Distributed Function Calculation via Linear Iterative Strategies in the Presence of Malicious Agents

Shreyas Sundaram and Christoforos N. Hadjicostis

***Abstract*—Given a network of interconnected nodes, each with its own value (such as a measurement, position, vote, or other data), we develop a distributed strategy that enables some or all of the nodes to calculate any arbitrary function of the node values, despite the actions of malicious nodes in the network. Our scheme assumes a broadcast model of communication (where all nodes transmit the same value to all of their neighbors) and utilizes a linear iteration where, at each time-step, each node updates its value to be a weighted average of its own previous value and those of its neighbors. We consider a node to be malicious or faulty if, instead of following the predefined linear strategy, it updates its value arbitrarily at each time-step (perhaps conspiring with other malicious nodes in the process). We show that the topology of the network completely characterizes the resilience of linear iterative strategies to this kind of malicious behavior. First, when the network contains $2f$ or fewer vertex-disjoint paths from some node x_j to another node x_i, we provide an explicit strategy for f malicious nodes to follow in order to prevent node x_i from receiving any information about x_j's value. Next, if node x_i has at least $2f+1$ vertex-disjoint paths from every other (non-neighboring) node, we show that x_i is guaranteed to be able to calculate any arbitrary function of all node values when the number of malicious nodes is f or less. Furthermore, we show that this function can be calculated after running the linear iteration for a finite number of time-steps (upper bounded by the number of nodes in the network) with almost any set of weights (i.e., for all weights except for a set of measure zero).**

***Index Terms*—Distributed function calculation, distributed consensus, fault-tolerant consensus, observability theory, structured systems, networked control, multi-agent systems, wireless broadcast model**

I. Introduction

In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of certain parameters. Examples include sensor networks where sink nodes are tasked with calculating the average measurement value of all the sensors [1], [2], and multi-agent systems where all agents communicate with each other to coordinate their speed and direction [3]. Various algorithms to accomplish function calculation in networks have been proposed by the computer science, communication, and control communities over the past few decades [4], [5], [6], [7]. The special case of *distributed consensus*, where all nodes in the network calculate the same function [4], has received extensive attention from the control community due to its applicability to topics such as cooperative control, multi-agent systems, and modeling flocking behavior in biological and physical systems [8]. In these cases, the approach to consensus is to use a linear iteration, where each node in the network repeatedly updates its value as a weighted linear combination of its own value and those of its neighbors [9], [10], [11], [12]. These works have revealed that if the network topology satisfies certain conditions, the weights for the linear iteration can be chosen so that all of the nodes asymptotically converge to the same value (even if the network connections are time-varying). Recently, it was shown in [13] that in networks with time-invariant topologies, this linear iterative strategy can also be applied to the more general function calculation problem, allowing any node to calculate any arbitrary function of the node values in a finite number of time-steps (upper bounded by the size of the network).

Due to the increasingly prevalent use of sensor networks and multi-agent systems in life– and mission–critical applications (e.g., [14]), it is imperative to analyze any proposed network algorithms to determine their resilience to nodes that behave in erroneous or unexpected ways. This might be the case, for example, if some nodes in the network are compromised by a malicious attacker whose objective is to disrupt the operation of the network [4]. Alternatively, the nodes might suffer from hardware malfunctions, thereby causing them to calculate their update value incorrectly [15]. The robustness of various existing information dissemination schemes to these types of abnormal behaviors has been investigated in the literature (e.g., see [6]); however, a similar analysis of the susceptibility of *linear iterative* strategies to malicious or faulty behavior has received scant attention, and this is the focus of our work. Specifically, we allow for the possibility that some nodes in the network update their values at each time-step in an arbitrary (possibly coordinated) manner, instead of following the predefined strategy of using a specific linear combination of their neighbors' (and own) values. The contribution of this paper is to show that the graph connectivity is the determining factor for the ability of linear iterative strategies to tolerate malicious (or faulty) agents. First, we demonstrate that if a

This material is based upon work supported in part by the National Science Foundation (USA), under NSF Career Award 0092696 and NSF ITR Award 0426831. The research leading to these results has also received funding from the European Community (EC) Seventh Framework Programme (FP7/2007-2013) under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or EC. Parts of this work were presented in preliminary form at the 2008 American Control Conference.

S. Sundaram is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. E-mail: ssundara@uwaterloo.ca. C. N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, and also with the Coordinated Science Laboratory, and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. E-mail: chadjic@ucy.ac.cy.

Address for correspondence: Shreyas Sundaram, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, Canada.

given node x_i has $2f$ or fewer vertex-disjoint paths from some other node x_j, then there exists a set of f or fewer nodes that can maliciously update their values so that x_i cannot obtain sufficient information to calculate any function that involves node x_j's value, regardless of the number of time-steps for which the linear iteration is run. The fact that only f malicious nodes will be required to disrupt a network with $2f$ connectivity is not surprising, given existing results in the distributed systems literature (which we will review later in the paper). What is *not* clear, however, is exactly *how* these f malicious nodes should behave in order to disrupt the linear iterative strategy and not be identified in the process. Our analysis solves this problem by providing an explicit choice of malicious nodes, along with a strategy for them to follow.

Next, we show that if a given node has at least $2f+1$ vertex-disjoint paths from any other node (with which it does not have a direct connection), it can work around up to f misbehaving nodes to correctly calculate any arbitrary function of the node values. Furthermore, we show that this can be achieved after running the linear iteration for a finite number of time-steps with almost any set of weights. To derive our results, we build upon the function calculation algorithm in [13], and exploit concepts from classical linear system theory (such as structured system theory, strong observability, and invariant zeros of linear systems) to overcome malicious or faulty nodes. As we will describe in further detail later, our results serve to narrow the gap between linear iterative schemes and existing fault-tolerant consensus algorithms in the literature (such as those described in [4]). In particular, we show that under the broadcast model of communication (which implies that nodes cannot send different information to different neighbors – e.g., in a wireless setting), simple linear-iterative schemes are as powerful as any other algorithm in terms of the number of malicious nodes they can handle (i.e., they impose the same constraints on the underlying network topology).

In our development, we use $\mathbf{e}_{i,N}$ to denote the $N\times 1$ column vector with a 1 in its i–th position and 0's elsewhere. The symbol $\mathbf{I}_N$ denotes the $N\times N$ identity matrix and $\mathbf{A}'$ indicates the transpose of matrix $\mathbf{A}$. We will denote the cardinality of a set $\mathcal{S}$ by $|\mathcal{S}|$, and for a pair of sets $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{S}\setminus\mathcal{T}$ denotes the elements of $\mathcal{S}$ that are not in $\mathcal{T}$. The set of nonnegative integers is denoted by $\mathbb{N}$.

II. Background

A. Graph Theory

We will require the following terminology in order to facilitate our discussion. Further details can be found in standard texts on graph theory, such as [16].

A graph is an ordered pair $\mathcal{G}=\{\mathcal{X},\mathcal{E}\}$, where $\mathcal{X}=\{x_1,x_2,\ldots,x_N\}$ is a set of vertices (or nodes), and $\mathcal{E}$ is a set of ordered pairs of different vertices, called directed edges. If, $\forall x_i,x_j\in\mathcal{X}$ with $i\neq j$, $(x_i,x_j)\in\mathcal{E}\Leftrightarrow(x_j,x_i)\in\mathcal{E}$, the graph is said to be undirected. The vertices in the set $\mathcal{N}_i=\{x_j|(x_j,x_i)\in\mathcal{E}\}$ are said to be neighbors of vertex x_i, and the in-degree of vertex x_i is denoted by $\deg_i=|\mathcal{N}_i|$. A *subgraph* of $\mathcal{G}$ is a graph $\mathcal{H}=\{\bar{\mathcal{X}},\bar{\mathcal{E}}\}$, with $\bar{\mathcal{X}}\subseteq\mathcal{X}$ and $\bar{\mathcal{E}}\subseteq\mathcal{E}$ (where all edges in $\bar{\mathcal{E}}$ are between vertices in $\bar{\mathcal{X}}$). A subgraph $\mathcal{H}$ of $\mathcal{G}$ is said to be *induced* if $(x_i,x_j)\in\bar{\mathcal{E}}\Leftrightarrow(x_i,x_j)\in\mathcal{E}$ whenever $x_i,x_j\in\bar{\mathcal{X}}$.

A *path* P from vertex x_{i_0} to vertex x_{i_t} is a sequence of vertices $x_{i_0},x_{i_1},\ldots,x_{i_t}$ such that $(x_{i_j},x_{i_{j+1}})\in\mathcal{E}$ for $0\leq j\leq t-1$. A path is called a *cycle* if its start vertex and end vertex are the same, and no other vertex appears more than once in the path. Paths P_1 and P_2 are *vertex-disjoint* if they have no vertices in common. Paths P_1 and P_2 are *internally vertex-disjoint* if they are vertex-disjoint, with the possible exception of the end vertices. A set of paths $P_1,P_2,\ldots,P_r$ are (internally) vertex-disjoint if the paths are pairwise (internally) vertex-disjoint. Given two subsets $\mathcal{X}_1,\mathcal{X}_2\subset\mathcal{X}$, an *$r$-linking* from $\mathcal{X}_1$ to $\mathcal{X}_2$ is a set of r vertex-disjoint paths, each with start vertex in $\mathcal{X}_1$ and end vertex in $\mathcal{X}_2$. Note that if $\mathcal{X}_1$ and $\mathcal{X}_2$ are not disjoint, we will take their common vertices to be vertex-disjoint paths between $\mathcal{X}_1$ and $\mathcal{X}_2$ of length zero.

A graph is said to be *strongly connected* if there is a path from vertex x_j to vertex x_i for every $x_i,x_j\in\mathcal{X}$. We will call a graph *disconnected* if there exists at least one pair of vertices $x_i,x_j\in\mathcal{X}$ such that there is no path from x_j to x_i. A *vertex-cut* in a graph is a subset $\mathcal{S}\subset\mathcal{X}$ such that removing the vertices in $\mathcal{S}$ (and the associated edges) from the graph causes the remaining graph to be disconnected. More specifically, a (j,i)-cut in a graph is a subset $\mathcal{S}_{ji}\subset\mathcal{X}$ such that removing the vertices in $\mathcal{S}_{ji}$ (and the associated edges) from the graph causes the graph to have no paths from vertex x_j to vertex x_i. We will denote the smallest size of a (j,i)-cut by κ_{ji}. If $(x_j,x_i)\in\mathcal{E}$ (i.e., vertex x_j is a neighbor of vertex x_i), we will take κ_{ji} to be infinite (since removing other vertices will not remove the direct path between x_j and x_i). We will also adopt the convention that κ_{ii} is infinite. Note that if $\min_j\kappa_{ji}$ is finite, then the in-degree of vertex x_i must be at least $\min_j\kappa_{ji}$ (since otherwise, removing all the neighbors of vertex x_i would disconnect the graph, thereby producing a (j,i)-cut of size less than $\min_j\kappa_{ji}$). Note that there are various efficient algorithms for computing the quantity κ_{ji} for any vertices x_i and x_j, such as the Ford-Fulkerson algorithm (which has run-time polynomial in the number of vertices) [16]. The *connectivity* of the graph is defined as $\min_{i,j}\kappa_{ij}$. The following classical result will play an important role in our derivations (e.g., see [16]).

Lemma 1 (Fan Lemma): Let x_i be a vertex in graph $\mathcal{G}$, and let c be a nonnegative integer such that $\kappa_{ji}\geq c$ for all j. Let $\mathcal{R}\subset\mathcal{X}$ be any subset of the vertices, with $|\mathcal{R}|=c$. Then there exists a set of c internally vertex-disjoint paths from $\mathcal{R}$ to x_i, where the only common vertex of each of these paths is x_i. □

Since all internally vertex-disjoint paths have to pass through different neighbors of x_i, the Fan Lemma implies that there will be a c-linking from $\mathcal{R}$ to $\mathcal{N}_i\cup\{x_i\}$. Note that some of the paths in this linking might have zero length (i.e., if x_i or some of its neighbors are in $\mathcal{R}$).

B. Distributed System Model

The interaction constraints in distributed systems and networks can be modeled via a directed graph $\mathcal{G}=\{\mathcal{X},\mathcal{E}\}$, where $\mathcal{X}=\{x_1,x_2,\ldots,x_N\}$ is the set of N nodes in the system

and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ represents the communication constraints in the network (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information directly from node x_j). Note that undirected graphs can be readily handled by treating each undirected edge as two directed edges.

We will deal with networks where information is disseminated via the *wireless broadcast* model, whereby each node (behaving or misbehaving) sends the same information to all of its neighbors. This model, while obviously applicable to wireless networks, also holds when information is obtained by *direct sensing* (i.e., where each node measures or senses the values of its neighbor, as opposed to receiving that value through a transmission). We assume that every node in the network has an identifier (so that nodes can associate each piece of information that they sense or receive with the corresponding neighbor). Each node is also assumed to have sufficient memory[1] (so that it can store the information that it receives or senses from its neighbors), and sufficient computational capability to perform mathematical operations on this stored information (such as calculating the rank of a matrix, multiplying matrices, etc.). We will assume either that nodes always transmit a value (even if they are faulty), or that messages are delivered in a fixed amount of time. This assumption is necessary because it would otherwise be impossible to perform fault-diagnosis – the receiving node will never be able to determine whether an expected message from another node is simply delayed, or if the transmitting node has failed [4]. When running a specified algorithm, we assume that nodes in the network wait until they have received transmissions from all of their neighbors, and then execute their transmission or update strategies before waiting for the next transmissions from their neighbors. We will capture this behavior by referring to the behavior of a node at *time-step* k, by which we mean the k–th transmission or update step executed by that node. We assume that all messages are either delivered in the order they were transmitted, or have an associated time-stamp or sequence number to indicate the order of transmission.

C. Function Calculation via Linear Iterations

Suppose that each node x_i has some initial value, given by $x_i[0]$. The goal is for certain nodes to gather enough information to calculate certain functions that depend on (some of) these initial values. To achieve this, nodes can update and/or exchange their values over several time-steps based on some strategy that adheres to the constraints imposed by the network topology. A common strategy is to "flood" the network with the needed information (where each node simply forwards any message that it gets from a neighbor to all of its neighbors) [4] or to use routing algorithms to route the value of each node to the nodes that need it [17]. Motivated by gossip and consensus algorithms that adopt nearest neighbor rules (e.g., [8]), the scheme that we study in this paper makes use of linear iterations; specifically, at each time-step (or iteration) k, each node updates its value as

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{x_j \in \mathcal{N}_i} w_{ij}x_j[k] \ , \qquad (1)$$

where the w_{ij} form a set of weights.[2] In other words, each node updates its value to be a linear combination of its own value and the values of its neighbors. For ease of analysis, the values of all nodes at time-step k can be aggregated into the value vector $\mathbf{x}[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}'$, and the update strategy for the entire system can be represented as

$$\mathbf{x}[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \mathbf{x}[k] \qquad (2)$$

for $k \in \mathbb{N}$, where $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i \cup \{x_i\}$.

Definition 1: Let $g : \mathbb{R}^N \mapsto \mathbb{R}^q$ be a function of the initial values of the nodes (note that $g(\cdot)$ will be a vector-valued function if $q \geq 2$). We say $g(x_1[0], x_2[0], \ldots, x_N[0])$ is *calculable by node* x_i if it can be calculated by node x_i after running the linear iteration for a sufficiently large number of time-steps, perhaps by using all the values that it sees over those time-steps. □

While there has been a great deal of work on choosing the weight matrix $\mathbf{W}$ for the linear iteration (2) in order to achieve objectives ranging from asymptotic consensus (e.g., [8]) to calculating arbitrary functions in finite time [13], there has been little investigation of what happens when some nodes do not follow the linear iterative strategy. In this paper, we will show that, under certain fundamental conditions on the underlying network topology (and assuming the nodes know the weight matrix $\mathbf{W}$), linear iterative schemes are also powerful enough to allow the nodes in the network to distributively calculate any arbitrary function *even when* some of the nodes in the network are malicious or faulty.

D. Previous Results on Fault Tolerant Function Calculation

The problem of transmitting information over networks (and specifically, reaching consensus) in the presence of faulty or malicious nodes has been studied thoroughly over the past several decades (e.g., see [4], [6] and the references therein). It is known in the literature that if there are only $2f$ vertex-disjoint paths between two nodes in a network, then there exists at least one set of f coordinated malicious nodes that can prevent at least one of the nodes from receiving any information about the other (*regardless* of the algorithm). The proofs of this general result are non-constructive,[3] and based on the following intuition: if there are only $2f$ independent paths between the two nodes, then a set of malicious nodes on f of the paths could collude to pretend that the transmitted

[1] As we will see later in the paper, the amount of memory required by each node will increase as a function of the number of nodes N in the network.

[2] The methodology for choosing the weights appropriately and the implications of this choice are discussed later in the paper.

[3] Indeed, the precise strategy used by the malicious nodes to disrupt the network will depend on the particular algorithm that is used to disseminate information.

value was something other than the true value, and the receiving node would not know whether to believe the f malicious nodes, or the f nodes transmitting the true value on the other f independent paths [4], [6], [18].

The topic of overcoming malicious behavior in distributed settings has been extensively studied in the computer science literature under the moniker of the *Byzantine Generals Problem* (e.g., [4], [19]). That problem deals with ensuring that all nodes in the network reach agreement on a certain value, even when there are up to f malicious (Byzantine) nodes that are allowed to send different values to different neighbors (i.e., the entire network, including both behaving and misbehaving nodes, are assumed to operate under a wired communication model). In this situation, it has been established that the necessary and sufficient conditions to tolerate f Byzantine nodes is that (1) the total number of nodes N in the network satisfies $N \geq 3f+1$ and (2) the connectivity of the network is at least $2f+1$. The second condition is necessary in order to reliably exchange information between two nodes in the network. By itself, the second condition is not sufficient under the wired communication model, since if node x_i receives node x_j's value reliably, it still does not know what x_j told other nodes in the network. This is where the first condition comes in: there must be a sufficient number of non-Byzantine nodes in the network in order for x_i to ascertain what x_j told 'most' of the nodes. When one considers a wireless communication model (as is the case with the linear iterative strategy), the first condition (i.e., $N \geq 3f+1$) is no longer necessary, since the malicious nodes can no longer provide conflicting information to their neighbors. However, the second condition (i.e., connectivity at least $2f+1$) is still necessary, since the problem of reliably communicating information between two nodes remains, as described above (e.g., see also [18], [20]).

As mentioned in the previous section, the vulnerability of linear iterative function-calculation schemes to malicious behavior has not received much attention in the literature.[4] In [11], it was shown that if a node in the network does not update its value at each time-step (i.e., it maintains a constant value), then a particular class of linear iterative strategies (where each node updates its value to be a convex combination of its neighborhood values) will cause all other nodes to asymptotically converge to the value of that node. A similar analysis was done in [22], where it was argued that since the asymptotic consensus scheme can be disrupted by a single node that maintains a constant value, it can also be disrupted by a single node that updates its values arbitrarily (since maintaining a constant value is a special case of arbitrary updates). Both of these works only considered a straightforward application of the linear iteration for asymptotic consensus, without making explicit use of the information available to the nodes via the iteration.

The problem was revisited in [23] for the specific case where all nodes have to reach agreement (not necessarily on any specific function of the initial values), and where the network contains only one malicious node that does not behave in a completely arbitrary manner (specifically, the additive errors introduced by the malicious node in that setting are not allowed to asymptotically decay faster than a certain rate). Under these special conditions, [23] showed that if every node in the system uses the information it receives from the linear iteration to try and estimate the malicious updates injected into the network (via an *unknown input observer*), then a graph connectivity of two is sufficient to detect and isolate the malicious node, and have the other nodes reach agreement. This result shows that linear iterative strategies actually have some degree of robustness against malicious nodes if one uses the information available to each node appropriately. It is also worth noting that although the result in [23] seems to contradict the intuition that only one malicious node should be required to disrupt a 2-connected network, this discrepancy arises due to the fact that updates applied by the malicious nodes in [23] are *restricted* to a certain class.

In this paper, we provide a comprehensive analysis of linear iterative strategies in the presence of malicious nodes. First, we show exactly how f malicious nodes (with the ability to update their values in a coordinated and arbitrary manner) should behave in a network of $2f$ connectivity in order to disrupt the linear iterative strategy. Next, we show that linear iterative strategies are able to achieve the minimum bound required to disseminate information reliably; specifically, when node x_i has $2f+1$ or more paths from the other nodes in the network, f malicious nodes will be unable to prevent x_i from calculating any function of the initial values (under the broadcast model of communication).[5]

III. System Model and Main Results

A. Modeling Malicious Behavior

Suppose the objective in the system is for node x_i to calculate $g_i(x_1[0], x_2[0], \ldots, x_N[0])$, for some function $g_i : \mathbb{R}^N \rightarrow \mathbb{R}^{q_i}$ (note that different nodes can be required to calculate different functions). When there are no malicious nodes in the network, we noted in the last section that this can be accomplished by having the nodes run the linear iteration $\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k]$ with an appropriate weight matrix $\mathbf{W}$ for a finite number of time-steps (e.g., following the method in [13]). Suppose, however, that instead of applying the update equation (1), some nodes update their values incorrectly; for example, node l updates its value at each time-step as

$$x_l[k+1] = w_{ll}x_l[k] + \sum_{x_j \in \mathcal{N}_l} w_{lj}x_j[k] + u_l[k] \ , \qquad (3)$$

where $u_l[k]$ is an additive error at time-step k.

[4] It is worth mentioning that linear iterative strategies have also been studied recently in the communications community for the purpose of transmitting streams of values through networks, under the moniker of *network coding* (e.g., [5]). In these cases, the network is assumed to contain a source node that is trying to send information at a certain *rate* to some destination nodes, and network codes have been developed to maximize this rate in the presence of malicious attackers [21]. The key difference in our work is that we are interested in the problem where (potentially) all nodes have a single initial value, and certain nodes want to calculate functions of these values, as opposed to recovering a new source value at each time-step.

[5] Preliminary versions of these results were presented in [24], [25], where the problem of allowing *every* node in the network to recover all of the initial values was studied. This paper generalizes that work by presenting necessary and sufficient conditions under which any node (or any subset of nodes) can obtain all of the initial values.

Definition 2: Suppose all nodes run the linear iteration for T time-steps in order to perform function calculation. Node x_l is said to be *malicious* (or *faulty*) if $u_l[k]$ is nonzero for at least one time-step k, $0 \leq k \leq T-1$. □

Note that the fault model considered here is extremely general, and allows node x_l to update its value in a completely arbitrary manner (via appropriate choices of the error $u_l[k]$ at each time-step). As such, this fault model is capable of encapsulating a wide variety of faults, under the condition that each malicious node sends the same value to all of its neighbors.[6] For example, suppose a node x_l in the network exhibits a stopping failure, whereby it stops transmitting information (so that the neighbors of node x_l simply receive the value zero from node x_l at each time-step). This stopping failure can be captured by our fault model by selecting the error $u_l[k]$ at each time-step to set node x_l's state $x_l[k+1]$ to zero. Another example of a fault is when the link from node x_j to node x_l drops out at time-step k. One can capture these errors in our fault model by simply selecting $u_l[k]$ to cancel out the term $w_{lj}x_j[k]$ in the update equation for node x_l. Note that if the above errors are not accidental, but intentional, the corresponding node will be called malicious. More generally, one could have multiple faulty and/or malicious nodes (the latter could be coordinating their actions to disrupt the network). In the rest of the paper, we will be using the terms faulty and malicious interchangeably. We will assume that malicious nodes are omniscient, and know the network topology along with the weights that are being used by the nodes in the network. They are also allowed to know all of the initial values in the network; as we will show, when the connectivity of the network is smaller than a certain value, this information is *not* required by the malicious nodes in order to disrupt the network. In fact, the malicious nodes will only need to agree initially on a certain vector, and then thereafter update their values in a manner that is consistent with the weight matrix $\mathbf{W}$, but that does not require further communication or coordination among them. However, we will also show that when the connectivity is sufficiently high, the malicious nodes will not be able to disrupt the network *even when* they have access to all of the above information (including all of the initial values in the network).

Note that a set of malicious nodes can obviously try to influence the result of a computation by changing their own *initial* values. Of course, if all initial values are equally valid, then it will be impossible for *any* algorithm to detect this type of malicious behavior, since any initial value that a malicious node chooses for itself is a legitimate value that could also have been chosen by a node that is functioning correctly. On the other hand, if there is a statistical distribution on the initial values, techniques that exploit these relationships among the initial values can potentially be used to identify outliers due to malicious nodes and eliminate their influence on the system [26]. We will not discuss the issue of verifying initial values further in our work because of its philosophically different nature, and because of the fact that our problem formulation remains valid in cases where malicious nodes do not contribute initial values (i.e., they function as routers, and the functions calculated in the network do not depend on the initial values of these nodes). Instead, our goal is to avoid the more pernicious case where malicious nodes spread confusion about the initial values of *non-malicious* nodes. For example, consider a distributed system consisting of agents that are trying to vote 1 ('yes') or 0 ('no') on a course of action. The agents wish to reach consensus on the majority vote in a distributed manner (i.e., by exchanging their values via other agents in the network). In this scenario, both 0 and 1 are equally valid votes for every agent, and so it is not important whether malicious agents change their own initial value (i.e., their vote). What *is* important, however, is to ensure that votes are not changed as they are passed through other agents in the network; in other words, the malicious nodes should not be able to affect the result of the majority vote other than by choosing their own vote for the course of action.

[6]This is not an impediment or limitation to the behavior of the malicious nodes, in the sense that the correctly functioning nodes are also required to operate under this condition. In other words, the malicious nodes are allowed to behave in an arbitrary manner within the confines of the communication modality for the entire system.

B. Modeling the Values Seen by Each Node

Let $\mathcal{F} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_f}\}$ denote the set of nodes that are malicious during a run of the linear iteration. Combining (3) with (2), the linear iteration can then be modeled as

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{W}\mathbf{x}[k] + \underbrace{\begin{bmatrix}\mathbf{e}_{i_1,N} & \mathbf{e}_{i_2,N} & \cdots & \mathbf{e}_{i_f,N}\end{bmatrix}}_{\mathbf{B}_{\mathcal{F}}} \underbrace{\begin{bmatrix} u_{i_1}[k] \\ u_{i_2}[k] \\ \vdots \\ u_{i_f}[k] \end{bmatrix}}_{\mathbf{u}_{\mathcal{F}}[k]} \\ \mathbf{y}_i[k] &= \mathbf{C}_i\mathbf{x}[k], \quad 1 \leq i \leq N \ . \end{aligned} \tag{4}$$

Here, $\mathbf{C}_i$ is a $(\deg_i + 1) \times N$ matrix with a single 1 in each row denoting the positions of the state-vector $\mathbf{x}[k]$ that are available to node x_i (i.e., these positions correspond to nodes that are neighbors of node x_i, along with node x_i itself). Thus, the vector $\mathbf{y}_i[k]$ denotes the set of outputs (or node values) seen by node x_i during time-step k of the linear iteration. The set $\mathcal{F}$ is unknown to the (non-malicious) nodes in the network, and thus the exact matrix $\mathbf{B}_{\mathcal{F}}$ is also unknown. However, the nodes do know that it is a matrix with a single 1 in the rows corresponding to the malicious nodes, and zeros elsewhere (recall that $\mathbf{e}_{l,N}$ denotes a vector of length N with a single nonzero entry with value 1 in its l–th position). Note that the above model readily captures the ability of the malicious nodes to cooperatively update their values (i.e., they can choose the vector $\mathbf{u}_{\mathcal{F}}[k]$) in order to disrupt the network.

The set of all values seen by node x_i during the first $L+1$ time-steps of the linear iteration (for any nonnegative integer L) is given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:L-1], \tag{5}$$

where $\mathbf{y}_i[0:L] = \begin{bmatrix}\mathbf{y}_i'[0] & \mathbf{y}_i'[1] & \mathbf{y}_i'[2] & \cdots & \mathbf{y}_i'[L]\end{bmatrix}'$ and $\mathbf{u}_{\mathcal{F}}[0:L-1] = \begin{bmatrix}\mathbf{u}_{\mathcal{F}}'[0] & \mathbf{u}_{\mathcal{F}}'[1] & \mathbf{u}_{\mathcal{F}}'[2] & \cdots & \mathbf{u}_{\mathcal{F}}'[L-1]\end{bmatrix}'$.

The matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}}$ can be expressed recursively as

$$\mathcal{O}_{i,L} = \begin{bmatrix} \mathbf{C}_i \\ \mathcal{O}_{i,L-1}\mathbf{W} \end{bmatrix}, \ \mathcal{M}_{i,L}^{\mathcal{F}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathcal{O}_{i,L-1}\mathbf{B}_{\mathcal{F}} & \mathcal{M}_{i,L-1}^{\mathcal{F}} \end{bmatrix}, \tag{6}$$

where $\mathcal{O}_{i,0} = \mathbf{C}_i$ and $\mathcal{M}_{i,0}^{\mathcal{F}}$ is the empty matrix (with zero columns). These matrices will characterize the ability of node x_i to calculate the required function of the initial values; the matrix $\mathcal{O}_{i,L}$ has the form of the *observability matrix* for the pair $(\mathbf{W}, \mathbf{C}_i)$, and we will call $\mathcal{M}_{i,L}^{\mathcal{F}}$ the *invertibility matrix*[7] for the triplet $(\mathbf{W}, \mathbf{B}_{\mathcal{F}}, \mathbf{C}_i)$. We will call upon the following simple lemma (the proof of which is omitted in the interest of space).

Lemma 2: Let $\mathcal{F}_1$ and $\mathcal{F}_2$ denote two subsets of $\mathcal{X}$, and define $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. Then, the column space of $\mathcal{M}_{i,L}^{\mathcal{F}}$ is the same as the column space of $\begin{bmatrix} \mathcal{M}_{i,L}^{\mathcal{F}_1} & \mathcal{M}_{i,L}^{\mathcal{F}_2} \end{bmatrix}$ for any nonnegative integer L. □

C. Main Result of the Paper

Our goal in this paper will be to study the robustness of linear-iterative strategies to malicious or faulty behavior by a subset of nodes in the network. Specifically, over the remainder of the paper, we demonstrate the following key result.

Theorem 1: Given a fixed network with N nodes described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let f denote the maximum number of malicious nodes that are to be tolerated in the network, and let κ_{ji} denote the size of the smallest (j,i)-cut between any two vertices x_j and x_i. Then, regardless of the actions of the malicious nodes, node x_i can uniquely determine all of the initial values in the network via a linear iterative strategy if and only if $\min_j \kappa_{ji} \geq 2f+1$. Furthermore, if this condition is satisfied, x_i will be able to recover the initial values after the nodes run the linear iterative strategy with almost any choice of weights for at most N time-steps. □

Due to the fact that the linear iterative strategy only requires that the connectivity be $2f+1$ in order to reliably disseminate information, and since this bound is fundamental for information dissemination under any algorithm (as discussed in Section II-D), the above theorem shows that linear iterative strategies are as powerful as any other strategy for information dissemination in the presence of malicious agents (under the wireless broadcast model of communication).

Remark 1: In order for a node x_i to overcome malicious behavior, we will assume that it knows its observability matrix $\mathcal{O}_{i,L}$ for some sufficiently large L (upper bounded by the size of the network, as we will show later in the paper). It is sufficient (but not necessary) for node x_i to know the weight matrix $\mathbf{W}$ in order for it to obtain the observability matrix. This assumption of some (or full) knowledge of the network topology via the observability matrix (or the weight matrix) is similar to the assumptions made by much of the literature on fault-tolerant distributed systems [6], [4], [19], and we will adopt it here to demonstrate the resilience of linear iterative strategies. As described in [13], it is possible for the nodes in the network to distributively determine their observability matrices when there are no malicious nodes; the extension of this result to networks with malicious nodes is an area for future research.

Note also that node x_i does not necessarily need to know precisely how many nodes were malicious during the linear iteration; it only needs to know that the number of malicious nodes is upper bounded by f. If node x_i does not know the value of f *a priori*, but does know the network topology (e.g., via the weight matrix $\mathbf{W}$), it can determine the maximum number of malicious nodes that it can tolerate by calculating the size of the minimum vertex-cut between itself and any other node x_j, and then making the assumption that the actual number of malicious nodes does not exceed this maximum value. □

IV. Attacking the Linear Iterative Strategy

The lower bound of $2f+1$ in Theorem 1 is established by existing results on fault-tolerant function calculation, as described in Section II-D. However, since those results are non-constructive, we will start by providing an explicit strategy for the malicious nodes to follow to disrupt the linear iterative strategy. To do this, consider node x_i in the network $\mathcal{G}$, and let node x_j be any other node that is not a neighbor of node x_i. Let $\mathcal{F}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|\mathcal{F}_1|}}\}$ and $\mathcal{F}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|\mathcal{F}_2|}}\}$ denote disjoint sets of vertices such that $\mathcal{F}_1 \cup \mathcal{F}_2$ forms a (j,i)-cut of $\mathcal{G}$. Let $\mathcal{H}$ denote the set of all nodes that have a path to node x_i in the graph induced by $\mathcal{X} \setminus (\mathcal{F}_1 \cup \mathcal{F}_2)$ (including node x_i), and let $\bar{\mathcal{H}} = \mathcal{X} \setminus (\mathcal{H} \cup \mathcal{F}_1 \cup \mathcal{F}_2)$.

Theorem 2: For any nonnegative integer L, the column space of the matrix $\mathcal{O}_{i,L}\mathbf{B}_{\bar{\mathcal{H}}}$ is contained in the column space of the matrix $\begin{bmatrix} \mathcal{M}_{i,L}^{\mathcal{F}_1} & \mathcal{M}_{i,L}^{\mathcal{F}_2} \end{bmatrix}$ (where $\mathbf{B}_{\bar{\mathcal{H}}}$ is defined in equation (4)). □

Proof: Let $\mathbf{x}_{\mathcal{H}}[k]$, $\mathbf{x}_{\mathcal{F}_1}[k]$, $\mathbf{x}_{\mathcal{F}_2}[k]$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vectors of values of nodes in sets $\mathcal{H}$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\bar{\mathcal{H}}$, respectively. Let n denote the number of nodes in set $\bar{\mathcal{H}}$ (i.e., $\mathbf{x}_{\bar{\mathcal{H}}}[k] \in \mathbb{R}^n$). Note that $x_i[k]$ is contained in $\mathbf{x}_{\mathcal{H}}[k]$, $x_j[k]$ is contained in $\mathbf{x}_{\bar{\mathcal{H}}}[k]$, and that the sets $\mathcal{H}, \mathcal{F}_1, \mathcal{F}_2$, and $\bar{\mathcal{H}}$ form a partition of the set of nodes $\mathcal{X}$. Assume without loss of generality that the vector $\mathbf{x}[k]$ in (4) is of the form $\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{F}_1}[k] & \mathbf{x}'_{\mathcal{F}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k] \end{bmatrix}'$ (it can always be put into this form via an appropriate permutation of the node indices). Then, since no node in set $\mathcal{H}$ has an incoming edge from any node in set $\bar{\mathcal{H}}$, the weight matrix for the linear iteration must necessarily have the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \mathbf{W}_{13} & \mathbf{0} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \mathbf{W}_{23} & \mathbf{W}_{24} \\ \mathbf{W}_{31} & \mathbf{W}_{32} & \mathbf{W}_{33} & \mathbf{W}_{34} \\ \mathbf{W}_{41} & \mathbf{W}_{42} & \mathbf{W}_{43} & \mathbf{W}_{44} \end{bmatrix}. \tag{7}$$

The $\mathbf{C}_i$ matrix in (4) for node x_i must be of the form $\mathbf{C}_i = \begin{bmatrix} \mathbf{C}_{i,1} & \mathbf{C}_{i,2} & \mathbf{C}_{i,3} & \mathbf{0} \end{bmatrix}$, again because node x_i has no neighbors in set $\bar{\mathcal{H}}$. Furthermore, from the definition of the matrix $\mathbf{B}_{\mathcal{F}}$ in (4), note that this partitioning of nodes implies that we can write $\mathbf{B}_{\mathcal{F}_1} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{|\mathcal{F}_1|} & \mathbf{0} & \mathbf{0} \end{bmatrix}'$, $\mathbf{B}_{\mathcal{F}_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I}_{|\mathcal{F}_2|} & \mathbf{0} \end{bmatrix}'$, $\mathbf{B}_{\bar{\mathcal{H}}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n \end{bmatrix}'$ (for some ordering of the nodes in $\mathcal{F}_1, \mathcal{F}_2$ and $\bar{\mathcal{H}}$).

[7] This terminology is due to the fact that matrices of the form $\mathcal{M}_{i,L}^{\mathcal{F}}$ arise in the study of *dynamic system inversion*, where the objective is to recover a set of unknown inputs from the output of a linear system [27].

Using the recursive definition of $\mathcal{O}_{i,L}$ in (6), and the fact that $\mathbf{C}_i \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \mathbf{0}$, we obtain

$$\begin{aligned}\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} &= \begin{bmatrix} \mathbf{C}_i \\ \mathcal{O}_{i,L-1}\mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{W} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_1}\mathbf{W}_{24} + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\mathcal{F}_2}\mathbf{W}_{34} \\ &\quad + \begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-1} \end{bmatrix} \mathbf{B}_{\bar{\mathcal{H}}}\mathbf{W}_{44} \ .\end{aligned}$$

Continuing the above procedure recursively for matrices of the form $\begin{bmatrix} \mathbf{0} \\ \mathcal{O}_{i,L-\alpha} \end{bmatrix} \mathbf{B}_{\bar{\mathcal{H}}}$, $1 \le \alpha \le L$, we obtain (after using some algebra and the recursive definition of the matrices $\mathcal{M}_{i,L}^{\mathcal{F}_1}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_2}$ from equation (6))

$$\mathcal{O}_{i,L} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_n \end{bmatrix} = \mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix}. \tag{8}$$

This concludes the proof of the theorem. ■

Note that the above theorem applies to *any* decomposition of a vertex-cut into sets $\mathcal{F}_1$ and $\mathcal{F}_2$ (including the case where $\mathcal{F}_1$ is the entire vertex-cut and $\mathcal{F}_2 = \emptyset$, if desired). The benefit of framing the theorem in this general manner is that expression (8) will now allow us to show how a certain set of nodes $\mathcal{F}_1$ (or $\mathcal{F}_2$) can maliciously update their values in order to disrupt the network.

Lemma 3: Let the initial values of nodes in the set $\bar{\mathcal{H}}$ be denoted by the vector $\mathbf{a}$. For any other vector $\mathbf{b}$, suppose that the nodes in $\mathcal{F}_1$ are malicious and apply the error sequence $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b}-\mathbf{a})$, $k \in \mathbb{N}$. Then, the values $\mathbf{y}_i[k], k \in \mathbb{N}$ seen by node x_i are exactly the same as if $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$ and nodes in set $\mathcal{F}_2$ are malicious with $\mathbf{u}_{\mathcal{F}_2}[k] = \mathbf{W}_{34}\mathbf{W}_{44}^k(\mathbf{a}-\mathbf{b})$, $k \in \mathbb{N}$. □

Proof: As in the proof of Theorem 2, let $\mathbf{x}_{\mathcal{H}}[k], \mathbf{x}_{\mathcal{F}_1}[k]$, $\mathbf{x}_{\mathcal{F}_2}[k]$, and $\mathbf{x}_{\bar{\mathcal{H}}}[k]$ denote the vector of values of nodes in sets $\mathcal{H}$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\bar{\mathcal{H}}$, respectively, and assume (without loss of generality) that the vector $\mathbf{x}[k]$ in (4) is of the form $\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}'_{\mathcal{H}}[k] & \mathbf{x}'_{\mathcal{F}_1}[k] & \mathbf{x}'_{\mathcal{F}_2}[k] & \mathbf{x}'_{\bar{\mathcal{H}}}[k] \end{bmatrix}'$. Let n be the number of nodes in set $\mathcal{H}$, and let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ be arbitrary vectors.

Suppose the nodes in set $\mathcal{F}_1$ are malicious. From (5), the values seen by node x_i over $L+1$ time-steps are given by $\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{F}_1}\mathbf{u}_{\mathcal{F}_1}[0:L-1]$. From Theorem 2 (specifically, equation (8)), this expression can be written as

$$\begin{aligned}\mathbf{y}_i[0:L] &= \mathcal{O}_{i,L} \begin{bmatrix} \mathbf{x}_{\mathcal{H}}[0] \\ \mathbf{x}_{\mathcal{F}_1}[0] \\ \mathbf{x}_{\mathcal{F}_2}[0] \\ \mathbf{0} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_1}\mathbf{u}_{\mathcal{F}_1}[0:L-1] \\ &\quad + \left(\mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \right) \mathbf{x}_{\bar{\mathcal{H}}}[0].\end{aligned} \tag{9}$$

Suppose $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$, and that nodes in $\mathcal{F}_1$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b}-\mathbf{a})$. Substituting this into the expression for $\mathbf{y}_i[0:L]$ (with $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$), the values seen by node x_i under this fault scenario are given by

$$\begin{aligned}\mathbf{y}_i[0:L] &= \mathcal{O}_{i,L} \begin{bmatrix} \mathbf{x}_{\mathcal{H}}[0] \\ \mathbf{x}_{\mathcal{F}_1}[0] \\ \mathbf{x}_{\mathcal{F}_2}[0] \\ \mathbf{0} \end{bmatrix} + \mathcal{M}_{i,L}^{\mathcal{F}_1} \begin{bmatrix} \mathbf{W}_{24} \\ \mathbf{W}_{24}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{24}\mathbf{W}_{44}^{L-1} \end{bmatrix} \mathbf{b} \\ &\quad + \mathcal{M}_{i,L}^{\mathcal{F}_2} \begin{bmatrix} \mathbf{W}_{34} \\ \mathbf{W}_{34}\mathbf{W}_{44} \\ \vdots \\ \mathbf{W}_{34}\mathbf{W}_{44}^{L-1} \end{bmatrix} \mathbf{a}.\end{aligned} \tag{10}$$

Now suppose that nodes in $\mathcal{F}_2$ are malicious (instead of nodes in $\mathcal{F}_1$). Again, from (5) and Theorem 2, the values seen by node x_i over $L+1$ time-steps will be given by equation (9), except with $\mathcal{M}_{i,L}^{\mathcal{F}_1}\mathbf{u}_{\mathcal{F}_1}[0:L-1]$ replaced by $\mathcal{M}_{i,L}^{\mathcal{F}_2}\mathbf{u}_{\mathcal{F}_2}[0:L-1]$. If $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$, and nodes in $\mathcal{F}_2$ update their values at each time-step k with the error values $\mathbf{u}_{\mathcal{F}_2}[k] = \mathbf{W}_{34}\mathbf{W}_{44}^k(\mathbf{a}-\mathbf{b})$, one can verify that the set of values seen by node x_i under this fault scenario will be identical to the expression in (10), and thus the values received by node x_i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{a}$ and the nodes in $\mathcal{F}_1$ are malicious will be indistinguishable from the values seen by node x_i when $\mathbf{x}_{\bar{\mathcal{H}}}[0] = \mathbf{b}$ and the nodes in $\mathcal{F}_2$ are malicious. Since this holds for all nonnegative integers L, this fault scenario makes it impossible for node x_i (and in fact, any node in set $\mathcal{H}$) to obtain the initial values of any node in set $\bar{\mathcal{H}}$, or to identify[8] the malicious nodes. ■

Remark 2: The additive errors for nodes in set $\mathcal{F}_1$ to use in order to disrupt the linear iterative strategy are $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k(\mathbf{b}-\mathbf{a})$ for $k \in \mathbb{N}$; note that these nodes do not actually need to know the initial values of the nodes in set $\bar{\mathcal{H}}$ (taken to be $\mathbf{a}$ in the above expression) in order to apply this strategy. If they simply choose *any* random vector $\bar{\mathbf{a}}$, and update their values at each time-step as $\mathbf{u}_{\mathcal{F}_1}[k] = \mathbf{W}_{24}\mathbf{W}_{44}^k\bar{\mathbf{a}}$, then they are effectively applying the same update as above with $\mathbf{b} = \mathbf{a} + \bar{\mathbf{a}}$, without knowing the value of $\mathbf{a}$. The same reasoning holds if nodes in set $\mathcal{F}_2$ are malicious. □

Based on the result in Lemma 3, we are now ready to prove the following key theorem (re-establishing the "only if" part of Theorem 1).

Theorem 3: Given a network described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let κ_{ji} denote the size of the smallest (j,i)-cut between vertices x_j and x_i. If $\kappa_{ji} \le 2f$ for some positive integer f, then there exists a set $\mathcal{F}$ of f malicious nodes along with a set of additive errors $u_l[k], x_l \in \mathcal{F}, k \in \mathbb{N}$, such that node x_i cannot calculate any function that involves node x_j's initial value (regardless of the number of time-steps for which the iteration is performed). □

[8]It may be possible for nodes in set $\mathcal{X} \setminus \mathcal{H}$ to identify the malicious nodes (e.g., under the conditions described later in the paper), but they cannot reliably provide this information to the nodes in set $\mathcal{H}$, since any such information must once again pass through the nodes in sets $\mathcal{F}_1$ or $\mathcal{F}_2$, both of which are suspicious from the perspective of nodes in set $\mathcal{H}$.

Proof: In Lemma 3, we saw that if the union of the disjoint sets of vertices

$$\mathcal{F}_1 = \{x_{l_1}, x_{l_2}, \ldots, x_{l_{|\mathcal{F}_1|}}\},\ \ \mathcal{F}_2 = \{x_{h_1}, x_{h_2}, \ldots, x_{h_{|\mathcal{F}_2|}}\}$$

forms a (j,i)-cut, then node x_i cannot distinguish a particular set of errors by nodes in $\mathcal{F}_1$ from another set of errors by nodes in $\mathcal{F}_2$. Furthermore, these errors make it impossible for node x_i to obtain any information about the initial value of node x_j. Choose $\mathcal{F}_1$ and $\mathcal{F}_2$ such that $|\mathcal{F}_1| = \lfloor \frac{\kappa_{ji}}{2} \rfloor$ and $|\mathcal{F}_2| = \lceil \frac{\kappa_{ji}}{2} \rceil$. Since $\kappa_{ji} \leq 2f$, we have $\lfloor \frac{\kappa_{ji}}{2} \rfloor \leq f$ and $\lceil \frac{\kappa_{ji}}{2} \rceil \leq f$, and so $\mathcal{F}_1$ and $\mathcal{F}_2$ are both legitimate sets of malicious nodes (if one is interested in tolerating a maximum of f malicious nodes in the system). Choosing the set $\mathcal{F}$ of malicious nodes to be either the set $\mathcal{F}_1$ or $\mathcal{F}_2$ (with the corresponding additive errors) completes the proof of the theorem. ■

Remark 3: Clearly, the above theorem can easily be modified to show that errors by a set $\mathcal{F}_1$ of $f+t$ malicious nodes would be indistinguishable from errors by a set $\mathcal{F}_2$ of $f-t$ malicious nodes, for any $0 \leq t \leq f$. The reason for focusing on the case $t = 0$ is that we are interested in dealing with a *maximum number* of malicious nodes in the network. In other words, if we know that a malicious attacker can only target at most f nodes (e.g., due to the costs incurred by attacking a node), then we can disregard all cases where more than f nodes are potentially malicious. Thus, even though we can show that a set $\mathcal{F}_1$ of $f-t$ malicious nodes can masquerade as a set $\mathcal{F}_2$ of $f+t$ malicious nodes, the latter case can be discounted because of the fact that $f+t$ nodes are not likely to be malicious, and so we can potentially identify the set $\mathcal{F}_1$. On the other hand, when $t = 0$, $\mathcal{F}_1$ and $\mathcal{F}_2$ are equally valid (and likely) sets of f malicious nodes, and thus we cannot discount one over the other. □

V. Calculating Functions in the Presence of Malicious Behavior

In this section, we will establish the "if" portion of Theorem 1 and show that if $\kappa_{ji} \geq 2f+1$ for all j, it will be impossible for any set of f or fewer malicious nodes to prevent node x_i from calculating any function of the initial values in the system. We start our development with the following theorem, which provides a procedure for node x_i to calculate functions in the presence of malicious nodes.

Theorem 4: Suppose that there exists an integer L and a weight matrix $\mathbf{W}$ such that, for all possible sets $\mathcal{J} \subset \mathcal{X}$ of $2f$ nodes, the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{J}}$ for node x_i satisfy

$$\text{rank}\left(\begin{bmatrix}\mathcal{O}_{i,L} & \mathcal{M}_{i,L}^{\mathcal{J}}\end{bmatrix}\right) = N + \text{rank}\left(\mathcal{M}_{i,L}^{\mathcal{J}}\right) \ . \qquad (11)$$

Then, if the nodes run the linear iteration for $L+1$ time-steps with the weight matrix $\mathbf{W}$, node x_i can calculate any arbitrary function of the initial values $x_1[0], x_2[0], \ldots, x_N[0]$, even when up to f nodes are malicious. □

Proof: Let $\mathbf{W}$ be a weight matrix that satisfies the conditions in the above theorem, and let the nodes run the linear iteration for $L+1$ time-steps. Suppose that the malicious nodes during the linear iteration are a subset of the set $\mathcal{F} = \{x_{j_1}, x_{j_2}, \ldots, x_{j_f}\}$. From (5), the values seen by node x_i over $L+1$ time-steps are given by

$$\mathbf{y}_i[0:L] = \mathcal{O}_{i,L}\mathbf{x}[0] + \mathcal{M}_{i,L}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:L-1] \ . \qquad (12)$$

Let $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{\binom{N}{f}}$ denote all possible sets of f nodes, and let $\mathcal{M}_{i,L}^{\mathcal{F}_1}, \mathcal{M}_{i,L}^{\mathcal{F}_2}, \ldots, \mathcal{M}_{i,L}^{\mathcal{F}_{\binom{N}{f}}}$ denote the corresponding invertibility matrices. With these matrices in hand,[9] suppose node x_i finds the first $j \in \{1, 2, \ldots, \binom{N}{f}\}$ such that the vector $\mathbf{y}_i[0:L]$ is in the column space of the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_j}$. This means that node x_i can find vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_j}[0:L-1]$ such that $\mathcal{O}_{i,L}\bar{\mathbf{x}} + \mathcal{M}_{i,L}^{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[0:L-1] = \mathbf{y}_i[0:L]$. Equating this to (12) and rearranging, we have

$$\mathcal{O}_{i,L}(\mathbf{x}[0]-\bar{\mathbf{x}}) + \mathcal{M}_{i,L}^{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[0:L-1] - \mathcal{M}_{i,L}^{\mathcal{F}_j}\mathbf{u}_{\mathcal{F}_j}[0:L-1] = \mathbf{0}.$$

Letting $\mathcal{J} = \mathcal{F} \cup \mathcal{F}_j$, we note from Lemma 2 that the above expression can be written as

$$\mathcal{O}_{i,L}(\mathbf{x}[0] - \bar{\mathbf{x}}) + \mathcal{M}_{i,L}^{\mathcal{J}}\mathbf{u}_{\mathcal{J}}[0:L-1] = \mathbf{0} \ ,$$

for some appropriately defined vector $\mathbf{u}_{\mathcal{J}}[0:L-1]$. From equation (11) in the statement of the theorem, the observability matrix is assumed to be of full column rank, and all its columns are linearly independent of the columns of the invertibility matrix $\mathcal{M}_{i,L}^{\mathcal{J}}$ (since the set $\mathcal{J}$ has $2f$ or fewer nodes). This means that $\bar{\mathbf{x}} = \mathbf{x}[0]$ in the above expression, and thus node x_i has recovered the entire initial value vector $\mathbf{x}[0]$, despite the efforts of the nodes in $\mathcal{F}$. This concludes the proof of the theorem. ■

Remark 4: The above procedure requires node x_i to check up to $\binom{N}{f}$ possibilities in order to determine the potential set of malicious nodes. Since $\left(\frac{N}{f}\right)^f \leq \binom{N}{f} \leq \left(\frac{eN}{f}\right)^f$ [28], the complexity of decoding increases at most exponentially with f (if N is fixed) and polynomially with N (if f is fixed). It is worth noting that this procedure is equivalent to the brute force method of determining up to f errors in an N-dimensional real-number codeword with distance $2f+1$. In coding theory, there exist efficient ways of performing this check for both structured and random real number codes (e.g., see [29] and the references therein), and one can potentially exploit those results to streamline the procedure in the proof of Theorem 4 (this is left for future research). □

Remark 5: Note that if transmissions between nodes are corrupted by noise, then the vector $\mathbf{y}_i[0:L]$ might not fall strictly within the column space of the matrices $\mathcal{O}_{i,L}$ and $\mathcal{M}_{i,L}^{\mathcal{F}_j}$ for any j. In particular, the line between noise and malicious errors becomes blurred as the magnitude of the noise increases, which makes it harder to detect and isolate malicious behavior. This issue has been investigated in the context of real-number error correcting codes in [29], and we will leave connections to such work for future research. □

In the sequel, we will show that when $\kappa_{ji} \geq 2f+1$ for all j, one can find a weight matrix $\mathbf{W}$ and an integer L so that

[9] Note from equation (6) that the columns of the invertibility matrix $\mathcal{M}_{i,L}^{\mathcal{F}_j}$ are simply a subset of the columns of $\mathcal{O}_{i,0}, \mathcal{O}_{i,1}, \ldots, \mathcal{O}_{i,L-1}$. In other words, node x_i can obtain the invertibility matrices simply from the knowledge of the matrix $\mathcal{O}_{i,L}$, and therefore does not necessarily need to store the invertibility matrices for every possible set of f nodes.

all columns of the observability matrix $\mathcal{O}_{i,L}$ will be linearly independent of each other, and of the columns in $\mathcal{M}_{i,L}^{\mathcal{J}}$, where $\mathcal{J}$ is any set of up to $2f$ nodes (i.e., equation (11) will be satisfied). From Theorem 4, node x_i will therefore be able to obtain all initial values from the outputs that it sees during the course of the linear iteration, and can calculate any arbitrary function of those values. To find such a weight matrix, we will first require some concepts from classical control theory.

A. Strong Observability

Consider a linear system of the form

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k] \ , \end{aligned} \tag{13}$$

with state vector $\mathbf{x} \in \mathbb{R}^N$, input $\mathbf{u} \in \mathbb{R}^m$, and output $\mathbf{y} \in \mathbb{R}^p$, with $p \geq m$. When the values of the inputs at each time-step are completely unknown and arbitrary, systems of the above form are termed *linear systems with unknown inputs* [30], [27]. For such systems, the following notions of *strong observability*, *matrix pencils* and *invariant zeros* have been established in the literature (e.g., see [30], [31], [32], [33]).

Definition 3 (Strong Observability): A linear system with unknown inputs (of the form (13)) is said to be *strongly observable* if $\mathbf{y}[k] = 0$ for all k implies $\mathbf{x}[0] = 0$ (regardless of the values of the unknown inputs $\mathbf{u}[k]$). □

Definition 4 (Matrix Pencil): For the linear system (13), the matrix $\mathbf{P}(z) = \left[\begin{smallmatrix} \mathbf{A}-z\mathbf{I}_N & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{smallmatrix}\right]$ is called the *matrix pencil* of the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. □

Definition 5 (Normal-Rank): The *normal-rank* of the matrix pencil $\mathbf{P}(z)$ is defined as $\text{rank}_n(\mathbf{P}(z)) \equiv \max_{z_0 \in \mathbb{C}} \text{rank}(\mathbf{P}(z_0))$. □

Definition 6 (Invariant Zero): The complex number $z_0 \in \mathbb{C}$ is called an *invariant zero* of the system (13) if $\text{rank}\,(\mathbf{P}(z_0)) < \text{rank}_n(\mathbf{P}(z))$. □

The following theorem provides a characterization of the strong observability of a system.

Theorem 5 ([30], [31], [32], [34]): The following statements are equivalent:

- The system (13) is strongly observable.
- The set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ has no invariant zeros.
- Denoting the output of the system (13) over N time-steps by

$$\underbrace{\begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \\ \vdots \\ \mathbf{y}[N-1] \end{bmatrix}}_{\mathbf{y}[0:N-1]} = \underbrace{\begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{N-1} \end{bmatrix}}_{\mathcal{O}_{N-1}} \mathbf{x}[0] + \underbrace{\begin{bmatrix} \mathbf{D} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{CB} & \mathbf{D} & \cdots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{N-2}\mathbf{B} & \mathbf{CA}^{N-3}\mathbf{B} & \cdots & \mathbf{D} \end{bmatrix}}_{\mathcal{M}_{N-1}} \underbrace{\begin{bmatrix} \mathbf{u}[0] \\ \mathbf{u}[1] \\ \mathbf{u}[2] \\ \vdots \\ \mathbf{u}[N-1] \end{bmatrix}}_{\mathbf{u}[0:N-1]},$$

the matrices $\mathcal{O}_{N-1}$ and $\mathcal{M}_{N-1}$ satisfy $\text{rank}\left(\begin{bmatrix}\mathcal{O}_{N-1} & \mathcal{M}_{N-1}\end{bmatrix}\right) = N + \text{rank}\,(\mathcal{M}_{N-1})$.

□

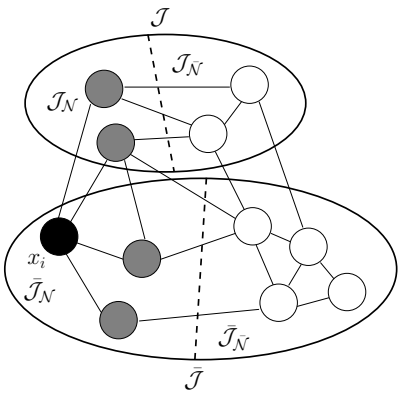

Fig. 1. A sample partition of the vertex set. The black node is x_i, the gray nodes are $\mathcal{N}_i$ (the neighbors of node x_i), and the white nodes are all the other nodes.

The above theorem indicates that if we can choose the weight matrix $\mathbf{W}$ so that the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ has no invariant zeros for all possible sets $\mathcal{J}$ of $2f$ nodes, then the rank condition in equation (11) of Theorem 4 will be satisfied with $L = N-1$; therefore, node x_i will be able to calculate any desired function of the initial values, even in the presence of up to f malicious or faulty nodes. We now focus on choosing $\mathbf{W}$ so that this is the case. In fact, our development will reveal that one can choose $\mathbf{W}$ so that multiple nodes in the system can simultaneously calculate any desired functions of the initial values (e.g., they can reach consensus, or calculate different functions of the initial values, all with the same $\mathbf{W}$).

B. Invariant Zeros

Let x_i be any given node in the network, let $\mathcal{J} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_{2f}}\}$ denote any set of $2f$ nodes (possibly containing x_i), and let $\bar{\mathcal{J}} = \mathcal{X} \setminus \mathcal{J}$. Further partition the sets $\mathcal{J}$ and $\bar{\mathcal{J}}$ as follows (an illustration of these sets is shown in Fig. 1):

- $\mathcal{J}_{\mathcal{N}} = \mathcal{J} \cap (\mathcal{N}_i \cup \{x_i\})$. This set contains all nodes in set $\mathcal{J}$ that are neighbors of node x_i (or node x_i itself).
- $\mathcal{J}_{\bar{\mathcal{N}}} = \mathcal{J} \setminus \mathcal{J}_{\mathcal{N}}$. This set contains all nodes in set $\mathcal{J}$ that are not neighbors of node x_i (nor node x_i itself).
- $\bar{\mathcal{J}}_{\mathcal{N}} = \bar{\mathcal{J}} \cap (\mathcal{N}_i \cup \{x_i\})$. This set contains all nodes in set $\bar{\mathcal{J}}$ that are neighbors of node x_i (or node x_i itself).
- $\bar{\mathcal{J}}_{\bar{\mathcal{N}}} = \bar{\mathcal{J}} \setminus \bar{\mathcal{J}}_{\mathcal{N}}$. This set contains all nodes in set $\bar{\mathcal{J}}$ that are not neighbors of node x_i (nor node x_i itself).

Note that these sets partition the entire set of nodes. For any choice of weights for the linear iteration, and for any subsets $\mathcal{A} \subseteq \mathcal{X}$ and $\mathcal{B} \subseteq \mathcal{X}$, let $\mathbf{W}_{\mathcal{A}}$ denote the square weight matrix corresponding to interconnections within the set $\mathcal{A}$, and let $\mathbf{W}_{\mathcal{A},\mathcal{B}}$ denote the weight matrix corresponding to connections from nodes in set $\mathcal{A}$ to nodes in set $\mathcal{B}$.

Lemma 4: For any set $\mathcal{J}$ of $2f$ nodes, the invariant zeros of the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}}, \bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}, \bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\mathcal{N}}, \bar{\mathcal{J}}_{\mathcal{N}}})$. □

The proof of the above lemma can be found in Appendix A. This lemma, along with Theorem 5, reveals that in order to

ensure that the rank condition (11) in Theorem 4 is satisfied for node x_i, we can focus on the problem of choosing the weights so that the set

$$(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$$

will have no invariant zeros for any set $\mathcal{J}$ of $2f$ nodes. To accomplish this, we will use techniques from control theory pertaining to *linear structured systems* [35], [36], [37]. Specifically, a linear system of the form (13) is said to be structured if each entry of the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ is either a fixed zero or an independent free parameter. It is known that linear systems have certain structural properties (such as observability, controllability, etc.), and these properties hold generically. In other words, if the structural property holds for some particular choice of free parameters, it will hold for almost any choice of parameters (i.e., the set of parameters for which the property does not hold has Lebesgue measure zero) [36], [37]. It turns out that the number of invariant zeros of a linear system is also a structured property (i.e., a linear system with a given zero/nonzero structure will have the same number of invariant zeros for almost any choice of the free parameters) [33].

To analyze structural properties (such as the number of invariant zeros) of linear systems, one first associates a graph $\mathcal{H}$ with the structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ as follows. The vertex set of $\mathcal{H}$ is given by $\mathcal{X} \cup \mathcal{U} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is the set of state vertices, $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ is the set of input vertices, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_p\}$ is the set of output vertices. The edge set of $\mathcal{H}$ is given by $\mathcal{E}_{xx} \cup \mathcal{E}_{ux} \cup \mathcal{E}_{xy} \cup \mathcal{E}_{uy}$, where

- $\mathcal{E}_{xx} = \{(x_j, x_i) |\ \mathbf{A}_{ij} \neq 0\}$ is the set of edges corresponding to interconnections between the state vertices,
- $\mathcal{E}_{ux} = \{(u_j, x_i) |\ \mathbf{B}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the state vertices,
- $\mathcal{E}_{xy} = \{(x_j, y_i) |\ \mathbf{C}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the state vertices and the output vertices, and
- $\mathcal{E}_{uy} = \{(u_j, y_i) |\ \mathbf{D}_{ij} \neq 0\}$ is the set of edges corresponding to connections between the input vertices and the output vertices.

The following theorems from [33] and [37] characterize the generic number of invariant zeros of a structured system and the normal-rank of a structured matrix pencil in terms of the associated graph $\mathcal{H}$. The terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem 6 ([33], [37]): Let $\mathbf{P}(z)$ be the matrix pencil of the structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, and let the normal-rank of $\mathbf{P}(z)$ be $N + m$, even after the deletion of an arbitrary row from $\mathbf{P}(z)$. Let $\mathcal{H}$ be the graph associated with the set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, and consider all possible subgraphs of $\mathcal{H}$ that consist of a disjoint union of a size m linking from $\mathcal{U}$ to $\mathcal{Y}$, a set of cycles in $\mathcal{X}$, and a set of $\mathcal{Y}$-topped paths. From these subgraphs, denote by $\bar{\mathcal{H}}$ the one that contains the largest number of vertices from $\mathcal{X}$. Then the generic number of invariant zeros of system (4) is equal to N minus the number of vertices of $\mathcal{X}$ contained in $\bar{\mathcal{H}}$. □

Theorem 7 ([33]): Let $\mathbf{P}(z)$ be the matrix pencil of the structured set $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$. Then the normal-rank of $\mathbf{P}(z)$ is generically equal to N plus the maximum size of a linking from $\mathcal{U}$ to $\mathcal{Y}$. □

To apply these results to the problem of determining the number of invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$, we note that all matrices in this set are essentially structured matrices (since all of the nonzero entries in the above matrices represent weights which can be chosen arbitrarily and independently). In particular, the nodes in set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ act as the state vertices for the structured system, the nodes in the set $\mathcal{J}_{\bar{\mathcal{N}}}$ act as the input vertices, and the nodes in the set $\bar{\mathcal{J}}_{\mathcal{N}}$ act as the output vertices. Recall that the above results on structured systems assumed that the number of outputs is larger than (or equal to) the number of inputs. The following lemma shows that this property does in fact hold for the sets $\bar{\mathcal{J}}_{\mathcal{N}}$ and $\mathcal{J}_{\bar{\mathcal{N}}}$ if the in-degree of node x_i is sufficiently high.

Lemma 5: If $\deg_i \geq 2f + 1$, then for any set $\mathcal{J}$ of $2f$ nodes, $|\bar{\mathcal{J}}_{\mathcal{N}}| > |\mathcal{J}_{\bar{\mathcal{N}}}|$. □

The proof of the above lemma can be found in Appendix B. Note that if $\kappa_{ji} \geq 2f + 1$ for every j, and there is at least one node that is not a neighbor of node x_i, then $\deg_i$ must necessarily be no smaller than $2f + 1$ (since otherwise, there would not be $2f + 1$ vertex-disjoint paths from any node x_j to node x_i). We can now use the above results on structured systems to prove the following lemma for the linear iteration in (4); the lemma will be used in conjunction with Lemma 4 and Theorem 5 to show that node x_i can uniquely determine the initial values $\mathbf{x}[0]$ after running the linear iteration for at most N time-steps, even with up to f malicious nodes.

Lemma 6: Consider node x_i in the network $\mathcal{G}$, and suppose that $\kappa_{ji} \geq 2f + 1$ for all j. Then for almost any real-valued choice of weights (with $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i \cup \{x_i\}$), the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$ will have no invariant zeros for any set $\mathcal{J}$ of $2f$ nodes. □

The proof of the above lemma appears in Appendix C. With the above lemma in hand, it is a simple matter to prove the following theorem.

Theorem 8: Given a fixed network with N nodes described by a graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, let f denote the maximum number of malicious nodes that are to be tolerated in the network, and let κ_{ji} denote the size of the smallest (j, i)-cut between any two vertices x_j and x_i. Define

$$\mathcal{T} = \{x_i \mid \kappa_{ji} \geq 2f + 1 \text{ for all } x_j \in \mathcal{X}\} \ .$$

Then, with almost any choice of real-valued weight matrix $\mathbf{W}$ (with $w_{ij} = 0$ if $x_j \notin \mathcal{N}_i \cup \{x_i\}$), every node in $\mathcal{T}$ can uniquely determine all of the initial values in the network after running the linear iteration for at most N time-steps, regardless of the updates applied by up to f malicious nodes. □

Proof: If node x_i is in set $\mathcal{T}$, then from Lemmas 4 and 6, we see that for almost any choice of weight matrix $\mathbf{W}$, and for any set $\mathcal{J}$ of cardinality $2f$, the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ will have no invariant zeros. Furthermore, since the set of weights for which this property does not hold has measure zero, it will hold generically for all nodes in set $\mathcal{T}$. From Theorem 5, we see that rank $\left(\begin{bmatrix}\mathcal{O}_{i,N-1} & \mathcal{M}^{\mathcal{J}}_{i,N-1}\end{bmatrix}\right) = N +$

rank $\left(\mathcal{M}^{\mathcal{J}}_{i,N-1}\right)$ for all $x_i \in \mathcal{T}$ and all sets $\mathcal{J}$ of $2f$ nodes. Thus, Theorem 4 indicates that every node in $\mathcal{T}$ can calculate any arbitrary function of the initial values after running the linear iteration for at most N time-steps. ■

Theorem 8 proves the "if" part of Theorem 1, and in conjunction with Theorem 3, we see that Theorem 1 is proved in full.

Remark 6: The above theorem provides an *upper bound* of N on the number of time-steps required by node x_i to obtain the initial values despite the presence of malicious agents. The actual number of time-steps required depends on the network topology, and there is not a general relationship describing how the actual number of time-steps grows with the number of agents N. As a trivial example, consider the fully-connected graph on N nodes; in this case, any node x_i can obtain all of the initial values accurately after just one time-step, regardless of N. It is easy to construct more complicated graphs where the number of time-steps is independent of N. A tighter upper bound on the number of time-steps required for any node to accumulate all of the initial values when there are *no* malicious nodes is presented in [38]. □

VI. Identifying the Malicious Nodes

Note that although the procedure described in the proof of Theorem 4 allows a given node x_i to determine the entire set of initial values, it does not necessarily require node x_i to determine the *actual* set of malicious nodes. Instead, node x_i is only required to find a *candidate* set $\mathcal{F}_j$ of malicious nodes, and then write the values that it has received over the past $L+1$ time-steps as a linear combination of the columns in the corresponding invertibility matrix and the observability matrix. In many cases, this candidate set will be the actual set of malicious nodes, but there can also be cases where the two sets are different. As a trivial example, consider the case of a network where two nodes x_i and x_j are separated by distance D, and suppose that node x_i requires $L+1$ time-steps in order to calculate the initial values of all nodes via the linear iteration. Now suppose that node x_j is malicious, but only commits its first additive error during one of the last $D-1$ time-steps. This additive error will not propagate to node x_i before time-step L, and thus node x_i will not be aware that node x_j has acted maliciously. In other words, as far as node x_i is concerned, the case where node x_j is malicious during the last $D-1$ time-steps of the linear iteration is indistinguishable from the case where node x_j behaves completely correctly for all time-steps. Note that this fact does not hamper node x_i from correctly obtaining the initial values of all the nodes. However, there may be applications when it is desirable for nodes to distributively identify the exact set of malicious nodes (e.g., in order to remove them from the network). The following theorem indicates that if node x_i examines the values that it receives from the linear iteration over a number of time-steps larger than that required purely for function calculation, it can determine the exact set of nodes that were malicious during the initial stages of the iteration.

Theorem 9: Suppose node x_i can calculate the entire set of initial values after running the linear iteration for L_i+1 time-steps, despite the actions of up to f malicious nodes (i.e., there is a weight matrix $\mathbf{W}$ that satisfies equation (11) with $L=L_i$). Let $\mathcal{F}$ be the set of nodes that are malicious during the course of the linear iteration (with cardinality f or lower). Then, for any nonnegative integer $\bar{L}$, after $L_i+1+\bar{L}$ time-steps of the linear iteration, node x_i can uniquely identify the nodes in set $\mathcal{F}$ that were malicious during the first $\bar{L}$ time-steps of the iteration. □

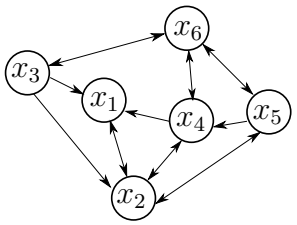

Fig. 2. Node x_1 needs to calculate the sum of squares of all initial values in the network, even if one node is malicious.

Proof: Note that the theorem holds trivially for $\bar{L}=0$, and so we focus on the case of $\bar{L} \geq 1$ in the rest of the proof. From Theorem 4, we know that node x_i can uniquely determine the correct value of $\mathbf{x}[0]$ based on the values $\mathbf{y}_i[0:L_i]$. Now, view $\mathbf{x}[1]$ as the vector of initial values in the network; once again, Theorem 4 indicates that node x_i can recover $\mathbf{x}[1]$ from the values $\mathbf{y}_i[1:L_i+1]$. From (4), node x_i can now obtain $\mathbf{x}[1]-\mathbf{W}\mathbf{x}[0]=\mathbf{B}_{\mathcal{F}}\mathbf{u}_{\mathcal{F}}[k]$. Thus, every nonzero component in the vector on the left hand side of the above equation indicates an additive error injected by the corresponding node, and so every node that is malicious during time-step 0 can be identified by the above method. The same process can be repeated to find all nodes that were malicious during the first $\bar{L}$ time-steps from the transmitted values $\mathbf{y}_i[0:L_i+\bar{L}]$. ■

VII. Example

Consider the network shown in Fig. 2. The objective in this network is for node x_1 to calculate the function $\sum_{j=1}^{6} x_j^2[0]$, even if there is up to $f=1$ malicious node in the network.

A. Network Design

Examining the network, we see that there are three internally vertex-disjoint paths from node x_5 to node x_1, and also from node x_6 to node x_1. Since all other nodes are neighbors of node x_1, we have that $\kappa_{j1} \geq 3$ for all j, and so Theorem 8 indicates that node x_1 can calculate the desired function after running the linear iteration (with almost any choice of weights) for at most $N=6$ time-steps, despite the presence of up to 1 malicious node. For this example, we will take each of the edge- and self-weights to be i.i.d. random variables chosen from the set[10] $\{-5,-4,-3,-2,-1,1,2,3,4,5\}$ with equal

[10]In general, the result in Theorem 8 will hold with probability 1 if one chooses the weights for the linear iteration from a continuous distribution over the real numbers (such as a Gaussian distribution). For this pedagogical example, however, it will suffice to consider a distribution on a small set of integers; this makes the presentation of the numerical values more concise.

probabilities. These weights produce the matrix

$$\mathbf{W} = \begin{bmatrix} -3 & 1 & -1 & 3 & 0 & 0 \\ 1 & 2 & -1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 4 & 0 & 2 & 2 & 5 \\ 0 & 2 & 0 & 0 & -1 & 5 \\ 0 & 0 & 1 & 3 & -4 & -4 \end{bmatrix}.$$

Since node x_1 has access to its own value, as well as those of its neighbors (nodes x_2, x_3 and x_4) at each time-step, the matrix $\mathbf{C}_1$ is given by $\mathbf{C}_1 = \begin{bmatrix} \mathbf{I}_4 & \mathbf{0} \end{bmatrix}$. With the above weight matrix, one can verify that equation (11) is satisfied with $i = 1$ and $L = L_1 = 2$ for all sets $\mathcal{J}$ of $2f = 2$ nodes, where $\mathcal{O}_{i,L_i}$ and $\mathcal{M}_{i,L_i}^{\mathcal{J}}$ are defined in equation (5) (the exact numerical values are omitted in the interest of space). We make the above matrices available to node x_1, and at this point, node x_1 has all the information it needs to be able to calculate the function $\sum_{j=1}^{6} x_j^2[0]$ (or indeed, any arbitrary function of the initial values) after $L_1 + 1 = 3$ time-steps of the linear iteration, even in the presence of one malicious node.

B. Performing Function Calculation

Suppose that the initial values of the nodes are $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$. In order for node x_1 to calculate the function $\sum_{j=1}^{6} x_j^2[0]$, the nodes run the linear iteration with the weight matrix provided above. Suppose that node x_4 is malicious, and at time-steps 1 and 2, it updates its value as $x_4[1] = 4x_2[0] + 2x_4[0] + 2x_5[0] + 5x_6[0] - 8$ and $x_4[2] = 4x_2[1] + 2x_4[1] + 2x_5[1] + 5x_6[1] - 12$, i.e., during the first time-step, it commits an additive error of $u_4[0] = -8$, and during the second time-step, it commits an additive error of $u_4[1] = -12$. All other nodes follow the predefined (correct) strategy of updating their values according to the weighted average specified by the weight matrix $\mathbf{W}$. The values of all nodes over the first three time-steps of the linear iteration are given by $\mathbf{x}[0] = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$, $\mathbf{x}[1] = \begin{bmatrix} -26 & 0 & 26 & 49 & 46 & -80 \end{bmatrix}'$, and $\mathbf{x}[2] = \begin{bmatrix} 199 & 43 & -134 & -222 & -446 & 309 \end{bmatrix}'$. The values seen by node x_1 at time-step k are given by $\mathbf{y}_1[k] = \mathbf{C}_1\mathbf{x}[k]$, and node x_1 can now use these values to calculate the vector of initial values, despite the efforts of the malicious node. As discussed in Theorem 4, node x_1 finds the first set $\mathcal{F}_j$ for which $\mathbf{y}_1[0:2]$ is in the column space of $\mathcal{O}_{1,2}$ and $\mathcal{M}_{1,2}^{\mathcal{F}_j}$; in this example, node x_1 finds that this holds for $j = 4$. It now finds vectors $\bar{\mathbf{x}}$ and $\mathbf{u}_{\mathcal{F}_4}[0:1]$ such that $\mathbf{y}_i[0:2] = \mathcal{O}_{1,2}\bar{\mathbf{x}} + \mathcal{M}_{1,2}^{\mathcal{F}_4}\mathbf{u}_{\mathcal{F}_4}[0:1]$ as $\bar{\mathbf{x}} = \begin{bmatrix} 3 & -1 & 4 & -4 & 7 & 11 \end{bmatrix}'$ and $\mathbf{u}_{\mathcal{F}_4}[0:1] = \begin{bmatrix} -8 & -12 \end{bmatrix}'$. Node x_1 now has access to $\mathbf{x}[0] = \bar{\mathbf{x}}$, and can calculate the function $\sum_{j=1}^{6} x_j^2[0]$ to obtain the value 212.

Note that in this example, the candidate set $\mathcal{F}_j$ found by node x_1 does, in fact, contain the actual malicious node x_4. This will not be true in general; if we want node x_1 to be guaranteed to locate any node that is malicious during the first $\bar{L}$ time-steps of the iteration, we should have node x_1 repeat the above procedure for $L_i + \bar{L} + 1 = 3 + \bar{L}$ time-steps of the linear iteration, as described in Theorem 9. The numerical details are very similar to those in the above example, and are omitted here.

It is worth noting that the network in Fig. 2 also has $\kappa_{j2} \geq 3$ for all j, and so node x_2 can also follow the same procedure to calculate any arbitrary function of all values in the system in the presence of up to one faulty node (with the same weight matrix, and in parallel to node x_1). On the other hand, consider node x_6 in the network. The set $\mathcal{F} = \{x_4, x_5\}$ forms a $(2,6)$-cut (i.e., removing nodes x_4 and x_5 removes all paths from node x_2 to node x_6), and so $\kappa_{26} = 2$. Thus, node x_6 is not guaranteed to be able to calculate any function of node x_2's value when there is a faulty node in the system. In fact, one can verify that in the example above, where node x_4 is malicious and updates its values with the errors $u_4[0] = -8$ and $u_4[1] = -12$, the values seen by node x_6 during the first three time-steps of the linear iteration are the same as the values seen by node x_6 when $x_1[0] = 4, x_2[0] = -3$ and node x_5 is malicious, with $u_5[0] = 4$ and $u_5[1] = 6$. In other words node x_6 cannot distinguish the case when node x_4 is faulty from the case where node x_5 is faulty (with different initial values in the network). As discussed in Section IV, node x_4 can continue to update its values erroneously so that node x_6 can never resolve this ambiguity, regardless of the number of time-steps for which the linear iteration is run, and thus it cannot correctly calculate a function of the values of node x_1 and node x_2.

VIII. Summary

We considered the problem of using linear iterative strategies for distributed function calculation in the presence of malicious agents that can update their values in an arbitrary and possibly coordinated manner, under the broadcast model of communication. We showed that the ability of linear iterative strategies to tolerate such malicious behavior is completely determined by the topology of the network. If there exists a pair of nodes x_i and x_j such that $\kappa_{ji} \leq 2f$, then we showed that it is possible for a particular set of f malicious nodes to update their values in a coordinated fashion so that node x_i cannot correctly determine the value of node x_j, and thus cannot calculate any function that involves that value, regardless of the number of time-steps for which the linear iteration is run. We then showed that it is possible for a given node x_i to calculate any arbitrary function in the presence of up to f malfunctioning or malicious nodes as long as the size of the smallest (j,i)-cut with any other node x_j is at least $2f+1$. For all nodes that satisfy the connectivity requirements, we showed that they can calculate their desired functions after running the linear iteration with almost any real-valued choice of weights for at most N time-steps. Furthermore, under these connectivity requirements, any node that is malicious or faulty during the first part of the linear iteration can be uniquely identified by the other nodes in the network if they run the linear iteration for more time-steps than that required to simply perform function calculation.

Acknowledgment

The authors thank the anonymous reviewers for their constructive comments and insights.

APPENDIX A
PROOF OF LEMMA 4

Proof: The matrix pencil for the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ is given by $\mathbf{P}(z) = \begin{bmatrix} \mathbf{W}-z\mathbf{I}_N & \mathbf{B}_{\mathcal{J}} \\ \mathbf{C}_i & \mathbf{0} \end{bmatrix}$. By a simple renumbering of the nodes,[11] we can assume without loss of generality that $\mathbf{W}, \mathbf{B}_{\mathcal{J}}$, and $\mathbf{C}_i$ are of the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\bar{\mathcal{J}}_{\mathcal{N}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}},\mathcal{J}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\bar{\mathcal{J}}_{\mathcal{N}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\mathcal{J}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\mathcal{N}}} \end{bmatrix},$$

$$\mathbf{B}_{\mathcal{J}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{|\mathcal{J}_{\bar{\mathcal{N}}}|} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} \end{bmatrix}, \quad \mathbf{C}_i = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{|\bar{\mathcal{J}}_{\mathcal{N}}|} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{|\mathcal{J}_{\mathcal{N}}|} \end{bmatrix},$$

since $\mathbf{B}_{\mathcal{J}}$ contains a single 1 in each row corresponding to a node in $\mathcal{J}$, and $\mathbf{C}_i$ measures nodes that are neighbors of node x_i, or node x_i itself. Based on this, one can see that

$$\begin{aligned} \text{rank}(\mathbf{P}(z)) = |\mathcal{J}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\mathcal{N}}| + |\bar{\mathcal{J}}_{\mathcal{N}}| + |\mathcal{J}_{\mathcal{N}}| \\ + \text{rank}\left(\begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} \end{bmatrix} \right), \end{aligned}$$

and so the invariant zeros of the set $(\mathbf{W}, \mathbf{B}_{\mathcal{J}}, \mathbf{C}_i, \mathbf{0})$ are exactly the invariant zeros of the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$. ∎

APPENDIX B
PROOF OF LEMMA 5

Proof: First, note from the definition of $\bar{\mathcal{J}}_{\mathcal{N}}$ and $\mathcal{J}_{\mathcal{N}}$ that $|\bar{\mathcal{J}}_{\mathcal{N}}| + |\mathcal{J}_{\mathcal{N}}| = \deg_i + 1$ (since the sets $\mathcal{J}$ and $\bar{\mathcal{J}}$ partition the vertex set of the graph, and therefore contain all neighbors of node x_i, along with node x_i itself). Furthermore $|\mathcal{J}_{\mathcal{N}}| + |\mathcal{J}_{\bar{\mathcal{N}}}| = 2f$ (since those two sets partition the set $\mathcal{J}$). Thus, $|\bar{\mathcal{J}}_{\mathcal{N}}| = \deg_i + 1 - |\mathcal{J}_{\mathcal{N}}| = \deg_i + 1 - 2f + |\mathcal{J}_{\bar{\mathcal{N}}}|$. Since $\deg_i \geq 2f+1$, this expression becomes $|\bar{\mathcal{J}}_{\mathcal{N}}| \geq |\mathcal{J}_{\bar{\mathcal{N}}}| + 2 > |\mathcal{J}_{\bar{\mathcal{N}}}|$, thereby proving the lemma. ∎

APPENDIX C
PROOF OF LEMMA 6

Proof: The matrix pencil for the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$ is

$$\mathbf{P}(z) = \begin{bmatrix} \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} - z\mathbf{I}_{|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}} \\ \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} & \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}} \end{bmatrix}.$$

We will show that this pencil has full normal-rank even after the deletion of an arbitrary row, and then use Theorem 6 to prove the lemma.

We start by constructing the graph $\mathcal{H}$ associated with the set $(\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}, \mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}}, \mathbf{W}_{\mathcal{J}_{\bar{\mathcal{N}}},\bar{\mathcal{J}}_{\mathcal{N}}})$. For this set of matrices, note that the state vertices in the graph $\mathcal{H}$ are given by the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, the input vertices are given by the set $\mathcal{J}_{\bar{\mathcal{N}}}$, and the output vertices are given by the set $\bar{\mathcal{J}}_{\mathcal{N}}$. In particular, $\mathcal{H}$ can be obtained by first taking the graph of the network $\mathcal{G}$ and removing all incoming edges to nodes in $\mathcal{J}_{\bar{\mathcal{N}}}$ (since the nodes in $\mathcal{J}_{\bar{\mathcal{N}}}$ are treated as inputs), and removing all outgoing edges from nodes in $\bar{\mathcal{J}}_{\mathcal{N}}$ (since those nodes are treated as outputs). Furthermore, remove all nodes that are in the set $\mathcal{J}_{\mathcal{N}}$ and their incident edges (since these nodes do not appear anywhere in the matrix pencil), and add a set of self loops to every state vertex to correspond to the nonzero entries on the diagonal of the weight matrix $\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}$.

We will now examine what happens when we remove a single row from $\mathbf{P}(z)$. Suppose we remove one of the rows of $\mathbf{P}(z)$ corresponding to a vertex $v \in \bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ (i.e., one of the top $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ rows of $\mathbf{P}(z)$), and denote the resulting matrix by $\bar{\mathbf{P}}(z)$. The generic rank of $\bar{\mathbf{P}}(z)$ can be found by examining the associated graph, which we will denote by $\bar{\mathcal{H}}$. Note that $\bar{\mathcal{H}}$ is obtained from $\mathcal{H}$ simply by removing all incoming edges to vertex v in $\mathcal{H}$, since we removed the row corresponding to v from $\mathbf{P}(z)$; however, all outgoing edges from v are still left in the graph (since the column corresponding to vertex v is left in matrix $\bar{\mathbf{P}}(z)$). Thus, we see that vertex v can be treated as an input vertex in $\bar{\mathcal{H}}$, leaving $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1$ state vertices (corresponding to the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}} \setminus \{v\}$). Now, note that the set $\mathcal{J} \cup \{v\}$ has cardinality $2f+1$, and thus Lemma 1 indicates that there is a linking of size $2f+1$ from $\mathcal{J} \cup \{v\}$ to $\{x_i\} \cup \mathcal{N}_i$ in $\mathcal{G}$. In this linking, consider the paths starting at vertices in the set $\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}$; none of these paths passes through the vertices in $\bar{\mathcal{J}}_{\mathcal{N}}$ (since these vertices are the start vertices of other paths in the linking). Graph $\mathcal{G}$ thus contains a linking from $\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}$ to $\bar{\mathcal{J}}_{\mathcal{N}}$, where each path in the linking only passes through vertices in the set $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, and therefore this linking also exists in the graph $\bar{\mathcal{H}}$. According to Theorem 7, $\bar{\mathbf{P}}(z)$ has generic normal-rank equal to the number of state vertices in $\bar{\mathcal{H}}$ (equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1$) plus the maximal size of a linking from the inputs to the outputs (equal to $|\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}|$), for a total of $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| - 1 + |\mathcal{J}_{\bar{\mathcal{N}}} \cup \{v\}| = |\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\bar{\mathcal{N}}}|$. The pencil $\bar{\mathbf{P}}(z)$ thus has generically full normal-rank.

Now, suppose that we remove a row of $\mathbf{P}(z)$ corresponding to a vertex $v \in \bar{\mathcal{J}}_{\mathcal{N}}$ (i.e., one of the bottom $|\bar{\mathcal{J}}_{\mathcal{N}}|$ rows of $\mathbf{P}(z)$), and again denote the resulting matrix by $\bar{\mathbf{P}}(z)$. The associated graph $\bar{\mathcal{H}}$ is obtained by simply removing vertex v from $\mathcal{H}$. Since $\kappa_{ji} \geq 2f+1$ for every j in graph $\mathcal{G}$, Lemma 1 indicates that there will be a linking of size $2f+1$ from the set $\mathcal{J} \cup \{v\}$ to the set $\{x_i\} \cup \mathcal{N}_i$ in $\mathcal{G}$. In particular, there will be a linking of size $|\mathcal{J}_{\bar{\mathcal{N}}}|$ from the set $\mathcal{J}_{\bar{\mathcal{N}}}$ to the set $\bar{\mathcal{J}}_{\mathcal{N}} \setminus \{v\}$, and none of the paths in this linking would go through any of the vertices in the set $\mathcal{J}_{\mathcal{N}} \cup \{v\}$ (since these vertices are the start vertices of other paths in the linking). The linking from the set $\mathcal{J}_{\bar{\mathcal{N}}}$ to the set $\bar{\mathcal{J}}_{\mathcal{N}} \setminus \{v\}$ will therefore also exist in graph $\bar{\mathcal{H}}$, and once again, Theorem 7 indicates that the matrix pencil will have generically full normal-rank equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}| + |\mathcal{J}_{\bar{\mathcal{N}}}|$.

We have thus shown that $\mathbf{P}(z)$ will have generically full normal-rank even after the deletion of an arbitrary row. From Theorem 6, the number of invariant zeros of $\mathbf{P}(z)$ is equal to $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ minus the maximal number of vertices in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$ contained in the disjoint union of a size $|\mathcal{J}_{\bar{\mathcal{N}}}|$ linking from $\mathcal{J}_{\bar{\mathcal{N}}}$ to $\bar{\mathcal{J}}_{\mathcal{N}}$, a cycle family in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$, and a $\bar{\mathcal{J}}_{\mathcal{N}}$-topped path family. If we simply take any $|\mathcal{J}_{\bar{\mathcal{N}}}|$-linking from $\mathcal{J}_{\bar{\mathcal{N}}}$ to $\bar{\mathcal{J}}_{\mathcal{N}}$, and include

[11] Note that this does not affect the invariant zeros of the system, since this renumbering of the nodes simply corresponds to pre– and post–multiplying the original matrix pencil by appropriate permutation matrices. Since permutation matrices are nonsingular, the rank of the new pencil matrix will be the same as the rank of the original matrix pencil for all values of z.

all the self-loops on state vertices in $\mathcal{H}$ that are not included in the linking (corresponding to the nonzero weights on the diagonal of the weight matrix $\mathbf{W}_{\bar{\mathcal{J}}_{\bar{\mathcal{N}}}}$), we will have a linking and a set of disjoint cycles that covers all $|\bar{\mathcal{J}}_{\bar{\mathcal{N}}}|$ vertices in $\bar{\mathcal{J}}_{\bar{\mathcal{N}}}$. Thus, the matrix pencil $\mathbf{P}(z)$ will generically have no invariant zeros, thereby proving the lemma. ■

REFERENCES

[1] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[2] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 20–27.

[3] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the American Control Conference*, 2005, pp. 1859–1864.

[4] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[6] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks*. Springer-Verlag, 2005.

[7] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, Mar. 2008.

[8] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[9] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[10] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[11] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[12] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, Feb. 2005.

[13] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation and consensus using linear iterative strategies," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 650–660, May 2008.

[14] K. Sampigethaya, M. Li, R. Poovendran, R. Robinson, L. Bushnell, and S. Lintelman, "Secure wireless collection and distribution of commercial airplane health data," in *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*, 2007, pp. 4.E.6–1–4.E.6.8.

[15] C. N. Hadjicostis, *Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems*. Kluwer Academic Publishers, 2002.

[16] D. B. West, *Introduction to Graph Theory*. Prentice-Hall Inc., Upper Saddle River, New Jersey, 2001.

[17] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[18] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *Journal of the Association for Computing Machinery*, vol. 40, no. 1, pp. 17–47, Jan. 1993.

[19] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982.

[20] V. Bhandari and N. H. Vaidya, "On reliable broadcast in a radio network," in *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2005, pp. 138–147.

[21] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros, "Resilient network coding in the presence of Byzantine adverseries," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2596–2603, June 2008.

[22] V. Gupta, C. Langbort, and R. M. Murray, "On the robustness of distributed algorithms," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 3473–3478.

[23] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 5594–5599.

[24] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterations in the presence of malicious agents – part I: Attacking the network," in *Proc. of the American Control Conference*, 2008, pp. 1350–1355.

[25] ——, "Distributed function calculation via linear iterations in the presence of malicious agents – part II: Overcoming malicious behavior," in *Proc. of the American Control Conference*, 2008, pp. 1356–1361.

[26] V. Barnett and T. Lewis, *Outliers in Statistical Data*. John Wiley and Sons Ltd., West Sussex, England, 1996.

[27] M. K. Sain and J. L. Massey, "Invertibility of linear time-invariant dynamical systems," *IEEE Transactions on Automatic Control*, vol. AC-14, no. 2, pp. 141–149, Apr. 1969.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2001.

[29] G. Takos and C. N. Hadjicostis, "Error correction in DFT codes subject to low-level quantization noise," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1043–1054, Mar. 2008.

[30] M. L. J. Hautus, "Strong detectability and observers," *Linear Algebra and its Applications*, vol. 50, pp. 353–368, Apr. 1983.

[31] D. Rappaport and L. M. Silverman, "Structure and stability of discrete-time optimal systems," *IEEE Transactions on Automatic Control*, vol. 16, no. 3, pp. 227–233, June 1971.

[32] W. Kratz, "Characterization of strong observability and construction of an observer," *Linear Algebra and its Applications*, vol. 221, pp. 31–40, May 1995.

[33] J. van der Woude, "The generic number of invariant zeros of a structured linear system," *SIAM Journal on Control and Optimization*, vol. 38, no. 1, pp. 1–21, Nov. 1999.

[34] L. M. Silverman, "Discrete Riccati equations: Alternative algorithms, asymptotic properties and system theory interpretations," in *Control and Dynamic Systems*, C. T. Leondes, Ed. Academic Press, 1976, vol. 12, pp. 313–386.

[35] C.-T. Lin, "Structural controllability," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 201–208, June 1974.

[36] K. J. Reinschke, *Multivariable Control: A Graph-Theoretic Approach*. Springer-Verlag, 1988.

[37] J.-M. Dion, C. Commault, and J. van der Woude, "Generic properties and control of linear structured systems: a survey," *Automatica*, vol. 39, no. 7, pp. 1125–1144, July 2003.

[38] S. Sundaram and C. N. Hadjicostis, "On the time complexity of information dissemination via linear iterative strategies," in *Proceedings of the American Control Conference*, 2010, pp. 6789–6794.

Shreyas Sundaram (M'09) is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Waterloo. He received his MS and PhD degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 2005 and 2009, respectively. He was a Postdoctoral Researcher in the GRASP Laboratory at the University of Pennsylvania from 2009-2010. His research interests include secure and fault-tolerant control of distributed systems and networks, linear system and estimation theory, and the application of algebraic graph theory to system analysis. He was a finalist for the Best Student Paper Award at the 2007 and 2008 American Control Conferences.

Christoforos N. Hadjicostis (M'99, SM'05) received S.B. degrees in Electrical Engineering, in Computer Science and Engineering, and in Mathematics, the M.Eng. degree in Electrical Engineering and Computer Science in 1995, and the Ph.D. degree in Electrical Engineering and Computer Science in 1999, all from the Massachusetts Institute of Technology, Cambridge, MA.

In 1999 he joined the Faculty at the University of Illinois at Urbana-Champaign where he served as assistant and then associate professor with the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Information Trust Institute. Since 2007, Dr. Hadjicostis has been with the Department of Electrical and Computer Engineering at the University of Cyprus. His research focuses on fault diagnosis and tolerance in distributed dynamic systems; error control coding; monitoring, diagnosis and control of large-scale discrete event systems; and applications to network security, anomaly detection, energy distribution systems, medical diagnosis, biosequencing, and genetic regulatory models.