Brief Announcement: On The Resilience of Decentralized Sensor Networks Under Byzantine Attacks

ANONYMOUS AUTHOR(S)

This paper explores issues that impact the resilience of a decentralized sensor network to malicious attack. We formulate a graph model for sensor data validation, where the data reported by a sensor is cross-checked with data from neighboring sensors. In this model, we show that perfect validation is impossible. Subsequent results aim to understand how the structure of the network influences its susceptibility to attacks that force validators to either incorrectly reject correct data from honest sensors, or incorrectly accept wrong data from attacker-controlled sensors. We show that natural formulations of these questions are NP-hard.

CCS Concepts: • **Computer systems organization** → **Reliability**.

Additional Key Words and Phrases: sensor networks, decentralized systems, resilience

The real point of the matter is that what we call a "wrong datum" is one which is inconsistent with all other known data. It is our only criterion of right and wrong. It is the Machine's as well.

The Evitable Conflict, Isaac Asimov, 1950.

1 INTRODUCTION

A sensor network is a composition of spatially scattered sensors that provide readings (*datums*) on physical quantities. We consider the *unstructured* and *decentralized* form of these networks. By unstructured, we mean that the sensors are arbitrarily distributed. By decentralized, we mean that the sensors are controlled by parties with possibly conflicting interests. In this adversarial setting, malicious parties may collude to provide wrong but mutually consistent sensor data that either advances their interests or harms that of others; we say that these parties conduct a Byzantine attack [5]. For instance, consider decentralized sensors that monitor a scarce resource, such as water. Sensor readings may be tampered with to suggest a (false) water scarcity that artificially inflates the price of water. Compounding the problem, it is often easy to tamper with sensor data without tampering with the sensor itself, simply by manipulating its physical environment.

Given the ease of malicious reporting, any system that gathers and acts upon decentralized sensor data must include mechanisms that filter out bad data: ideally, accepting all correct datums and rejecting all incorrect ones. But validation faces several challenges. For instance, a validator that relies only upon the distribution of a single datum may be fooled by values that follow the distribution but are, in fact, incorrect. For instance, an incorrectly reported water level reading may still lie within the expected range of values for that location. One may hope that cross-checking the data provided by one sensor against the data provided by other sensors leads to more accurate validation, as suggested in the quote above.

In this paper, we consider the extent to which cross-checking makes the validation process resistant to malicious attacks. We model a sensor network as a graph, where a sensor is associated with each node, and the data at that node is cross-checked against data from its graph neighbors during validation. The identities of the sensors (nodes) under malicious control are not known.

We first argue that perfect validation is unachievable. One is thus forced to consider imperfect validators. However, we rule out fragile validators, by assuming that the validation of a sensor reading can be subverted only if at least half of the validator's inputs are provided by Byzantine (i.e., maliciously controlled) sensors.

The model leads naturally to the following network analysis question: Given a network graph and a budget k on the number of Byzantine sensors, could an attacker succeed in subverting enough validations to either reject correct data from a large fraction of the honest nodes, or accept fake data from a large number of the malicious nodes? The answers to these questions naturally depend on the structure of the network. We show that while the questions can be answered exactly for some regular networks, they are NP-hard to resolve in general.

The contributions of this work are in the modeling of Byzantine attacks, the framing of the network analysis questions, and in the hardness results. These results lead to new questions on the structure of resilient networks, which are the subject of ongoing work.

2 MODEL AND RESULTS

A *Decentralized Sensor Network* (DSN) is defined as a pair (G, F). Here, $G = (V, E)$ is an undirected graph with a set of nodes V and an irreflexive edge relation E. Each node represents a sensor. The edge relation E represents the "neighborhood" of a sensor (which need not be related to proximity in space). An edge between nodes u and v implies that the validation of data provided by sensor u depends on data from sensor v (and vice-versa). The set $F = \{f_v : \mathsf{D}^{deg(v)+1} \longrightarrow \mathsf{Bool} \mid v \in V\}$ is a collection of validation procedures, one for each sensor. For simplicity, we suppose that all sensors provide data from a single domain D.

We assume a *synchronous* model where sensor readings are collected and validated in *rounds*. At the beginning of a round, every sensor v reports a reading $\rho(v)$ to its validator. The validation procedure f_v queries the neighbors of v for their readings, and determines whether $\rho(v)$ should be accepted, based on $\rho(v)$ and the reported neighboring readings. Validation may rely on external knowledge (e.g., climate data) and past history of inputs. We focus here on the decision for a single round, leaving the analysis of multi-round attacks for future work.

Call a sensor node *Byzantine* if, during a round, it either (1) *may not* report any value, or (2) *may* report an incorrect value for the physical quantity the sensor is supposed to measure. Specifically, a Byzantine sensor is allowed to report different values to different neighbors when queried for its reading. An *honest* sensor is one that is not Byzantine. That is, it reports the correct value of the physical quantity that it measures.

A *perfect validator* is one that accepts every physically correct reading (Liveness) and rejects every incorrect reading (Soundness). A validator that accepts every reading is trivially live, while one that rejects every reading is trivially sound. The challenge is in meeting both:

Theorem 2.1 (Informal). *A perfect validator is unachievable.*

The argument is by contradiction. We show that given a perfect validator, one can define a perfect computational simulation of the physical world. Hence, if a perfect validator were to exist, there would be no need for sensors!

Athough validation must be imperfect, we assume that it is not fragile. We assume that reaching a wrong decision is possible only if at least half of the validator's inputs are provided by Byzantine nodes. Precisely, for validator f_v: (1) If strictly more than half of the inputs to f_v are from honest nodes, then f_v makes the correct decision on $\rho(v)$ (i.e., it accepts a correct value for $\rho(v)$ and rejects an incorrect one); and (2) On the other hand, if at least half the inputs to f_v are from Byzantine nodes then, regardless of the inputs from honest nodes, it is possible for an attacker to choose the Byzantine inputs to force an incorrect decision.

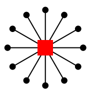
In a real system, validation may also have inherent accuracy limitations. We ignore this issue to better focus on the role of the Byzantine attacker. We assume further that the computation of f_v cannot be tampered with, which may be ensured through replication and Byzantine fault-tolerant consensus. Thus, the only way to subvert a validation is by providing misleading inputs.

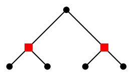
The model opens up two very different ways in which a Byzantine attacker may disrupt the network. First, some notation. Let $E(v)$ represent the set of neighbors of v; $B(v)$ the set of Byzantine neighbors of v; and $H(v)$ the set of honest neighbors of v. (Thus, $|E(v)| = |H(v)| + |B(v)|$.)

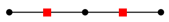
Consider a node v. There are two possibilities, depending on whether v is honest or malicious.

(1) **Blocking**: If node v is honest, its Byzantine neighbors may force the validator f_v to reject the correctly reported reading $\rho(v)$ by maliciously reporting values that are inconsistent with $\rho(v)$. For the attack to be effective, at least half the inputs to f_v must be Byzantine; i.e., $|B(v)| \geq 1 + |H(v)|$, as v is itself honest. The honest node v is said to be *blocked.*

(2) **Poisoning**: If node v is Byzantine, its Byzantine neighbors may force the validator f_v to accept the incorrect datum $\rho(v)$ by reporting incorrect readings which support $\rho(v)$. For the attack to be effective, at least half the inputs to f_v must be Byzantine; i.e., $1 + |B(v)| \geq |H(v)|$, as v is itself Byzantine. The Byzantine node v is said to be *poisoned.*


(a) Star


(b) Tree


(c) Line

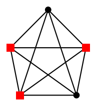

(d) Clique

Fig. 1. Examples of blocking attacks where Byzantine sensors are represented as red squares.

At this stage, it may be helpful to see examples of networks and blocking attacks, illustrated in Figure 1. For the star network shown in part (a), a single Byzantine node at the center suffices to block *all* other nodes, each of which is honest. However, as all neighbors of the central node are honest, this node cannot fool its validator into accepting incorrect data. For a complete binary tree shown in part (b), it suffices to choose the last-but-one row of nodes and every alternate row to block all remaining (honest) nodes. For a line shown in part (c), making every alternate node Byzantine suffices to block all honest nodes. But each Byzantine node has two honest neighbors, so it cannot be poisoned. Finally, for the complete graph shown in part (d), an attacker must control at least half the graph nodes to block the rest. Moreover, for each Byzantine node, two out of its four neighbors are also Byzantine, which is sufficient support for poisoning.

We see that in some networks (such as the star) many honest nodes can be blocked with a small number of well-chosen Byzantine nodes, so the network is susceptible to attack. In other cases, blocking a large number of honest nodes requires choosing a large number of Byzantine nodes, so those networks have higher resilience. Thus, the resilience of a network is (not surprisingly) sensitive to its topology.

That gives rise to a framing of resilience in terms of network analysis: Can a large fraction of the nodes of a DSN be blocked or poisoned, provided the attacker has a budget k of Byzantine nodes? We show that both questions are NP-hard.

Theorem 2.2 (Blocking). *Given a* DSN $G = (V, E)$ *and* $k < |V|$, *it is NP-hard to determine if there exists a choice of* k *Byzantine nodes such that at least half of the remaining honest nodes are blocked.*

Theorem 2.3 (Poisoning). *Given a* DSN $G = (V, E)$ *and numbers* $t \leq k < |V|$, *it is NP-hard to determine if there exists a choice of* k *Byzantine nodes such that at least* t *of those are poisoned.*

The proof of the first result is through a reduction from the Positive Influence Dominating Set (PIDS) question, known to be NP-hard and even APX-hard [13, 14, 16]. The main observation is that Blocking is similar to VertexCover and PIDS, in the sense that Blocking essentially requires *some* proportion of vertices being *half-covered*. Our proof of the second result is inspired by constructions in Agrawal and Maity's study on the Small Set Vertex Expansion (SSVE) problem [1]. The key idea here is that we can rephrase the problem of Poisoning in a way that is reminiscent to Vertex Expansion, and therefore mimic Agrawal and Maity's construction. Full proofs are available in the anonymized report at [**?**].

These results are, is in a way, encouraging. They suggest that it is difficult for an attacker to find an optimal attack, one that requires subverting the least number of sensors. On the other hand, they also suggest that it may be difficult to determine whether a network is sufficiently resilient to attack. This conundrum suggests more questions: (Network design) Which classes of sparse networks have high resiliency? and (Network augmentation) How can the resiliency of a given network be strengthened by augmenting its structure? These questions are practically important and mathematically fascinating.

3 RELATED WORK

The resilience of sensor networks to Byzantine attacks has been studied from several perspectives including Byzantine node detection, game-theoretic analysis of Byzantine behavior, and establishment of a chain of trust. We discuss the most closely related work.

Distributed Attack Detection. Marano, Matta and Tong [7] considered distributed detection problem in the presense of cooperative Byzantine attack. Their model differs from ours in several aspects: (1) each sensor can only fake their own data and cannot impact the other sensors' data, while we allow *blocking* and *poisoning*, and (2) the objective of each adversarial sensor is to maximize the difference between the true and the accepted fake value, while we focus on the number of accepted fake values.

Likelihood-based Attack Detection. The work of Sun, Zhang and Fan [11] identifies adversarial nodes based on a likelihood metric. In contrast to our model, this assumes that the validators know the detection efficiency for each sensor, the total number of adversarial sensors, and the probability of receiving the information from the sensor.

PeerReview Detection System. Haeberlen, Kouznetsov and Druschel [3] designed a detection system called PeerReview assuming a certain degree of observability of Byzantine behaviour. Our model does not aim for detection, but rather supports the analysis of resilience under Byzantine attacks.

Byzantine Attacks as Noncooperative Games. Hespanha [4] models Byzantine attacks as a noncooperative game where the objective of the adversaries is to force validators to accept wrong readings. That work focuses on the construction of Nash equilibria for this game; while our results focus on how the resilience of the network is influenced by its structure.

Off-Chain Data Providers. Several so-called "oracle" data providers (e.g., Chainlink [2]) seek to supply trustworthy real-world data to smart contracts by relying on cryptographic signatures of intermediate data providers (*i.e.,* "end-to-end integrity"). However, the integrity of the data path does not suffice when the validity of the original raw data is at issue, which is a serious concern in distributed sensor networks [10]. Our work analyzes the degree to which the raw data can be trusted.

REFERENCES

[1] Garima Agrawal and Soumen Maity. 2021. The Small Set Vertex Expansion Problem. *Theor. Comput. Sci.* 886 (2021), 84–93.

[2] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. 2021. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs* 1 (2021), 1–136.

[3] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2006. The Case for Byzantine Fault Detection.. In *HotDep*.

[4] João P Hespanha. 2021. Sensor manipulation games in cyber security. *Game Theory and Machine Learning for Cyber Security* (2021), 137–148.

[5] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (jul 1982), 382–401. https://doi.org/10.1145/357172.357176

[6] Pangfeng Liu and Sandeep N. Bhatt. 2000. Experiences with Parallel N-Body Simulation. *IEEE Trans. Parallel Distributed Syst.* 11, 12 (2000), 1306–1323.

[7] Stefano Marano, Vincenzo Matta, and Lang Tong. 2008. Distributed detection in the presence of Byzantine attacks. *IEEE Transactions on Signal Processing* 57, 1 (2008), 16–29.

[8] John H. Reif. 2009. Mechanical Computing: The Computational Complexity of Physical Devices. In *Encyclopedia of Complexity and Systems Science*. Springer, 5466–5482.

[9] John H. Reif and Stephen R. Tate. 1993. The Complexity of N-body Simulation. In *ICALP (Lecture Notes in Computer Science, Vol. 700)*. Springer, 162–176.

[10] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A. Selcuk Uluagac. 2021. A Survey on Sensor-Based Threats and Attacks to Smart Devices and Applications. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1125–1159. https://doi.org/10.1109/COMST.2021.3064507

[11] Ziteng Sun, Chuang Zhang, and Pingyi Fan. 2016. Optimal Byzantine attack and Byzantine identification in distributed sensor networks. In *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 1–6.

[12] N. N. Vasiliev and D. A. Pavlov. 2017. Computational complexity of the initial value problem for the three body problem. *CoRR* abs/1704.08762 (2017).

[13] Feng Wang, Erika Camacho, and Kuai Xu. 2009. Positive Influence Dominating Set in Online Social Networks. In *COCOA (Lecture Notes in Computer Science, Vol. 5573)*. Springer, 313–321.

[14] Feng Wang, David Hongwei Du, Erika Camacho, Kuai Xu, Wonjun Lee, Yan Shi, and Shan Shan. 2011. On positive influence dominating sets in social networks. *Theor. Comput. Sci.* 412, 3 (2011), 265–269.

[15] Xu Zhu, Jieun Yu, Wonjun Lee, Donghyun Kim, Shan Shan, and Ding-Zhu Du. 2010. New dominating sets in social networks. *J. Glob. Optim.* 48, 4 (2010), 633–642.

[16] Feng Zou, Zhao Zhang, and Weili Wu. 2009. Latency-Bounded Minimum Influential Node Selection in Social Networks. In *WASA (Lecture Notes in Computer Science, Vol. 5682)*. Springer, 519–526.

A PROOF OF MAIN RESULTS

In this section, we prove our main results.

A.1 Infeasibility Result

We first formalize Theorem 2.1 and prove our main infeasibility result.

THEOREM A.1. *Building a perfect validator is equivalent to defining a perfect physical sensor model.*

We postulate that perfect physical modelling is infeasible in practice; combined with Theorem A.1, we conclude that it is infeasible to build a perfect validator in practice. Note that it is trivial to build a validator that achieves *perfect liveness* or *perfect soundness* respectively, by imposing the validator to accept every report or reject every report. The impossibility lies in achieving both at the same time.

For a sensor v, we denote by $W(v)$ the true but hidden value for the sensor in real world. A simulation M is a program that, upon the input a sensor v, outputs its simulation $M(v)$. A simulation M is said to be *ϵ-close* if $W(v) - \epsilon \leq M(v) \leq W(v) + \epsilon$ for all v's. We postulate that, when ϵ is sufficiently small, a ϵ-close simulation is not feasible. We support our postulate by an abundance of references. A famous result in physical simulation by Reif and Tate [9] states that simulating the n-body problem is PSPACE-hard to approximate within poly(n) bits of accuracy. Here, the n-body simulation problem is, "given initial positions and velocities of n particles that have pair-wise force interactions, simulate the movement of these particles so as to determine the positions of the particles at a future time" [9]. For infeasibility of simulating the n-body problem in practice, see [6] which attempts parallel classical computing. For more infeasibility result on simulating physical systems, we refer readers to Reif's survey [8]. In particular, Vasiliev and Pavlov showed that complexity lower bound for the three-body problem is not bounded by any polynomial, and the authors reckoned that the choice of the three-body problem is not crucial and the results can be extended to "other systems where trajectories with a complex dynamical behavior can be constructed and analyzed by methods of symbolic dynamics" [12].

We say a validator is *ϵ-perfect* if, upon input a sensor v's reading and its neighbors' readings, it accepts v if its reading is ϵ-close to $W(v)$. Suppose for now that we can build a ϵ-perfect validator, then we can build an ϵ-close simulation M as follows. First, fix a feasible range for a sensor reading; e.g. 0 to 100 degrees when water temperature. Then, the machine iterates $0, \epsilon, 2\epsilon, \ldots,$ through 100, and submits the iterated value to the ϵ-perfect validator When the validator accepts, the machine outputs that value as its simulation. It is clear that the machine we constructed is an ϵ-close simulation. Since that is postulated to be impossible in practice, an ϵ-perfect validator is impossible in practice.

A.2 Proof of Theorem 2.2

Recall the statement of Theorem 2.2: Given a DSN $G = (V, E)$, it is NP-hard to determine if there exists an arrangement of k Byzantine sensors such that at least half of the remaining honest sensors are blocked.

Recall that a Positive Influence Dominating Set (PIDS) for a graph $G = (V, E)$ is a subset $D \subseteq V$ such that any node v in $V \setminus D$ has at least $\left\lceil \frac{d(v)}{2} \right\rceil$ neighbors in D, where $d(v)$ is the degree of node v [16]. A closely related variant is called Total Positive Influence Dominating Set (TPIDS), where the domination property holds for all $v \in V$, i.e. including those in D [13] (In fact, [14] calls this variant PIDS). Consider the decision problem for PIDS (*resp.* TPIDS): given a graph $G = (V, E)$ and a positive integer k, does there exist a PIDS (*resp.* TPIDS) D of size at most k? Both decision problems are shown to be NP-complete and even APX-hard [14–16]. We are now ready to prove Theorem 2.2.

Proof of Theorem 2.2. We show this by reducing the 4-constrained PIDS problem to Blocking. Given an instance (G, k) of the 4-constrained PIDS problem, i.e., where n (number of nodes in G) $> 4k$, we construct G' as follows:

G' is G together with a clique W of $n - k + 1$ nodes. In W, there is a distinguished node w_0 that is connected to every node of G. We set $k' := k + 1$. G' has $2n - k + 1$ nodes. In any assignment, $k + 1$ nodes are Byzantine, so there are $2n - k + 1 - (k + 1) = 2(n - k)$ honest nodes. For the assignment to be a Blocking set, at least half of those honest nodes must be blocked, i.e., there should be at least $(n - k)$ blocked nodes.

Let D be a dominating set for G. We note that:

Claim: Any dominating set for G induces a blocking set for G'.

Proof. Let $D+\{w_0\}$ be the Byzantine nodes creates a blocking set for G'. Consider a honest node x in G. This node is not in D. As D is a dominating set for G, we know that $2|\text{neighbors of } x \text{ from } G \text{ in } D| \geq \text{degree}(x, G)$.

Now we calculate:

$$\begin{aligned}
2|\text{byzantine neighbors of } x \text{ in } G'| &= 2(|\text{neighbors of } x \text{ from } G \text{ in } D| + 1) \quad (\text{ the } +1 \text{ is node } w_0)\\
&= 2|\text{neighbors of } x \text{ from } G \text{ in } D| + 2\\
&\geq \text{degree}(x, G) + 2 \quad (\text{ by PIDS property for } G)\\
&= \text{degree}(x, G') + 1 \quad (\text{ as } x \text{ is connected to } w_0 \text{ in } G')
\end{aligned}$$

Therefore, x is blocked. The number of nodes in G that are not in D is $(n - k)$. As all of these are blocked, $D + \{w_0\}$ is a blocking set for G'. □

Conversely, we note that:

Claim: Any blocking set for G' induces a dominating set for G

Proof. Let B' be a blocking set for G'. From B', we show how to construct a (possibly different) solution B'' which consists of w_0 and k nodes of G, and where every node in G that is not in B'' is blocked.

In such a solution, we have that for any node x of G not in B'':

$$2|\text{neighbors of } x \text{ in } B''| \geq 1 + \text{degree}(x, G'), \quad \text{i.e.,}$$
$$2 * (|\text{neighbors of } x \text{ in } B'' \text{ and in } G| + 1) \geq 1 + 1 + \text{degree}(x, G)$$

Simplifying, we get:

$$2|\text{neighbors of } x \text{ in } B'' \text{ and in } G| \geq \text{degree}(x, G)$$

Hence, the nodes in B'' and G form a dominating set for G.

□

We now consider the given solution B' and show how to transform it to a solution B'' with the properties above. Note that every node of W is either Byzantine or unblocked. Consider a honest node y in W. If y is w_0, then to block w_0, we must have

$$\begin{aligned}
2|\text{Byzantine neighbors of } w_0 \text{ in } G'| &\geq 1 + \text{degree}(w_0, G') \quad \text{i.e.,} \\
2|\text{Byzantine neighbors of } w_0 \text{ in } G'| &\geq 1 + n + (n-k) \quad (\text{as } w_0 \text{ is connected to all of } G \text{ and the other } w\text{'s}) \\
&\geq 1 + 2n - k \\
&\geq 1 + 8k + 2 - k \quad (\text{as } n > 4k), \text{ i.e.,} \\
n &\geq 4k+1) \\
&> 2(k+1)
\end{aligned}$$

Consider any other node y in W. To block y, we must have

$$\begin{aligned}
2|\text{byzantine neighbors of } y \text{ in } G'| &\geq 1 + \text{degree}(y, G') \quad \text{i.e.,} \\
2|\text{byzantine neighbors of } y \text{ in } G'| &\geq 1 + (n-k) \quad (\text{as } n > 4k, \text{ i.e., } n \geq 4k+1) \\
&\geq 3k+2 \\
&> 2(k+1) \quad (\text{as } k > 0)
\end{aligned}$$

But this means that y must have strictly more than $k+1$ Byzantine neighbors, which breaks the budget. So y must be unblocked. The proof in (a) implies that all the blocked nodes are in G. As B' is a blocking set, there must be at least $(n-k)$ blocked nodes in G. Let k_0 be the number of Byzantine nodes in G in B'.

As at least $n-k$ nodes are honest and blocked in G, k_0 must be at most k. Indeed, if it is $k+1$, then G has at least $n-k+k+1 > n$ nodes, a contradiction. Hence, W has k_1+1 Byzantine nodes, for $k_1 \geq 0$ such that $k_1 + k_0 = k$.

G has $(n-k_0)$ honest nodes and at least $(n-k)$ of those are blocked. We pick a subset X of G that contains exactly $(n-k)$ blocked nodes. If w_0 is not Byzantine in B', move the Byzantine label from some node of W to w_0. This cannot unblock any node in X as that node only gains a Byzantine neighbor.

Now mark the remaining k_1 honest nodes in G outside X as Byzantine, moving the Byzantine labels from the k_1 non-w_0 Byzantine nodes in W. This cannot unblock any node in X, as any such node only gains Byzantine neighbors. In this new assignment B'', the Byzantine nodes are either w_0 or in G, and $n-k$ honest nodes are blocked. Hence, B'' is a blocking set, and it also meets the requirements above.

□

Remark. Note that the proof actually shows the following slightly stronger result, since the constructed G' is connected and the reduction is clearly an L-reduction.

Corollary A.2. *Given a connected* DSN $G = (V, E)$*, it is APX-hard to determine if there exists an arrangement of k Byzantine sensors such that at least half of the remaining honest sensors are blocked.*

A.3 Proof of Theorem 2.3

Recall the statment of Theorem 2.3: Given a DSN $G = (V, E)$ and numbers $t \leq k < n$, it is NP-hard to determine if there exists an arrangement of k Byzantine sensors such that at least t of them are supported (i.e., poisoned).

Our proof of Theorem 2.3 is inspired by Agrawal and Maity's study on the Small Set Vertex Expansion (SSVE) problem [1].

Proof of Theorem 2.3. We denote an input to our problem to be (G, k, t). A moment of reflection should convince the reader that it suffices to prove the NP-hardness of the following problem: given a DSN $G = (V, E)$, does there exist a set of t sensors which can be simultaneously supported by $k - t$ Byzantine sensors? We shall present a polynomial time reduction from CLIQUE which will finish the proof.

Given an input (G, k) to CLIQUE, we construct the input $(G', \frac{k^2+k}{2}, \frac{k^2-k}{2})$ to our problem: $G' = (V', E')$ where $V' = \{e \in E\} \cup \{v^i \mid v \in V, i \in \{1,2\}\} \cup \{p^i_j \mid i \in \{1,2\}, j \in \{1,2,\dots,k^2+k\}\}$ and $E' = \{\{p^i_j, v^i\} \mid i \in \{1,2\}, j \in \{1,2\dots,k^2+k\}, v \in V\} \cup \{\{e, v^i\} \mid i \in \{1,2\}, v \in e\}$. See Figure 2 for a visualization of the construction. It remains to show this is a valid reduction.

Suppose (G, k) is an Yes instance to CLIQUE and let C_k be the k-clique subgraph in G, then by choosing the Byzantine sensors to be $\{e \in C_k\} \cup \{v^1 \mid v^1 \in e\}$, we find a Byzantine set of size $\frac{k^2-k}{2} + k = \frac{k^2+k}{2}$ with supported set $\{e \in C_k\}$ of size $\frac{k^2-k}{2}$. On the other hand, suppose the input $(G', \frac{k^2+k}{2}, \frac{k^2-k}{2})$ is an Yes instance to our problem, we claim that the supported set must be all "edge" vertices and therefore the remaining Byzantine sensors form a $\frac{k^2+k}{2} - \frac{k^2-k}{2} = k$-clique in G. Indeed, suppose this is not the case, then there must exist some v^i or some p^i_j in the Byzantine sensors. If p^i_j for some i, j is Byzantine, then since the only neighbors of p^i_j are v_i's, to suppose such p^i_j we must have some v^i in the Byzantine sensors, therefore we reduce to the case that there exists some v^i is Byzantine. However, the degree of v^i is greater than $k^2 + k$, and therefore to support such vertex we need at least $\frac{k^2+k}{2}$ other Byzantine sensors, which will exceed the budget since we now have at least $\frac{k^2+k}{2} + 1$ Byzantine sensors. This finishes the proof. □

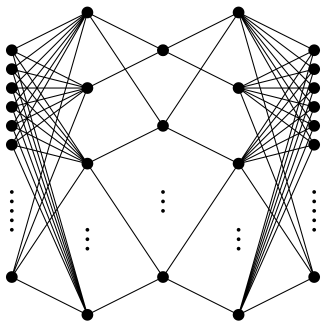


Fig. 2. Construction in the proof of Theorem 2.3

Remark. The proof of Theorem 2.3 can be easily adapted so that a Byzantine sensor is said to be $\frac{1}{r}$*-supported*, where $r \in \mathbb{N}$, if more than $\frac{1}{r}$ neighbors are Byzantine. The number $\frac{1}{r}$ is called the *validator robustness*: A higher $\frac{1}{r}$ means implies that more Byzantine neighbors are needed to support a single Byzantine sensor, which yields a higher security of the network. We simply define $G' = (V', E')$ where $V' = \{e \in E\} \cup \{v^i \mid v \in V, i \in [r]\} \cup \{p^i_j \mid i \in [r], j \in [k^2+k]\}$ and $E' = \{\{p^i_j, v^i\} \mid i \in [r], j \in [k^2+k], v \in V\} \cup \{\{e, v^i\} \mid i \in [r], v \in e\}$ while the input is $(G', \frac{k^2+k}{2}, \frac{k^2-k}{2})$. The proof of the following theorem will follow the same reasoning in the proof of Theorem 2.3.

Theorem A.3. *Given a* DSN $G = (V, E)$*, it is NP-hard to determine if there exists an arrangement of k Byzantine sensors such that at least t of them are $\frac{1}{r}$-supported.*