IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 11, 2021, accepted April 24, 2021, date of publication April 29, 2021, date of current version May 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076538

Reinforcement Learning-Based Detection for State Estimation Under False Data Injection

WEILIANG JIANG, WEN YANG, JIAYU ZHOU, WENJIE DING, YUE LUO, AND YUN LIU
Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China

Corresponding author: Wen Yang (weny@ecust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61973123, in part by the projects sponsored by the Development Fund for Shanghai Talents, in part by the Shanghai Natural Science Foundation under Grant 18ZR1409700, in part by the Programme of Introducing Talents of Discipline to Universities (the 111 Project) under Grant B17017, in part by the Shuguang Program supported by the Shanghai Education Development Foundation and Shanghai Municipal Education Commission, and in part by the Fundamental Research Funds for the Central Universities.

ABSTRACT We consider the problem of network security under false data injection attacks over wireless sensor networks.To resist the attacks which can inject false data into communication channels according to a certain probability, we formulate the online attack detection problem as a partially observable Markov decision process problem and design a detector for each sensor based on the framework of model-free reinforcement learning. By numerical simulations, we illustrate the effectiveness of the proposed reinforcement learning algorithm and show the performance of the proposed detector compared with the typical detector in the existing works.

INDEX TERMS Wireless sensor network, false data injection attack, reinforcement learning, partially observable Markov decision process.

I. INTRODUCTION

Wireless sensor network (WSN) is a distributed sensor network whose terminals are sensors that can sense and inspect the outside world. It is a network formed by the free organization and combination of tens of thousands of sensor nodes through wireless communication technology. Nowadays, WSN has practical applications in many areas such as explosion protection, medical treatment, health care, home furnishing, industry [1], and so on.

One of the promising applications is power grid. The next generation power grid, i.e., the smart grid, can effectively overcome various problems existing in the traditional power system. However, this critical cyber infrastructure makes the smart grid vulnerable to different security attacks [2]. Main purpose of attackers is to disrupt the state estimation mechanism in the smart grid to cause wide-area power blackouts or to manipulate electricity market prices [3].

There are many types of cyber-attacks, one of such attacks is the false data injection (FDI) attack that changes the data integrity of packages by modifying their payloads [4]–[6]. To mitigate the physical overloads due to the judiciously designed false data injection attack, Che *et al.* [7] proposed a cyber-secured corrective dispatch scheme which can secure the flow levels. On the other hand, in the power system, a set of transmission lines can be more vulnerable to the attack. Therefore, Che *et al.* [8] proposed a high-risk line fast screening (HLFS) approach to identify these lines. Li *et al.* [9] proposed a sequential detector based on the generalized likelihood ratio to address the challenge, which aims to manipulate the state estimation procedure by injecting malicious data to the monitoring meters. Distributed attack detectin has long been pursued in a wireless sensor network setting. Yang *et al.* [10] proposed a detector with a stochastic protection rule based on the innovation from its neighboring sensors to reduce the impact of FDI attack. Liu *et al.* [11] utilized the relative entropy to detect whether the data are attacked and explored the tradeoff between the performance degradation and attack stealthiness level. Owing to the fact that Kullback-Leibler (KL) divergence is effective to detect FDI attack, Hua *et al.* [12] proposed a distributed adaptive algorithm over KL to weaken the impact of FDI attack. Although the χ^2 detector is often used in the area of fault diagnosis, it may be bypassed by some carefully crafted FDI attacks. Therefore, Ye *et al.* [13] proposed the summation detector to detect these types of attacks. Moreover, compared with the

The associate editor coordinating the review of this manuscript and approving it for publication was Usama Mir.

χ^2 detector, the effectiveness of the summation detector is further verified for existing attack and the improved attack. However, most of the detection methods are based on the gaussianity of the innovations, which leads to the limitation of the detectors. When the FDI attacks make innovations no longer maintain gaussianity, the detectors mentioned above, like the χ^2 detector, will lose efficacy. Furthermore, if the attacker keeps changing the attack strategy, the KL divergence detector will also fail to detect the attacks. Then methods that can cope with these challenges become crucial. One of the strategies is reinforcement learning.

Reinforcement learning is used to describe and solve problems in which agents use learning strategies to maximize returns or achieve specific goals in their interactions with the environment. According to the given conditions, reinforcement learning can be divided into model-based and model-free reinforcement learning. Consider that the attacker strategy and the transition probability between the hidden states are unknown in general, the exact model is unknown. Hence, we will apply the model-free reinforcement learning to design detector. Typical model-free methods are Monte-Carlo method and Temporal-Difference method. The Monte-Carlo method obtains the sequence of state, action and benefit from real or simulated environment interaction, and it carries out value estimation and strategy improvement at the end of the whole episode, and obtains the estimation of value function through continuous iteration. The Temporal-Difference method can also learn strategies directly from the experience of interacting with the environment. However, the Temporal-Difference method is quite different from the Monte-Carlo method. Unlike the Monte-Carlo method, the Temporal-Difference method does not need to wait for the final result of the interaction, but can update the value function of the current state based on the value estimation of other states [14]–[17]. Kurt *et al.* [18] formulated the attack detection problem in the smart grid as a partially observable Markov decision process (POMDP) problem and proposed a detector using the framework of model-free reinforcement learning (RL) for POMDPs. Reinforcement learning method is based on the amplitude of the past innovations to train the detector (the larger norm of innovation, the higher probability of data being attacked), which has nothing to do with the statistical characteristics of innovation, that is, it does not require the gaussianity of innovation. Moreover, the detector trained by reinforcement learning can cope with the FDI attacks whose statistical characteristics keep changing. In control area, RL has been applied for solving optimal control problems and has obtained well performance [19]–[23]. Extension of single-agent RL to multiple agents is the multi-agent RL. Moreover, stochastic games extend the Markov decision processes to multi-agent case. Several RL-based solution approaches have been proposed for stochastic games [24]–[28].

In this paper, we propose an online cyber-attack detection algorithm using the framework of model-free RL for POMDPs based on WSN. The proposed detection algorithm does not require attack models. Therefore, the proposed scheme is widely applicable. In the training phase, the defender learns a mapping from observations to actions by trial-and-error. To make the detector adapt to any attack, we follow a detection approach by training the agent (defender) with low-magnitude attacks. This is the worst-case scenario from the defender's perspective since such attack is difficult to detect. Then, in the detection phase, the trained defender becomes sensitive to detect slight attacks. That is, to avoid detection, the attacker has to exploit very low attack magnitudes which makes little damage on the system.

Notation: $\mathbb{R}$ denotes the set of real numbers. $tr(\cdot)$ denotes the trace of a matrix and $(\cdot)^T$ denotes the transpose of a matrix or a vector. $|A|$ denotes the cardinality of a set A. $\mathbb{E}[\cdot]$ refers to the expectation, and $p(\cdot)$ represents the probability.

II. MODEL

A. SYSTEM MODEL

Consider a discrete linear time-invariant system described by

$$x(k+1) = Ax(k) + \omega(k) \tag{1}$$

$$y_i(k) = H_i x(k) + v_i(k) \tag{2}$$

where $x(k) \in \mathbb{R}^m$ is the system state, $y_i \in \mathbb{R}^m$ is the measurement of sensor i, $A \in \mathbb{R}^{m\times m}$ is the system matrix, $H_i \in \mathbb{R}^{m\times m}$ is the measurement matrix of sensor i, the process noise $\omega(k)$ and the measurement noise $v_i(k)$ are mutually uncorrelated white Gaussian noises with covariances $Q > 0$ and $R_i > 0$, respectively.

The sensor network is modeled as a directed graph $\mathfrak{G} = (\mathcal{V}, \mathcal{D})$ with a set of nodes $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of edges $\mathcal{D} \subset \mathcal{V} \times \mathcal{V}$. The edge $(i, j) \in \mathcal{D}$ means that the data can be transmitted from the jth sensor to the ith sensor. Define the in-neighbors of the ith sensor as $N_i = \{j : (i, j) \in \mathcal{D}\}$, i.e., the in-neighbors of the ith sensor are those sensors who can send data to the ith sensor. Let $d_i = |N_i|$ be the number of in-neighbors of sensor i. Denote the out-neighbors of sensor i as $\overline{N}_i = \{j : (i, j) \in \mathcal{D}\}$, i.e., the sensors can receive data from the ith sensor.

Next, we define some quantities which will be used in the remainder of the paper:

$$\hat{x}_i(k+1) \triangleq \mathbb{E}[x(k+1)|y_i(k)] \tag{3}$$

$$e_i(k+1) \triangleq x(k+1) - \hat{x}_i(k+1) \tag{4}$$

$$P_i(k+1) \triangleq \mathbb{E}[(x(k+1) - \hat{x}_i(k+1))(\cdot)^T | y_i(k)] \tag{5}$$

We propose a distributed state estimator for sensor i:

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + K_p^i(k)[y_i(k) - H_i\hat{x}_i(k)] - \varepsilon A \sum_{j\in N_i} \left(\hat{x}_i(k) - \hat{x}_j(k)\right) \tag{6}$$

where $\hat{x}_i(k)$ is the estimate of sensor i at time step k, $K_p^i(k)$ is the estimator gain to be designed, and ε is the consensus gain in the range of $\left(0, \frac{1}{\Delta}\right)$, with $\Delta = \max_i d_i$. Define the innovation of sensor i as $\Delta_i(k) = y_i(k) - H_i\hat{x}_i(k)$,

$i = 1, 2, \ldots, n$. At each time step, sensor i sends its estimate $\hat{x}_i(k)$ and innovation $\Delta_i(k)$ to its out-neighbors.

B. ATTACK MODEL

Consider a kind of false data injection attack. Let $q_{ij}(k)$, $j \in N_i$ indicate whether the attacker launches attack on the edge (i, j), i.e.,

$$q_{ij}(k) = \begin{cases} 1, & \text{if the edge } (i,j) \text{ is attacked}, j \in N_i \\ 0, & \text{otherwise} \end{cases}$$

where the attack probability $p\Big(q_{ij}(k) = 1\Big) = \xi_q$, and $q_{ij}(k)$ is independent of $q_{rs}(k)$ while $(i, j) \neq (r, s)$.

When edge (j, i) is attacked, i.e., $q_{ji}(k) = 1$, the transmitted data $\hat{x}_i(k)$ and $\Delta_i(k)$ become $\hat{x}_i^*(k) = T(k)\hat{x}_i(k) + \rho\alpha_{ji}(k)$, $\Delta_i^*(k) = T(k)\Delta_i(k) + \rho\alpha_{ji}(k)$, respectively. Here, $T(k)$ is a matrix that changes randomly with k, ρ is a given constant quantifying the amplitude of the injected false data, and $\alpha_{ji}(k) \in \mathbb{R}^m$ is zero-mean white Gaussian with covariance Θ_{ji}, i.e., $\alpha_{ji}(k) \sim \mathcal{N}(0, \Theta_{ji})$. Assume that $\alpha_{ij}(k)$ is independent of $\alpha_{sr}(t)$, when $(i, j) \neq (r, s)$ or $k \neq t$.

C. DISTRIBUTED ESTIMATOR WITH DETECTOR

In the previous sections, we have introduced a healthy system model and an attack model. In the following, a detector based on Q-table is proposed to detect the abnormal data. The specific design of the detector will be given in the later section. Here, we design a distributed estimator with the detector for each sensor as follows:

$$\hat{x}_i(k+1) = A\hat{x}_i(k) + K_p^i(k)[y_i(k) - H_i\hat{x}_i(k)] - \varepsilon A\sum_{j\in N_i}\gamma_{ij}(k)\Big(\hat{x}_i(k) - \hat{x}_j^*(k)\Big) \quad (7)$$

where $\gamma_{ij}(k)$ is a binary variable representing the protection decision. If the detector of sensor i regards the edge (i, j) being attacked, i.e., the received data from sensor j is suspicious, then $\gamma_{ij}(k) = 0$; otherwise, $\gamma_{ij}(k) = 1$. Note that $\gamma_{ij}(k)$ is independent of $\gamma_{rs}(t)$ when $i \neq r, j \neq s$ or $k \neq t$.

Remark 1: The cross estimation error covariance between sensor i and sensor j is

$$\begin{aligned} P_{ij}(k+1) &= \mathbb{E}\{e_i(k+1)e_j^T(k+1)\} \\ &= F_i(k)P_{ij}(k)F_j^T(k) + Q \\ &\quad +\varepsilon F_i(k)\sum_{r\in N_j}\gamma_{jr}(k)[P_{ir}(k) - P_{ij}(k)]A^T \\ &\quad +\varepsilon A\sum_{s\in N_i}\gamma_{is}(k)[P_{sj}(k) - P_{ij}(k)]F_j^T(k) \\ &\quad +\varepsilon^2 A\sum_{s\in N_i}\sum_{r\in N_j}\gamma_{is}(k)\gamma_{jr}(k)\Big[P_{sr}(k) \\ &\quad +P_{ij}(k) - P_{sj}(k) - P_{ir}(k) + \Big(\hat{x}_i(k) - \hat{x}_s(k)\Big) \\ &\quad \hat{x}_r^T(k)\Big(I - T(k)\delta_{jr}(k)\Big)^T + \Big(I - T(k)\delta_{is}(k)\Big) \\ &\quad \hat{x}_s(k)\Big(\hat{x}_j(k) - \hat{x}_r(k)\Big)^T + \Big(I - T(k)\delta_{is}(k)\Big) \\ &\quad \hat{x}_s(k)\hat{x}_r^T(k)\Big(I - T(k)\delta_{jr}(k)\Big)^T\Big]A^T \end{aligned} \quad (8)$$

We can derive the optimal estimator gain of the proposed estimator (7).

Lemma 1: The optimal estimator gain in (7) which minimizes the estimation error covariance $P_i(k)$ is given by

$$K_p^{i*}(k) = A\Big\{P_i(k) + \varepsilon\sum_{r\in N_i}\gamma_{ir}(k)[P_{ri}(k) - P_i(k)]\Big\} H_i^T M_i^{-1}(k) \quad (9)$$

where $M_i(k) = H_iP_i(k)H_i^T + R_i$, $i = 1, 2, \ldots, n$.

Proof: Let $F_i(k) = A - K_p^i(k)H_i$,

$$\delta_{ij}(k) = \begin{cases} I, & \text{if } q_{ij}(k) = 1, j \in N_i \\ T(k)^{-1}, & \text{otherwise} \end{cases}$$

Then, we have

$$\begin{aligned} P_i(k+1) &= \mathbb{E}\{e_i(k+1)e_i^T(k+1)\} \\ &= F_i(k)P_i(k)F_i^T(k) + Q + K_p^i(k)R_iK_p^{iT}(k) \\ &\quad +\varepsilon F_i(k)\sum_{r\in N_i}\gamma_{ir}(k)[P_{ir}(k) - P_i(k)]A^T \\ &\quad +\varepsilon A\sum_{r\in N_i}\gamma_{ir}(k)[P_{ri}(k) - P_i(k)]F_i^T(k) \\ &\quad +\varepsilon^2 A\sum_{r\in N_i}\gamma_{ir}^2(k)\Big[P_i(k) + P_r(k) - P_{ri}(k) \\ &\quad -P_{ir}(k) + \Big(\hat{x}_i(k) - \hat{x}_r(k)\Big)\hat{x}_r^T(k) \\ &\quad \Big(I - T(k)\delta_{ir}(k)\Big)^T + \Big(I - T(k)\delta_{ir}(k)\Big)\hat{x}_r(k) \\ &\quad \Big(\hat{x}_i(k) - \hat{x}_r(k)\Big)^T + \Big(I - T(k)\delta_{ir}(k)\Big)\hat{x}_r(k) \\ &\quad \hat{x}_r^T(k)\Big(I - T(k)\delta_{ir}(k)\Big)^T\Big]A^T + \varepsilon^2 A \\ &\quad \sum_{\substack{r,s\in N_i \\ r\neq s}}\gamma_{ir}(k)\gamma_{is}(k)\Big[P_i(k) + P_{rs}(k) \\ &\quad -P_{ri}(k) - P_{is}(k) + \Big(\hat{x}_i(k) - \hat{x}_r(k)\Big) \\ &\quad \hat{x}_s^T(k)\Big(I - T(k)\delta_{is}(k)\Big)^T + \Big(I - T(k)\delta_{ir}(k)\Big) \\ &\quad \hat{x}_r(k)\Big(\hat{x}_i(k) - \hat{x}_s(k)\Big)^T + \Big(I - T(k)\delta_{ir}(k)\Big) \\ &\quad \hat{x}_r(k)\hat{x}_s^T(k)\Big(I - T(k)\delta_{is}(k)\Big)^T\Big]A^T \\ &\quad +\rho^2\varepsilon^2 A\sum_{r\in N_i}\gamma_{ir}(k)q_{ir}(k)\Theta_r A^T \end{aligned} \quad (10)$$

Due to the random sequence $\{\gamma_{ij}\}_0^\infty$, the equation (10) is stochastic and cannot be determined. Rewrite (10) as

$$\begin{aligned} P_i(k+1) &= A\Big\{P_i(k) + \varepsilon\sum_{r\in N_i} \\ &\quad \gamma_{ir}(k)[P_{ir}(k) + P_{ri}(k) - 2P_i(k)] \\ &\quad +\varepsilon^2\sum_{r\in N_i}\gamma_{ir}^2(k)\Big[P_i(k) + P_r(k) - P_{ri}(k) \end{aligned}$$

$$
\begin{aligned}
&-P_{ir}(k)+\Big(\hat{x}_i(k)-\hat{x}_r(k)\Big)\hat{x}_r^T(k)\\
&\Big(I-T(k)\delta_{ir}(k)\Big)^T+\Big(I-T(k)\delta_{ir}(k)\Big)\hat{x}_r(k)\\
&\Big(\hat{x}_i(k)-\hat{x}_r(k)\Big)^T+\Big(I-T(k)\delta_{ir}(k)\Big)\hat{x}_r(k)\\
&\hat{x}_r^T(k)\Big(I-T(k)\delta_{ir}(k)\Big)^T\Big]\\
&+\varepsilon^2\sum_{\substack{r,s\in N_i\\ r\neq s}}\gamma_{ir}(k)\gamma_{is}(k)\Big[P_i(k)+P_{rs}(k)\\
&-P_{ri}(k)-P_{is}(k)+\Big(\hat{x}_i(k)-\hat{x}_r(k)\Big)\\
&\hat{x}_s^T(k)\Big(I-T(k)\delta_{is}(k)\Big)^T+\Big(I-T(k)\delta_{ir}(k)\Big)\\
&\hat{x}_r(k)\Big(\hat{x}_i(k)-\hat{x}_s(k)\Big)^T+\Big(I-T(k)\delta_{ir}(k)\Big)\\
&\hat{x}_r(k)\hat{x}_s^T(k)\Big(I-T(k)\delta_{is}(k)\Big)^T\Big]\\
&+\rho^2\varepsilon^2\sum_{r\in N_i}\gamma_{ir}(k)q_{ir}(k)\Theta_r\Big\}A^T+Q\\
&-A\Big\{P_i(k)+\varepsilon\sum_{r\in N_i}\gamma_{ir}(k)[P_{ri}(k)-P_i(k)]\Big\}\\
&H_iM_i^{-1}(k)H_i^T\Big\{P_i(k)+\varepsilon\sum_{r\in N_i}\gamma_{ir}(k)\\
&[P_{ir}(k)-P_i(k)]\Big\}A^T+[K_p^i(k)-K_p^{i*}(k)]\\
&M_i(k)[K_p^i(k)-K_p^{i*}(k)]^T
\end{aligned}
\tag{11}
$$

From above equation, it is clear that $P_i(k+1)$ is minimized at $K_p^i(k)=K_p^{i*}(k)$. Then, the proof is completed.

Using the above formulas, the state estimation can be updated.

III. PROBLEM FORMULATION

In this paper, we formulate the online attack detection problem as a partially observable Markov decision process (POMDP) problem. A discrete-time POMDP simulates the relationship between an agent and its environment. Formally, POMDP is defined by the seven-tuple $(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R},\mathcal{O},\mathcal{G},\lambda)$ where $\mathcal{S}$ denotes the set of states of the environment, $\mathcal{A}$ denotes the set of actions of the agent, $\mathcal{T}$ denotes the set of conditional transition probabilities between states, $\mathcal{R}:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ denotes the reward function, $\mathcal{O}$ denotes the set of observations of the agent, $\mathcal{G}$ denotes the set of conditional observation probabilities, and $\lambda\in[0,1]$ denotes a discount factor that determines the present value of future rewards.

At each time t, the environment is in a certain state $s_t\in\mathcal{S}$. The agent takes an action $a_t\in\mathcal{A}$, which results in a transition of the environment to the next state s_{t+1} with the probability $\mathcal{T}(s_{t+1}|s_t,a_t)$. At the same time, the agent obtains an observation $o_t\in\mathcal{O}$ depending on the current state of the environment with the probability $\mathcal{G}(o_t|s_t)$. Finally, the agent receives a reward $r_t=\mathcal{R}(s_t,a_t)$ from the environment based on its action and the current state of the environment. The process is repeated until a terminal state is achieved. In this process, the goal of the agent is to determine an optimal policy $\pi:\mathcal{O}\to\mathcal{A}$ that maximizes the agent's expected total discounted rewards, i.e., $\mathbb{E}[\sum_{k=0}^{\infty}\lambda^k r_{t+k+1}]$. Similarly, if an agent receives costs instead of rewards from the environment, then the goal is to minimize the excepted total discounted costs. Considering the latter, the POMDP problem can be rewritten as follows:

$$
\min_{\pi:\mathcal{O}\to\mathcal{A}}\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\lambda^k r_{t+k+1}\right] \tag{12}
$$

The false data injection attacks mentioned above are applied to online attack detection. This problem can be expressed as a POMDP problem (see Fig. 1). We assume that before an unknown time τ, the system is operated normally, the attacker does not launch attacks. After τ, the attacker launches false data injection attacks to the sensor's communication channel stochastically. It will change the state estimation and innovation. Due to unknown attack launch time and random attack, there are three hidden states: *pre-attack-time*, *post-attack-time without attack* and *post-attack-time with attack*. At each time, after obtaining the averaged measurement, two actions are available for the agent: *stop* and declare an attack or *continue* to have further measurements. Whenever the action *stop* is chosen, the system moves into a *terminal* state, and stays there afterwards. The whole process is plotted as Fig. 1.

Next, we explain the online attack detection problem in a POMDP setting. Different costs should be assigned in different states with different actions:

Step1: If the current moment is before the time τ, or the current moment is after the time τ, but at the moment, the attacker does not launch attacks on either edge while the system regards that it is under attack, i.e., the agent takes the action *stop*, then the agent receives a cost of 2. Skip to Step 5.

Step2: If the current moment is before the time τ, or the current moment is after the time τ, and at the moment, the attacker does not launch attacks on either edge while the system does not regard that it is under attack, i.e., the agent takes the action *continue*, then the agent receives zero cost. Move the process forward to the next moment.

Step3: If the current moment is after the time τ and the attacker launches attacks on m direct-edges, while the system does not regard that it is under attack, then the agent receives a cost of $\frac{cost\_single\times m}{num}$, where *num* is the number of direct-edges in the sensor network and *cost_single* is the unit cost of each edge. Move the process forward to the next moment.

Step4: If the current moment is after the time τ, at the moment the attacker launches attacks and the system thinks it is under attack, then the agent receives zero cost. Skip to Step 5.

Step5: Terminal. End the current process.

Due to the unknown attack launch time and the unknown attacking strategies, the exact POMDP model is unknown. And since the RL algorithms are known to be effective under

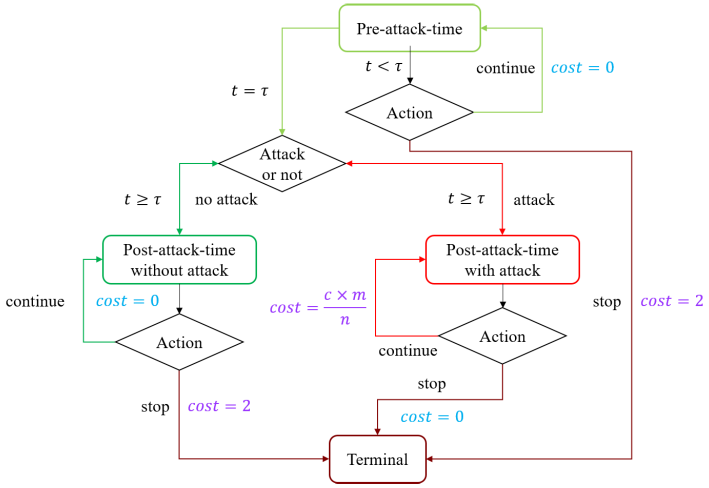


FIGURE 1. The relevant settings of POMDP.

uncertain environments, we follow a model-free RL algorithm to obtain a solution of (12). Then a mapping from observations to the actions, i.e., Q-table, needs to be learned.

Since different hidden states require different optimal actions, additional information obtained from the historical data of observations should be used to further reduce the uncertainty of the underlying states. Due to the limitation of resources, only finite data can be used in practice to obtain the approximately optimal solution. There are several ways to deal with the above situation, one of which is to use a finite-size sliding window of observations as a cache and map the recent window to an action. Let $f(\cdot)$ be the function that processes a finite historical data of measurements and produces the observation signal, so that the observation signal at time k is $o_k = f(\{\Delta_1(k), \Delta_2(k), \ldots, \Delta_n(k)\})$. At each time, the detector of the sensor i observes $\Delta_j(k), j \in N_i$ and decides whether to use the information from the neighbors of sensor i. The subsequent section details how to use an RL algorithm to obtain a solution of (12).

IV. SOLUTION APPROACH

A. INTRODUCTION OF ALGORITHM

Firstly, we define the observation signal $o_k = f(\{\Delta_1(k), \Delta_2(k), \ldots, \Delta_n(k)\})$. Let

$$\eta_k = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{d_i}\sum_{j\in N_i}\Delta_j^2(k) \tag{13}$$

In the above formula, the innovation plays a key role in judging whether the received data has been attacked. A larger norm of innovation corresponds to a higher probability of data being attacked. Hence, η_k could reduce the uncertainty of the underlying states to some extent.

Since η_k can take any nonnegative value, the observation space is continuous so that the RL problem cannot be solved in limited time with limited resources. To address this issue, we can quantify the observation space into I mutually exclusive and disjoint intervals using the quantization thresholds $\beta_0 = 0 < \beta_1 < \cdots < \beta_{I-1} < \beta_I = \infty$. When the observation value η_k falls into the interval $[\beta_{i-1}, \beta_i)$, i.e., $\beta_{i-1} \leq \eta_k < \beta_i, i \in 1, \ldots, I$, the observation at time k is represented with θ_i. In this way, all possible observations are $\theta_1, \theta_2, \ldots, \theta_I$. Each θ_i needs to be assigned to a different value.

Furthermore, η_k is possible to obtain identical observations in different states. For this reason, as mentioned above, we use a finite historical data of observations. The size of the sliding window is denoted as M so that there are I^M possible observation windows and the sliding window at time k is $\{\eta_j : k - M + 1 \leq j \leq k\}$. Based on the above, we regard an observation o as an observation window so that the observation space $\mathcal{O}$ contains all possible observation windows.

For each possible observation-action pair (o, a), we try to learn a $Q(o, a)$ value using an RL algorithm where all $Q(o, a)$ values are stored in a Q-table of size $I^M \times 2$. After learning the Q-table, for each observation o, the detector will choose the action a that minimizes the $Q(o, a)$ value.

The considered RL-based detection scheme consists of learning and online detection phases. Using the Sarsa algorithm [29] to train the Q-table, and then the trained Q-table will be the basis for online attack detection to defend the attacker. A simulation environment is created for training procedure. At each time, the agent (detector) takes an action based on its observation and receives a cost in return of its action from the simulation environment. Then, in the online detection phase, for each time, according to the observations, the Q-table learned before is used to choose the action with the lowest expected future cost. If the action *stop* is chosen by the detector, then the sensor will not use the information from the corresponding neighbor, otherwise the sensor will use the information from the neighbor to have further measurements.

The learning and online attack detection algorithms are summarized in Algorithm 1 and 2, respectively.

Algorithm 1 Learning Phase—SARSA Algorithm

Initialize $Q(o, a)$ arbitrarily, $\forall o \in \mathcal{O}, \forall a \in \mathcal{A}$.
for $e = 0 : E - 1$ **do**
  $k \leftarrow 0$
  $s \leftarrow$*pre-attack*
  $cost \leftarrow 0$
  Choose an initial o based on the *pre-attack-time* state and choose the initial $a = continue$.
  while $s \neq terminal$ and $k < T$ **do**
    if $a == stop$ **then**
      $s \leftarrow terminal$
      if $k < \tau$ or $cost == 0$ **then**
        $cost \leftarrow 2$
      else
        $cost \leftarrow 0$
      end if
      $Q(o, a) \leftarrow Q(o, a) + \alpha\Big(cost - Q(o, a)\Big)$
    else if $a == continue$ **then**
      $cost \leftarrow 0$
      if $k \geq \tau$ **then**
        Launch the attack to each edge stochastically. Assume that there are m direct-edges attacked and num edges in total.
      end if
      $cost \leftarrow \frac{cost\_single \times m}{num}$
      if $k \geq \tau$ and $cost == 0$ **then**
        $s \leftarrow$*post-attack-time with no attack*
      else if $k \geq \tau$ and $cost \neq 0$ **then**
        $s \leftarrow$*post-attack-time with attack*
      end if
      Collect the measurements $y_i(k)$.
      Employ the Kalman filter using (7),(8),(9), (10) and obtain the innovation at the moment.
      Compute η_k using (13) and quantize it to obtain θ_i if $\beta_{i-1} < \eta_k < \beta_i$.
      Update the sliding observation window o with the most recent entry θ_i and obtain o'.
      Choose action a' from o' using the ϵ-greedy policy based on the Q-table (that is being learned).
      $Q(o, a) \leftarrow Q(o, a) + \alpha\Big(cost + Q(o', a') - Q(o, a)\Big)$
      $o \leftarrow o'$
      $a \leftarrow a'$
    end if
    $k \leftarrow k + 1$
  end while
end for
Output: Q-table

Algorithm 2 Online Attack Detection

Input: Q-table learned in Algorithm 1.
for $loop = 0 : fre - 1$ **do**
  Choose an initial o based on the *pre-attack-time* state and choose the initial $a = continue$ for each sensor.
  for $k = 0 : T - 1$ **do**
    for $i = 0 : n - 1$ **do**
      if $k \geq \tau$ **then**
        Launch attack to each edge stochastically.
      end if
      Compute the square of the innovation of r, $r \in N_i$, i.e., $\Delta_r^2(k)$ and quantize it to obtain θ_j if $\beta_{j-1} < \Delta_r^2(k) < \beta_j$.
      Update the sliding observation window o_r with the most recent entry θ_j and obtain o_r'. Choose the action a with the minimum $Q(o_r', a)$.
      if $a == stop$ **then**
        Sensor i will not use the information of sensor r.
      else
        Sensor i will use the information of sensor r.
      end if
      Collect the measurements $y_i(k)$.
      Employ the Kalman filter using (7),(8),(9), (10) and obtain the innovation at the moment.
    end for
    Compute $inno\_avg_i = \frac{1}{d_i}\sum_{j \in N_i} \Delta_j^2(k)$ for each sensor i. Quantize it to obtain θ_j if $\beta_{j-1} < inno\_avg_i < \beta_j$. Update the sliding observation window o_i with the most recent entry θ_j. Let $o_i \leftarrow o_i'$.
  end for
end for
Compute the trace of the averaged estimation error covariance $J(k) = \frac{1}{n}\sum_{i=0}^{n-1} tr\Big(P_i(k)\Big)$ and plot the averaged trace over time.

In Algorithm 1, E denotes the number of learning episodes, T denotes the maximum length of a learning episode, α is the learning rate, and ϵ is the exploration rate. The agent (defender) takes ϵ-greedy policy to choose the action that minimizes Q value with probability $1 - \epsilon$ and the other action with probability ϵ. In Algorithm 2, to verify the performance of the system with detector, we compute and plot the trace of the averaged estimation error covariance overtime.

B. COMPLEXITY ANALYSIS

In order to obtain the time complexity of SARSA algorithm, we firstly consider the time complexity of a single iteration. Since SARSA algorithm only updates Q-table once at each time, and the update of Q-table requires the distributed state estimation algorithm, and considering that the maximum time limit for a learning episode is T, the time complexity of a single iteration is $O(n^4 \times T)$, where n is the number of sensors. Furthermore, the time complexity of the whole algorithm is

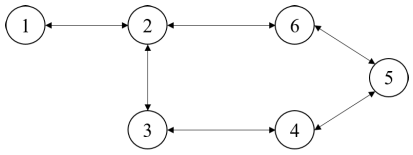
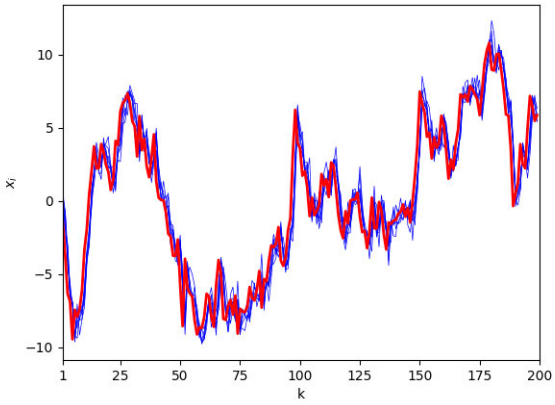
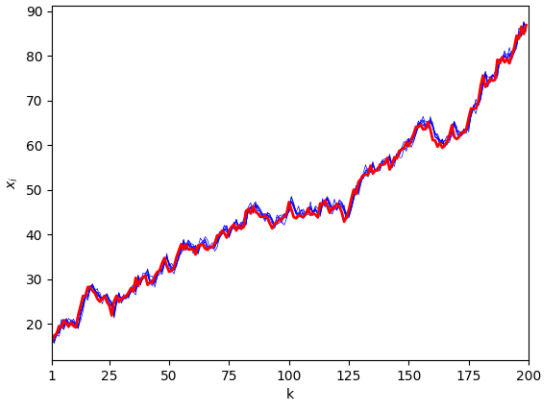


FIGURE 2. Network topology.



(a) The first dimension



(b) The second dimension

FIGURE 3. The performance of sensors (blue curve) tracking the considered system (red curve).

$O(n^4 \times ET)$, where E is the number of learning episodes. On the other hand, due to the size of the sliding window M and the size of the Q-table $I^M \times 2$, the space complexity of Algorithm 1 is $M + 2I^M$.

In Algorithm 2, the detection phase are repeated for *fre* times. For each time, the observation o is obtained from each sensor and the Q-table trained by Algorithm 1 is used to choose the action a that minimizes the expected total discounted costs. Due to the distributed state estimation algorithm, the time complexity of a single iteration is $O(n^4 \times T)$. Hence, the total time complexity of the detection phase is $O(fre \times n^4 \times T)$. Furthermore, as the size of the Q-table is $I^M \times 2$, and for each sensor, the size of the sliding window is M, the space complexity of Algorithm 2 is $nM + 2I^M$, where n is the number of sensors.

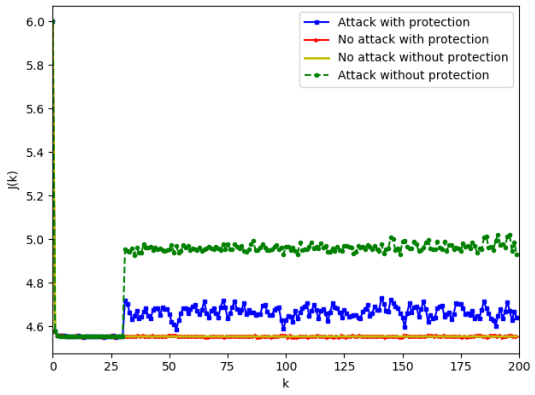
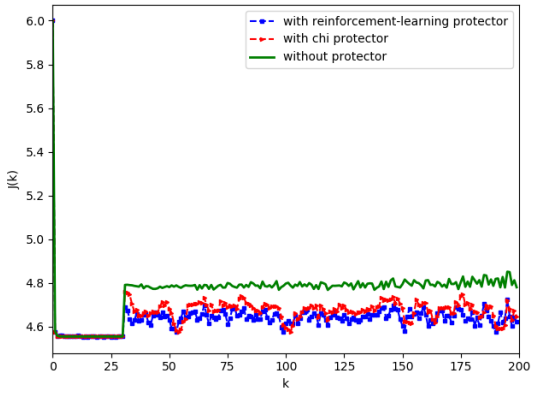
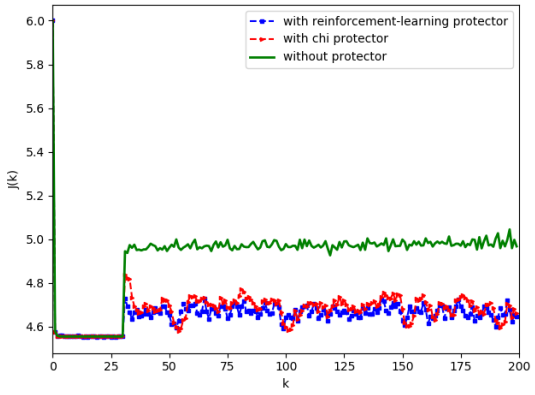


FIGURE 4. Comparison of estimation performance in different cases. All the curves are obtained after averaged over 100 runs.



(a) $\rho = 3$



(b) $\rho = 4$

FIGURE 5. The comparison between the reinforcement learning detector and the χ^2 detector under different attacker power.

V. SIMULATION

We present simulation examples to illustrate the performance of the proposed estimator with the detector under data injection attacks. Consider a sensor network with 6 sensors (see Fig. 2). The system parameters are defined as follows:

$$A = \begin{bmatrix} 1.01 & 0 \\ 0 & 1.01 \end{bmatrix}, \quad H_i = \begin{bmatrix} 1+\zeta_i & 0 \\ 0 & 1+\zeta_i \end{bmatrix}$$

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad R_i = \begin{bmatrix} 2\varsigma_i & 0 \\ 0 & 2\varsigma_i \end{bmatrix}$$

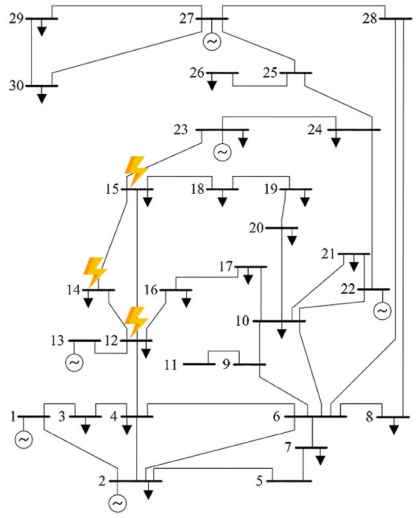


FIGURE 6. IEEE 30-bus test case.

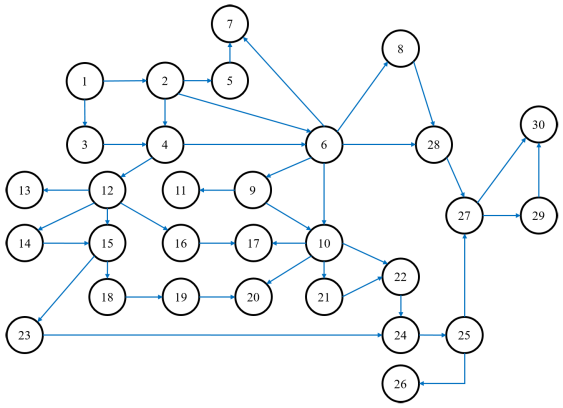


FIGURE 7. IEEE 30-bus network topology.

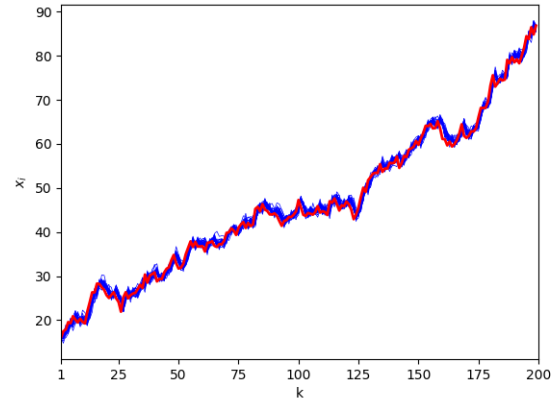
$$\Theta_i = \begin{bmatrix} 2\delta_i & 0 \\ 0 & 2\delta_i \end{bmatrix}, \quad T(k) = \begin{bmatrix} 1+0.02\xi(k) & 0.02\xi(k) \\ 0.02\xi(k) & 1+0.02\xi(k) \end{bmatrix}$$

where $\zeta_i, \varsigma_i, \delta_i, \xi(k) \in [0, 1), \forall i, k$, and $\varepsilon = 0.1$.

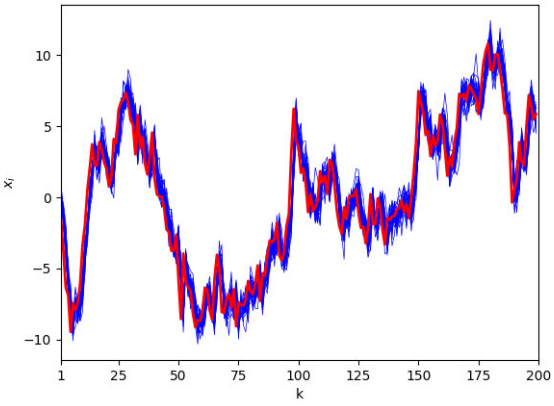
Set the attack probability on each edge to be $\lambda_q = 0.6$, the amplitude of the injected false data to be $\rho = 3$, $\rho = 4$.

For the proposed RL-based online attack detection scheme, the number of quantization levels is $I = 4$ and the quantization thresholds are chosen as $\beta_1 = 10, \beta_2 = 20, \beta_3 = 30$. Further, sliding observation window is chosen to be $M = 2$, and the episode length of learning phase and detection phase is $T = 100$ and $T = 200$, respectively. Moreover, the learning rate $\alpha = 0.1$ and the exploration rate $\epsilon = 0.1$. In the learning phase, firstly, the defender is trained over 5000 episodes with stochastic attack launch time $\tau = 50$, and then trained over 5000 episodes with $\tau = 5$ to ensure that the defender can fully explore the observation space under normal operation conditions and stochastic attack conditions. The cost of accepting a single attacked channel is set as 0.5.

First, we verify the stability of system with detector. As Fig. 3 shows, all the sensors (blue curve) can track the unstable system (red curve) as time evolves. Further, we verify the effectiveness of the proposed protector with



(a) The first dimension



(b) The second dimension

FIGURE 8. The performance of sensors (blue curve) tracking the IEEE 30-bus system (red curve).

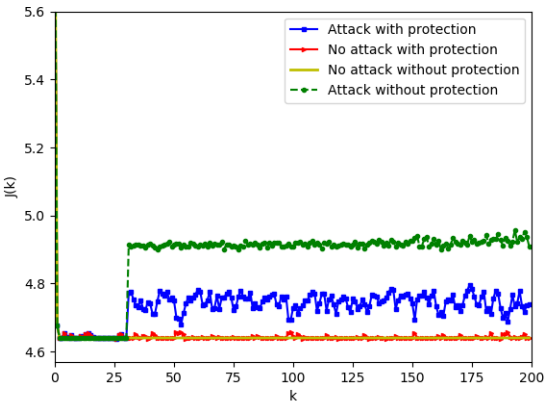
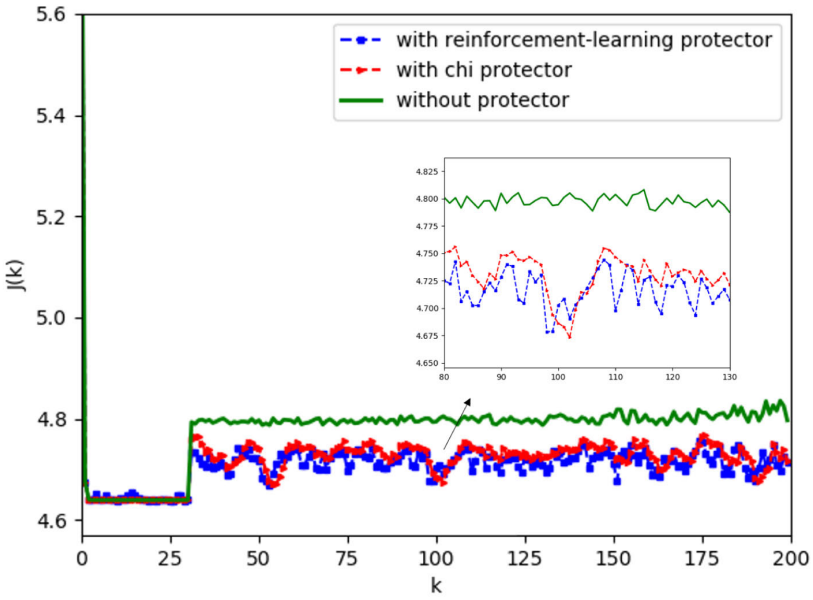


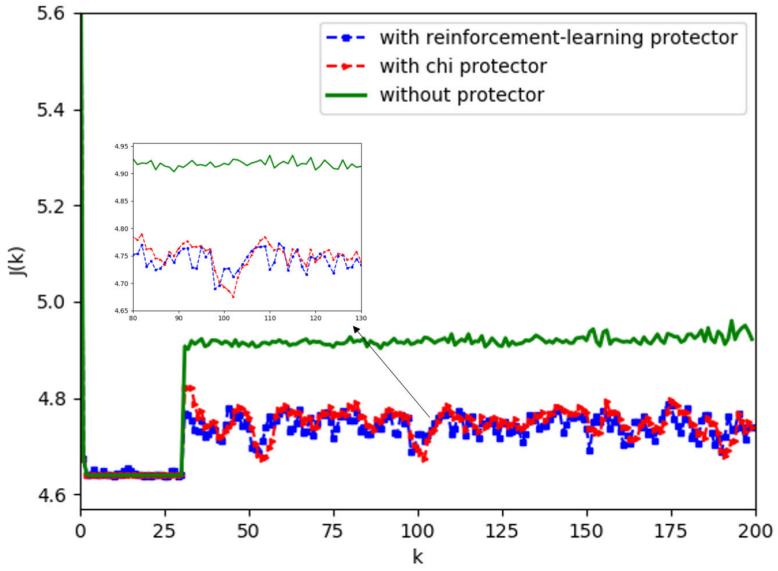
FIGURE 9. Comparison of estimation performance under different cases in the IEEE 30-bus system. All the curves are obtained after averaged over 100 runs.

the amplitude $\rho = 4$ by comparing the performances of four different cases, i.e., no attack without protection, no attack with protection, having attack without protection, having attack with protection. To do so, we define the trace of the averaged estimation error covariance (EEC), as

$$J(k) = \frac{1}{n} \sum_{i=1}^{n} tr\Big(P_i(k)\Big)$$



(a) $\rho = 3$



(b) $\rho = 4$

FIGURE 10. The comparison between the reinforcement learning detector and the χ^2 detector under different attacker power in the case of the IEEE 30-bus.

As Fig. 4 shows, the curve corresponding to the case of no attack with protection almost overlaps with the curve corresponding to the case of no attack without protection, except some time with false positives, which implies that the proposed protector has little impact on the estimation performance when the network is not being attacked. Obviously, the curve corresponding to the case of having attack without protection diverges asymptotically as time evolves. When the proposed protector is employed for each sensor, the trace of the estimation error covariance drops significantly and converge asymptotically, which implies that the proposed protector can resist attacks effectively.

In the following, we compare the detectors under three different cases, i.e., having attack without protection, having attack with the χ^2 failure detector, having attack with the proposed detector. As Fig. 5 shows, the averaged EEC of the healthy system converges to a steady state quickly. The attacker begins to launch the false data injection attacks at 30 time step. Furthermore, we discuss two cases, i.e., the amplitude of the injected false data $\rho = 3$ and $\rho = 4$. In the first case, the χ^2 failure detector and the detector proposed in this paper both have better performance than the case without protection. Furthermore, the reinforcement learning detector has a slight advantage over the χ^2 failure

TABLE 1. The negative false probability and the positive false probability.

		Reinforcement Learning Detector	χ^2 Detector
$\rho = 3$	False Positive Probability	26.72%	27.16%
	False Negative Probability	43.12%	49.28%
$\rho = 4$	False Positive Probability	27.70%	37.37%
	False Negative Probability	32.75%	33.49%

detector. In the second case, as the attacker power becomes larger, the performance of the two detectors is still better than the case without protection. However, compared to the first case, the reinforcement learning detector has a weaker advantage over the other detector, which indicates that the smaller attack, the more obvious advantage of reinforcement learning detector.

Then we study the IEEE 30-bus system as shown in the Fig. 6. The corresponding network topology is given according to the voltage (see Fig. 7). The parameters of the system and the parameters of the attacker are the same as the previous simulation. Furthermore, the online attack detection scheme remains unchanged.

Fig. 8 shows a similar result to Fig. 3. As Fig. 9 shows, the curve corresponding to the case of no attack with protection implies that the proposed protector has little impact on the estimation performance when the network of IEEE 30-bus system is not being attacked. When the curve corresponding to the case of having attack without protection diverges asymptotically, the proposed protector can resist attacks effectively and make sure that the trace of the estimation error covariance converge as time evolves.

Same as the previous simulation, we compare the detectors under three different cases, i.e., having attack without protection, having attack with the χ^2 failure detector, having attack with the proposed detector. As Fig. 10 shows, when the amplitude is 3, the detector proposed in this paper has a slight advantage over the χ^2 failure detector. However, when the amplitude increases to 4, the reinforcement learning detector and the χ^2 failure detector have almost the same performance, which indicates that whether in the case of i6 sensors or the IEEE 30-bus system, the reinforcement learning detector has a significant advantage when the attack is small.

Finally, we explore the false negative probability and the false positive probability of the two detectors. As TABLE 1 shows, with the increase of attack amplitude, the false positive rate increases and the false negative rate decreases. Meanwhile, the false negative rate of reinforcement learning detector is always lower than that of χ^2 square detector and the false positive rate of χ^2 detector is greater than that of reinforcement learning detector. As discussed above, the smaller the attack, the greater advantage of reinforcement learning.

VI. CONCLUSION

In this paper, we investigate the problem of network security over wireless sensor networks under false data injection attacks. We formulate the online attack detection problem as a POMDP problem and propose a solution based on the model-free reinforcement learning. The detector of a sensor decides whether to use the received data from its neighboring sensors for the next estimate update. The simulation results illustrate the performance of the proposed detector on the trace of the averaged estimation error covariance (EEC) under the case of a network with 6 sensors and IEEE-30 bus test case. The proposed online detection method is widely applicable to any quickest change detection problem where the normal system can be modeled sufficiently accurately and the goal is to detect attacks online that are generally difficult to model. Moreover, we have considered a single-agent reinforcement learning setting with optimizing the policy of the defender where the policy of the attacker does not change.

In the future, we will develop more sophisticated memory techniques instead of the finite-size sliding window approach. The reinforcement learning method used in this paper is tabular method. That is, the continuous observation space is discretized to compute the Q values. We will try functional approximation methods, e.g., neural networks, to compute the Q values. As mentioned above, in the current setting, the attacker does not learn. Therefore, the current setup can be extended to multi-agent reinforcement learning in the future, where there is a game with multiple states in which the feedback, actions and state transitions are determined jointly by the defender and the attacker. In this setting, the optimal attack strategy and the optimal defense strategy will be obtained eventually.

REFERENCES

[1] A. A. Kumar, K. Ovsthus, and L. M. Kristensen, "An industrial perspective on wireless sensor networks—A survey of requirements, protocols, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1391–1412, 3rd Quart., 2014.

[2] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1630–1638, Jul. 2017.

[3] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, Oct. 2010, pp. 226–231.

[4] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2218–2234, May 2020.

[5] F. Li and Y. Tang, "False data injection attack for cyber-physical systems with resource constraint," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 729–738, Feb. 2020.

[6] R. Zhang and P. Venkitasubramaniam, "False data injection and detection in LQG systems: A game theoretic approach," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 1, pp. 338–348, Mar. 2020.

[7] L. Che, X. Liu, and Z. Li, "Mitigating false data attacks induced overloads using a corrective dispatch scheme," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3081–3091, May 2019.

[8] L. Che, X. Liu, and Z. Li, "Fast screening of high-risk lines under false data injection attacks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4003–4014, Jul. 2019.

[9] S. Li, Y. Yilmaz, and X. Wang, "Quickest detection of false data injection attack in wide-area smart grids," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2725–2735, Nov. 2015.

[10] W. Yang, Y. Zhang, G. Chen, C. Yang, and L. Shi, "Distributed filtering under false data injection attacks," *Automatica*, vol. 102, pp. 34–44, Apr. 2019.
[11] H. Liu, B. Niu, and Y. Li, "False-data-injection attacks on remote distributed consensus estimation," *IEEE Trans. Cybern.*, early access, Apr. 17, 2020, doi: 10.1109/TCYB.2020.2977056.
[12] Y. Hua, F. Chen, S. Deng, S. Duan, and L. Wang, "Secure distributed estimation against false data injection attack," *Inf. Sci.*, vol. 515, pp. 248–262, Apr. 2020.
[13] D. Ye and T.-Y. Zhang, "Summation detector for false data-injection attack in cyber-physical systems," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2338–2345, 2020.
[14] H. Yu, A. R. Mahmood, and R. S. Sutton, "On generalized bellman equations and temporal-difference learning," *J. Mach. Learn. Res.*, vol. 19, no. 48, pp. 1–49, 2018. [Online]. Available: http://jmlr.org/papers/v19/17-283.html
[15] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 2177–2182.
[16] D. Pandey and P. Pandey, "Approximate Q-learning: An introduction," in *Proc. 2nd Int. Conf. Mach. Learn. Comput.*, 2010, pp. 317–320.
[17] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Comput. Chem. Eng.*, vol. 139, Aug. 2020, Art. no. 106886.
[18] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5174–5185, Sep. 2019.
[19] B. Kiumarsi, F. L. Lewis, and Z.-P. Jiang, "H_∞ control of linear discrete-time systems: Off-policy reinforcement learning," *Automatica*, vol. 78, pp. 144–152, Apr. 2017.
[20] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Aug. 2009.
[21] R. Moghadam and F. L. Lewis, "Output-feedback H_∞ quadratic tracking control of linear systems using reinforcement learning," *Int. J. Adapt. Control Signal Process.*, vol. 33, no. 2, pp. 300–314, Feb. 2019.
[22] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.
[23] V. G. Lopez and F. L. Lewis, "Dynamic multiobjective control for continuous-time systems using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2869–2874, Jul. 2019.
[24] A. Nowé, P. Vrancx, and Y.-M. D. Hauwere, *Game Theory and Multiagent Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 441–470.
[25] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1994, pp. 157–163. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781558603356500271
[26] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. 15th Nat. Conf. Artif. Intell. (AAAI). 10th Conf. Innov. Appl. Artif. Intell.* Palo Alto, CA, USA: AAAI Press/MIT Press, 1998, pp. 746–752.
[27] M. Weinberg and J. S. Rosenschein, "Best-response multiagent learning in non-stationary environments," in *Proc. 3rd Int. Joint Conf. Auto. Agents Multiagent Syst.*, 2004, pp. 506–513.
[28] S. Zuo, Y. Song, F. L. Lewis, and A. Davoudi, "Optimal robust output containment of unknown heterogeneous multiagent system using off-policy reinforcement learning," *IEEE Trans. Cybern.*, vol. 48, no. 11, pp. 3197–3207, Nov. 2018.
[29] R. S. Sutton and A. G. Barto, "Temporal-difference learning," in *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998, pp. 133–160.

WEILIANG JIANG received the B.S. degree from the School of Science, East China University of Science and Technology, in 2019, where he is currently pursuing the M.S. degree with the School of Information Science and Engineering.

WEN YANG received the B.Sc. degree in mineral engineering and the M.Sc. degree in control theory and control engineering from Central South University, China, in 2002 and 2005, respectively, and the Ph.D. degree in control theory and control engineering from Shanghai Jiao Tong University, Shanghai, China, in 2009. She was a Visiting Student with the University of California at Los Angeles, from 2007 to 2008. She is currently a Professor with the East China University of Science and Technology (ECUST). Her research interests include information fusion, state estimation, network security, coordinated control, complex networks, and reinforcement learning.

JIAYU ZHOU received the M.S. degree from the School of Automation, Guangdong University of Technology, in 2019. He is currently pursuing the Ph.D. degree from East China University of Science and Technology. His research interests include state estimation, cyber-physical system security, and distributed filtering.

WENJIE DING received the B.S. degree in automation from the East China University of Science and Technology, in 2018, where he is currently pursuing the Ph.D. degree in control science and engineering.

YUE LUO received the B.S. degree in automation from the East China University of Science and Technology, in 2019, where he is currently pursuing the Ph.D. degree in control science and engineering.

YUN LIU received the M.S. degree in control science and engineering from the Guangdong University of Technology, Guangzhou, China, in 2020. He is currently pursuing the Ph.D. degree in control science and engineering with the East China University of Science and Technology, Shanghai, China. From August 2018 to November 2018, he was a Research Assistant with the Department of Mechanical Engineering, The University of Hong Kong. His current research interests include periodic systems, network security, and distributed filtering.