Exact Quantum Amplitude Amplification for the Leader Election Problem

Seiichiro Tani[*†]
tani@theory.brl.ntt.co.jp

Hirotada Kobayashi[‡]
hirotada@nii.ac.jp

Keiji Matsumoto[‡†]
keiji@nii.ac.jp

[*] NTT Communication Science Laboratories, NTT Corporation
[†]Quantum Computation and Information Project, ERATO, JST
[‡]Foundations of Information Research Division, National Institute of Informatics

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. The first algorithm uses quantum amplitude amplification in a distributed manner under the anonymous condition, and runs in $O(n^2)$ rounds for any network topology, where n is the number of parties. The second one is for the case where the number of parties is a power of two, but it takes only linear rounds in n.

1 Introduction

Quantum computation and communication are turning out to be much more powerful than the classical equivalents for various computational tasks. Perhaps the most exciting developments in quantum computation would be polynomial-time quantum algorithms for factoring integers and computing discrete logarithms [15], 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, 16, 17]. Many other algorithms and protocols have been proposed that show the strength of quantum computation and communication, such as cryptographic results (e.g., [8, 7, 1]) and communication complexity results (e.g., [14, 6, 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 a unique leader if 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 the network is anonymous, i.e., each party has the same identifier [2, 10, 19, 20], 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 [19]. In the quantum setting, the situation is quite different. It was recently proved that the problem can be exactly solved even when the network is anonymous, if it consists of quantum links [18].

This paper gives two exact quantum algorithms that exactly solve the leader election problem in an anonymous network on the basis of approaches that are quite different from those of the algorithms in [18]. Each algorithm has its own flavor, which would give some characteristic aspect of quantum computing under the anonymous setting. Our first algorithm elects a unique leader from among n parties by using quantum amplitude amplification in a distributed manner under the anonymous condition. It attains the same performance as that of the algorithm in [18], i.e., it takes $O(n^2)$ rounds and $O(n^4)$ communication complexity for synchronous network of *any* topology. Our second algorithm is restricted to the case wherein the number of parties is a power of two. It takes at most just $3n$ rounds if the topology is a cycle, and at most $6n$ rounds for a general graph, while the communication complexity is $O(n^5 \log n)$ for both cases. Both algorithms are easily modified to support their use in asynchronous networks.

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 locally 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 $\{1, 2, \dots, d_v\}$. 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 every party does not have their unique local/global information except for the number of ports the parties have, the system is said to be *anonymous*. As an extreme case, 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 detail).

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 other parties to 0. In the case of anonymous networks, Yamashita and Kameda [19] proved that, if the "symmetricity" (defined in [19]) 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 $O(n^2)$-round Quantum Algorithm for General Case

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 corresponding 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.

Finally, we quote the exact quantum amplitude amplification theorem, which our algorithm is based on.

Theorem 1 ([9]) *Let $\mathcal{A}$ be any quantum algorithm that uses no measurements, and let $\chi : \mathbb{Z} \to \{0, 1\}$ be any Boolean function. Given the initial success probability $a > 0$ of $\mathcal{A}$, $Q(\mathcal{A}, \chi, \phi, \psi) Q^{m-1}(\mathcal{A}, \chi, \pi, \pi) \mathcal{A}|0\rangle$ gives a good solution with certainty by setting ψ and ϕ ($0 \leq \psi, \phi \leq 2\pi$) to some appropriate values depending on a, where $m = \Theta(\frac{1}{\sqrt{a}})$ and $Q(\mathcal{A}, \chi, \phi, \psi) = -\mathcal{A} F_0(\phi) \mathcal{A}^{-1} F_\chi(\psi)$ such that:*

$$F_\chi(\psi) : |x\rangle \mapsto \begin{cases} e^{i\psi}|x\rangle & \text{if } \chi(x) = 1 \\ |x\rangle & \text{if } \chi(x) = 0, \end{cases} \qquad F_0(\phi) : |x\rangle \mapsto \begin{cases} e^{i\phi}|x\rangle & \text{if } x = 0 \ldots 0 \\ |x\rangle & \text{otherwise.} \end{cases}$$

In particular, when $a > 0.5$, the theorem holds for $m = 1$ if we set ϕ and ψ [9] so that

$$e^{i\psi}(1 - e^{i\phi})\sqrt{a} \sin(\theta_a) = ((1 - e^{i\phi})a + e^{i\phi}) \frac{1}{\sqrt{1-a}} \cos(\theta_a),$$

where $\sin^2 \theta_a = a$. By a bit calculation, one can verify that such ϕ and ψ exist.

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 parties eligible to be the leader 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. Each phase prepares the uniform superposition of $2^{|E_i|}$ of x's where $x = x_1 \ldots x_{|E_i|}$ and x_l is a one-bit value possessed by the lth party in E_i. The phase then amplifies the amplitude of any inconsistent state over E_i so that we can get an inconsistent string over E_i by measurement. Once the parties in E_i share an inconsistent string, the number of eligible parties can be reduced with certainty by excluding $l \in E_i$ from E_i such that party l does not have the maximum one-bit value among all one-bit values in the string.

This is possible if every party knows the number $|E_i|$ of eligible parties by using Theorem 1, since he/she can know the initial probability of getting an inconsistent string, which is clearly $1 - \frac{2}{2^{|E_i|}}$. Actually, every party does not know $|E_i|$, however, we can detour to avoid this issue as described later.

A more precise description of the phase is as follows. First, each eligible party prepares the state $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ in one-qubit register $\mathbf{R}_0$ where H is the Hadamard operator over $\mathbb{C}^2$, while each ineligible party prepares the state $|0\rangle$ in $\mathbf{R}_0$. Notice that $\mathcal{A} = H^{\otimes |E_i|}$ over the qubits in $\mathbf{R}_0$s of all eligible parties. To realize $F_\chi(\psi)$ in a distributed manner where $\chi = 1$ if x is inconsistent over E_i and $\chi = 0$ otherwise, each party multiplies the amplitude of any inconsistent state by a factor of $e^{i\frac{1}{n}\psi}$. At this point, no party can check the inconsistency of each basis state, since he/she has only a part of information of the state. Every party, hence, computes the "witness" of the inconsistency: every party calls Subroutine A with $\mathbf{R}_0$ and another one-qubit register $\mathbf{S}$ whose content is initialized to "consistent," and Subroutine A runs a classical algorithm for every basis state that sets the content of $\mathbf{S}$ to "consistent" if parties share a consistent string over E_i and to "inconsistent" otherwise. Every party then multiplies the amplitude of any inconsistent state by the factor of $e^{i\frac{1}{n}\psi}$, which multiplies the amplitude of any inconsistent state as a whole by the factor of $e^{i\psi}$. Next every party inverts the computation and communication of Subroutine A to disentangle $\mathbf{S}$. This completes $F_\chi(\psi)$ on the qubits in $\mathbf{R}_0$s of all parties.

As $\mathcal{A}^{-1}$, every party in E_i applies H to the qubit in $\mathbf{R}_0$.

To realize $F_0(\psi)$, every party prepares the "witness" of the all-zero state: every party calls Subroutine B with $\mathbf{R}_0$ and $\mathbf{S}$ whose content is initialized to "allzero," and Subroutine B runs a classical algorithm for every basis state that sets the content of $\mathbf{S}$ to "allzero" if the parties in E_i share the all-zero string, i.e., $00\cdots 0$, as the contents of $\mathbf{R}_0$s, and to "nonallzero" otherwise. Every party then multiplies the amplitude of any inconsistent state such that by the factor of $e^{i\frac{1}{n}\phi}$ if s is "allzero." This multiplies the amplitude of the all-zero state $|00\ldots 0\rangle$ as a whole by a factor of $e^{i\phi}$. By inverting the computation and communication of Subroutine B to disentangle $\mathbf{S}$, $F_0(\psi)$ is completed.

Finally, every party in E_i applies H to the qubit in $\mathbf{R}_0$ to realize $\mathcal{A}$.

In each phase, every party can use k as $|E_i|$ to set ψ and ϕ to some appropriate values. Hence, the eligible parties may share a consistent state since k does not necessarily represent $|E_i|$. In this case, E_i is not changed, since our strategy is to exclude $l \in E_i$ from E_i such that party l does not have the maximum one-bit value among all one-bit values in the string. In the case of $k = |E_i|$ by chance, $|E_i|$ is reduced by at least one (but not to zero), since the eligible parties always succeed in sharing an inconsistent state (although they cannot verify it). It is clear from this observation that k is at least $|E_i|$ since k is initially $n = |E_i|$ and is decreased by 1 after each phase. This means that exactly one leader is elected after the last phase. More strictly, (1) in the first phase, obviously $k = |E_i| = n$, (2) if we assume that $k \geq |E_i|$ in the ith phase, $|E_i|$ decreases with certainty when $k = |E_i|$, and either decreases or is unchanged when $k > |E_i|$.

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

Algorithm QLE

```
Input: a classical variable status ∈ {"eligible", "ineligible"}, integers $n, d$

Output: a classical variable status ∈ {"eligible", "ineligible"}

1. Prepare one-qubit quantum registers $\mathbf{R}_0$ 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 To realize $F_\chi(\psi)$, perform the next operations.
    2.2.1 Perform Subroutine A with $\mathbf{R}_0$, $\mathbf{S}$, status, $n$, and $d$.
    2.2.2 Apply the next unitary operator on $\mathbf{R}_0$ and $\mathbf{S}$:
          $$|r\rangle|s\rangle \mapsto \begin{cases} e^{i\frac{1}{n}\psi}|r\rangle|s\rangle & \text{if } s \text{ is "inconsistent"} \\ |r\rangle|s\rangle & \text{if } s \text{ is "consistent"}, \end{cases}$$
          where $r$ and $s$ are the contents of $\mathbf{R}_0$ and $\mathbf{S}$, respectively.
    2.2.3 Invert every computation and communication of 2.2.1 to disentangle $\mathbf{S}$.
  2.3 If status="eligible," apply $H$ to the qubit in $\mathbf{R}_0$.
  2.4 To realize $F_0(\phi)$, do the next operations.
    2.4.1 Perform Subroutine B with $\mathbf{R}_0$, $\mathbf{S}$, status, $n$, and $d$, which sets the content of $\mathbf{S}$ to "allzero" if all
          parties have 0 as the contents of their $\mathbf{R}_0$s, and to "nonallzero" otherwise.
    2.4.2 Apply the next unitary operator on $\mathbf{R}_0$ and $\mathbf{S}$:
          $$|r\rangle|s\rangle \mapsto \begin{cases} e^{i\frac{1}{n}\phi}|r\rangle|s\rangle & \text{if } s \text{ is "allzero"} \\ |r\rangle|s\rangle & \text{if } s \text{ is "nonallzero"}, \end{cases}$$
          where $r$ and $s$ are the contents of $\mathbf{R}_0$ and $\mathbf{S}$, respectively.
```

```
        2.4.3 Invert every computation and communication of 2.4.1 to disentangle $\mathbf{S}$.
    2.5 If **status**="eligible," apply $H$ to the qubits in $\mathbf{R}_0$.
    2.6 If **status** = "eligible," measure the qubit in $\mathbf{R}_0$ in the $\{|0\rangle, |1\rangle\}$ basis to obtain one-bit value $z$; otherwise
        let $z := -1$.
    2.7 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**.
```

3.1.1 Subroutine A:

Subroutine A is used basically for the purpose of checking the consistency of all strings that are superposed to a quantum state shared by the parties. We use the commute operator "$\circ$" over 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 the 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.

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	$\times$	$\times$	$*$	$\times$	$\times$	$\times$	$\times$	$\times$

Subroutine A

Input: one-qubit quantum registers $\mathbf{R}_0$ and $\mathbf{S}$, a classical variable **status** $\in$ {"eligible", "ineligible"}, integers n, d

Output: one-qubit quantum registers $\mathbf{R}_0$ and $\mathbf{S}$

```
1. Prepare two-qubit quantum registers $\mathbf{X}_0^{(1)}, \ldots, \mathbf{X}_d^{(1)}, \ldots, \mathbf{X}_0^{(n-1)}, \ldots, \mathbf{X}_d^{(n-1)}, \mathbf{X}_0^{(n)}$.
   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 $\mathbf{X}_1^{(t)}, \ldots, \mathbf{X}_d^{(t)}$.
        (Perform CNOT targeted to the qubit in each of $\mathbf{X}_1^{(t)}, \ldots, \mathbf{X}_d^{(t)}$ controlled by the qubit in $\mathbf{X}_0^{(t)}$).
    2.2 Exchange the qubit in $\mathbf{X}_i^{(t)}$ with the party connected via port $i$ for $1 \leq i \leq d$ (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 content of $\mathbf{X}_i^{(t)}$ for $0 \leq i \leq d$.
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 $\mathbf{S}$.
```

As one can see from the description of Algorithm QLE, 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 $\mathbf{S}$, 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.

For the proof of correctness, see appendix A. Subroutine A takes $\Theta(n)$ rounds and the total communication complexity over all parties is $\Theta(|E|n)$, where $|E|$ and D are the number of edges and the maximum degree of the underlying graph, respectively (see appendix B).

3.1.2 Subroutine B

Subroutine B works in almost the same way as Subroutine A except it uses commute operator $\diamond$ over set $\mathcal{S} = \{0, 1, *, \times\}$ instead of $\circ$ and, "allzero" and "nonallzero" instead of "eligible" and "ineligible." Operator $\diamond$ is defined as follows: $x \diamond y$ is $\times$ if both of x and y is not in $\{0, *\}$, $*$ if $x = y = *$, and 0 otherwise (see appendix C).

3.1.3 Subroutine C

Subroutine C is a simple classical algorithm based on flooding that computes the maximum value of z values 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 in appendix D).

3.2 Complexity Analysis and Discussion

From lemma 7, Subroutine A takes $\Theta(n)$ rounds and has communication complexity $\Theta(|E|n)$. Subroutine B has the same round/communication complexity. Subroutine C also takes $\Theta(n)$ rounds and has communication complexity $\Theta(|E|n)$ except that the communication is classical. Algorithm QLE requires $(n - 1)$ phases. Thus, we have the next theorem.

Theorem 2 *Let $|E|$ be the number of edges of the underlying graph. Given the number n of parties, Algorithm QLE exactly elects a unique leader in $\Theta(n^2)$ rounds. 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 QLE with N instead of n. The complexity in this case is described simply by replacing every n by N in Theorem 2.

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

The amplitude of desirable states can be amplified to one also by decreasing the success probability of $\mathcal{A}$ [9]. Consider preparing $\alpha|0\rangle + \beta|1\rangle$ in step 2.1 for some complex numbers α and β such that $|\alpha|^2 + |\beta|^2 = 1$ instead of preparing $(|0\rangle + |1\rangle)/\sqrt{2}$. The probability of observing inconsistent states becomes $1 - (|\alpha^n|^2 + |\beta^n|^2)$. Thus, if we set α and β such that $1 - (|\alpha^n|^2 + |\beta^n|^2) = 1/4$, i.e., $|\alpha^n|^2 + |\beta^n|^2 = 3/4$, the amplitude of the inconsistent states can be amplified to exactly one by applying $Q(\mathcal{A}, \chi, \pi, \pi)$ to $\mathcal{A}|0\rangle$. Such α and β exists, since $|\alpha^n|^2 + |\beta^n|^2 = 1 > 3/4$ if $\alpha = 0$ and $\beta = 1$ and $|\alpha^n|^2 + |\beta^n|^2 = 2/2^n < 3/4$ if $\alpha = \beta = \frac{1}{\sqrt{2}}$.

4 Linear-round Quantum Algorithm for a Special Case

As in the previous section, 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. This section describes another algorithm that takes only $6n$ rounds if n is a power of two.

4.1 View and Folded View

First, we briefly review the classical technique, *view* [19]. Let $G = (V, E)$ be the underlying network topology and let $n = |V|$. Assume that each party corresponding to node $v \in V$ has a value $x_v \in S$ for some set S, and consider the mapping $X: V \to S$ 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 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 $T_X^h(v)$. 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 $T_X^h(v)$.

We denote the set of non-isomorphic views by $\Gamma_{G,\sigma,X}$, i.e., $\Gamma_{G,\sigma,X}=\{T_{G,\sigma,X}(v)\mid v\in V\}$, and the set of non-isomorphic views truncated to depth h by $\Gamma^h_{G,\sigma,X}$, i.e., $\Gamma^h_{G,\sigma,X}=\{T^h_{G,\sigma,X}(v)\mid v\in V\}$. For simplicity, we may use Γ_X and Γ_X^h instead of $\Gamma_{G,\sigma,X}$ and $\Gamma^h_{G,\sigma,X}$, respectively. The number of views isomorphic to $T_X\in\Gamma_X$ is known to be constant over all T_X. Hence, this number c_X is equal to $n/|\Gamma_X|$. For any subset S' of S, let $\Gamma_X(S')$ be the subset of Γ_X in which the root of each view $T_X\in\Gamma_X(S')$ is labeled with a value in S'. The number of parties having values in S' becomes $c_X|\Gamma_X(S')|=n|\Gamma_X(S')|/|\Gamma_X|$. Once the party corresponding to node v constructs $T_X^{2(n-1)}(v)$, it can compute $\Gamma_X^{(n-1)}$ and, hence, $|\Gamma_X|$ and $|\Gamma_X(S')|$ (and the number of parties having values in S), since $T_X(u)$ is isomorphic to $T_X(u')$ if and only if $T_X^{n-1}(u)$ is isomorphic to $T_X^{n-1}(u')$ for any u and u'

4.2 The Algorithm

In what follows, we assume that the underlying graph is a cycle of length n for simplicity. Basically the same approach will work for any connected graph, as will be discussed later.

The idea is as follows. The algorithm creates an inconsistent state $|\phi\rangle$ shared by all parties such that $|\phi\rangle$ is a superposition of only the n-bit strings whose have Hamming weights of odd values. Suppose that every party i obtains a single-bit value y_i by measuring $|\psi\rangle$. As we will prove later, if every party i uses $T_X^{n-1}(v_i)$ as its identifier, a unique leader can be elected deterministically when n is a power of 2, where v_i is the node of the underlying graph corresponding to party i. Thus, only a single run of the above process is sufficient to elect a unique leader, which takes just linear rounds in n.

First, every party prepares $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|0\rangle$ in one-qubit registers $\mathbf{R}_0$ and $\mathbf{S}$, respectively. They then call Subroutine A' with $\mathbf{R}_0$, $\mathbf{S}$, n and d, to set to $\mathbf{S}$ the Hamming weight (mod 2) of the contents in all $\mathbf{R}_0$s in n rounds. Next every party measures the qubit in $\mathbf{S}$ in the $\{|0\rangle,|1\rangle\}$ basis and stores the result into variable y. If $y=0$ (1), the resulting state $|\psi\rangle$ is the uniform superposition of the n-bit strings that have the Hamming weights of even (resp. odd) values. In the case of $y=0$, every party applies two kinds of unitary operators V and U_n to the qubit in $\mathbf{R}_0$ to share a superposition of only the strings that have the Hamming weights of odd values. By measuring the qubit in $\mathbf{R}_0$, every party gets classical value z. Finally, every party calls Subroutine C', which elects a leader by constructing the view for the mapping naturally induced by z values of all parties in $(n-1)$ rounds.

More precisely, each party l performs Algorithm QLE' described below with parameters "eligible," n, 2. The party who obtains output "eligible" is the unique leader.

Algorithm QLE'

Input: classical variable **status**, integers n,d

Output: classical variable **status**

1. Prepare $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|0\rangle$ in one-qubit registers $\mathbf{R}_0$ and $\mathbf{S}$, respectively.

2. Call Subroutine A' with $\mathbf{R}_0$, $\mathbf{S}$, n and d, to set to $\mathbf{S}$ the Hamming weight (mod 2) of the contents of all parties' $\mathbf{R}_0$s .
   Measure the qubit in $\mathbf{S}$ in the $\{|0\rangle,|1\rangle\}$ basis and store the result into variable y.

3. If $y=0$, apply $U_n\cdot V$ to the qubit in $\mathbf{R}_0$, where for any positive integer k,
$$U_k=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & e^{-i\frac{\pi}{k}}\\ -e^{i\frac{\pi}{k}} & 1\end{pmatrix},\qquad V=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\\ 1 & 1\end{pmatrix}.$$
   Measure the qubit in $\mathbf{R}_0$ in the $\{|0\rangle,|1\rangle\}$ basis and store the result into variable z.

4. Call Subroutine C' with **status**, z, n and d.

5. Output **status**.

4.2.1 Subroutine A'

Subroutine A' just runs the following simple classical algorithm for a superposition of n-bit strings. Suppose that each party i has a single-bit value r_i and prepares variable s initialized to 0. Every party sends a copy of r_i via port 1 and 0 via port 2. Notice that we assume that the underlying graph is a cycle. After receiving values x_1 and x_2 via ports 1 and 2, respectively, every party computes the XOR of the values and s, and sets s to the result. Every party then sends x_1 and x_2 via ports 2 and 1, respectively. After repeating this n times, every party has in s the Hamming weight $\pmod 2$ of the n-bit string consisting of all r_is. For a more precisely, see appendix E.

4.2.2 Subroutine C'

In Subroutine C', each party (corresponding to node v of the underlying graph) constructs view $T_Z^{2(n-1)}(v)$ where Z is the underlying mapping naturally induced by the z values of all parties. Each party then computes Γ_Z^{n-1} from $T_Z^{2(n-1)}(v)$, and selects view $T_{\min} \in \Gamma_Z^{n-1}$ such that $T_{\min}$ has the minimum binary expression (under a fixed natural binary encoding scheme of view) among Γ_Z^{n-1}. Finally, only the party that has its view isomorphic to $T_{\min}$ remains eligible.

Subroutine C'

Input: a classical variable **status** $\in$ {"eligible", "ineligible"}, a single-bit value z, integers n, d

Output: a classical variable **status** $\in$ {"eligible", "ineligible"}

1. Construct view $T_Z^{2(n-1)}(v)$ where Z is the underlying mapping naturally induced by the z values of all parties.
2. Compute Γ_Z^{n-1} from $T_Z^{2(n-1)}(v)$.
   Set $T_{\min}$ to $T \in \Gamma_Z^{n-1}$ such that $\overline{T} = \min\{\overline{T'} \mid T' \in \Gamma_Z^{n-1}\}$, where $\overline{T}$ and $\overline{T'}$ are the binary expressions of T and T' under a fixed natural binary encoding scheme of view.
3. If $T_Z^{(n-1)}(v)$ is NOT isomorphic to $T_{\min}$, set **status** to "ineligible"
4. Output **status**.

4.3 Analysis

After steps 1 and 2 of QLE', it is easy to see that the contents of all $\mathbf{R}_0$s are the uniform superposition of the n-bit strings that have the Hamming weights of even (odd) values, if $y = 0$ (resp. 1). In the case of $y = 0$, Step 3 transforms the state to a superposition of the strings that have the Hamming weights of *odd* values, due to lemmas 3 and 4. In Subroutine C', no pair of views $T_Z^{(n-1)}(v)$ over all v is isomorphic at step 3, if the string consisting of the z values has the Hamming weight of an odd value and n is a power of 2, due to Proposition 5. For only one v, hence, $T_Z^{(n-1)}(v)$ is isomorphic to $T_{\min}$. Thus, only one leader is elected.

Communication occurs only in Subroutine A' and C', which take n and $2(n-1)+1 = 2n-1$ rounds. Thus the algorithm takes at most $3n$ rounds in total. As for communication complexity, Subroutine A' needs $O(n^2)$ qubit quantum communication, since n qubits go along the circle. The communication in Subroutine C' is needed to construct the view. Note that the size of $T_X^h(v)$ is exponential in h, which results in exponential communication complexity. The technique called f-view [18] can reduce the communication complexity required to construct $T_X^h(v)$ down to $O(|E|h^2n^2 \log n)$, where E is the set of edges of the underlying graph. Thus Subroutine C' has $O(n^5 \log n)$ classical communication complexity.

The proofs of the next two lemmas are given in appendices F and G.

Lemma 3 *Suppose that* $|\phi_k\rangle = (U_k)^{\otimes k}(\frac{|0^k\rangle+|1^k\rangle}{\sqrt{2}})$. *Then* $\langle\phi_k|j\rangle = 0$, *for any positive integer* k *and any non-negative integer* j $(0 \le j \le 2^k - 1)$ *such that the Hamming weight of (the binary expression of)* j *is* $0 \pmod 2$.

Lemma 4 $(V)^{\otimes k}\frac{1}{\sqrt{2^{k-1}}}\sum_{j=0, j\in \mathrm{HW}(j)=0 \pmod 2}^{2^k-1}|j\rangle = \frac{|0^k\rangle+|1^k\rangle}{\sqrt{2}}$ *for any positive integer* k, *where* $\mathrm{HW}(j)$ *denotes the Hamming weight of (the binary expression of)* j.

Proposition 5 *For any* 2^m*-party distributed system with underlying graph* $G = (V, E)$, *if* X *is a mapping such that* $X : V \to \{0, 1\}$ *and* $|\{v \mid X(v) = 1, v \in V\}| = 1 \pmod 2$, *no pair of any two* $T_{X,\sigma,G}^{(n-1)}(v)$*s over all* v *is isomorphic.*

Proof. The number of isomorphic views is constant over all views [19]. Since for any u and v such that $X(u) \neq X(v)$, $T_{X,\sigma,G}(u)$ is not isomorphic to $T_{X,\sigma,G}(v)$, the number of views isomorphic to any view T is a common divisor of $|\{v \mid X(v) = 1, v \in V\}|$ and 2^m. When $|\{v \mid X(v) = 1, v \in V\}| = 1 \pmod 2$, 1 is the unique common divisor. Since $T_{X,\sigma,G}(v)$ is isomorphic to $T_{X,\sigma,G}(v')$ if and only if $T^{(n-1)}_{X,\sigma,G}(v)$ is isomorphic to $T^{(n-1)}_{X,\sigma,G}(v')$ for any $v, v' \in V$ [13], the proof is completed. □

For a general graph, we need change only Subroutine A'. If every party has a one-bit value, which forms an n-bit string, every party can compute the Hamming weight of the string by constructing a view of depth $2(n-1)$. We use this classical algorithm in Subroutine A' for each basis state in the contents of all $\mathbf{R}_0$s. Hence Subroutine A' takes $2n-1$ rounds. Furthermore, we need to erase all garbage created by Subroutine A' by inverting every computation and communication performed by Subroutine A', in order for step 3 of QLE' to work well. Therefore the total number of rounds required is at most $6n$ rounds.

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] H. M. Buhrman, R. E. Cleve, J. H. Watrous, and R. de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87(16):167902, 2001.

[7] C. Crépeau, D. Gottesman, and A. D. Smith. Secure multi-party quantum computation. In *Proc. of 34th ACM STOC*, pages 643–652.

[8] 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.

[9] M. M. Gilles Brassard, Peter Hoyer and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information: A Millennium Volume*, 305, 2002.

[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] N. Norris. Universal covers of graphs: Isomoriphism to depth n-1 implies isomoriphism to all depths. *Discrete Applied Mathematics*, 56(1):61–74, 1995.

[14] R. Raz. Exponential separation of quantum and classical communication complexity. In *Proc. of 31st ACM STOC*, pages 358–367, 1999.

[15] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[16] 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.

[17] K. Tamaki, M. Koashi, and N. Imoto. Unconditionally secure key distribution based on two nonorthogonal states. *Phys. Rev. Lett.*, 90(16):167904, 2003.

[18] S. Tani, H. Kobayashi, and K. Matsumoto. Exact quantum algorithms for the leader election problem. In *Proceedings of 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, volume 3404 of *Lecture Notes in Computer Science*, pages 581–592. Springer, 2005.

[19] 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.

[20] 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.

A Correctness of Subroutine A

Lemma 6 *Suppose that n parties share n-qubit state $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle$. If each party runs Subroutine A with the following objects as input: (1) one-qubit quantum registers $\mathbf{R}_0$, which stores one of the qubits whose contents are $|\psi\rangle$, and $\mathbf{S}$, (2) a classical variable* **status** $\in \{\text{eligible}, \text{ineligible}\}$, *(3) integers n and d, which are the number of parties and the number of neighbors of the party, respectively, Subroutine A outputs $\mathbf{R}_0$ and $\mathbf{S}$ whose contents are $\sum_{i=0}^{2^n-1} \alpha_i|i\rangle \otimes |C_i\rangle^{\otimes n}$, where C_i is "consistent" if $|i\rangle$ is consistent over E, i.e., the set of indices of parties whose* **status** *is "eligible," and "inconsistent" otherwise.*

Proof. Subroutine A just superposes an application of a reversible classical algorithm to each basis state. Furthermore, no interference occurs since the contents of $\mathbf{R}_0$s are never changed during the execution of the algorithm. Thus, it is sufficient to prove the classical algorithm. Below, we do not explicitly consider the reversibility of the algorithm since we can easily make the algorithm reversible.

Suppose that we are given one-bit classical registers $\mathbf{R_0}$ and $\mathbf{S}$, classical variable **status**, and integers n and d. For any party i and a positive integer t, the content of $\mathbf{X}_{0,i}^{(t+1)}$ is set to $x_{0,i}^{(t)} \circ x_{1,i}^{(t)} \circ \cdots \circ x_{d,i}^{(t)}$ in step 2.3, where $\mathbf{X}_{j,i}^{(k)}$ is $\mathbf{X}_j^{(k)}$ of party i, and $x_{j,i}^{(k)}$ is the content of $\mathbf{X}_{j,i}^{(k)}$. For any i, by expanding this recurrence relation, the content of $\mathbf{X}_{0,i}^{(n)}$ can be expressed in the form of $x_{0,i_1}^{(1)} \circ \cdots \circ x_{0,i_m}^{(1)}$ for some $m \leq (n-1)^{(n-1)}$. Since the diameter of the underlying graph is at most $n-1$, there is at least one $x_{0,j}^{(1)}$ in $x_{0,i_1}^{(1)}, \ldots, x_{0,i_m}^{(1)}$ for each j. Thus $x_{0,i_1}^{(1)}, \ldots, x_{0,i_m}^{(1)}$ is equal to $x_{0,1}^{(1)}, x_{0,2}^{(1)}, \ldots, x_{0,n}^{(1)}$, since $\circ$ is associative and $x \circ x = x$ for any $x \in \{0, 1, *, \times\}$.

Therefore, we can derive the following facts: (1) if there are both 0 and 1 in the the contents of $\mathbf{R}_0$s of all parties, the algorithm outputs $\mathbf{S}$ ='$\times$'; (2) if only one of 0 or 1 is in the the contents of $\mathbf{R}_0$s except $*$, the algorithm outputs $\mathbf{S}$ ='0' or '1', respectively; (3) if there is at least one eligible party, the algorithm does not output $\mathbf{S}$ ='$*$'. $\square$

B Complexity of Subroutine A

Lemma 7 *Let $|E|$ and D be the number of edges and the maximum degree of the underlying graph, respectively. Subroutine A takes $\Theta(n)$ rounds. The total communication complexity over all parties is $\Theta(|E|n)$.*

Proof.

It is sufficient to consider just step 2 since only steps 2 and 4 communicate and step 4 is the inversion of step 2. It is easy to see that step 2 takes $\Theta(n)$ rounds. With regard to communication complexity, every party sends one qubit via each link for each repetition in step 2. Hence every party i needs to communicate $\Theta(nd_i)$ qubits in step 2. By summing up the number of qubits communicated over all parties, we have $\Theta(n|E|)$. $\square$

C Definition of operator $\diamond$

Table 2: The definitions of commute operator "$\diamond$"

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

D Subroutine C

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 $1 \le i \le d$.
        Set $y_i$ to the value received via port $i$ for $1 \le i \le d$.
    2.3 Let $z_{\max} := \max_{0 \le i \le d} y_i$.
3. Output $z_{\max}$.
```

E Subroutine A'

Subroutine A'

Input: one-qubit quantum registers $\mathbf{R}_0$ and $\mathbf{S}$, integers $n, d = 2$

Output: one-qubit quantum registers $\mathbf{R}_0$ and $\mathbf{S}$

```
1. Prepare one-qubit quantum registers $\mathbf{X}_1, \mathbf{X}_2$.
   Copy the content of $\mathbf{R}_0$ to $\mathbf{X}_2$
   (Perform CNOT targeted to the qubit in $\mathbf{X}_2$ controlled by the qubit in $\mathbf{R}_0$).
   Set the content of $\mathbf{X}_1$ to $|0\rangle$.
2. For $t := 1$ to $n$, do the following:
    2.1 Swap the contents of $\mathbf{X}_1$ and $\mathbf{X}_2$
    2.2 Exchange the qubit in $\mathbf{X}_i$ with the party connected via port $i$ for $1, 2$ (i.e., the original qubit in $\mathbf{X}_i$ is
        sent via port $i$, and the qubit received via that port is newly set in $\mathbf{X}_i$).
    2.3 Set the contents of $\mathbf{S}$ to $s \oplus x_1 \oplus x_2$ where $s$ and $x_i$ denote the contents of $\mathbf{S}$ and $\mathbf{X}_i$ for $i = 1, 2$,
        respectively.
3. Copy the content of $\mathbf{R}_0$ to $\mathbf{X}_2$ to disentangle $\mathbf{X}_2$.
4. Output quantum registers $\mathbf{R}_0$ and $\mathbf{S}$.
```

F Proof of lemma 3

Proof. Fix a k-bit string j such that the Hamming weight of j is $2m$ for any nonnegative integer m ($\le \lfloor k/2 \rfloor$). The amplitude of $|j\rangle$ in $(U_k)^{\otimes k}(\frac{|0^k\rangle + |1^k\rangle}{\sqrt{2}})$ is as follows:

$$\frac{1}{\sqrt{2}} \left\{ \left(\frac{1}{\sqrt{2}}\right)^{k-2m} \left(\frac{-e^{i\frac{\pi}{k}}}{\sqrt{2}}\right)^{2m} + \left(\frac{e^{-i\frac{\pi}{k}}}{\sqrt{2}}\right)^{k-2m} \left(\frac{1}{\sqrt{2}}\right)^{2m} \right\} = 0.$$

□

G Proof of lemma 4

Proof.

Let $|\phi\rangle$ be

$$(V^{\dagger})^{\otimes k} \frac{|0^k\rangle + |1^k\rangle}{\sqrt{2}} = \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \right)^{\otimes k} \frac{|0^k\rangle + |1^k\rangle}{\sqrt{2}}.$$

Fix a k-bit string j such that the Hamming weight of j is $2m$ for any nonnegative integer m ($\leq \lfloor k/2 \rfloor$). The amplitude of $|j\rangle$ in $|\phi\rangle$ is

$$\frac{1}{\sqrt{2}}\left\{\left(\frac{1}{\sqrt{2}}\right)^{k-2m}\left(\frac{-1}{\sqrt{2}}\right)^{2m}+\left(\frac{1}{\sqrt{2}}\right)^{k-2m}\left(\frac{1}{\sqrt{2}}\right)^{2m}\right\}=\frac{1}{\sqrt{2^{k-1}}}.$$

Fix a k-bit string j such that the Hamming weight of j is $2m-1$ for any nonnegative integer m ($\leq \lfloor (k+1)/2 \rfloor$). The amplitude of $|j\rangle$ in $|\phi\rangle$ is

$$\frac{1}{\sqrt{2}}\left\{\left(\frac{1}{\sqrt{2}}\right)^{k-(2m-1)}\left(\frac{-1}{\sqrt{2}}\right)^{(2m-1)}+\left(\frac{1}{\sqrt{2}}\right)^{k-(2m-1)}\left(\frac{1}{\sqrt{2}}\right)^{(2m-1)}\right\}=0.$$

$\square$