Exact Quantum Algorithms for the Leader Election Problem

Seiichiro Tani[1][2], Hirotada Kobayashi[2], and Keiji Matsumoto[3][2]

[1] NTT Communication Science Laboratories, NTT Corporation,
3-1 Morinosato-Wakamiya, Atsugi, Kanagawa 243-0198, Japan
`tani@theory.brl.ntt.co.jp`
[2] Quantum Computation and Information Project, ERATO,
Japan Science and Technology Agency,
5-28-3 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
`hirotada@qci.jst.go.jp`
[3] Foundations of Information Research Division, National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
`keiji@nii.ac.jp`

Abstract. It is well-known that no classical algorithm can solve exactly (i.e., in bounded time without error) the leader election problem in anonymous networks. This paper proposes two quantum algorithms that, when the parties are connected by quantum communication links, can exactly solve the problem for any network topology in polynomial rounds and polynomial communication/time complexity with respect to the number of parties. Our algorithms work well even in the case where only the upper bound of the number of parties is given.

1 Introduction

Quantum computation and communication are turning out to be much more powerful than the classical equivalents in various computational tasks. Perhaps the most exciting developments in quantum computation would be polynomial-time quantum algorithms for factoring integers and computing discrete logarithms [14], and the most remarkable ones in quantum communication would be quantum key distribution protocols [5, 4] that have been proved to be unconditionally secure [12, 15, 16]. Many other algorithms and protocols have been proposed to show the strength of quantum computation and communication, such as cryptographic results (e.g., [9, 8, 1]) and communication complexity results (e.g., [13, 7, 3]). This paper sheds light on another significant superiority of quantum computing over the classical equivalent in the setting of traditional distributed computing.

The leader election problem is a core problem in traditional distributed computing in the sense that, once it is solved, it becomes possible to efficiently solve many substantial problems in distributed computing (see, e.g., [11]). The goal of the leader election problem is to elect a unique leader from among distributed parties. Obviously, it is possible to deterministically elect the unique leader when each party has a unique identifier, and many classical deterministic algorithms with this assumption have been proposed. As the number of parties grows, however, it becomes difficult to preserve the uniqueness of the identifiers. Thus, other studies have examined the cases wherein each party is anonymous, i.e., each party has the same identifier [2, 10, 17, 18], as an extreme case. In this setting, no classical exact algorithm (i.e., an algorithm that runs in bounded time and solves the problem with zero error) exists for a broad class of network topologies including regular graphs, even if the network topology (and thus the number of parties) is known to each party prior to algorithm invocation [17]. Moreover, to the best of our knowledge, no zero-error probabilistic algorithm is known that works for *any* topology and runs in time/communication expected polynomial in the number of parties. Here, and throughout this paper, we denote by time complexity the maximum number of steps, including steps for the local computation, necessary for each party to execute the protocol, where the maximum is taken over all parties. In synchronous networks, the number of simultaneous message passing is also an important measure. Each turn of simultaneous message passing is referred to as a *round*.

This paper considers the model in which the network is anonymous and consists of quantum links, and proposes two exact quantum algorithms both of which elect a unique leader from among n parties in polynomial time for *any* topology of synchronous networks. Our first algorithm is simple and runs in $O(n^3)$ time. The total communication complexity of this algorithm is $O(n^4)$, but this includes the quantum communication of $O(n^4)$ qubits. To reduce the quantum communication complexity, our second algorithm incurs $O(n^6(\log n)^2)$ time complexity, but demands the quantum communication of only $O(n^2 \log n)$ qubits (plus classical communication of $O(n^6(\log n)^2)$ bits). While our first algorithm needs $\Theta(n^2)$ rounds of quantum communication, our second algorithm needs only one round of quantum communication at the beginning of the protocol to share sufficient amount of entanglement, and after the first round, the protocol performs only local quantum operations and classical communications (LOCCs) of $O(n \log n)$ rounds. Both algorithms are easily modified to support their use in asynchronous networks. Furthermore, both algorithms can be easily modified so that they work well even when each party initially knows only the upper bound of the number of parties. This implies that the exact number of parties can be computed when its upper bound is given. No classical zero-error algorithm exists in such cases for any topology that has a cycle as its subgraph [10].

2 Preliminaries

A *distributed system* (or *network*) is composed of multiple parties and bidirectional classical communication links connecting parties. In a quantum distributed system, every party can perform quantum computation and communication and each adjacent pair of parties has a bidirectional quantum communication link between them. When the parties and links are viewed as nodes and edges, respectively, the topology of the distributed system is expressed by an undirected connected graph, say, $G = (V, E)$. In what follows, we may identify each party/link with its corresponding node/edge in the underlying graph for the system, if it is not confusing. Every party has *ports* corresponding one-to-one to communication links incident to the party. Every port of party l has a unique label i, $(1 \leq i \leq d_l)$, where d_l is the number of parties adjacent to l. More formally, G has a *port numbering*, which is a set σ of functions $\{\sigma[v] \mid v \in V\}$ such that, for each node v of degree d_v, $\sigma[v]$ is a bijection from the set of edges incident to v to $\{1, 2, \ldots, d_v\}$. It is stressed that each function $\sigma[v]$ may be defined independently of the others. Just for ease of explanation, we assume that port i corresponds to the link connected to the ith adjacent party of l. In our model, each party knows the number of its ports and the party can appropriately choose one of its ports whenever it transmits or receives a message.

Initially, every party has local information, such as its internal state, and global information, such as the number of nodes in the system (or its upper bound). Every party runs the same algorithm, which has local and global information as its arguments. If all parties have the same local and global information except for the number of ports the parties have, the system is said to be *anonymous*. This is essentially equivalent to the situation in which every party has the same identifier since we can regard the local/global information of the party as his identifier. If message passing is performed synchronously, such a distributed system is called *synchronous*. The unit interval of synchronization is called a *round* (see [11] for more detailed descriptions).

Next we define the *leader election (LE) problem*. Suppose that there is a distributed system and each party in the system has a variable initialized to 0. The task is to set the variable of exactly one of the parties to 1 and the variables of all the other parties to 0. In the case of anonymous networks, Yamashita and Kameda [17] proved that, if the "symmetricity" (defined in [17]) of the network topology is more than one, LE cannot be solved exactly (more rigorously speaking, there are some port numberings for which LE cannot be solved exactly) by any classical algorithm even if all parties know the topology of the network (and thus the number of nodes). In fact, for a broad class of graphs such as regular graphs, the "symmetricity" is more than one. When the parties initially know only the upper bound of the number of the parties, the result by Itai and Rodeh [10] implies that LE cannot be solved with zero error by any classical algorithm (including the one that may not always halt).

3 Quantum Leader Election Algorithm I

For simplicity, we assume that the network is synchronous and each party knows the number n of parties prior to the algorithm. It is easy to generalize our algorithm to the asynchronous case and to the case where only the upper bound N of the number of parties is given, as will be discussed at the end of this section.

Initially all parties are eligible to become the unique leader. The key to solving the leader election problem in an anonymous network is to break symmetry, i.e., to have just a single party possess a certain state coressponding to the leader.

First we introduce the concept of *consistent* and *inconsistent* strings. Suppose that each party l has a c-bit string x_l. That is, the n parties share cn-bit string $x = x_1 x_2 \cdots x_n$. For convenience, we may consider that each x_l expresses an integer, and identify string x_l with the integer it expresses. Given a set $E \subseteq \{1, \ldots, n\}$, string x is said to be *consistent* over E if x_l has the same value for all l in E. Otherwise x is said to be *inconsistent* over E. We also say that a cn-qubit pure state $|\psi\rangle = \sum_x \alpha_x |x\rangle$ shared by the n parties is *consistent (inconsistent)* over E if $\alpha_x \neq 0$ only for x that is consistent (inconsistent) over E. Further, for a positive integer m, we denote the state that is of the form of $(|0^m\rangle + |1^m\rangle)/\sqrt{2}$, by the m-cat state.

3.1 The Algorithm

The algorithm repeats one procedure exactly $(n-1)$ times, each of which is called a *phase*. In each phase, the number of eligible parties either decreases or remains the same, but never increases or becomes zero. After $(n-1)$ phases the number of eligible parties becomes one with certainty.

Each phase has a parameter denoted by k, whose value is $(n - i + 1)$ in the ith phase. In each phase i, let $E_i \subseteq \{1, \ldots, n\}$ be the set of all ls such that party l is still eligible. First, each eligible party prepares the state $(|0\rangle + |1\rangle)/\sqrt{2}$, while each ineligible party prepares the state $|0\rangle$. Next every party calls Subroutine A, followed by partial measurement. This transforms the system state into either the cat state $(|0^{|E_i|}\rangle + |1^{|E_i|}\rangle)/\sqrt{2}$ shared only by eligible parties, or a state that is inconsistent over E_i. In the former case, each eligible party calls Subroutine B. If k equals $|E_i|$, Subroutine B always succeeds in transforming the $|E_i|$-cat state into a state that is inconsistent over E_i. Now, each eligible party l measures his qubits in the computational basis to obtain (a binary expression of) some integer z_l. Parties then compute the maximum value of z_l over all eligible parties l, by calling Subroutine C. Finally, parties with the maximum value remain eligible, while the other parties become ineligible. More precisely, each party l performs Algorithm I described below with parameters "eligible," n, and d_l. The party who obtains the output "eligible" is the unique leader.

Subroutine A: Subroutine A is essentially for the purpose of checking the consistency of each string that is superposed to a quantum state shared by parties. We use a commute operator "$\circ$" over a set $\mathcal{S} = \{0, 1, *, \times\}$ whose operations are summarized in Table 1. Intuitively, "0" and "1" represent the possible values all eligible parties will have when the string finally turns out to be consistent; "$*$" represents "don't care," which means that the corresponding party has no information about the values any of eligible parties have; and "$\times$" represents "inconsistent," which means that the corresponding party already knows that the string is inconsistent. Subroutine A is precisely described below.

As one can see from the description of Algorithm I, the content of $\mathbf{S}$ is initially "consistent" whenever Subroutine A is called. Therefore, after every party finishes Subroutine A, the state shared by parties in their $\mathbf{R}_0$s is decomposed into a consistent state for which each party has the content "consistent" in his

Algorithm I

```
Input: a classical variable status, integers $n, d$
Output: a classical variable status
1. Prepare one-qubit quantum registers $\mathbf{R}_0$, $\mathbf{R}_1$, and $\mathbf{S}$.
2. For $k := n$ down to 2, do the following:
   2.1 If status = “eligible,” prepare the states $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|$“consistent”$\rangle$ in
       $\mathbf{R}_0$ and $\mathbf{S}$, otherwise prepare the states $|0\rangle$ and $|$“consistent”$\rangle$ in $\mathbf{R}_0$ and $\mathbf{S}$.
   2.2 Perform Subroutine A with $\mathbf{R}_0$, $\mathbf{S}$, status, $n$, and $d$.
   2.3 Measure the qubit in $\mathbf{S}$ in the $\{|$“consistent”$\rangle, |$“inconsistent”$\rangle\}$ basis.
       If it results in $|$“consistent”$\rangle$ and status = “eligible,” prepare the state $|0\rangle$ in
       $\mathbf{R}_1$ and perform Subroutine B with $\mathbf{R}_0$, $\mathbf{R}_1$, and $k$.
   2.4 If status = “eligible,” measure the qubits in $\mathbf{R}_0$ and $\mathbf{R}_1$ in the $\{|0\rangle, |1\rangle\}$ basis
       to obtain a nonnegative integer $z$ expressed by the two bits; otherwise let
       $z := -1$.
   2.5 Perform Subroutine C with $z$, $n$, and $d$ to know the maximum value $z_{\max}$ of $z$
       over all parties.
       If $z \neq z_{\max}$, let status := “ineligible.”
3. Output status.
```

Table 1. The definitions of commute operator “$\circ$”

x	y	$x \circ y$	x	y	$x \circ y$	x	y	$x \circ y$	x	y	$x \circ y$
0	0	0	1	0	$\times$	$*$	0	0	$\times$	0	$\times$
0	1	$\times$	1	1	1	$*$	1	1	$\times$	1	$\times$
0	$*$	0	1	$*$	1	$*$	$*$	$*$	$\times$	$*$	$\times$
0	$\times$	$\times$	1	$\times$	$\times$	$*$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{S}$, and an inconsistent state for which each party has the content “inconsistent” in his $\mathbf{S}$. Steps 4 and 5 are performed so that the output quantum registers $\mathbf{R}_0$ and $\mathbf{S}$ are disentangled from work quantum registers $\mathbf{X}_i^{(t)}$s.

Subroutine B: Suppose k parties are still eligible and share the k-cat state $(|0^k\rangle + |1^k\rangle)/\sqrt{2}$. Subroutine B has purpose of changing the k-cat state to a superposition of inconsistent strings, if k is given. Subroutine B is precisely described as follows, where $\{U_k\}$ and $\{V_k\}$ are two families of unitary operators,

$$U_k = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-i\frac{\pi}{k}} \\ -e^{i\frac{\pi}{k}} & 1 \end{pmatrix}, V_k = \frac{1}{\sqrt{R_k+1}} \begin{pmatrix} 1/\sqrt{2} & 0 & \sqrt{R_k} & e^{i\frac{\pi}{k}}/\sqrt{2} \\ 1/\sqrt{2} & 0 & -\sqrt{R_k}e^{-i\frac{\pi}{k}} & e^{-i\frac{\pi}{k}}/\sqrt{2} \\ \sqrt{R_k} & 0 & \frac{e^{-i\frac{\pi}{2k}} I_k}{i\sqrt{2}R_{2k}} & -\sqrt{R_k} \\ 0 & \sqrt{R_k+1} & 0 & 0 \end{pmatrix},$$

where R_k and I_k are the real and imaginary parts of $e^{i\frac{\pi}{k}}$, respectively.

The point is that the amplitudes of the states $|00\rangle^{\otimes k}$, $|01\rangle^{\otimes k}$, $|10\rangle^{\otimes k}$, and $|11\rangle^{\otimes k}$ shared by k eligible parties in their registers $\mathbf{R}_0$ and $\mathbf{R}_1$ are simultaneously zero after each eligible party applies Subroutine B with parameter k, if the particles in $\mathbf{R}_0$s of all eligible parties form the k-cat state.

```
Subroutine A
Input: one-qubit quantum registers $\mathbf{R}_0, \mathbf{S}$, a classical variable status, integers $n, d$
Output: one-qubit quantum registers $\mathbf{R}_0, \mathbf{S}$
1. Prepare two-qubit quantum registers $\mathbf{X}_0^{(1)}, \dots, \mathbf{X}_d^{(1)}, \dots, \mathbf{X}_0^{(n-1)}, \dots, \mathbf{X}_d^{(n-1)}, \mathbf{X}_0^{(n)}$.
   If status = "eligible," copy the content of $\mathbf{R}_0$ to $\mathbf{X}_0^{(1)}$, otherwise set the content of
   $\mathbf{X}_0^{(1)}$ to "$*$."
2. For $t := 1$ to $n-1$, do the following:
   2.1 Copy the content of $\mathbf{X}_0^{(t)}$ to each of $\mathbf{X}_1^{(t)}, \dots, \mathbf{X}_d^{(t)}$.
   2.2 Exchange the qubit in $\mathbf{X}_i^{(t)}$ with the party connected via port $i$ for $1 \le i \le d$
       (i.e., the original qubit in $\mathbf{X}_i^{(t)}$ is sent via port $i$, and the qubit received via
       that port is newly set in $\mathbf{X}_i^{(t)}$).
   2.3 Set the content of $\mathbf{X}_0^{(t+1)}$ to $x_0^{(t)} \circ x_1^{(t)} \circ \cdots \circ x_d^{(t)}$, where $x_i^{(t)}$ denotes the con-
       tent of $\mathbf{X}_i^{(t)}$ for $0 \le i \le d$.
3. If the content of $\mathbf{X}_0^{(n)}$ is "$\times$," turn the content of $\mathbf{S}$ over (i.e., if initially the content
   of $\mathbf{S}$ is "consistent," it is flipped to "inconsistent," and vice versa).
4. Invert every computation and communication in Step 2.
5. Invert every computation in Step 1.
6. Output quantum registers $\mathbf{R}_0$ and $\mathbf{S}$.
```

```
Subroutine B
Input: one-qubit quantum registers $\mathbf{R}_0, \mathbf{R}_1$, an integer $k$
Output: one-qubit quantum registers $\mathbf{R}_0, \mathbf{R}_1$
1. If $k$ is even, apply $U_k$ to the qubit in $\mathbf{R}_0$; otherwise perform CNOT controlled by
   the qubit in $\mathbf{R}_0$ to that in $\mathbf{R}_1$, and then apply $V_k$ to the qubits in $\mathbf{R}_0$ and $\mathbf{R}_1$.
2. Output quantum registers $\mathbf{R}_0$ and $\mathbf{R}_1$.
```

Subroutine C: Subroutine C is a classical algorithm that computes the maximum value over parties. The procedure is very similar to Subroutine A. In fact, Subroutines A and C can be merged into one subroutine, although they are separately explained for simplicity. Subroutine C is precisely described as follows.

3.2 Complexity Analysis

First we state the complexity of Algorithm I without proof.

Theorem 1. *Let $|E|$ and D be the number of edges and the maximum degree of the underlying graph, respectively. Given the number n of parties, Algorithm I exactly elects a unique leader in $\Theta(n^2)$ rounds and $O(Dn^2)$ time. The total communication complexity over all parties is $\Theta(|E|n^2)$.*

If each party initially knows only the upper bound N of the number of parties, each party has only to perform Algorithm I with N instead of n. The complexity in this case is described simply by replacing every n by N in Theorem 1.

Furthermore, Algorithm I is easily modified so that it works well even in the asynchronous settings. Note that all parties receive messages via each port at

```
Subroutine C
Input: integers $z$, $n$, $d$
Output: an integer $z_{\max}$
 1. Let $z_{\max} := z$.
 2. For $t := 1$ to $n-1$, do the following:
    2.1 Let $y_0 := z_{\max}$.
    2.2 Send $y_0$ via port $i$ for $1 \leq i \leq d$.
        Set $y_i$ to the value received via port $i$ for $1 \leq i \leq d$.
    2.3 Let $z_{\max} := \max_{0 \leq i \leq d} y_i$.
 3. Output $z_{\max}$.
```

each round. Now, let each party wait to perform the operations of the $(i+1)$st round until he finishes receiving all messages that are supposed to be received at the ith round. This modification enables us to simulate synchronous behavior in asynchronous networks. In order to know at which round the received message was originally sent, we tag every message. This modification increases the communication and time complexity by the multiplicative factor $\log n$.

4 Quantum Leader Election Algorithm II

To reduce the amount of quantum communication, our second algorithm make use of a classical technique, called *view*, which was introduced by Yamashita and Kameda [17]. However, a naïve application of view exponentially increases the classical time/communication complexity. To reduce this complexity, this paper introduces the new technique of *folded view*, with which the algorithm still runs in time/communication polynomial with respect to the number of parties.

4.1 View and Folded View

First, we briefly review the classical technique, *view*. Let $G=(V,E)$ be the underlying network topology and let $n=|V|$. Suppose each party corresponding to node $v \in V$ has a value $x_v \in S$ for some set S, and consider a mapping $X\colon V \rightarrow S$ defined by $X(v)=x_v$. For each v and port numbering σ, the *view* $T_{G,\sigma,X}(v)$ is a labeled, rooted tree with infinite depth defined recursively as follows: (1) $T_{G,\sigma,X}(v)$ has the root w with label $X(v)$, corresponding to v, (2) for each vertex v_i adjacent to v in G, $T_{G,\sigma,X}(v)$ has vertex w_i labeled with $X(v_i)$, and an edge from root w to w_i with label $l(v,v_i)$ given by $l(v,v_i)=(\sigma[v](v,v_i),\sigma[v_i](v,v_i))$, and (3) w_i is the root of $T_{G,\sigma,X}(v_i)$. It should be stressed that v, v_i, w, and w_i are not identifiers of parties and are introduced just for definition. For simplicity, we often use $T_X(v)$ instead of $T_{G,\sigma,X}(v)$, because we usually discuss views of some fixed network with some fixed port numbering. The *view of depth* h is the subtree of $T_X(v)$ of depth h with the same root as is that of $T_X(v)$, which is denoted by $T_X^h(v)$. For any value $x \in S$, the number of parties having x can be computed from $T_X^{2(n-1)}(v)$ [17]. Each party v can construct $T_X^h(v)$ as follows. In the first round, each party v constructs $T_X^0(v)$, i.e., the root of $T_X(v)$. For each party v, if v has $T_X^{i-1}(v)$ in the ith round, v can construct $T_X^i(v)$ in the $(i+1)$st round by exchanging $T_X^{i-1}(v)$ with his neighbors. By induction, in the $(h+1)$st round, each party v can construct $T_X^h(v)$.

Note that the size of $T_X^h(v)$ is exponential in h, which results in exponential time/communication complexity when we construct it. To reduce the time/communication complexity to something bounded by a polynomial, we introduce the new technique called *folded view* by generalizing the OBDD-reduction algorithm [6]. A *folded view (f-view) of depth h*, denoted by $\widetilde{T}_X^h(v)$, is a vertex- and edge-labeled directed acyclic multigraph obtained by merging nodes at the same level in $T_X^h(v)$ into one node if the subtrees rooted at them are isomorphic. The number of nodes in each level of an f-view is obviously bounded by n, and thus the total number of nodes in an f-view of depth h is at most hn. Actually, an f-view of depth $(h+1)$ can be efficiently constructed from a given f-view of depth h without unfolding it. Here we state without proof that every f-view of depth h is constructed in $O(h^2n^3(\log n)^2)$ time for each party and with $O(|E|h^2n^2\log n)$ bits of classical communication. Once $\widetilde{T}_X^{2(n-1)}(v)$ is constructed, each party can count without communication the number of parties having a value x in $O(n^6\log n)$ time.

4.2 The Algorithm

As in the previous section, we assume that the network is synchronous and each party knows the number n of parties prior to the algorithm. Again our algorithm is easily generalized to the asynchronous case. Although it needs a bit more elaboration, which is not mentioned in this version, it is also possible to modify our algorithm to work well even if only the upper bound N of the number of parties is given.

The algorithm consists of two stages, which we call Stages 1 and 2 hereafter. Stage 1 aims to have the n parties share a certain type of entanglement, and thus, this stage requires the parties to exchange quantum messages. In Stage 1, each party performs Subroutine Q $s = \lceil \log n \rceil$ times in parallel to share s pure quantum states $|\phi^{(1)}\rangle, \ldots, |\phi^{(s)}\rangle$ of n qubits. Here, each $|\phi^{(i)}\rangle$ is of the form $(|x^{(i)}\rangle + |\overline{x^{(i)}}\rangle)/\sqrt{2}$ for an n-bit string $x^{(i)}$ and its bitwise negation $\overline{x^{(i)}}$, and the lth qubit of each $|\phi^{(i)}\rangle$ is possessed by the lth party. It is stressed that only one round of quantum communication is necessary in Stage 1.

In Stage 2, the algorithm decides a unique leader among the n parties only by local quantum operations and classical communications with the help of the shared entanglement prepared in Stage 1. This stage consists of at most s phases, each of which reduces the number of eligible parties by at least half. In each phase i, let $E_i \subseteq \{1, \ldots, n\}$ be the set of all ls such that party l is still eligible. First every party runs Subroutine $\tilde{\mathrm{A}}$ to decide if state $|\phi^{(i)}\rangle$ is consistent or inconsistent over E_i. If state $|\phi^{(i)}\rangle$ is consistent, every party performs Subroutine $\tilde{\mathrm{B}}$, which first transforms $|\phi^{(i)}\rangle$ into the $|E_i|$-cat state $(|0^{|E_i|}\rangle + |1^{|E_i|}\rangle)/\sqrt{2}$ shared only by eligible parties and then calls Subroutine B described in the previous section to obtain an inconsistent state. Now each party l measures his qubits to obtain a label x_l and performs Subroutine $\tilde{\mathrm{C}}$ that reduces the number of eligible parties by at least half via minority voting.

More precisely, each party l performs Algorithm II described below with parameters "eligible," n, and d_l. The party who obtains output "eligible" is the unique leader.

```
                              Algorithm II
Input: a classical variable status, integers $n, d$
Output: a classical variable status

Stage 1:
    Let $s := \lceil \log n \rceil$ and prepare one-qubit quantum registers $\mathbf{R}_0^{(1)}, \ldots, \mathbf{R}_0^{(s)}$ and
    $\mathbf{R}_1^{(1)}, \ldots, \mathbf{R}_1^{(s)}$, each of which is initialized to the $|0\rangle$ state.
    Perform $s$ attempts of Subroutine Q in parallel, each with $\mathbf{R}_0^{(i)}$ and $d$ for $1 \leq i \leq s$,
    to obtain each $y^{(i)}$ and to share each $|\phi^{(i)}\rangle = (|x^{(i)}\rangle + |\overline{x^{(i)}}\rangle)/\sqrt{2}$ of $n$ qubits.
Stage 2:
    Let $k := n$.
    For $i := 1$ to $s$, repeat the following:
     1. Perform Subroutine $\tilde{\mathrm{A}}$ with status, $n$, $d$, and $y^{(i)}$ to obtain its output
        consistency.
     2. If consistency = "consistent," perform Subroutine $\tilde{\mathrm{B}}$ with $\mathbf{R}_0^{(i)}$, $\mathbf{R}_1^{(i)}$, status,
        $k$, $n$, and $d$.
     3. If status = "eligible," measure the qubits in $\mathbf{R}_0^{(i)}$ and $\mathbf{R}_1^{(i)}$ in the $\{|0\rangle, |1\rangle\}$
        basis to obtain a nonnegative integer $z$; otherwise set $z := -1$.
        Perform Subroutine $\tilde{\mathrm{C}}$ with status, $z$, $n$, and $d$ to compute nonnegative inte-
        gers $z_{\mathrm{minor}}$ and $c_{z_{\mathrm{minor}}}$.
     4. If $z \neq z_{\mathrm{minor}}$, let status := "ineligible."
        Let $k := c_{z_{\mathrm{minor}}}$.
     5. If $k = 1$, terminate and output status.
```

Subroutine Q: Subroutine Q is mainly for the purpose of sharing a cat-like quantum state $|\phi\rangle = (|x\rangle + |\overline{x}\rangle)/\sqrt{2}$. It also outputs a classical string, which is used in Stage 2 for each party to obtain the information on $|\phi\rangle$ via just classical communication. This subroutine can be performed in parallel by tagging messages, and thus Stage 1 involves only one round of quantum communication. The precise description of Subroutine Q is found below. Step 6 is necessary to disentangle the qubit in output register $\mathbf{R}_0$ from that in every $\mathbf{R}'_i$.

Subroutine $\tilde{\mathbf{A}}$: Suppose $|\phi\rangle = (|x\rangle + |\overline{x}\rangle)/\sqrt{2}$ is shared by the n parties. Let x_l be the lth bit of x and let X and $\overline{X}$ be mappings defined by $X(v_l) = x_l$ and $\overline{X}(v_l) = \overline{x_l}$ for each l, respectively, where v_l is the node corresponding to the lth party. Similar to Subroutine A in the previous section, Subroutine $\tilde{\mathrm{A}}$ checks the consistency of $|\phi\rangle$. Hereafter v denotes the node corresponding to the party invoking the subroutine. The output y of Subroutine Q is useful to construct

Subroutine Q

```
Input: a one-qubit quantum register $\mathbf{R}_0$, an integer $d$
Output: a one-qubit quantum register $\mathbf{R}_0$, a string $y$ of length $d$
1. Prepare $d$ one-qubit quantum registers $\mathbf{R}'_1, \ldots, \mathbf{R}'_d$ and $\mathbf{S}_1, \ldots, \mathbf{S}_d$, each of which
   is initialized to the $|0\rangle$ state.
2. Create the $(d+1)$-cat state $(|0^{d+1}\rangle + |1^{d+1}\rangle)/\sqrt{2}$ in registers $\mathbf{R}_0$, $\mathbf{R}'_1, \ldots, \mathbf{R}'_d$.
3. Exchange the qubit in $\mathbf{R}'_i$ with the party connected via port $i$ for $1 \le i \le d$ (i.e.,
   the original qubit in $\mathbf{R}'_i$ is sent via port $i$, and the qubit received via that port is
   newly set in $\mathbf{R}'_i$).
4. Set the content of $\mathbf{S}_i$ to $x_0 \oplus x_i$, for $1 \le i \le d$, where $x_0$ and $x_i$ denote the contents
   of $\mathbf{R}_0$ and $\mathbf{R}'_i$, respectively.
5. Measure the qubit in $\mathbf{S}_i$ in the $\{|0\rangle, |1\rangle\}$ basis to obtain a bit $y_i$, for $1 \le i \le d$.
   Set $y := y_1 \cdots y_d$.
6. Clear the content of $\mathbf{R}'_i$ by using $y_i$, for $1 \le i \le d$.
7. Output $\mathbf{R}_0$ and $y$.
```

f-view $\widetilde{T}_X^{n-1}(v)$ or $\widetilde{T}_{\overline{X}}^{n-1}(v)$, which can replace quantum communications in the consistency check. The precise description of Subroutine Ã is found below.

Subroutine Ã

```
Input: a classical variable status, integers $n, d$, a string $y$ of length $d$
Output: a classical variable consistency
1. Set $\widetilde{T}_Y^0(v)$ to the node labeled with $(0, \textbf{status})$, where $Y$ is either $X$ or $\overline{X}$.
2. For $i := 1$ to $(n-1)$, do the following:
   2.1 Send $\widetilde{T}_Y^{i-1}(v)$ and receive $\widetilde{T}_Y^{i-1}(v_j)$ via port $j$, for $1 \le j \le d$, where $v_j$ is the
       node corresponding to the party connected via port $j$.
   2.2 If the $j$th bit $y_j$ of $y$ is 1, negate the first element of every node label in
       $\widetilde{T}_Y^{i-1}(v_j)$, for $1 \le j \le d$.
   2.3 Set the root of $\widetilde{T}_Y^i(v)$ to the node labeled with $(0, \textbf{status})$.
       Set the $j$th child of the root of $\widetilde{T}_Y^i(v)$ to $\widetilde{T}_Y^{i-1}(v_j)$, for $1 \le j \le d$.
       For every level of $\widetilde{T}_Y^i(v)$, merge nodes at that level into one node if the f-views
       rooted at them are isomorphic.
3. If both label $(0, \text{“eligible”})$ and label $(1, \text{“eligible”})$ are found among the
   node labels in $\widetilde{T}_Y^{n-1}(v)$, let consistency := “inconsistent”; otherwise let
   consistency := “consistent.”
4. Output consistency.
```

Subroutine B̃: Suppose $|\phi\rangle = (|x\rangle + |\overline{x}\rangle)/\sqrt{2}$ shared by the n parties is consistent over the set E of eligible parties. After every ineligible party performs Step 2 in Subroutine B̃, the state shared by the eligible parties is either $\pm(|0^{|E|}\rangle + |1^{|E|}\rangle)/\sqrt{2}$ or $\pm(|0^{|E|}\rangle - |1^{|E|}\rangle)/\sqrt{2}$. The state $\pm(|0^{|E|}\rangle - |1^{|E|}\rangle)/\sqrt{2}$ is shared if and only if the number of ineligible parties that measured $|-\rangle$ in Step 1 is odd, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, respectively.

Steps 3 and 4 are for the purpose of having the eligible parties always share $\pm(|0^{|E|}\rangle + |1^{|E|}\rangle)/\sqrt{2}$. Again let v denote the node corresponding to the party that invoked the subroutine, and define the family $\{W_k\}$ of unitary transformations by $W_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{k}} \end{pmatrix}$. The precise description of Subroutine $\tilde{\mathrm{B}}$ is found below.

Subroutine $\tilde{\mathbf{B}}$

Input: one-qubit quantum registers $\mathbf{R}_0, \mathbf{R}_1$, a classical variable **status**, integers k, n, d
Output: one-qubit quantum registers $\mathbf{R}_0, \mathbf{R}_1$

```
1. Let $w := 0$.
2. If status = "ineligible," measure the qubit in $\mathbf{R}_0$ in the $\{|+\rangle, |-\rangle\}$ basis.
   If this results in $|-\rangle$, let $w := 1$.
3. Construct f-view $\widetilde{T}_W^{(2n-1)}(v)$ to count the number $p$ of parties with $w = 1$, where
   $W$ is the underlying mapping naturally induced by the $w$ values of all parties.
4. If $p$ is odd and status = "eligible," apply $W_k$ to the qubit in $\mathbf{R}_0$.
5. Perform Subroutine B with $\mathbf{R}_0$, $\mathbf{R}_1$ and $k$.
6. Output quantum registers $\mathbf{R}_0$ and $\mathbf{R}_1$.
```

Subroutine $\tilde{\mathbf{C}}$: Suppose each party l has value z_l. Subroutine C is a classical algorithm that computes value z_{minor} such that the number of parties with value z_{minor} is nonzero and the smallest among all possible z values. The precise description of Subroutine $\tilde{\mathrm{C}}$ is found below.

Subroutine $\tilde{\mathbf{C}}$

Input: integers z, n, d
Output: integers z_{minor}, $c_{z_{\mathrm{minor}}}$

```
1. Construct f-view $\widetilde{T}_Z^{(2n-1)}(v)$, where $Z$ is the underlying mapping naturally induced
   by the $z$ values of all parties.
2. For $i := 0$ to 3, count the number $c_i$ of parties having a value $z = i$ using $\widetilde{T}_Z^{(2n-1)}(v)$.
   If $c_i = 0$, let $c_i := n$.
3. Let $z_{\mathrm{minor}} \in \{m \mid c_m = \min_{0 \le i \le 3} c_i\}$.
4. Output $z_{\mathrm{minor}}$ and $c_{z_{\mathrm{minor}}}$.
```

4.3 Complexity Analysis

Here we only give the complexity of Algorithm II without proof.

Theorem 2. *Let $|E|$ and D be the number of edges and the maximum degree of the underlying graph, respectively. Given the number n of parties [the upper bound*

N of it], Algorithm II exactly elects a unique leader in $O(n \log n)$ [$O(N \log N)$] rounds and $O(n^6 (\log n)^2)$ [$O(N^7 (\log N)^2)$] time of which only the first round requires quantum communication. The total communication complexity over all parties is $O(|E| n^4 (\log n)^2)$ [$O(|E| N^5 (\log N)^2)$] which includes only $O(|E| \log n)$ [$O(|E| N \log N)$] qubits of quantum communication.

References

1. A. Ambainis, H. M. Buhrman, Y. Dodis, and H. Röhrig. Multiparty quantum coin flipping. In *Proc. of 19th IEEE Conf. on Computational Complexity*, pages 250–259, 2004.
2. D. Angluin. Local and global properties in networks of processors (extended abstract). In *Proc. of 20th ACM STOC*, pages 82–93, 1980.
3. Z. Bar-Yossef, T. S. Jayram, and I. Kerenidis. Exponential separation of quantum and classical one-way communication complexity. In *Proc. of 36th ACM STOC*, pages 128–137, 2004.
4. C. H. Bennett. Quantum cryptography using any two nonorthogonal states. *Phys. Rev. Lett.*, 68(21):3121–3124, 1992.
5. C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proc. of IEEE Conf. on Computers, Systems and Signal Processing*, pages 175–179, 1984.
6. R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, 1986.
7. H. M. Buhrman, R. E. Cleve, J. H. Watrous, and R. de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87(16):167902, 2001.
8. C. Crépeau, D. Gottesman, and A. D. Smith. Secure multi-party quantum computation. In *Proc. of 34th ACM STOC*, pages 643–652.
9. P. Dumais, D. Mayers, and L. Salvail. Perfectly concealing quantum bit commitment from any quantum one-way permutation. In *Proc. of EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 300–315, 2000.
10. A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Inf. Comput.*, 88(1):60–87, 1990.
11. N. A. Lynch. *Distributed Algorithms.* Morgan Kaufman Publishers, 1996.
12. D. Mayers. Unconditional security in quantum cryptography. *J. ACM*, 48(3):351–406, 2001.
13. R. Raz. Exponential separation of quantum and classical communication complexity. In *Proc. of 31st ACM STOC*, pages 358–367, 1999.
14. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
15. P. W. Shor and J. Preskill. Simple proof of security of the BB84 quantum key distribution protocol. *Phys. Rev. Lett.*, 85(2):441–444, 2000.
16. K. Tamaki, M. Koashi, and N. Imoto. Unconditionally secure key distribution based on two nonorthogonal states. *Phys. Rev. Lett.*, 90(16):167904, 2003.
17. M. Yamashita and T. Kameda. Computing on anonymous networks: Part I – characterizing the solvable cases. *IEEE Trans. Parallel Distrib. Syst.*, 7(1):69–89, 1996.
18. M. Yamashita and T. Kameda. Computing on anonymous networks: Part II – decision and memobership problems. *IEEE Trans. Parallel Distrib. Syst.*, 7(1):90–96, 1996.