion, we know that each unitary operator $V_{0||x,1||y}$ can be implemented by a sequence of d-qubit controlled-NOT gates. As a result, the unitary operator $\widehat{V}$ can be decomposed naturally into a sequence of d-qubit controlled-NOT gates.

As above, we have proved that any d-value permutation V can be accomplished only by a sequence of d-qubit controlled-

NOT gates. We next show that any other k-qubit controlled-NOT gates can be also used to construct some d-qubit value permutation circuits, where $0 \leq k < d$. First, considering a special value permutation V such that given any two distinct strings $x, y \in \{0,1\}^d$ it satisfies that $|y\rangle = V|x\rangle$ and $|x\rangle = V|y\rangle$ then this value permutation can be easily decomposed into the following form:

$$V = \prod_{x\in\{0,1\}^d} V_{x,p(x)}. \tag{A8}$$

Clearly, each $V_{x,p(x)}$ consists of a sequence of $(d-1)$-qubit controlled-NOT gates, thus V can be generated from the $(d-1)$-qubit controlled-NOT gates. From this special example we can infer that some value permutations may have more simplified constructions which are related mainly to the concrete structure of the permutation p. Second, we can further prove that if V_1 is an n-qubit value permutation and V_2 is an m-qubit value permutation, then the composite value permutations $V_1 \otimes V_2$ is an $(n+m)$-qubit value permutation. This is because that for any different $x, y \in \{0,1\}^{n+m}$ we can easily check that

$$\begin{aligned}\langle x|(V_1\otimes V_2)^\dagger(V_1\otimes V_2)|y\rangle &= \langle x_1,x_2|(V_1^\dagger V_1)\otimes(V_2^\dagger V_2)|y_1,y_2\rangle \\ &= \langle x_1|V_1^\dagger V_1|y_1\rangle\langle x_2|V_2^\dagger V_2|y_2\rangle \\ &= 0,\end{aligned} \tag{A9}$$

where we have defined $x = x_1||x_2, y = y_1||y_2$ and $x_1, y_1 \in \{0,1\}^n, x_2, y_2 \in \{0,1\}^m$. Clearly, since we know $x \neq y$, then there must have $x_1 \neq y_1$ or $x_2 \neq y_2$, thus according to Definition 1 the composite operation $V_1 \otimes V_2$ is also a value permutation. As a result, any k-qubit controlled-NOT gates $\{\mathrm{C^kX}\}_{k=0}^d$ can be used to construct the d-qubit value permutation V. As a special case, we can only make use of two simple gates, e.g., Pauli X gate and CX gate, then by tensor multiplication we can always generate some appropriate d-qubit value permutations.

Appendix B: The proof of inequality (18)

First, according to the quantum information theory,[22] we know that any quantum operation admits an operator-sum representation, thus the deviation $\mathcal{E}_{\mathrm{dev}}$ in Eq. (16) can be expressed as follows:

$$\sigma_x = \mathcal{E}_{\mathrm{dev}}(\rho_{\mathrm{out}}^x) = \sum_i E_i \rho_{\mathrm{out}}^x E_i^\dagger, \quad \text{s.t.} \quad \sum_i E_i^\dagger E_i = \mathcal{I}_m, \tag{B1}$$

where the operator $\mathcal{I}_m$ denotes the identity operator in Pauli group $\mathbb{P}_m$. Moreover, since the Pauli group can be treated as a basis to the matrices, so each operator E_i can be uniquely expressed as a combination of n-qubit Pauli operators, that is,

$$E_i = \sum_{P\in\mathbb{P}_m} \alpha_{i,P} P, \tag{B2}$$

where all coefficients $\{\alpha_{i,P}\}$ are possibly complex numbers and it is easy to check that the numbers satisfy the following equation:

$$\sum_i \sum_{P\in\mathbb{P}_m} |\alpha_{i,P}|^2 = \sum_{i,P\in\mathbb{P}_m} |\alpha_{i,P}|^2 = 1. \tag{B3}$$

Substituting Eq. (B2) to Eq. (B1), we obtain immediately that

$$\sigma_x = \sum_i \sum_{P,P'\in\mathbb{P}_m} \alpha_{i,P}\alpha_{i,P'}^* P\rho_{\mathrm{out}}^x P'. \tag{B4}$$

Then, we compute the upper bound of the quantity $\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sigma_x\right)$, note that we can rewrite this expression as follows:

$$\begin{aligned}&\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sum_i \sum_{P=P'} \alpha_{i,P}\alpha_{i,P'}^* P\rho_{\mathrm{out}}^x P'\right) \\ &+\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sum_i \sum_{P\neq P'} \alpha_{i,P}\alpha_{i,P'}^* P\rho_{\mathrm{out}}^x P'\right) \\ =\ &\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sum_i \sum_P |\alpha_{i,P}|^2 P\rho_{\mathrm{out}}^x P\right) \\ &+\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sum_i \sum_{P\neq P'} \alpha_{i,P}\alpha_{i,P'}^* P\rho_{\mathrm{out}}^x P'\right).\end{aligned} \tag{B5}$$

In the following content we show that the second term in the above expression is exactly equal to zero. To do this, let $P = P_1 \otimes P_2$ and $P' = P_1' \otimes P_2'$, where $P_1, P_1' \in \mathbb{P}_n$ and $P_2, P_2' \in \mathbb{P}_d$, then we can see that

$$\begin{aligned}&\frac{1}{2^d}\sum_x \mathrm{Tr}\left(P_{\mathrm{incor}}^x \sum_i \sum_{P\neq P'} \alpha_{i,P}\alpha_{i,P'}^* P\rho_{\mathrm{out}}^x P'\right) \\ =\ &\frac{1}{2^d} \sum_{\substack{i,x \\ P_1,P_1'\in\mathbb{P}_n \\ P_2,P_2'\in\mathbb{P}_d}} \alpha_{i,P_1,P_2}\alpha_{i,P_1',P_2'}^* \mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_c P_1'] \\ &\times \mathrm{Tr}(\rho_s^x P_2 \rho_s^x P_2'),\end{aligned} \tag{B6}$$

where $P_1 \otimes P_2 \neq P_1' \otimes P_2'$, which means there only exist two possibilities: $P_1 \neq P_1'$ or $P_2 \neq P_2'$. Now we consider those conditions separately. For the latter case, note that $\rho_s^x P_2 \rho_s^x P_2' = \rho_s^x P_2 \rho_s^x P_2 P_2 P_2'$ and $\rho_s^x = |x\rangle\langle x|$ where $x \in \{0,1\}^d$, which means if $P_2 \in \{I,Z\}^{\otimes d}$ then $\rho_s^x P_2 \rho_s^x P_2' = \rho_s^x P_2 P_2'$, otherwise $\rho_s^x P_2 \rho_s^x P_2' = 0$, thus we know that

$$\begin{aligned}&\frac{1}{2^d} \sum_{\substack{i,x,P_1,P_1'\in\mathbb{P}_n \\ P_2,P_2'\in\mathbb{P}_d \\ P_2\neq P_2'}} \alpha_{i,P_1,P_2}\alpha_{i,P_1',P_2'}^* \mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_c P_1']\mathrm{Tr}(\rho_s^x P_2\rho_s^x P_2') \\ =\ &\frac{1}{2^d} \sum_{\substack{i,P_1,P_1'\in\mathbb{P}_n \\ P_2\in\{I,Z\}^{\otimes d},P_2'\in\mathbb{P}_d \\ P_2\neq P_2'}} \alpha_{i,P_1,P_2}\alpha_{i,P_1',P_2'}^* \mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_c P_1'] \\ &\times \mathrm{Tr}\left(\sum_x \rho_s^x P_2 P_2'\right).\end{aligned} \tag{B7}$$

It is easy to check that $\sum_x \rho_s^x = \mathcal{I}_d$ and $\mathrm{Tr}(P_2 P_2') = 0$, if

$P_2 \neq P_2'$. Thus, in this case we get that

$$\frac{1}{2^d}\sum_{\substack{i,x,P_1,P_1'\in\mathbb{P}_n\\ P_2,P_2'\in\mathbb{P}_d\\ P_2\neq P_2'}}\alpha_{i,P_1,P_2}\alpha^*_{i,P_1',P_2'}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']\mathrm{Tr}(\rho_s^xP_2\rho_s^xP_2') = 0. \tag{B8}$$

Then, we consider the case where $P_1 \neq P_1'$. According to the above analysis, we can first simplify the expression as follows:

$$\begin{aligned}
&\frac{1}{2^d}\sum_{\substack{i,x,P_1,P_1'\in\mathbb{P}_n\\ P_2\in\mathbb{P}_d,P_1\neq P_1'}}\alpha_{i,P_1,P_2}\alpha^*_{i,P_1',P_2}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']\mathrm{Tr}(\rho_s^xP_2\rho_s^xP_2)\\
&=\frac{1}{2^d}\sum_{\substack{i,x,P_1,P_1'\in\mathbb{P}_n\\ P_2\in\{I,Z\}^{\otimes d},P_1\neq P_1'}}\alpha_{i,P_1,P_2}\alpha^*_{i,P_1',P_2}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']\mathrm{Tr}(\rho_s^x)\\
&=\sum_{\substack{i,P_1,P_1'\in\mathbb{P}_n\\ P_2\in\{I,Z\}^{\otimes d},P_1\neq P_1'}}\alpha_{i,P_1,P_2}\alpha^*_{i,P_1',P_2}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']\\
&=\sum_{\substack{i,P_1,P_1'\in\mathbb{P}_n\\ P_1\neq P_1'}}\beta_{i,P_1,P_1'}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1'],
\end{aligned}\tag{B9}$$

where $\beta_{i,P_1,P_1'}$ is defined as follows:

$$\beta_{i,P_1,P_1'}=\sum_{P_2\in\{I,Z\}^{\otimes d}}\alpha_{i,P_1,P_2}\alpha^*_{i,P_1',P_2}. \tag{B10}$$

We next show that the sum of all $\beta_{i,P_1,P_1'}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']$ can be only a pure imaginary number. Assume the output state is $\rho_c=|c\rangle\langle c|$ where $c\in\{0,1\}^n$ (although this state ρ_c generally is a superposition state, one can always make the algorithm output one of this outcome), in this case we can see that

$$\begin{aligned}
(\mathcal{I}_n-\rho_c)P_1\rho_cP_1' &= (\mathcal{I}_n-\rho_c)P_1\rho_cP_1P_1P_1'\\
&=\begin{cases}0, & \text{if } P_1\in\{I,Z\}^{\otimes n},\\ P_1\rho_cP_1', & \text{otherwise.}\end{cases}
\end{aligned}\tag{B11}$$

According to the above conclusions, we can immediately get

$$\begin{aligned}
&\beta_{i,P_1,P_1'}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1']+\beta_{i,P_1',P_1}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1'\rho_cP_1]\\
&=(\beta_{i,P_1,P_1'}-\beta^*_{i,P_1,P_1'})\mathrm{Tr}(P_1\rho_cP_1'),
\end{aligned}\tag{B12}$$

where we use the simple fact that $\beta_{i,P_1',P_1}=\beta^*_{i,P_1,P_1'}$ and $\mathrm{Tr}(P_1\rho_cP_1')=-\mathrm{Tr}(P_1'\rho_cP_1)$ if $P_1\neq P_1'$. One can easily verify that $\mathrm{Tr}(P_1\rho_cP_1')\in\{-1,0,+1\}$ if $\rho_c=|c\rangle\langle c|, c\in\{0,1\}^n$. Since the number $\beta_{i,P_1,P_1'}-\beta^*_{i,P_1,P_1'}$ is exactly pure imaginary, so the result $(\beta_{i,P_1,P_1'}-\beta^*_{i,P_1,P_1'})\mathrm{Tr}(P_1\rho_cP_1')$ is also a pure imaginary number. However, we know that the value of $\frac{1}{2^d}\sum_x\mathrm{Tr}(P^x_{\mathrm{incor}}\sigma_x)$ should be a real number in $[0,1]$ and the first term in Eq. (B5) is a non-negative number, which means the above sum vanishes in the end. As a result, we know that

$$\frac{1}{2^d}\sum_x\mathrm{Tr}\left(P^x_{\mathrm{incor}}\sum_i\sum_{P\neq P'}\alpha_{i,P}\alpha^*_{i,P'}P\rho^x_{\mathrm{out}}P'\right)=0. \tag{B13}$$

Based on the aforementioned analysis, we are left with the following equation:

$$\frac{1}{2^d}\sum_x\mathrm{Tr}(P^x_{\mathrm{incor}}\sigma_x)=\frac{1}{2^d}\sum_x\mathrm{Tr}\left(P^x_{\mathrm{incor}}\sum_i\sum_{P=P'}\alpha_{i,P}\alpha^*_{i,P'}P\rho^x_{\mathrm{out}}P'\right), \tag{B14}$$

which means for all $x\in\{0,1\}^d$ the state σ_x can be replaced by the new one σ_x' defined as follows:

$$\sigma_x'=\sum_{i,P\in\mathbb{P}_m}\alpha_{i,P}\alpha^*_{i,P}P\rho^x_{\mathrm{out}}P. \tag{B15}$$

On the other hand, note that for any Clifford gate $C\in\mathbb{C}_m$, given a fixed Pauli operator $P\in\mathbb{P}_m$, there exists another Pauli operator $Q\in\mathbb{P}_m$ such that $CQC^\dagger=P$. Furthermore, given two different Pauli operators $P,P'\in\mathbb{P}_m$, the corresponding Pauli operators $Q,Q'\in\mathbb{P}_m$ are also different. Thus, for any $C\in\mathbb{C}_m$, the following equation holds true:

$$\sum_{P\in\mathbb{P}_m}P=\sum_{Q\in\mathbb{P}_m}CQC^\dagger, \tag{B16}$$

then substituting the above equation into Eq. (B15) and considering all operators in Clifford group $\mathbb{C}_m$, we finally obtain that

$$\sigma_x'=\frac{1}{|\mathbb{C}_m|}\sum_{C\in\mathbb{C}_m}\sum_{i,Q\in\mathbb{P}_m}\alpha_{i,Q}\alpha^*_{i,Q}CQC^\dagger\rho^x_{\mathrm{out}}CQC^\dagger. \tag{B17}$$

According to the conclusion in Ref. [9], the above state σ_x' can be simplified into the following form:

$$\sigma_x'=\beta\rho^x_{\mathrm{out}}+\frac{1-\beta}{4^m-1}\sum_{P\in\mathbb{P}_m\setminus\{I_m\}}P\rho^x_{\mathrm{out}}P, \tag{B18}$$

where $\beta=\sum_i|\alpha_{i,I_m}|^2$. Clearly, it satisfies $0\leq\beta\leq1$. Now we turn back to Eq. (B14) and get that

$$\frac{1}{2^d}\sum_x\mathrm{Tr}(P^x_{\mathrm{incor}}\sigma_x)=\frac{1}{2^d}\frac{1-\beta}{4^m-1}\sum_{x,P\in\mathbb{P}_m\setminus\{I_m\}}\mathrm{Tr}(P^x_{\mathrm{incor}}P\rho^x_{\mathrm{out}}P). \tag{B19}$$

Again, let $P=P_1\otimes P_2$, where $P_1\in\mathbb{P}_n$ and $P_2\in\mathbb{P}_d$. Substitute $P^x_{\mathrm{incor}}=(\mathcal{I}_n-\rho_c)\otimes\rho^x_s$ and $\rho^x_{\mathrm{out}}=\rho_c\otimes\rho^x_s$ into the above equation, then it is easy to check that

$$\begin{aligned}
&\frac{1}{2^d}\sum_x\mathrm{Tr}(P^x_{\mathrm{incor}}\sigma_x)\\
&=\frac{1-\beta}{2^d(4^m-1)}\sum_{\substack{x,P_1\in\mathbb{P}_n,P_2\in\mathbb{P}_d\\ P_1\otimes P_2\neq I_m}}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1]\mathrm{Tr}(\rho_s^xP_2\rho_s^xP_2)\\
&=\frac{2^d(1-\beta)}{4^m-1}\sum_{P_1\in\mathbb{P}_n\setminus\{I_n\}}\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1].
\end{aligned}\tag{B20}$$

Now considering ρ_c is an arbitrary superposition state, using the simple fact that $0\leq\mathrm{Tr}[(\mathcal{I}_n-\rho_c)P_1\rho_cP_1]\leq1$, we finally obtain the following result:

$$\frac{1}{2^d}\sum_x\mathrm{Tr}(P^x_{\mathrm{incor}}\sigma_x)\leq\frac{2^d(1-\beta)}{4^m-1}(4^n-1)\leq2^d\frac{4^n-1}{4^m-1}\leq2^d\frac{4^n}{4^m}=2^{-d}. \tag{B21}$$

References

[1] Deutsch D 1985 *Proc. R. Soc. Lond. A* **400** 97
[2] Feynman R P 1982 *Int. J. Theor. Phys.* **21** 476
[3] Yao X W, Xue F, Pang W M, *et al.* 2006 *Chin. Phys. Lett.* **23** 1996
[4] Feng Z B and Yan R Y 2010 *Chin. Phys. Lett.* **27** 010301
[5] Qü F M, Ji Z Q, Tian Y, *et al.* 2018 *Chin. Phys. B* **27** 070301
[6] Wu C, Fang M F, Xiao X, *et al.* 2011 *Chin. Phys. B* **20** 020305
[7] Xin T, Wang B X, Li K R, *et al.* 2018 *Chin. Phys. B* **27** 020308
[8] Arute F, Arya K, Babbush R, *et al.* 2019 *Nature* **574** 505
[9] Gheorghiu A, Kapourniotis T and Kashefi E 2019 *Theory of Computing Systems* **63** 715
[10] Goldwasser S, and Micali S and Rackoff C 1989 *SIAM J. Comput.* **18** 186
[11] Raussendorf R and Briegel H J 2001 *Phys. Rev. Lett.* **86** 5188
[12] Broadbent A, Fitzsimons J and Kashefi E 2009 *50th Annual IEEE Symposium on Foundations of Computer Science*, Atlanta, GA, pp. 517–526
[13] Kashefi E and Pappa A 2017 *Cryptography* **1** 12
[14] Cao X and Shang Y 2014 *Chin. Phys. Lett.* **31** 110302
[15] Diao D S 2013 *Chin. Phys. Lett.* **30** 010303
[16] Mayers D and Yao A 2004 arXiv:quant-ph/0307205
[17] Supic I and Bowles J 2019 arXiv:1904.10042 [quant-ph]
[18] Magniez F, Mayers D, Mosca M and Ollivier H 2005 arXiv:quant-ph/0512111
[19] Mckague M 2016 *Theory of Computing* **12** 3
[20] Aharonov D, Ben-Or M and Eban E 2008 arXiv:0810.5375 [quant-ph]
[21] Fitzsimons J F and Kashefi E 2017 *Phys. Rev. A* **96** 012303
[22] Nielsen M and Chuang I 2011 *Quantum Computation and Quantum Information: 10th Anniversary Edition*
[23] Gottesman D 1998 arXiv:quant-ph/9807006
[24] Broadbent A 2018 *Theory of Computing* **12** 11
[25] Goldreich O 2005 *Electronic Colloquium on Computational Complexity*
[26] Rosgen B and Watrous J 2005 *20th Annual IEEE Conference on Computational Complexity*, San Jose, CA, USA, pp. 344–354
[27] Childs A M 2005 *Quantum Info. Comput.* **5** 6
[28] Mahadev U 2018 *59th Annual Symposium on Foundations of Computer Science*, Paris, pp. 259–267
[29] Cojocaru A, Colisson L, Kashefi E and Wallden, P 2019 *Advances in Cryptology - ASIACRYPT 2019*, pp. 615–645
[30] Reichardt B W, Unger F and Vazirani U 2013 *Nature* **496** 456
[31] Gheorghiu A, Kashefi E and Wallden P 2015 *New J. Phys.* **17** 083040
[32] Morimae T and Koshiba T 2019 arXiv:1407.1636 [quant-ph]