IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 13, 2020, accepted February 25, 2020, date of publication March 4, 2020, date of current version March 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978458

Neural Networks-Aided Insider Attack Detection for the Average Consensus Algorithm

GANGQIANG LI[1], SISSI XIAOXIAO WU[1,2], (Member, IEEE), SHENGLI ZHANG[1], (Senior Member, IEEE), AND QIANG LI[2,3], (Member, IEEE)

[1]College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China
[2]Peng Cheng Laboratory, Shenzhen 518055, China
[3]School of Information and Communication Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu 611731, China

Corresponding author: Sissi Xiaoxiao Wu (xxwu.eesissi@szu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61701315, in part by the Shenzhen Technology Research and Development Fund under Grant JCYJ20170817101149906, and in part by the Shenzhen University Launch Fund under Grant 2018018.

ABSTRACT To support the big-data processing needs of large-scale deployments of smart devices, there is significant interest in moving from cloud-computing to multi-agent (fog-computing) models, given these algorithms scalability and self-healing properties with respect to nodes and link failures. However, these algorithms are often based on the average consensus primitive, which is, unfortunately, vulnerable to data injection attacks. Recognizing this challenge, this work proposes three novel methods for detecting and localizing adversarial nodes using neural network (NN) models. The methods proposed are based on fully distributed algorithms, wherein each node locally updates its local states by exchanging information with its neighbors without supervision. Compared to the state-of-the-art, the proposed approach leverages the automatic learning characteristics of NN to reduce the dependence on pre-designed models and human expertise in complex internal attack scenarios. Simulation results show that the NN-based methods can significantly improve the attacker detection and localization performance.

INDEX TERMS Gossip algorithm, average consensus, neural network (NN), insider attack, detection and localization.

I. INTRODUCTION

Nowadays, distributive resource allocation is expanding its footprint and attracting worldwide research and development efforts [1]–[4]. Typical examples of cloud computing [5], [6] applied to the Internet of Things (IoT) are 5G bandwidth management [7], [8] and wireless network utility maximization [9], [10], and federated learning [11], [12]. However, to achieve better responsiveness and scalability, multi-agent algorithms are considered a preferable alternative to cloud-based solutions for a range of applications such as real-time control, resource management, [13], [14], and also resource allocation problems [15]–[17]. When they aim at a coordinated response, the majority of these algorithms incorporate a peer-to-peer consensus primitive, that leads to a consistent globally optimum decision [18]–[20]. In consensus protocols, interacting nodes are randomly selected and locally execute computations with a node-to-node message passing protocol. Specifically, each node independently runs a consensus protocol in an iterative manner, where at each iteration it exchanges local information with its neighbors and then updates its local states. Such algorithms are inherently robust against intermittent communication and provide a certain degree of privacy for the participating agents compared to cloud-based centralized solutions. Despite such appealing features, decentralized algorithms are inherently vulnerable to insider data injection attacks, since each node implements the computation and message passing protocol iterations without any supervision [21], [22].

Generally speaking, the way malicious agents can be detected depends on the specific attack strategy. Attackers may simply jam the distributed algorithm by injecting random data that impede the convergence [23], while in *Sybil-type attacks* they simultaneously assume multiple

The associate editor coordinating the review of this manuscript and approving it for publication was Thanh Ngoc Dinh.

unique IDs, overwhelming the rest of the system and steering it towards specific objectives [24], [25]. This behavior at the application or at the physical communication layer depart significantly from the norm. In an insider attack, however, nodes can collude to force the system to converge to their target outcome following strategies that are harder to discriminate [22], [26], [27].

This is the kind of attack model we focus on in this paper, in which attackers nodes are coordinated and act as stubborn agents [28] sending messages to their peers that contain a constant bias [21], [22], [27], [29], [30]. As indicated[1] in [22], [27], under such coordinated attack the network always converges to a final state equal to the bias.

In the literature, a popular approach for the detection of the nodes anomalous behavior is to have each node compute a neighbor dependent score using the messages received during the protocol execution. For instance, in [23], the authors leveraged the insight that the presence of attackers may slow down the network convergence to design a score that signals a possible attack. In [32], the authors designed a comparative score, searching for a node message patterns that are dissimilar relative to the messages of other neighbors. In [33] the authors showed analytically and experimentally that it is possible to determine if messages come from the same source at the physical layer, providing a solution for the detection of Sybil-type attacks. In [22], [27] the authors proposed several strategies based on carefully crafted metrics for detecting and localizing attackers. While these methods have reasonable performance, the score design is somewhat ad-hoc and other characteristics of the peer-to-peer interactions may reveal the attack more efficiently and accurately. Our idea is to leverage deep-learning to detect attacks in multi-agent algorithms. In the area of network security, and network intrusion detection in particular, neural networks (NNs) have been extremely popular tools for classifying anomalous behavior. For example, in [34], use NN for external intrusion detection. Similarly, [35] used a NN classifier in a hierarchical network intrusion detection system, pre-processing the data to produce a statistical score as the input of the NN. In [36], a NN-based predictor, fusing past and present neighbors states, was proposed to identify malicious nodes in a wireless sensor network. The authors in [37] utilized NNs to detect malicious website visitors and defend against the distributed denials of service (DDoS) attacks. In [38], deep learning algorithms including NN are utilized to detect the internet intrusion for vehicles, while their focus is based on detecting different types of cyber-attack. The work in [39] utilized NN to detect insider attacker. However, they simply explored neighbors' information from their steady states and thus the performance gain is restricted. While the few examples above are insufficient to survey the literature on NN approaches for network intrusion detection, to the best of our knowledge the application of NN classifiers to detect attacks to multi-agent algorithms is new. The purpose of this work is to fill in this gap, by providing evidence of the potential of NN in this context.

While our NN model and our training philosophy can be applied to a wide set of multi-agent algorithms and attack scenarios, we focus on testing the approach on a case that has been thoroughly studied in [22], [27], to facilitate the comparison. The learning mechanism in this work is supervised such that NNs are trained in an offline manner with labeled data collecting from the normal nodes. Specifically, our strategy applies NN to three type of features extracted from the training data. The first two features are analogous to the ones used in [22], [27]: we call our classifiers a NN-based time difference (TDNN) strategy (a simple version was first introduced in [39]), a NN-based spatial difference (SDNN) strategy. We add to that a hybrid model that combines the discrete Fourier transform (DFT) of the sequence of messages received by a neighbor, and call this method the NN-based frequency difference (FDNN) strategy. These three methods run in a fully distributed manner wherein each normal node detects and localizes attackers independently. As our numerical results in Section IV confirm, compared with the statistical score-based methods, the proposed NN-based methods: 1) can automatically learn from the training data and reduce the dependence on pre-designed models and human expertise; 2) can adaptively scale the output metrics such that the decision thresholds for detection and localization tasks can be conveniently set between 0 and 1; 3) can significantly improve the performance for attacker detection and localization as NN has the ability to fully explore information underlying the data; 4) are more robust in the case that the statistics of the real environment is mismatched with the prior information which we have used to design the score-based method or train the NN. Of course, possible disadvantages are that they incur in training cost, and that these approaches are not suitable for zero-day attacks.

The paper is organized as follows. In the Section II, the attacker model based on the average consensus algorithm and two score-based strategies for detecting and localizing the internal attack are described. In Section III, we propose the generalized NN architecture for the detection/localization tasks and introduce three detection and localization methods

Algorithm 1 The Randomized Average Consensus Protocol

Input: initial states: $x_i^k(0) = \gamma_i^k \ \forall\, i \in \mathcal{V}$.
for $t = 1, 2, \cdots, T$
- Uniformly wake up a random node $i \in \mathcal{V}$
- Node i select a node $j \in \mathcal{N}_i$ with probability P_{ij}
- Update the states of node i and node j

$$x_i^k(t+1) = x_j^k(t+1) = (x_i^k(t) + x_j^k(t))/2 \tag{1}$$

- The other nodes keep their original states
  $\forall v \neq i, j, x_v^k(t+1) = x_v^k(t)$

end for

[1]The convergence properties of average consensus gossiping with stubborn agents were studied in [31].

based on NNs. Simulation results are illustrated in Section IV. We conclude and further discuss with simulation results in Section V.

II. SYSTEM MODEL BASED ON AVERAGE CONSENSUS

We model the network as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_N\}$ represents the set of all nodes with $|\mathcal{V}| = N$. The set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(v_i, v_j) \in \mathcal{E}$ represents the communication link in the network. v_i and v_j can directly communicate with each other if there is a communication link (v_i, v_j) between them. The neighbor set of a node i is defined as:

$$\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}\}, \quad \forall i \in \{1, 2, \cdots, N\}. \tag{2}$$

Normally, all the nodes in the network follow a randomized gossip protocol. That is, the node i interacts with its direct neighbor $j \in \mathcal{N}_i$ to update its local state information. Without loss of generality, we assume that this consensus algorithm runs a total of K instances independently. In each instance, we let $x_i^k(0) = \gamma_i^k$ represents the initial state of a normal node i, γ_i^k is a stationary discrete random variable. The average consensus protocol is described by the recursion:

$$\boldsymbol{x}^k(t+1) = \boldsymbol{A}^k(t)\boldsymbol{x}^k(t), \quad t = 1, 2, \ldots, T. \tag{3}$$

where the random vector $\boldsymbol{x}^k(t) = (x_1^k(t), \ldots, x_N^k(t))^\top \in \mathbb{R}^N$ represents the states of all the nodes at the tth iteration, $\boldsymbol{A}^k(t) \in \mathbb{R}^{N\times N}$ is the weight matrix at instance k and time t. To simplify notations, we shall drop the superscript k on $\boldsymbol{A}(t)$ from now on. The asynchronous average consensus protocol we consider (in the absence of attackers) is summarized in Algorithm 1. In the tth time slot, a node i select a node j at random, and both update their respective states to be equal to the average. The weight matrix is:

$$\boldsymbol{A}_{ij}(t) = \boldsymbol{I} - \frac{(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^\top}{2} \tag{4}$$

where $\boldsymbol{e}_i = [0, \cdots, 0, 1, 0, \cdots, 0]^\top$ is an $N \times 1$ unit vector with the ith element equal to 1. By defining $[\boldsymbol{P}]_{ij} = P_{ij}$ and $\boldsymbol{\Sigma} = diag([\Sigma_1, \cdots, \Sigma_N])$ as a diagonal matrix with $\Sigma_i = \sum_{j=1}^N (P_{ij} + P_{ji})$, the expected weight matrix is:

$$\overline{\boldsymbol{A}} = \mathbb{E}\left[\boldsymbol{A}(t)\right] = \boldsymbol{I} - \frac{1}{2N}\boldsymbol{\Sigma} + \frac{\boldsymbol{P} + \boldsymbol{P}^\top}{2N}. \tag{5}$$

It can be verified that $\overline{\boldsymbol{A}}$ is non-negative, symmetric and doubly stochastic. The expected states can be expressed as

$$\mathbb{E}[\boldsymbol{x}^k(t)|\boldsymbol{x}^k(0)] = \overline{\boldsymbol{A}}\mathbb{E}\left[\boldsymbol{x}^k(t-1)|\boldsymbol{x}^k(0)\right] = \overline{\boldsymbol{A}}^t\boldsymbol{x}^k(0),$$

Fact 1 [40]: Suppose the network is connected in expectation, i.e., *if* $\lambda_2(\overline{\boldsymbol{A}}) < 1$. *The state of node* $i \in \mathcal{V}$ *converges to the average of network initial states*

$$\lim_{t\to\infty} \boldsymbol{x}^k(t) = \boldsymbol{1}^\top\boldsymbol{x}^k(0)/N. \tag{6}$$

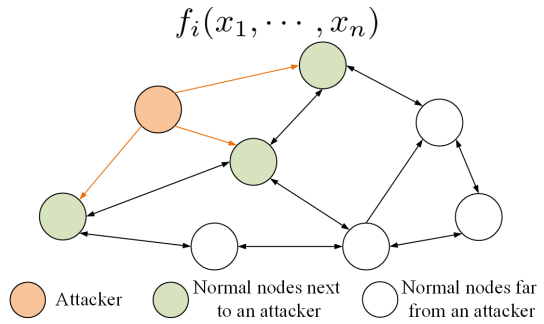


FIGURE 1. An attacked-communication network.

A. THE ATTACK MODEL

In our model, attackers are defined as nodes whose states cannot be changed by the others. As illustrated in Fig. 1, we divide the nodes into a normal nodes set $\mathcal{V}_r$ and an attacker nodes set $\mathcal{V}_s$, i.e., $\mathcal{V} = \mathcal{V}_r \cup \mathcal{V}_s$ and $N = |\mathcal{V}_r| + |\mathcal{V}_s|$. If node j is an attacker, its update will follow a modified rule:

$$x_j^k(t+1) = \alpha^k + m_j^k(t). \tag{7}$$

where α^k represents the desired value of the attackers in kth instance, and $m_j^k(t)$ is an artificial noise generated by the attackers to confuse the network. In order to disguise the behavior of the attackers as normal nodes, the artificial noise decays exponentially with time, emulating the converge trends expected in the algorithm. To describe the attack model, let us partition the full state vector as:

$$\boldsymbol{x}^k(t) = \begin{pmatrix}\boldsymbol{s}^k(t)^\top & \boldsymbol{r}^k(t)^\top\end{pmatrix}^\top \tag{8}$$

where $\boldsymbol{s}^k(t) \in \mathbb{R}^{|\mathcal{V}_s|}$ and $\boldsymbol{r}^k(t) \in \mathbb{R}^{|\mathcal{V}_r|}$ are the state vectors of attackers and normal nodes respectively. To model the fact that an attacker's state cannot be changed by the others, the weight matrix follows that $A_{ij}(t) = 1$ if $i = j \in \mathcal{V}_s$, and $A_{ij}(t) = 0$ if $i \in \mathcal{V}_s, j \in \mathcal{V}$. As such, the expected weight matrix $\overline{\boldsymbol{A}}$ admits the block structure:

$$\overline{\boldsymbol{A}} = \begin{pmatrix}\boldsymbol{I} & \boldsymbol{0} \\ \overline{\boldsymbol{B}} & \overline{\boldsymbol{D}}\end{pmatrix}, \tag{9}$$

where $\overline{\boldsymbol{B}} \in \mathbb{R}^{|\mathcal{V}_r|\times|\mathcal{V}_s|}$, $\overline{\boldsymbol{D}} \in \mathbb{R}^{|\mathcal{V}_r|\times|\mathcal{V}_r|}$ are the sub-matrices between attacker and normal nodes, and between normal and normal nodes, respectively. In this scenario, the attackers, if undetected, will succeed in steering the final states:

Fact 2 [27]: Under the attack model, the state of node $i \in \mathcal{V}$ *in each* k *instance converges to the desired value* α^k, i.e.,

$$\lim_{t\to\infty} \boldsymbol{x}^k(t) = \alpha^k\boldsymbol{1}. \tag{10}$$

We remark that (7) describes a coordinated attack model where the desired value of attackers are the same. As pointed out by [27], this is a challenging scenario as the attackers' identities are not easily revealed. Fig.3 in [27] has shown the evolution of all node states in an example of the average consensus algorithm when one attacker is present. Fact 2 proved that the average consensus algorithm will converge when the insider attacker is present in the network.

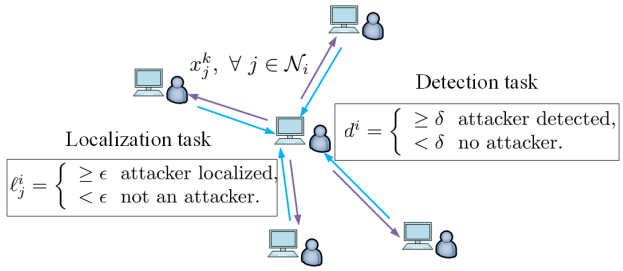


FIGURE 2. An illustration of the detection and localization tasks in [27].

B. ATTACKER DETECTION AND LOCALIZATION

The detection and localization tasks in this paper are considered under the same hypotheses as those in our previous work [27]. The first task is to detect if there is attacker(s) in the network. This network detection task runs in a fully decentralized fashion at each normal node $i \in \mathcal{V}_r$, *i.e.,*

$$\begin{aligned} &\mathcal{H}_0 : \mathcal{V}_s = \emptyset, \text{There is no attacker in the network,} \\ &\mathcal{H}_1 : \mathcal{V}_s \neq \emptyset, \text{Attacker is present in the network.} \end{aligned} \tag{11}$$

Under $\mathcal{H}_1$, the second task is to observe whether there are attackers in the neighborhood of node i. The *neighborhood detection* task is defined as follows:

$$\begin{aligned} &\mathcal{H}_0^i : \mathcal{N}_i \cap \mathcal{V}_s = \emptyset, \text{No neighbor is an attacker,} \\ &\mathcal{H}_1^i : \mathcal{N}_i \cap \mathcal{V}_s \neq \emptyset, \text{At least an attacker neighbor.} \end{aligned} \tag{12}$$

When $\mathcal{H}_1^i$ is detected at node i, the third task is to locate the attacker from the neighbors. We define the *neighborhood localization* task as follows:

$$\begin{aligned} &\mathcal{H}_0^{ij} : \ j \notin \mathcal{V}_s, \text{Node } j \text{ is not an attacker,} \\ &\mathcal{H}_1^{ij} : \ j \in \mathcal{V}_s, \text{Node } j \text{ is an attacker.,} \quad \forall\, j \in \mathcal{N}_i. \end{aligned} \tag{13}$$

If $\mathcal{H}_1^{ij}$ is true, we say that the attacker is localized. Then node i will disconnect from the attacker in the future communication.

An illustration of the detection and localization tasks is shown in Fig. 2. We denote $\boldsymbol{x}_i^k$ as the state vector collected by node i in kth instance where x_j^k is the state of node $j \in \mathcal{N}_i$, which can be directly obtained by node i. The detection and localization tasks are described below:

$$\boldsymbol{x}_i^k := [x_i^k, x_1^k, \cdots, x_j^k, \cdots, x_{|\mathcal{N}_i|}^k]^\top \quad \forall\, j \in \mathcal{N}_i. \tag{14}$$

$$d^i = \mathrm{DT}(\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^K) \underset{\mathcal{H}_0^i}{\overset{\mathcal{H}_1^i}{\gtrless}} \delta, \qquad \ell_j^i = \mathrm{LT}(\boldsymbol{x}_i^1, \cdots, \boldsymbol{x}_i^K) \underset{\mathcal{H}_0^{ij}}{\overset{\mathcal{H}_1^{ij}}{\gtrless}} \epsilon.$$

where $\boldsymbol{\ell}^i = [\ell_1^i, \cdots, \ell_{|\mathcal{N}_i|}^i]^\top$ is the metric vector of neighbors for the purpose of localization, K is the total instances of the average consensus algorithms observed, $\delta > 0$ and $\epsilon > 0$ are some pre-designed thresholds.

Remark 1: $\mathrm{DT}(\bullet)$ *and* $\mathrm{LT}(\bullet)$ *are detection statistical decision functions judiciously designed for detection and localization tasks. A node* $i \in \mathcal{V}_r$ *through these decision functions to calculate the criterion indexes to discern malicious nodes.*

Next we review the score based strategies for detection and localization tasks proposed in [22], [27].

C. SCORE-BASED STRATEGIES FOR INTERNAL ATTACKS

1) TEMPORAL DIFFERENCE STRATEGY

The first score-based strategy in [22], [27] is called the temporal difference method (TDM). Assume that we already have some prior information about the mean of the attacked value and the mean of the intial states of the normal nodes. Usually, $\mathbb{E}[x_s^k(0)] = \overline{\alpha} \neq \overline{\gamma} = \mathbb{E}[x_j^k(0)]$, the expected initial value of an attacker $s \in \mathcal{V}_s$ is different from the normal node $j \in \mathcal{V}_r$. When $t \to \infty$ the network converges to $\mathbb{E}[x_s^k(\infty)]$. Then each normal node $i \in \mathcal{V}_r$, in a decentralized fashion, evaluates the following score[2]:

$$\xi_{ij} := \tfrac{1}{K} \textstyle\sum_{k=1}^K (x_j^k(T) - x_j^k(0)), \quad j \in \mathcal{N}_i. \tag{15}$$

Herein, $x_j^k(T), x_j^k(0)$ are respectively the last and the first observed state value for node j in the neighborhood of node i. To identify whether there is an attacker on the network, the following detection criteria is used:

$$D_1^i := \tfrac{1}{|\mathcal{N}_i|} \textstyle\sum_{j \in \mathcal{N}_i} |\xi_{ij} - \overline{\xi}_i| \underset{\mathcal{H}_1^i}{\overset{\mathcal{H}_0^i}{\lessgtr}} \delta_I, \tag{16}$$

where $\overline{\xi}_i = (1/|\mathcal{N}_i|) \sum_{j \in \mathcal{N}_i} \xi_{ij}$, $\delta_I > 0$ is a pre-designed threshold. The intuition underlying (16) is that $\mathbb{E}[D_1^i] = 0$ when the attacker is not present while $\mathbb{E}[D_1^i] \neq 0$ otherwise.

For localization task, the $\mathcal{H}_1^{ij}$ and $\mathcal{H}_0^{ij}$ events are distinguished by checking the criteria:

$$L_1^{ij} := |\xi_{ij}| \underset{\mathcal{H}_0^{ij}}{\overset{\mathcal{H}_1^{ij}}{\lessgtr}} \epsilon_I, \qquad \forall\, j \in \mathcal{N}_i\,. \tag{17}$$

where $\epsilon_I > 0$ is a pre-designed threshold.

2) SPATIAL DIFFERENCE STRATEGY

The second strategy, proposed in [27], is the spatial difference method (SDM). Note the expected state in each t satisfies $\mathbb{E}[x_i^k(t) - x_j^k(t)|\mathcal{H}_0] = 0$ when there are no attackers in the network, while $\mathbb{E}[x_i^k(t) - x_j^k(t)|\mathcal{H}_1] \neq 0, j \in \mathcal{N}_i$ when attacker is present in the network. For the detection task, the criterion is:

$$\varphi_{ij}^k := \sum_{t=0}^{T} \Big(x_j^k(t) - \overline{x}_i^k(t) \Big). \tag{18}$$

$$D_2^i := \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \Big(\frac{1}{K} \sum_{k=1}^{K} \varphi_{ij}^k \Big)^2 \underset{\mathcal{H}_1^i}{\overset{\mathcal{H}_0^i}{\lessgtr}} \delta_{II}, \tag{19}$$

where $\overline{x}_i^k(t) = (1/|\mathcal{N}_i|) \sum_{j \in \mathcal{N}_i} x_j^k(t)$ is the average value of node i' neighbors at time t in the kth instance. Notice that φ_{ij}^k is the difference between the value of a neighboring node j and the average value $\overline{x}_i^k(t)$, and then sum up this difference from all the observed consensus iterations. In the above, δ_{II} is a pre-designed threshold.

[2]In practice, each agent evaluates $\Delta_j(t) \triangleq x_j^k(t+1) - x_j^k(t)$ at time tick t and sum it over all the time ticks to obtain ξ_{ij}

For localization task, we consider the following criteria:

$$\tilde{\varphi}_{ij}^{k} := \sum_{t=0}^{T} \left(x_j^k(t) - x_i^k(t)\right) - \varphi_{ii}^{k}, \tag{20}$$

$$L_2^{ij} := \left(\frac{1}{K}\sum_{k=1}^{K} \tilde{\varphi}_{ij}^{k}\right)^2 \underset{\mathcal{H}_1^{ij}}{\overset{\mathcal{H}_0^{ij}}{\lesseqgtr}} \epsilon_{II} \tag{21}$$

where φ_{ii}^k is calculated by node i according to the equation (18). The metric $\tilde{\varphi}_{ij}^k$ is designed to compare a neighbor node j to the node i itself. Similarly, ϵ_{II} are some pre-designed threshold for localizing attackers.

III. NEURAL NETWORKS AIDED DETECTION AND LOCALIZATION

As shown in [27], a well-constructed detection and localization functions is crucial to the detection performance in data injection attacks. However, we observe that in the above score-based strategies, D_1^i, L_1^{ij}, D_2^i and L_2^{ij} are roughly linear functions which fuse the state vector obtained by node i into a scalar score for classification. A natural question that arises is whether there exists *nonlinear* function that yields better statistics for the classification problem at hand. This is a natural application of NN which concatenates layers of neurons to approximate complicated functions. Next, we consider the detection and localization process as a classification problem with the state vector of node i. Specifically, our learning mechanism here is supervised. In particular, we assume that the training center has perceived some side information about the mean of the attacker value and the mean of the normal nodes' initial states. Thus, it can mimic the attacked algorithm in an offline manner and collect the neighbors' data from the normal node which is conducting detection and localization tasks. In this process, since we have known which one is the attacker, the training data is therefore labeled. Once we have trained the NN, it could be deployed on each normal node in the network.

Let $M = \max_i |\mathcal{N}_i|$ be the input dimension of these NNs. In what follows, we describe the general NN architecture for the detection/localization tasks, as well as their training mechanisms. Then, we describe three types of novel features to be used for training the NNs — including the temporal difference NN (TDNN), spatial difference NN (SDNN), and a Fourier transform-based NN (FDNN). For simplicity, we assume that the graph degrees are regular such that $M = |\mathcal{N}_i|$ for all i and relegate the discussions with irregular degree-graphs to Section III-E.

A. DETECTION AND LOCALIZATION TASKS VIA NN

For the detection task, the NN used for the detection is illustrated in Fig. 3 and described below:

$$\boldsymbol{a}^h = f(\boldsymbol{W}^h \boldsymbol{a}^{h-1} + \boldsymbol{b}^h), \quad h = 1, \ldots, n-1; \tag{22}$$

$$d^i = g(\boldsymbol{W}^n \boldsymbol{a}^{n-1} + \boldsymbol{b}^n), \quad d^i \underset{\mathcal{H}_0^i}{\overset{\mathcal{H}_1^i}{\gtrless}} \delta, \tag{23}$$

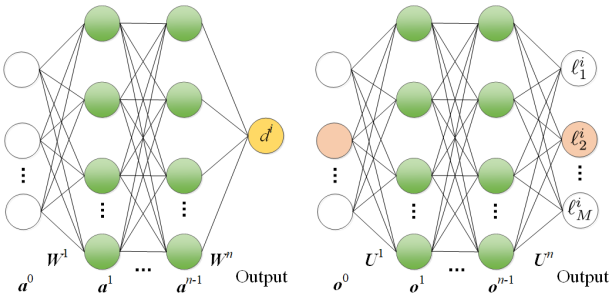


FIGURE 3. NNs method at node i: (Left) NN for detection task, (Right) NN for localization task.

where $\boldsymbol{a}^0$ is the input feature vecctor, $\boldsymbol{W}^h \in \mathbb{R}^{L_h \times L_{h-1}}$ is a weight matrix between hidden layer h and hidden layer $h-1$. $\boldsymbol{b}^h$ is a bias vector, and L_h is the number of neurons in the hidden layer h. f is the activation function and g is a sigmoid function. $\boldsymbol{a}^h, h > 0$ represents the state of the hth hidden layer, and $d^i \in \mathbb{R}$ is the expected output of node i. Lastly, $\delta \in [0, 1]$ is a prescribed threshold for detection task.

For the localization task, a similar NN is used. We assume that an attacker is detected in the neighborhood of node i, *i.e.,* $\mathcal{H}_1^i$ is true. The localization network is:

$$\boldsymbol{o}^h = f(\boldsymbol{U}^h \boldsymbol{o}^{h-1} + \boldsymbol{c}^h), \quad h = 1, \ldots, n-1; \tag{24}$$

$$\boldsymbol{\ell}^i = g(\boldsymbol{U}^n \boldsymbol{o}^{n-1} + \boldsymbol{c}^n), \quad \ell_j^i \underset{\mathcal{H}_0^{ij}}{\overset{\mathcal{H}_1^{ij}}{\gtrless}} \epsilon, \tag{25}$$

wher $\boldsymbol{o}^0$ is the input feature, $\boldsymbol{U}^h$, $\boldsymbol{o}^{h-1}$ and $\boldsymbol{c}^h$ are the weight matrix, hidden layer state and bias vector, respectively. In this task, we train the NN using a one-hot encoding for the output. For example, when the neighbor $j \in \mathcal{N}_i$ is an attacker, the NN should be trained to produce the output $\boldsymbol{\ell}^i = \boldsymbol{e}_j$. Here, $\boldsymbol{\ell}^i \in \mathbb{R}^M$ and $\epsilon \in [0, 1]$ is a pre-described threshold for the localization task. A normal node i will only run this task if $\mathcal{H}_1^i$ is true.

Given the labeled training set, the parameters $\boldsymbol{W}^h$ and $\boldsymbol{b}^h, \forall h$ are randomly initialized and then optimized through minimization of regression error over the training set via the back propagation method [41]. For the localization models, the parameters $\boldsymbol{U}^h$ and $\boldsymbol{c}^h, \forall h$ are initialized and updated in a similar fashion.

B. TEMPORAL DIFFERENCE STRATEGY BASED ON NN (TDNN)

Recall that in TDM, the temporal difference values of the neighborhood are evaluated at each normal node. Based on the metric ξ_{ij} in (15), we propose a TDNN method that, similar to the TDM method, uses the following input feature:

$$\boldsymbol{a}^0 = \boldsymbol{o}^0 = \boldsymbol{\xi}_i = [\xi_{i1}, \xi_{i2}, \cdots, \xi_{iM}]^\top, \tag{26}$$

which can be obtained by node i. Compared to the TDM method, we highlight that the TDNN method uses nonlinear functions (22), (24) leading to higher expressive power.

C. SPATIAL DIFFERENCE STRATEGY BASED ON NN (SDNN)

With the TDNN method, we detect and localize attackers by observing neighbors' information in the initial state and the steady state. A natural extension is to include transient states in the detection task. In fact, a similar strategy has been explored in the SDM method [27] and it is natural to use transient state information as a NN input feature.

For the detection task, as the neighbors' states of node i will be affected by attackers when $0 < t < \infty$, and the malicious node will directly steer the normal nodes to deviate from their true average. The node state in the network only changes slightly over time. We consider the following metric[3]:

$$S_{ij}^k(t) := \left| x_j^k(t) - x_j^k(t-1) \right|, \quad j \in \mathcal{N}_i. \tag{27}$$

$$X_{ij} := \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T} S_{ij}^k(t), \quad j \in \mathcal{N}_i. \tag{28}$$

The input feature for the detection model is

$$\boldsymbol{a}^0 = \boldsymbol{X}_i = [X_{i0}, X_{i1}, \cdots, X_{iM}]^\top. \tag{29}$$

For the localization task, we identify when attackers appear directly in the neighborhood of node i. We have $\mathbb{E}[x_i^k(t) - x_j^k(t)|\mathcal{H}_1^{ij}] \neq \mathbb{E}[x_i^k(t) - x_j^k(t)|\mathcal{H}_0^{ij}]$. This motivates us to consider the following metric:

$$I_{ij}^k(t) := \left| x_i^k(t) - x_j^k(t) \right|, \quad j \in \mathcal{N}_i, \tag{30}$$

$$Y_{ij} := \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T} I_{ij}^k(t), \quad j \in \mathcal{N}_i, \tag{31}$$

where $I_{ij}^k(t)$ is the absolute difference between node i and its neighbor node j. We use the following input feature:

$$\boldsymbol{o}^0 = \boldsymbol{Y}_i = [Y_{i1}, Y_{i2}, \cdots, Y_{iM}]^\top. \tag{32}$$

D. A HYBRID METHOD BASED ON DFT AND NN

Given a length T time series observations obtained for each neighbor of a normal node, both TDNN and SDNN methods accomplish the detection/localization tasks using a one-dimensional summary of the time series. Doing so may fail to capture some of the important transient properties of the sequences of messages that are exchanged. On the other hand, as $T \gg 1$, using the complete time series as input can be computational prohibitive. To remedy this problem, we consider input features that are computed mapping the signals in *frequency domain*, and sampling at a few frequency indices.

Consider a normal node $i \in \mathcal{V}_r$ and his/her neighbor $j \in \mathcal{N}_i$. We define the following metric for the detection task.

$$D_{ij}[f] := \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \left| \sum_{t=1}^{T} S_{ij}^k(t) e^{-\mathbf{i}2\pi tf/T} \right|, \quad j \in \mathcal{N}_i, \tag{33}$$

[3]Each normal node i evaluates $S_{ij}^k(t) \triangleq \left| x_j^k(t) - x_j^k(t-1) \right|$ at time tick t and sum it over all the time ticks to obtain X_{ij}.

where $S_{ij}^k(t)$ was defined in (27), $f \in \mathcal{F} = \{0, \ldots, T-1\}$ is the frequency index, and $\mathbf{i} = \sqrt{-1}$ is the imaginary number.

We employ the following input feature matrix for the Fourier Transform based NN (FDNN) detection model running at node i:

$$\boldsymbol{a}^0 = \boldsymbol{D}_i = [\boldsymbol{D}_{i1}^\top \boldsymbol{D}_{i2}^\top \cdots \boldsymbol{D}_{iM}^\top]^\top \tag{34}$$

where $\boldsymbol{D}_{ij} = [D_{ij}[0], D_{ij}[1], \cdots, D_{ij}[F-1]]^\top, j \in \mathcal{N}_i$ with $F \leq T-1$. Note that the input dimension is now MF as depicted in Fig. 3 (left). It is worth noting that the input feature used by the SDNN method is a special case of FDNN with $F = 1$. Compared to the SDNN method, the FDNN method employs a richer input feature for the NN model.

Similarly for the localization task, we employ the DFT to transform the time series to generate the input feature. We let

$$L_{ij}[f] := \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \left| \sum_{t=0}^{T} x_j^k(t) e^{-i2\pi tf/T} \right|, \quad j \in \mathcal{N}_i. \tag{35}$$

and define $\boldsymbol{L}_{ij} = [L_{ij}[0], L_{ij}[1], \cdots, L_{ij}[F-1]]^\top$ such that each node has F frequency parameters. The input feature is defined as

$$\boldsymbol{o}^0 = \boldsymbol{L}_i = [\boldsymbol{L}_{i1}^\top \boldsymbol{L}_{i2}^\top \cdots \boldsymbol{L}_{iM}^\top]^\top \tag{36}$$

Similar to the detection task, we note that the localization model has an input dimension equal to MF. To summarize the above three NN methods, we remark here that the main differences among these three NN methods are 1) the features of neighboring data they use are different, i.e., TDNN only uses the data from initial states and steady states while SDNN and FDNN both use the data from initial states, steady states and transient states; 2) the pre-processing of the data is different, i.e., FDNN explores the frequency domain of the data while TDNN and SDNN consider the time domain of the data sequence; 3) the number of NN inputs is different, which has been specified in equations (26), (29), (32), (34) and (36). Besdies, it is worth noting that the advantage of NN is to approximate a nonlinear function that yields better statistics for the classification problem at hand. However, it becomes difficult for us to directly analyze the probability of detection and false alarm since NNs are dealing with a nonlinear function without a closed form.

E. THE NN MODEL FOR DIFFERENT DEGREE-$|\mathcal{N}_i|$ NODES

We conclude this section by tailoring our M-input NNs to fit into the scenario when the normal nodes have an heterogeneous number of neighbors. Two scenarios are considered:

1) SCENARIO 1

When $|\mathcal{N}_i| > M$, the $|\mathcal{N}_i|$ neighbors will be divided into $\lceil |\mathcal{N}_i|/M \rceil$ potentially overlapping groups. In the detection and localization tasks, each group contains exactly M nodes when using the TDNN, SDNN and FDNN methods. Each group can be treated as a standard neighbor set of the observe node and the two tasks can be implemented with the unified NN model.

2) SCENARIO 2

When $|\mathcal{N}_i| < M$, we fill the deficient value in the input vector with a reference value to fit a Degree-$|\mathcal{N}_i|$ node. For instance, with the TDNN method, the input feature of the NN can be expressed by $\boldsymbol{\xi}_i = [\xi_{i1}, \cdots, \xi_{i|\mathcal{N}_i|}, \underbrace{\xi_{ii}, \ldots, \xi_{ii}}_{M-|\mathcal{N}_i|}]^\top$, where ξ_{ii} is the temporal difference value of node i. Moreover, we can fill ξ_{ii} in any position of the inputs in both the two tasks. For SDNN method, X_{ii} is used to fill the deficient value to adapt different degree-$|\mathcal{N}_i|$ in the detection task. Similarly, we utilize a special value to fill the inputs vector of localization task. $\boldsymbol{Y}_i = [Y_{i1}; \cdots; Y_{i\mathcal{N}_i}; \underbrace{Y_i^*; \cdots; Y_i^*}_{M-|\mathcal{N}_i|}]^\top$ when $|\mathcal{N}_i| < M$. Where Y_i^* is the reference point, *i.e.*, $Y_i^* = \min\{Y_{ij} | j \in \mathcal{N}_i\}$ for all samples with the same K in the simulation.

Similar to the TDNN and SDNN methods, we consider two unified models of the FDNN method for detection and localization tasks, respectively. We utilize two special vectors to fill the input vectors for two tasks, $\boldsymbol{D}_i = [\boldsymbol{D}_{i1}; \cdots; \boldsymbol{D}_{i\mathcal{N}_i}; \underbrace{\boldsymbol{D}_{ii}; \cdots; \boldsymbol{D}_{ii}}_{M-|\mathcal{N}_i|}]$, $\boldsymbol{L}_i = [\boldsymbol{L}_{i1}; \cdots; \boldsymbol{L}_{i\mathcal{N}_i}; \underbrace{\boldsymbol{L}_{ii}; \cdots; \boldsymbol{L}_{ii}}_{M-|\mathcal{N}_i|}]$ when $|\mathcal{N}_i| < M$. With a slight abuse of notation, the $\boldsymbol{D}_{ii}$ and $\boldsymbol{L}_{ii}$ are calculated independently by node i. The simulation results fit different degree-$|\mathcal{N}_i|$ node are presented in Section IV-C.

IV. NUMERICAL RESULTS

In this section, we compare the TDNN, SDNN and FDNN algorithms detection and localization performance with those of the TDM and SDM from [22], [27]. We simulate the case of a simple Manhattan network also considered in [22], [27] (c.f. Fig. 4) with $N = 9$ and one attacker. The randomized average consensus protocol (cf. Algorithm 1) is run with $P_{ij} = 1/|\mathcal{N}_i|$, and is terminated with $T = 500$. In the simulations, we set $\alpha^k \sim \mathcal{N}(0, 1)$, $\gamma_i^k \sim \mathcal{U}[-0.5, 1.5]$ and $m_j^k(t) \sim \mathcal{U}[-\left(\lambda_2(\overline{\boldsymbol{A}})\right)^t \left(\lambda_2(\overline{\boldsymbol{A}})\right)^t]$. We let the network run decentralized consensus algorithm for K instances, each time starting from a new initial state. Generally speaking, the detection and localization performance will improve as K increases as we have revealed in our previous works [22], [27], [39]. Since different methods exhibit different detection and localization performance, we have to choose different K so that herein different methods can keep the same level of probability of detection and probability of false alarm to produce comparable ROC curves. For the detection, each normal node collects data from those K sets of iterations, and fuses them to compute the features/scores used for the detection and localization, as explained in Sections II and III.

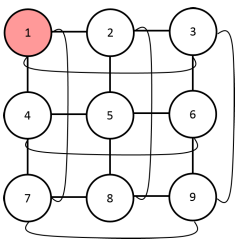


FIGURE 4. The Manhattan network topology.

We consider each of these NNs with three hidden layers, and the neurons in each hidden layer are 60, 40 and 20 respectively. To provide the training data for the NNs, we run 4000 samples of the scenario with one attacker in the network and 4000 samples of the scenario without attacker in the network. Notice that for each sample, data for K trails are collected to fuse at the detect node. We label the sample with '0' if the network has no attackers and '1' if any attacker is present in the network. Similarly, the testing data is formed by 2000 samples of '1' and 2000 samples of '0'. Thus, the size of training and testing data in our simulations are set to be 8000 and 4000, respectively. As for the localization task, we encode the indexes of neighbors by one-hot coding where the neighboring attacker is labeled by '1' and normal neighbors are labeled by '0'. The training samples for the localization are collected under the condition that we have confirmed that one attacker is present in the network and thus our purpose is to identify which one within the neighborhood is the attacker. In order to ensure the fairness of the test results, we randomly select the position of the attacker in the input vector of NNs. The size for the training and testing data for localization are set to be 4000 and 2000, respectively.

For all the FDNN methods, we have set $\mathcal{F} = \{0, 1, 2\}$ for the detection task and $\mathcal{F} = \{1, 2, 3\}$ for the localization task. In this first set of simulations, we have the training data and test the data coming from the same attackers. The simulations in Section IV-D, are done to explore the efficacy of NNs in classifying attacks not seen in the training data. This is important because, in practice, there will be little prior information about the mean of the initial states of the attackers (i.e. the value of α^k). Hence, it is important for us to test the NNs in scenarios that are statistically mismatched with the training data.

In this work, the detection and localization models are actually classifiers for which neural networks produce continuous quantities to predict class membership through different thresholds; see Fig. 2 for an illustration. To evaluate the performance for these classifiers, we follow [22], [27] to define the probabilities:

$$P_{nd}^i := P(\hat{\mathcal{H}}^i = \mathcal{H}_1^i | \mathcal{H}_1^i), P_{nf}^i := P(\hat{\mathcal{H}}^i = \mathcal{H}_1^i | \mathcal{H}_0^i),$$
$$P_{ld}^i := P(\hat{\mathcal{H}}^{ij} = \mathcal{H}_1^{ij} | \mathcal{H}_1^{ij}), P_{lf}^i := P(\hat{\mathcal{H}}^{ij} = \mathcal{H}_1^{ij} | \mathcal{H}_0^{ij}),$$

where P_{nd}^i (P_{nf}^i) and P_{ld}^i (P_{lf}^i) are the probability of detection (false alarm) for the neighborhood detection and localization tasks, respectively. More specifically, we calculate those probabilities by

$$P_{nd}^i = \frac{1}{\mathcal{D}_p} \sum\nolimits_{n=1}^{\mathcal{D}_p} I(d^{i,n} = \hat{d}^{i,n} = 1), \tag{37}$$

$$P_{nf}^i = \frac{1}{\mathcal{D}_n} \sum\nolimits_{n=1}^{\mathcal{D}_n} I(d^{i,n} = 0 \wedge \hat{d}^{i,n} = 1), \tag{38}$$

$$P^i_{ld} = \frac{1}{\mathcal{L}_p}\sum\nolimits_{n=1}^{\mathcal{L}_p} I(\ell^{i,n}_j = \hat{\ell}^{i,n}_j = 1), \tag{39}$$

$$P^i_{lf} = \frac{1}{\mathcal{L}_n}\sum\nolimits_{n=1}^{\mathcal{L}_n} I(\ell^{i,n}_j = 0 \wedge \hat{\ell}^{i,n}_j = 1). \tag{40}$$

Herein, $\mathcal{D}_p$ ($\mathcal{D}_n$) is the positive (negative) samples in the ground-truth of the detection task, $I(\cdot)$ is an indicator function. $d^{i,n}$ and $\hat{d}^{i,n}$ are the ground-truth class label and the predicted class label, respectively. A similar fashion is applied to the localization task.

Based on these probabilities, the receiver operating characteristic (ROC) is adopted as a performance measurement for evaluating the NN detection and localization ability of different methods. ROC is an important evaluation metric, in which the probability of successful detection or localization is plotted on the Y-axis and the probability of false alarm for detection or localization is plotted on the X-axis. This measurement is widely used in statistical hypothesis testing, and also for neural network when it is used as a classifier. It is obvious that the best possible prediction method would yield a point in the upper left corner at the coordinate (0, 1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). Hence, curves which approach upper left corner outperforms those far from it.

A. DETECTION AND LOCALIZATION FOR ONE ATTACKER

In this subsection, we consider the neighborhood detection and localization process. We consider only one attacker present in the Manhattan network in Figure 4. Without loss of generality, we assume that node 1 is the attacker given that the network topology is symmetrical. Our training data comes from the node which is next to the attacker.

The neighborhood detection performance and localization performance of TDNN are shown in Fig. 5, where we set the score-based TDM method in [22], [27] as a benchmark. The detection performance is depicted in Fig. 5 (left), while the localization performance is depicted in Fig. 5 (right), where we assume that the neighborhood detection test was completed without errors (by an 'Oracle') . From Fig. 5, it is obvious that both the detection and localization performances improve as K increases. The TDNN method improves significantly over the TDM method, having good performance when $K \geq 25$.

In Fig. 6, we plot the ROC curves of the SDNN method with $K = 5$, where the SDM method in [27] is chosen as a benchmark. From the plots we see that even when $K = 5$, the spatial methods (SDM and SDNN) already provide very good ROCs in detection and localization, and thus outperform the temporal method (e.g., TDM and TDNN). This implies that considering the transient states of the dynamical process do provide us more information to identify the attacker. Moreover, we found that both the detection and localization performances for SDNN outperform those for SDM. Hence, the NNs improve the detection and localization performance.

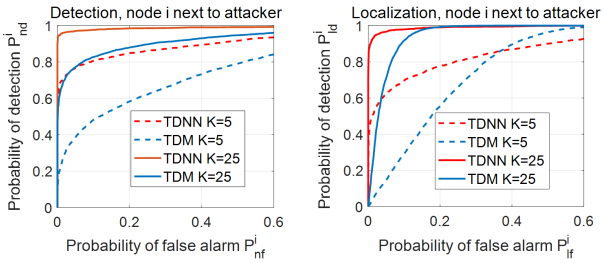


FIGURE 5. ROCs for comparing TDNN with TDM: (Left) ROCs for neighborhood detection, (Right) ROCs for neighborhood localization of attacker. Localization of the attacker with oracle for neighborhood attack detection.

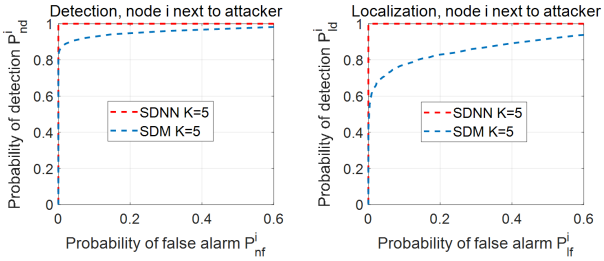


FIGURE 6. ROCs comparing SDNN with SDM: (Left) ROCs for neighborhood detection, (Right) ROCs for neighborhood localization of attacker.

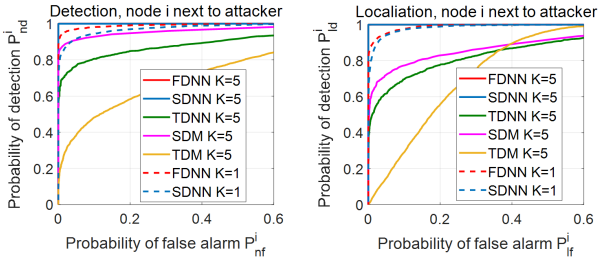


FIGURE 7. The performance for FDNN, SDNN, TDNN, SDM and TDM: (Left) ROCs for neighborhood detection, (Right) ROCs for neighborhood localization of attacker. Dotted lines show the performance for SDNN and FDNN in $K = 1$.

In Fig. 7, we plot the ROC curves of FDNN method with $K = 5$, where we set the SDNN, TDNN, SDM and TDM methods as benchmarks. From the plots we see that when $K = 5$, it is obvious that the spatial methods based on NNs (FDNN and SDNN) provide very good performance in detection and localization. The SDM method performs better than TDM and TDNN, since TDM and TDNN only involve initial and steady states. Moreover, we also plot the ROCs of FDNN and SDNN methods with $K = 1$ (the dotted lines) for a further comparison since they both involve transient states and NNs. It shows that both the FDNN and SDNN methods in $K = 1$ perform well in the two tasks. Actually, they even outperform the SDM, TDNN and TDM methods for $K = 5$. It is interesting to notice that FDNN exhibits better performance than SDNN confirming the intuition that the DFT is more informative than the average.

In Fig. 8, we try to investigate the optimal thresholds δ and ϵ defined in (22) and (24) respectively, for the SDNN and FDNN models when $K = 1$. Our purpose is to let

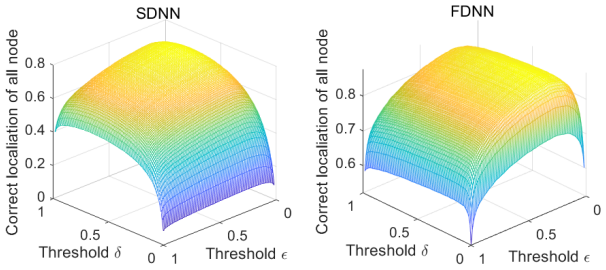


FIGURE 8. Probability of correct localization of all nodes for $K = 1$: (Left) SDNN, (Right) FDNN.

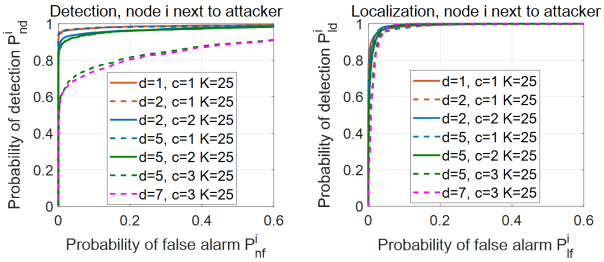


FIGURE 9. ROCs for multiple attackers of TDNN with $K = 25$, where d is the number of attackers in the network and c is the number of attackers in the detecting node's neighborhood.

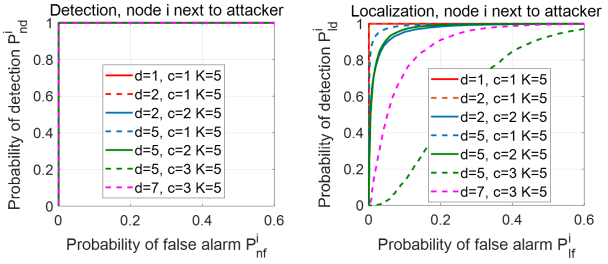


FIGURE 10. ROCs for multiple attackers of SDNN with $K = 5$, where d is the number of attackers in the network and c is the number of attackers in the detecting node's neighborhood.

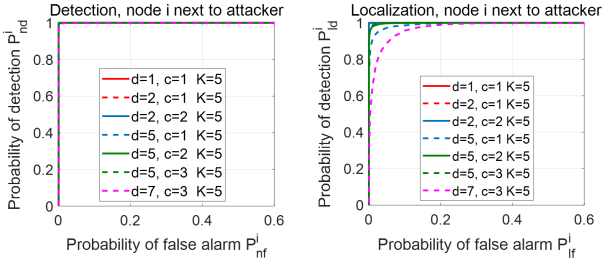


FIGURE 11. ROCs for multiple attackers of FDNN with $K = 5$, where d is the number of attackers in the network and c is the number of attackers in the detecting node's neighborhood.

nodes which are next to the attacker not only to detect and localize the attackers, but also classify correctly nodes that are normal ensuring that nodes that are not next to an attacker, i.e. those with only normal neighbors, will not falsely detect a neighborhood attacker. We then plot the "all-correct" probability that all neighbors are classified correctly. The plots show that for SDNN the thresholds $(\delta, \epsilon) = (0.68, 0.22)$ provides an "all-correct" probability of 78%, and for FDNN the thresholds $(\delta, \epsilon) = (0.23, 0.27)$ yields an "all-correct" rate of 88%.

B. DETECTION AND LOCALIZATION FOR MULTIPLE ATTACKERS

In this subsection, we consider the case of multiple attackers. The simulation parameters are the same as those in subsection IV-A and TDNN, SDNN and FDNN methods are adopted to detect and localize attackers with different scales. Specially, α^k is the initial state shared by all cooperative attackers, and each attacker has a random and independent noise m_j^k. Suppose that we have in total d attackers in the network and the detecting node has c attackers in its neighborhood. Thus, the legend with *'d attackers and c neighbors'* in the figures means that the network contains d attackers and the detecting node has c attacker as its neighbors.

In Fig. 9, we show the detection and localization performance of TDNN with $K = 25$ in the case of multiple collusive attackers. It shows that the number of the neighboring attacker (c) has a direct impact on the TDNN detection performance. However, when we fix c, the total number of attackers (d) has a slight impact on the TDNN detection performance. This implies that the neighboring attacker dominates the detection process. For the localization task, as shown in Fig. 9 (right), normal nodes exhibit similar performance in different attack scenarios. One reason is that TDNN uses only the initial and terminal states, the localization performance is independent of the number of attackers and the neighboring attackers.

Fig. 10 illustrates the ROC curves of SDNN for detection and localization of multiple attackers. As seen in Fig. 10 (left), SDNN has excellent detection performance in the case of multiple attackers for $K = 5$. In Fig. 10 (right), the localization performance of SDNN degrades with both the number of attackers and the number of neighboring attackers. It cannot work well when $d = 5, c = 3$ and $d = 7, c = 3$. One possible reason is that when there are more neighboring attackers, the influence of an individual attacker becomes less pronounced.

The ROC curves of FDNN for detection and localization of multiple attackers are shown in Fig. 11. One can observer in Fig. 11 (left) that the detection performance of FDNN is similar to that of SDNN shown in Fig. 10 (left). On the other hand, Fig. 11 (right) shows that, unlike SDNN, FDNN performs very well in the attacker localization task, especially for the scenario $d = 5, c = 3$ and $d = 7, c = 3$.

Furthermore, looking comparatively at the performance of FDNN, SDNN and TDNN plotted in Fig. 12 we notice that FDNN works relatively well in both the attacker detection and localization tasks with $K = 1$ and does not experience the same degradation of localization performance of SDNN as the number of neighboring attackers increases. We observe

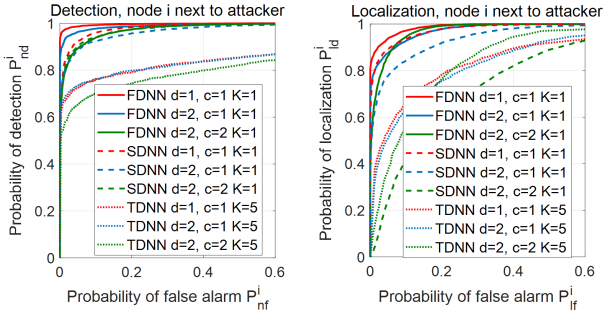


FIGURE 12. The performance for FDNN, SDNN and TDNN in the presence of multiple attackers, where d is the number of attackers in the network and c is the number of attackers in the detecting node's neighborhood.

that the performance of TDNN is limited due to the ignorance of transient state information. Generally speaking, FDNN performs the best, followed by SDNN and TDNN, which is consistent with previous results in one attacker network.

C. TAILORING THE NEURAL NETWORK TO FIT A DEGREE-$|\mathcal{N}_i|$ NODE

In this subsection, we will describe how to tailor an M-inputs NN to fit a degree-d node. We denote the number of mismatched inputs that do not matching the unified model is $p = M - d$. In this setting, we have set $M = 4$ and $d = 2, 3$ or 4. We take the Manhattan network topology as the target example, and the scenario of $d = 1, c = 1$ is applied to verify our proposed method. To set up the simulation, we choose the normal node 2 in the Manhattan network as the testing node, and thus the neighbors are nodes 1, 3, 5 and 8. When $p = 1$, we cut off the connection between node 2 and node 3, and thus neighbors of node 2 remain node 1, 5 and 8. In the case of $p = 2$, the testing node has two neighbors which are node 1 and 8, since we cut off the connection between node 2 and node 3, as well as the connection between node 2 and node 5. Note that we would set node 1 as the attacker when the attacker is present in the network. The parameters set in TDNN, SDNN and FDNN are the same as those in previous subsections. The testing data of different models are generated from a modified Manhattan network with $p = 1$ or $p = 2$. It is worth noting that, more judicious operation are needed to deal with the problem when the locations of missing inputs are not the same. In general, we can randomly select the locations of missing inputs and replace the values of the missing locations with some specific values, such as ξ_{ii} for TDNN (ref. Section III-E), X_{ii} and Y_i^* for SDNN (ref. Section III-E), $\boldsymbol{D}_{ii}$ and $\boldsymbol{L}_{ii}$ for FDNN (ref. Section III-D).

In the left and right parts of Fig. 13, 14 and 15, we show the detection and localization mismatched ROCs curves for TDNN ($K = 5$ and $K = 25$), SDNN ($K = 1$) and FDNN ($K = 1$) respectively. The results show a slight degradation in detection and localization performance as p increases. Our model fits well with irregular degree networks.

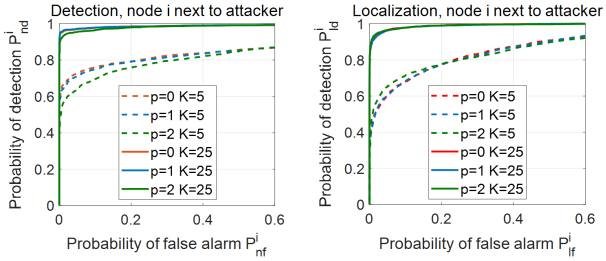


FIGURE 13. TDNN with different deficient size: $p = M - d$ means that the number of deficient inputs is p.

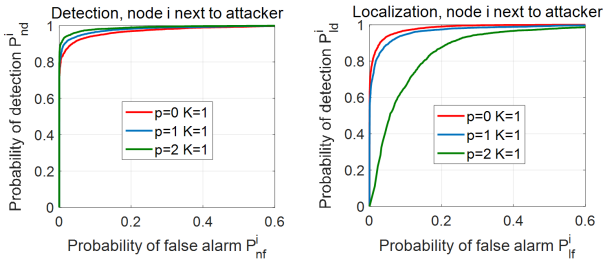


FIGURE 14. SDNN with different deficient size: $p = M - d$ means that the number of deficient inputs is $M - d$.

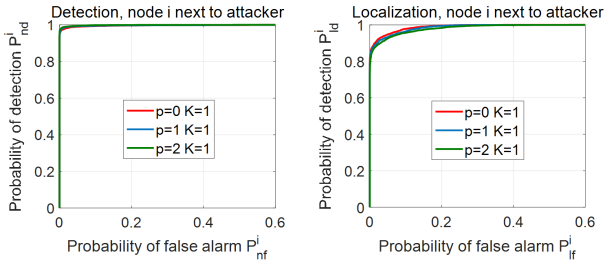


FIGURE 15. FDNN with different deficient size: $p = M - d$ means that the number of deficient inputs is $M - d$.

D. ROBUSTNESS OF INSIDER ATTACK DETECTION AND LOCALIZATION

In this part, we test the robustness of the NN methods under the situation that when the prior information we have is mismatched with the actual environment. We simulate scenarios with mismatched training and testing data. Assume that there is only one attacker in the network. We train the NNs with specific α^k, γ_i^k and then test it by varying statistic of α^k and γ_i^k, respectively.

In particular, the training data is collected from a normal node with only one next-to attacker under the scenario with $\alpha^k \sim \mathcal{N}(0, 1)$, $\gamma_i^k \sim \mathcal{U}[-0.5, 1.5]$. Following the order in the legend in Fig. 16, the first curve is the ROC of the TDNN matched model. For the third and the fourth curves, under the trained NN model, we generate the testing data by keeping $\alpha^k \sim \mathcal{N}(0, 1)$ and changing γ_i^k to $\gamma_i^k \sim \mathcal{U}[-0.2, 1.2]$ and $\gamma_i^k \sim \mathcal{U}[-0.8, 1.8]$, respectively. The mean of γ_i^k is unchanged while the deviation decreases for the third curve and increases for the fourth curve. The results show that the performance of the detection and localization both deteriorate when the deviation of γ_i^k increases and improve when the

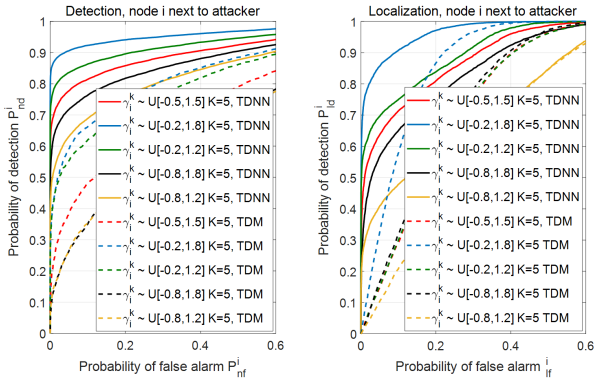


FIGURE 16. ROCs of TDNN and TDM for the mismatch model: $\alpha^k \sim \mathcal{N}(0, 1)$ and each entry of γ_i^k distributed as legended in the Fig. s for testing data.

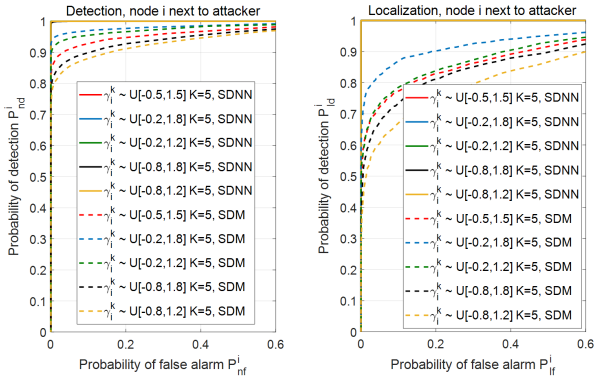


FIGURE 17. ROCs of SDNN and SDM for the mismatch model: $\alpha^k \sim \mathcal{N}(0, 1)$ and each entry of γ_i^k is distributed as legended for testing data.

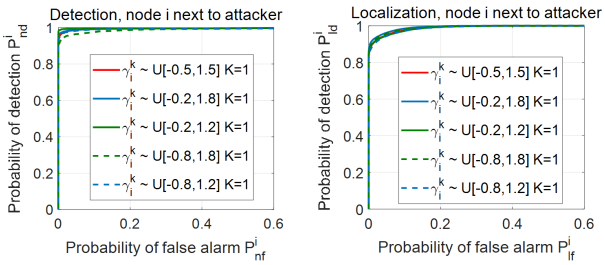


FIGURE 18. ROCs of FDNN for the mismatch model: $\alpha^k \sim \mathcal{N}(0, 1)$ and each entry of γ_i^k is distributed as legended for testing data.

deviation of γ_i^k decreases. To generate the second and the fifth curves, we changed the mean of γ_i^k. It shows that the performance of the detection and localization both improve when $\left|\mathbb{E}[\gamma_i^k] - \mathbb{E}[\alpha_i^k]\right|$ increases and deteriorate when the gap decreases. Meanwhile, the robustness performance of TDM is also given in Fig. 16. Obviously, the TDNN method has better robustness performance in both detection and localization tasks when dealing with a test set that is not seen in training.

In Fig. 17, we investigated the robustness of SDNN and SDM methods with $K = 5$ under an attack not seen in the training data. The curves of SDM follow the same trends as that in TDNN. Fig. 18 plots the ROC curves of FDNN with $K = 1$. It is worth mentioning that, even if the mismatch between training data and attack scenario degrades the performance, SDNN and FDNN methods inherently exhibit very good detection and localization performance.

V. CONCLUSION AND FURTHER DISCUSSION

In this paper, we have proposed three new defense strategies for randomized gossip algorithm for average consensus based on NNs to detect and localize insider adversarial nodes. The NNs are trained centrally, and the models are deployed at the different network nodes for online detection. The experiments clearly show that the NN methods provide better performance compared to statistical detectors that rely on simplified expert models; also, they exhibit good robustness against model mismatch and remain effective in the presence of coordinated attacks. At the end of this paper, we want to point it out that while the focus of this work is on the average consensus algorithm, the NN-based approach is also adapted to other decentralized algorithms; some preliminary work has been done in [42]. As a future work, it would be interesting to try the NN-based approach on more complicated attack models and other decentralized algorithms.

REFERENCES

[1] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018.

[2] T. T. Doan and C. L. Beck, "Distributed resource allocation over dynamic networks with uncertainty," 2017, *arXiv:1708.03543*. [Online]. Available: http://arxiv.org/abs/1708.03543

[3] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proc. 7th Int. Workshop Qual. Service. (IWQoS)*, Jun. 1999, pp. 27–36.

[4] B. Orostica and F. Nunez, "Robust gossiping for distributed average consensus in IoT environments," *IEEE Access*, vol. 7, pp. 994–1005, 2019.

[5] A. Singh and Y. Viniotis, "Resource allocation for IoT applications in cloud environments," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 719–723.

[6] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 7–12.

[7] N. Wang, E. Hossain, and V. K. Bhargava, "Backhauling 5G small cells: A radio resource management perspective," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 41–49, Oct. 2015.

[8] D. Amendola, N. Cordeschi, and E. Baccarelli, "Bandwidth management VMs live migration in wireless fog computing for 5G networks," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Oct. 2016, pp. 21–26.

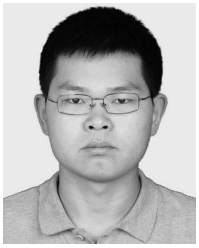
[9] D. Gesbert, S. G. Kiani, A. Gjendemsjo, and G. E. Oien, "Adaptation, coordination, and distributed resource allocation in interference-limited wireless networks," *Proc. IEEE*, vol. 95, no. 12, pp. 2393–2409, Dec. 2007.

[10] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[11] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported Internet of Things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.

[12] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*. [Online]. Available: http://arxiv.org/abs/1602.05629

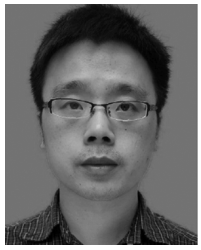
[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.
[14] Q. Wang and J. Wang, "Fully distributed fault-tolerant consensus protocols for Lipschitz nonlinear multi-agent systems," *IEEE Access*, vol. 6, pp. 17313–17325, 2018.
[15] G. Colistra, V. Pilloni, and L. Atzori, "Task allocation in group of nodes in the IoT: A consensus approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 3848–3853.
[16] V. Pilloni and L. Atzori, "Consensus-based resource allocation among objects in the Internet of Things," *Ann. Telecommun.*, vol. 72, nos. 7–8, pp. 415–429, May 2017.
[17] N. N. Srinidhi, S. M. D. Kumar, and K. R. Venugopal, "Network optimizations in the Internet of Things: A review," *Eng. Sci. Technol., Int. J.*, vol. 22, no. 1, pp. 1–21, 2018.
[18] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw. (IPSN)*. New York, NY, USA: ACM, 2004, pp. 20–27.
[19] T. Wang, H. Zhang, and Y. Zhao, "Average consensus of multi-agent systems under directed topologies and binary-valued communications," *IEEE Access*, vol. 6, pp. 55995–56006, 2018.
[20] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[21] S. Sundaram and B. Gharesifard, "Consensus-based distributed optimization with malicious nodes," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 244–249.
[22] R. Gentz, H.-T. Wai, A. Scaglione, and A. Leshem, "Detection of data injection attacks in decentralized learning," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 350–354.
[23] Q. Yan, M. Li, T. Jiang, W. Lou, and Y. T. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," in *Proc. Proc. IEEE INFOCOM*, Mar. 2012, pp. 900–908.
[24] D. Moser, P. Leu, V. Lenders, A. Ranganathan, F. Ricciato, and S. Capkun, "Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*. New York, NY, USA: ACM, 2016, pp. 375–386.
[25] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in two-tiered sensor networks," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 46–50.
[26] C. Zhao, J. He, and J. Chen, "Resilient consensus with mobile detectors against malicious attacks," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 4, no. 1, pp. 60–69, Mar. 2018.
[27] R. Gentz, S. Wu, H.-T. Wai, A. Scaglione, and A. Leshem, "Data injection attacks in randomized gossiping," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 2, no. 4, pp. 523–538, 2016.
[28] M. Mobilia, "Does a single zealot affect an infinite group of voters?" *Phys. Rev. Lett.*, vol. 91, no. 2, Jul. 2003.
[29] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," 2016, *arXiv:1606.08939*. [Online]. Available: http://arxiv.org/abs/1606.08939
[30] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 3, no. 1, pp. 145–158, Mar. 2017.
[31] M. E. Yildiz and A. Scaglione, "Computing along routes via gossiping," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3313–3327, Jun. 2010.
[32] B. Kailkhura, S. Brahma, and P. K. Varshney, "Consensus based detection in the presence of data falsification attacks," *IEEE Trans. Signal Process.*, to be published.
[33] S. Gil, S. Kumar, D. Katabi, and D. Rus, "Adaptive communication in multi-robot systems using directionality of signal strength," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 946–968, 2015.
[34] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.
[35] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: A hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proc. IEEE Workshop Inf. Assurance Secur.*, Jun. 2001, pp. 85–90.
[36] A. Doboli, "Discovery of malicious nodes in wireless sensor networks using neural predictors," *Wseas Trans. Comput. Res.*, vol. 2, pp. 38–43, Jan. 2007.
[37] D. Stevanovic, N. Vlajic, and A. An, "Detection of malicious and non-malicious Website visitors using unsupervised neural network learning," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 698–708, Jan. 2013.
[38] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018.
[39] G. Li, S. Xiaoxiao Wu, S. Zhang, H.-T. Wai, and A. Scaglione, "Detecting and localizing adversarial nodes usig neural networks," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.
[40] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.
[41] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometric Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, Nov. 1997.
[42] S. X. Wu, H.-T. Wai, A. Scaglione, A. Nedic, and A. Leshem, "Data injection attack on decentralized optimization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 3644–3648.



GANGQIANG LI received the B.Eng. degree in electronic engineering from the Henan University of Urban Construction, Pingdingshan, China, in 2014, and the M.Eng. degree in control science and engineering from Shenzhen University, Shenzhen, China, in 2017, where he is currently pursuing the Ph.D. degree in communication and information engineering. His research interests include machine learning, data mining, and distributed protocols.

SISSI XIAOXIAO WU (Member, IEEE) received the B.Eng. degree in electronic information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005, the M.Phil. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2009, and the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2013. From December 2013 to November 2015, she was a Postdoctoral Fellow of the Department of Systems Engineering and Engineering Management with CUHK. From December 2015 to March 2017, she was a Postdoctoral Fellow of the Signal, Information, Networks, and Energy Laboratory with Arizona State University, Tempe, AZ, USA, supervised by Prof. A. Scaglione. Since March 2017, she has been an Assistant Professor with the Department of Communication and Information Engineering, Shenzhen University, Shenzhen, China. Her research interests include wireless communication theory, optimization theory, stochastic process, channel coding theory, and with a recent emphasis on the modeling and data mining of opinion diffusion in social networks.

SHENGLI ZHANG (Senior Member, IEEE) received the B.Eng. degree in electronic engineering and the M.Eng. degree in communication and information engineering from the University of Science and Technology of China, Hefei, China, in 2002 and 2005, respectively, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2008. After that, he joined the Department of Communication Engineering, Shenzhen University. He is currently a full professor. From March 2014 to March 2015, he was a Visiting Associate Professor with Stanford University. He is the pioneer of physical-layer network coding. He has authored or coauthored more than 20 IEEE top journal articles and ACM top conference papers, including the IEEE JSAC, the IEEE TWC, the IEEE TMC, the IEEE TCom, and ACM MobiCom. His research interests include physical layer network coding, interference cancellation, and cooperative wireless networks. He served as an Editor for the IEEE TVT, the IEEE WCL, and *IET Communications*. He has also served as a TPC Member of several IEEE conferences, including WCNC 2012, WCNC 2014, ICC 2014, GLOBECOM 2014, ICC 2015, and the IEEE GLOBECOM 2016.



QIANG LI (Member, IEEE) received the B.Eng. and M.Phil. degrees in communication and information engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2005 and 2008, respectively, and the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2012. From August 2011 to January 2012, he was a Visiting Scholar with the University of Minnesota, Minneapolis, MN, USA. From February 2012 to October 2013, he was a Research Associate with the Department of Electronic Engineering and the Department of Systems Engineering and Engineering Management, CUHK. Since November 2013, he has been with the School of Communication and Information Engineering, UESTC, where he is currently an Associate Professor. His research interests include convex optimization and its applications in signal processing with an emphasis on physical-layer security and full-duplex communications. He received the First Prize Paper Award from the IEEE Signal Processing Society Postgraduate Forum, Hong Kong Chapter, in 2010, the Best Paper Award from the IEEE PIMRC 2016, and the Best Paper Award for the IEEE SIGNAL PROCESSING LETTERS, in 2016.