Catch Me If You Learn: Real-Time Attack Detection and Mitigation in Learning Enabled CPS

Ipsita Koley
Dept. of CSE
IIT Kharagpur
Kharagpur, India
ipsitakoley@iitkgp.ac.in

Sunandan Adhikary
Dept. of CSE
IIT Kharagpur
Kharagpur, India
sunandana@iitkgp.ac.in

Soumyajit Dey
Dept. of CSE
IIT Kharagpur
Kharagpur, India
soumya@cse.iitkgp.ac.in

2021 IEEE Real-Time Systems Symposium (RTSS) | 978-1-6654-2802-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/RTSS52674.2021.00023

***Abstract*—Increased dependence on networked, software-based control has escalated the vulnerabilities of Cyber-Physical Systems (CPSs). Detection and monitoring components developed leveraging dynamical systems theory are often employed as lightweight security measures for protecting such safety-critical CPSs against false data injection attacks. However, existing approaches do not correlate attack scenarios with parameters of detection systems. In this work, we propose real-time attack detection and mitigation strategies for safety-critical CPSs. A Reinforcement Learning (RL) based framework is presented which adaptively sets the parameters of such detectors based on experience learned from attack scenarios. The detection system is provided with a suitable training environment to learn how to maximize the detection rate while minimizing false alarms. Along with the objective of attack detection, our framework also attempts to preserve system performance by judiciously choosing control actions based on the operating region. Our proposed method i) mathematically establishes a detection strategy for fast and accurate FDI attack detection, ii) correlates the key parameters of the detection system by learning from attack scenarios, and iii) incorporates a real-time attack mitigation strategy that uses formally synthesized fast controllers, thus creating an end-to-end defense against FDI attacks for safety-critical CPSs. We evaluate our proposed method using well-known safety-critical CPS examples.**

***Index Terms*—cyber-physical systems, security, adaptive threshold, reinforcement learning, real time recovery, formal methods**

I. Introduction

While security is of paramount importance in CPS [1], the real time hard deadlines of safety critical functions and availability of limited computing resources constitute the major constraints to secure CPS design. CPSs are usually designed as closed loop feedback systems with both controller and estimator (like Luenberger or Kalman filter) in place. Both work together to ensure stability while minimizing the effect of noise. We consider that the communication channel between the plant and the controller-estimator unit is vulnerable, i.e. an attacker can gain access to the network and add spurious data to every communication between plant and controller. Such an attack is called *false data injection* (FDI) attack. Aim of an attacker would be to remain stealthy and destabilize the closed loop dynamics by failing the efficacy of the controller and the estimator. Traditional cryptographic techniques (for example, message authentication code (MAC) along with some encryption techniques like, RAS, AES etc.) that ensure integrity and authenticity of data, cannot be used in most of the cases to catch an FDI attack effort as they lead to both computation and communication overhead [2]. Therefore, it is difficult to secure every communication between the plant and controller.

An alternative design paradigm is to secure the data packets intermittently rather than regularly. For example, [3], [4] budget a worst case estimation error for the CPS and decide the number of consecutive iterations where security measures can be relaxed. This results in a choice of (l,m)-sporadic security implying every l consecutive sense/actuation values will be secured after every m consecutive insecure samples and the system can tolerate attacks with a theoretical guarantee of never violating the maximum error bound. Further refinements of the same philosophy naturally chooses suitable system parameters so that a schedulable sporadic security solution can be arrived at, such that the worst case tolerable error is minimized. This clearly is a correct-by-construction but worst case design method where the system continues executing in a degraded performance mode as long as attacks continue.

Another possible alternative to securing every communicated packet is designing light-weight attack detectors by exploiting control-theoretic properties. Such detectors leverage the features of the state-estimators. These estimators estimate the state of the plant by computing residue as the difference between estimated and observed sensor data. The detector either compares this residue directly with a predefined threshold to identify an anomaly [5] or applies statistical change detection methods, like χ^2-test, Cumulative Sum (CUSUM) [6], [7], etc. on the residue before the comparison. However, the constant threshold used by these residue based detectors may increase the false alarm rates (FAR) by misinterpreting measurement noise as attack, leading to unnecessary degradation in control performance. Moreover, recent research [8], [9] have shown how a *stealthy* attacker can fool such detectors by crafting perturbation sequences which create residues that are small enough (i.e. below threshold) to be classified as noise.

In light of the above discussion, we can say the primary challenges while designing a performance aware end-to-end

The authors acknowledge generous grant received from "IHUB NTIHAC Foundation - IIT Kanpur" for partially supporting this work.

978-1-6654-2802-6/21/$31.00 ©2021 IEEE
DOI 10.1109/RTSS52674.2021.00023

secure CPS are: *i) what would be the optimal threshold of an attack monitor so that FAR is minimized while even small attack efforts can be detected, and ii) if an attack effort is detected, how we can mitigate its effect at the earliest to preserve the real time performance of the system.* In the present work, we aim to address these questions. We present an intelligent secure CPS model that consists of: i) an adaptive attack monitor that thwarts an FDI attack, and ii) an efficient and fast controller that mitigates the effect of such attacks in minimum time. The overall framework is depicted in Fig. 1. We discuss each component of the proposed framework and thereby, the major contributions of this work as follow.

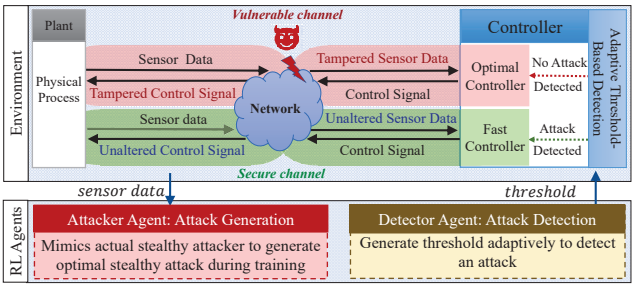


Fig. 1: RL based adaptive monitoring strategy

1. Adaptive Attack Detection: We propose an adaptive threshold based attack detector that leverages reinforcement learning (RL) for detecting whether a closed-loop dynamical system is under such stealthy FDI attack. The proposed RL framework learns from the affected system dynamics and adaptively tunes the threshold of the residue-based anomaly detector. The main motivation behind considering RL-based strategy comes from the following challenge. The false data injected into sensors and actuators by stealthy attacker are highly system-specific, random and does not follow any statistical distribution. Therefore, the parameters of the adaptive attack detector cannot be directly derived from the injected false data signature. The performance of the proposed detector depends on how well it is trained against optimal attack vectors. Thus, we also design an RL agent for mimicking the attacker's behavior during the training phase.

Similar attack detection strategies that balance between FAR and attack detectability has also been considered by [9], [10]. In [9], authors have defined the safety requirement of a CPS in terms of settling time. In such works, it is often straightforward to design FDI attacks through constraint solving such that the settling time property is satisfied while some other safety property gets violated causing critical damage to the system. On the contrary, our approach interprets system safety in terms of a safe operating region. Though, the authors of [10] have drawn similar motivation like ours and presented an attacker-defender game to solve the adaptive threshold selection problem, their work lacks consideration of the closed loop system dynamics of a CPS while modeling the attacker and the detector. Moreover, the optimization problem that they solve online to compute the correct threshold may incur timing overhead. On the other hand, the applicability of a well-trained RL algorithms on real time systems has already been established in the context of attacked state estimation [11] and energy efficiency [12] of safety-critical CPSs with real time requirements.

2. Fast Attack Mitigation: In an ideal scenario, when no attack takes place, the system operates with an optimal controller that ensures high performance by restricting the trajectory inside some *preferable* operating region. In the presence of an attack, system states are more likely to go beyond the preferable region. An FDI attacker can intelligently modify both sensor and actuation data in such a way that the controller is hoodwinked to think the system is still within preferable operating region. Therefore, instead of the tampered sensor measurements, we rely on the proposed detection system to identify an attack and accordingly the affected control loop is switched to a secure mode (like [4], [3]). During this mode, every communication between plant and controller is secured using some standard cryptographic methods, like MAC. Due to resource constraints incurred by the cryptographic method, it is safe to assume that only a subset of such control loops can be secured at the same time to ensure none of them misses their respective deadlines. Here, we implicitly also assume that not more than such a subset of loops are under attack simultaneously. Our objective here is to design an attack mitigation strategy that enables quicker transition to the preferable operating region so that the system can switch back to the optimal controller thus minimizing prolonged performance degradation. Moreover, this would imply spending minimum possible time in the secure mode so that other control loops can be accommodated in that mode. With the above motivation, we present a formal method for *offline* synthesis of control inputs that guarantee alleviating the effect of FDI attacks at the earliest as long as the attack is detected within the defined safety boundary of the system. This essentially lets the proposed system follow the Simplex architecture model [13], [14]. The adaptive attack detector works as the *decider* unit which determines which among the optimal controller (i.e. *high performance controller*) and the fast controller (i.e. *high assurance controller*) will be used based on whether it suspects an undesired behavior of the system.

Some researchers have also proposed attack mitigation strategies in context of secure CPS ([11], [15], [16], [17], [18]). The proposed method in [11] is applicable to a specific CPS i.e. connected vehicles and also the authors do not consider any active security primitive like detection systems. The approaches adopted by [15], [17] use a constant threshold based attack detector which may suffer from high FAR. A formal method based approach presented in [16] synthesizes controllers in the presence of an adversary. However, their method does not guarantee that the system can be brought back to the preferable operating region starting from anywhere within safety boundary. A recent work [18] proposes a real time recovery mechanism exploiting linear programming and formal methods. The major limitations of this work are as follows. They consider overly simplistic FDI attack scenarios,

for example, constant impurities being added to sensors for a limited window of consecutive samples. Any performance evaluation of their proposed online recovery control synthesis method against worst case attack scenarios is absent in their discussions. The criticality of an attack also depends on its onset time, state space region, attack length etc. Further, their system modeling does not explicitly consider stealthy attacks which bypass detection systems. Our proposed method of pre-synthesizing attack-mitigating fast control inputs takes care of all these limitations.

3. The attacker and detector agent constitute a ***multi-agent RL setup*** for training, where they collaborate with each other to learn the system environment better. A trained RL based attack detection module detects an FDI attack effort as early as possible. The fast controller module synthesizes system specific fast control inputs beforehand, which formally ensure fast recovery of system states back to the performance region. On deployment, the attack detection and mitigation modules operate in a complimentary fashion. Detection of an attack triggers the fast controller in secure channel. An early detection promises minimum deviation from desired trajectory and thereby, can reduce the fast control effort.

We discuss the principal components of our proposed intelligent secure CPS model in the following causal order. We present a mathematical formulation for synthesizing optimal detector thresholds in Sec. III-A. To define attacker's objective we need to distinguish between safe and unsafe trajectory of the system states. Therefore, in Sec. III-B, we formally define the operating regions of the system states and discuss a method to compute these regions. Following this, in Sec. III-C we mathematically formalize an optimal false data injection attack that aims to bypass the detector threshold (presented in Sec. III-A) and steer the system to unsafe region (as defined in Sec. III-B). Using formal methods, we present in Sec. III-D a region-based offline fast control input synthesis algorithm as demonstrated in contribution 2. Next, leveraging the mathematical formulations of Sec. III-A and Sec. III-C, we demonstrate the proposed multi-agent RL framework of contribution 3 in Sec. III-E. Finally, in Sec. III-F, we present an algorithm to describe the overall framework of the proposed learning enabled secure CPS model.

II. Secure CPS Model

General architecture of a secure CPS is presented in Fig. 2. The plant and the controller communicate between themselves over a network, which we consider is vulnerable to FDI attacks. In the absence of an adversary, the closed loop dynamics of a CPS can be presented as a discrete linear time-invariant (LTI) system.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k;\ y_k = Cx_k + Du_k + v_k; \\ \hat{y}_k &= C\hat{x}_k + Du_k;\ r_k = y_k - \hat{y}_k; \\ \hat{x}_{k+1} &= A\hat{x}_k + Bu_k + Lr_k;\ u_k = -K\hat{x}_k; \end{aligned} \tag{1}$$

where, $x_k \in \mathbb{R}^n$ is the system state vector, $y_k \in \mathbb{R}^m$ is the measurement vector obtained from available sensors at

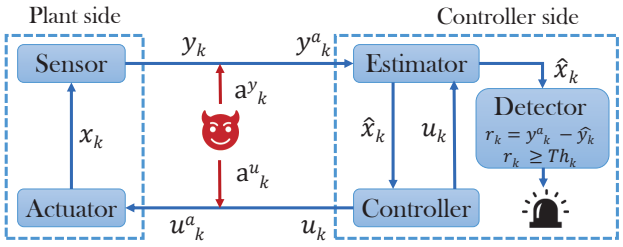


Fig. 2: Secure CPS architecture

k-th time stamp; A, B, C, D are the system matrices. We consider that the initial state $x_0 \in \mathcal{N}(\bar{x}_0, \Sigma)$, the process noise $w_k \in \mathbb{R}^n \sim \mathcal{N}(0, \Sigma_w)$ and the measurement noise $v_k \in \mathbb{R}^m \sim \mathcal{N}(0, \Sigma_v)$ are independent Gaussian random variables. Further, in every k-th sampling instant, the observable system state $\hat{x}_k$ is estimated using system output y_k while minimizing the effect of noise, and used for computing the control input $u_k \in \mathbb{R}^l$. The symbol r_k denotes the residue i.e. the difference between the measured and the estimated outputs. The estimator gain L and controller gain K ensure that both $(A-LC)$ and $(A-BK)$ are stable. The system has a detector unit (Fig. 1) which computes a function $f(r_k)$ and compares it with the current threshold Th_k to identify any anomalous behavior of the system. In case of a constant threshold based attack detector, $Th_k = Th\ \forall k$ where Th is a predefined fixed value. Considering an FDI attack, where the attacker injects false data a^y_k and a^u_k (Fig. 2) to the sensor data and control signal respectively, the equation of the system dynamics will become,

$$\begin{aligned} x^a_{k+1} &= Ax^a_k + B\tilde{u}^a_k + w_k;\ y^a_k = Cx^a_k + D\tilde{u}^a_k + v_k + a^y_k \\ \hat{y}^a_k &= C\hat{x}^a_k + Du^a_k;\ r^a_k = y^a_k - \hat{y}^a_k \\ \hat{x}^a_{k+1} &= A\hat{x}^a_k + Bu^a_k + Lr^a_k; u^a_k = -K\hat{x}^a_k;\ \tilde{u}^a_k = u^a_k + a^u_k \end{aligned} \tag{2}$$

Here, x^a_k, $\hat{x}^a_k$, y^a_k, r^a_k, u^a_k, $\tilde{u}^a_k$ represent plant state, estimated plant state, forged sensor data, residue, control signal, and forged control signal respectively in an attack scenario. In the present work we consider f as the popular χ^2-test commonly employed in existing works on secure CPS [5]. Note that even though we discussed about intrusion through network, physical level sensor data tampering [19] can also happen. The above attack model is generic to all such kind of falsification attacks.

III. Proposed Methodology

A. Optimal Threshold Synthesis

We present a residue based attack detection system where we consider the Kalman filter as the estimator. The proposed detector will adaptively select a threshold Th_k at every k-th sample. Let, for the discrete LTI system shown earlier (Eq. 1), the estimation error e_k be defined as $e_k = (x_k - \hat{x}_k)$. The Gaussian assumptions of noise and initial states ensure that e_k follows a normal distribution with 0 mean. We denote the steady state covariance matrix of this estimation error with Σ_e. Therefore, the system residue r_k, calculated in the Kalman estimator can be expressed as $r_k = Ce_k + v_k$ (see Eq. 1).

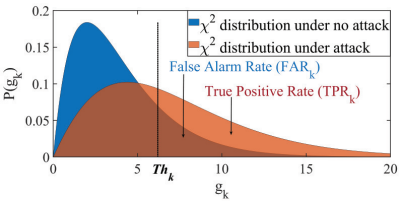


Fig. 3: χ^2-Distribution

Given that both estimation error and measurement noise are Gaussian distributed with zero mean and are independent of each other, the residue is normally distributed with 0 mean and covariance matrix given by $\Sigma_r = E[r_k r_k^T] - E[r_k]E[r_k]^T = E[(Ce_k)(Ce_k)^T] + E[v_k v_k^T] = C\Sigma_e C^T + \Sigma_v$. We use χ^2-test on r_k to find out how much the distribution of actual plant state x_k and its estimate $\hat{x}_k$ differ from each other. Let g_k denote the χ^2-test result at k-th sample and $g_k = \sum_{i=k-l_k+1}^{k} r_i^T \Sigma_r^{-1} r_i$. Here, l_k is the window size of χ^2-test while considering k-th sampling instance. In this case, the degree of freedom is ml_k, where m is the number of available sensors in the plant. In a normal scenario (no attack), g_k follows χ^2 distribution with mean ml_k (Fig. 3). Let Th_k be the threshold that is currently (at k-th sampling instance) being used by our variable threshold based detector unit. Then, g_k's probability density function (p.d.f) along with its cumulative distribution function w.r.t. Th_k can be defined as, $P(g_k) = \frac{g_k^{\frac{ml_k}{2}-1} e^{-\frac{g_k}{2}}}{2^{\frac{ml_k}{2}}\Gamma(\frac{ml_k}{2})}$; $P(g_k \leq Th_k) = \frac{\gamma(\frac{ml_k}{2}, \frac{Th_k}{2})}{\Gamma(\frac{ml_k}{2})}$. Here, Γ and γ are ordinary and lower incomplete gamma functions respectively. We say it is a *false alarm* when $g_k > Th_k$ even in the absence of an attacker. So, the *false alarm rate* (FAR) is the ratio of the number of times when the alarm has been raised falsely and the total number of alarms raised. We denote FAR_k as the FAR at k-th sample where the χ^2-test result g_k is compared with the threshold Th_k. In Fig. 3, the blue curve and the maroon curve represent the distribution of g_k under no attack and attack respectively. Therefore, FAR_k should be the fraction of area under the probability distribution curve of un-attacked g_k that is contained by the part beyond $g_k = Th_k$; thus computed as $FAR_k = 1 - P(g_k \leq Th_k)$.

Now, an FDI attack, where spurious data a_k^y and a_k^u are added to the sensor and the actuator data respectively, leads to non-centrality of the χ^2-test (the maroon curve in Fig. 3) result g_k^a obtained for the residue r_k^a (Eq. 2). The resulting g_k^a is compared to Th_k to flag an attack. The main motivation here is to understand that the system is under attack more often in the true positive case and reduce false alarms. Therefore, the pertinent problem becomes *how to achieve the above by suitable choice of threshold Th_k and the test parameter l_k*. To solve this, we leverage the non-centrality property of g_k^a which is proved in the following theorem.

***Theorem** 1:* Considering an FDI attack on an LTI system as specified in Eq. 2, the χ^2-test on residue r_k^a follows a non-central χ^2-distribution.

Proof: Under FDI attack, the estimation error at k-th sample is given by $e_k^a = x_k^a - \hat{x}_k^a$. Let $\triangle e_k = e_k^a - e_k$ be the difference between the estimation error under attack and no attack scenarios. Thus, $r_k^a = Ce_k^a + v_k + a_k^y = Ce_k + C\triangle e_k + v_k + a_k^y$. Considering that the mean of the estimation error is 0 and measurement noise is independent of the estimation error and sensor attack, the covariance Σ_{r^a} of the residue r_k^a generated due to an FDI attack can be computed as ,

$$\begin{aligned}
E[r_k^a r_k^{a^T}] &= E[Ce_k e_k^T C^T + Ce_k \triangle e_k^T C^T + C\triangle e_k e_k^T C^T \\
&\quad + C\triangle e_k \triangle e_k^T C^T + C\triangle e_k a_k^{y^T} + v_k v_k^T \\
&\quad + a_k^y \triangle e_k^T C^T + a_k^y a_k^{y^T}] \\
E[r_k^a]E[r_k^a]^T &= (CE[\triangle e_k] + E[a_k^y])(E[\triangle e_k]^T C^T + E[a_k^y]^T) \\
&= C\mu_{\triangle e}\mu_{\triangle e}^T C^T + C\mu_{\triangle e}\mu_{a^y}^T C^T + \mu_{a^y}\mu_{\triangle e}^T C^T + \\
&\quad \mu_{a^y}\mu_{a^y}^T C^T
\end{aligned}$$

Let the notations μ_i and Σ_i denote the mean and variance respectively of any variable i, and $\Sigma_{i,j}$ denotes the covariance of any i, j. Using the expressions of $E[r_k^a r_k^{a^T}]$ and $E[r_k^a]E[r_k^a]^T$ we get,

$$\begin{aligned}
\Sigma_{r^a} &= C\Sigma_e C^T + C\Sigma_{e,\triangle e^T} C^T + C\Sigma_{e,a^{y^T}} + C\Sigma_{\triangle e,e^T} \\
&\quad + C\Sigma_{\triangle e} C^T + C\Sigma_{\triangle e,a^y} + \Sigma_{a^y,e^T} + \Sigma_{a^y,\triangle e} C^T \\
&\quad + \Sigma_v + \Sigma_{a^y} = \Sigma_r + P
\end{aligned} \tag{3}$$

Here, $P = C\Sigma_{e,\triangle e^T} C^T + C\Sigma_{e,a^{y^T}} + C\Sigma_{\triangle e,e^T} + C\Sigma_{\triangle e} C^T + C\Sigma_{\triangle e,a^y} + \Sigma_{a^y,e^T} + \Sigma_{a^y,\triangle e} C^T + \Sigma_{a^y}$, $\Sigma_r = C\Sigma_e C^T + \Sigma_v$. Since, by definition, covariance is positive semi-definite and variance is positive, both Σ_r, P and consequently Σ_{r^a} are positive definite. Considering an FDI attack on the both control signal and sensor output, we apply χ^2-test on r_k^a which gives $g_k^a = \sum_{i=k-l_k+1}^{k} r_i^{a^T} \Sigma_r^{-1} r_i^a$. Therefore, the mean μ_k of the χ^2 statistics g_k^a of r_k^a at k-th sampling instance over an observation window of length l_k can be computed as,

$$\begin{aligned}
\mu_k &= E[g_k^a] = E[\sum_{i=k-l_k+1}^{k} r_i^{a^T} \Sigma_r^{-1} r_i^a] \\
&= \sum_{i=k-l_k+1}^{k} trace[\Sigma_{r^a} \times \Sigma_r^{-1}] \\
&= \sum_{i=k-l_k+1}^{k} trace(\Sigma_r \Sigma_r^{-1}) + \sum_{i=k-l_k+1}^{k} trace(P\Sigma_r^{-1}) \\
&= ml_k + \sum_{i=k-l_k+1}^{k} trace(P\Sigma_r^{-1}) > ml_k
\end{aligned} \tag{4}$$

The last inequality follows from the fact that P is positive definite. Hence, the mean of g_k^a is strictly greater than the mean ml_k of g_k. This makes the distribution of g_k^a a non-central one with non-centrality parameter at k-th sample be $\lambda_k = \sum_{i=k-l_k+1}^{k} trace(P\Sigma_r^{-1})$. □

Having proved the non-central property of the χ^2 distribution of g_k^a, we now leverage this to establish that the attack detectability can be improved with respect to an optimal threshold of the detector.

***Corollary** 1:* Consider that the true positive rate and false alarm rate at k-th sampling instances are TPR_k and FAR_k re-

spectively. Under attack scenario, its possible to have $TPR_k > FAR_k$. □

Proof: As shown in Theorem 1, under attack scenario, we have a non-central χ^2 distribution of g_k^a with mean μ_k and non-centrality parameter λ_k. Following [20], the p.d.f. of g_k^a will be, $P(g_k^a) = \frac{1}{2}e^{-\frac{(g_k^a+\lambda_k)}{2}}(\frac{g_k^a}{\lambda_k})^{\mu_k/4-1/2}I_{\mu_k/2-1}(\sqrt{\lambda_k g_k^a})$ with I denoting Bessel function. With respect to Th_k, we say an FDI attack is detected if $g_k^a > Th_k$. Under attack, this is a true positive case. We compute TPR_k as $TPR_k = 1 - P(g_k^a \leq Th_k)$ where $P(g_k^a \leq Th_k) = 1 - Q_{\mu_k/2}(\sqrt{\lambda_k}, Th_k)$. Essentially, this is the fraction of area under the distribution curve (Fig. 3) of g_k^a beyond $g_k^a = Th_k$. Here, Q is Marcum Q-function [20]. In Theorem 1, we have proved that $\mu_k > ml_k$ where ml_k is the mean of g_k. This causes the non-central χ^2 distribution of g_k^a to be more shifted towards the right than the χ^2 distribution of g_k.

Moreover, the variance of g_k is $\sigma_k = 2ml_k$ and variance of g_k^a is $\sigma_k^a = 2(ml_k + 2\lambda_k)$, where $\lambda_k > 0$. Clearly, $\sigma_k^a > \sigma_k$. Therefore, the expected deviation of g_k^a from μ_k is more than the expected deviation of g_k from ml_k which makes the distribution of P(g_k^a) wider and thereby flatter (since the area under both curves is unity). Hence, P($g_k^a > Th_k$) > P($g_k > Th_k$) as shown in Fig. 3. So, the non-central χ^2 distribution improves TPR_k i.e. attack detectability thus leading to $TPR_k > FAR_k$ for a properly chosen threshold parameter. □

With this intuition of having $TPR_k > FAR_k$ when the g_k^a distribution is non-central, we can optimally choose *tunable* parameters like l_k, Th_k to attain maximum possible TPR_k during attack and minimum possible FAR_k in absence of attack (i.e. only noise is present). The problem of synthesizing an optimal detector at every k-th simulation step can thus be formulated as the following optimization problem.

$$J_t = \max_{l_k, Th_k} \; w_1 \times TPR_k - w_2 \times FAR_k \text{ s.t. } FAR_k < \epsilon,\; l_k < l_{max} \tag{5}$$

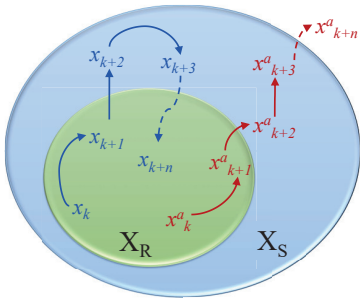


Fig. 4: Operating regions of the system: Safe region X_S, Preferable Operating Region X_R

The above cost function aims minimization of FAR_k and maximization of TPR_k at every simulation step. Here, w_1, w_2 are respective non-negative weights assigned to TPR and FAR depending on attacked (TPR increment gets more importance) and non-attacked (FAR reduction gets more importance) situations. ϵ is the maximum allowable FAR and l_{max} is maximum allowed χ^2 window length. At each k-th step, given the current measurement y_k^a, the solution of the above optimization problem is a pair $< l_k^*, Th_k^* >$, where l_k^* and Th_k^* are the optimal χ^2 window length and threshold respectively that lead to maximum TPR_k and minimum FAR_k w.r.t. current measurement of the system states.

B. Operating Regions of A CPS In Context of Secure CPS

Considering the discrete LTI system specified in Eq. 1, we classify the operating region of the system, as demonstrated in Fig. 4, in two primary sub-regions: i) *safe region* X_S, and ii) *preferable operating region* X_R where $X_R \subset X_S$. The system becomes unsafe when it goes beyond the outer region X_S. The inner region X_R defines the set of possible states in which system operation is preferred due to safety considerations. By preferable operating region, we mean that starting from anywhere within $\in X_R$, the controller ensures that the system will always remain within the safety region X_S in the absence of any FDI attack. With respect to the notion of safety invariance, we formally define such preferable operating regions as follows.

***Definition** 1:* Considering the discrete LTI system specified in Eq. 1 and controller with gain K, the **preferable operating region** of the system is defined as:

$$X_R = \{x \mid \forall\, w \sim \mathcal{N}(\Sigma_w, 0),\; v \sim \mathcal{N}(\Sigma_v, 0),\; and\; x \in X_R \rightarrow \forall n \in \mathbb{N},\; f^n(x, K) \in X_S\}$$

where, $X_R \subset X_S$ and the function f represents the dynamical relation of the LTI system (Eq. 1) in no attack scenario. □

However, an intelligent FDI attack can take the system beyond X_S even if the system initial states remain within X_R (see Fig. 4). Therefore, the job of the adaptive detector would be detecting such intelligent attack effort as soon as possible before the system becomes unsafe. And, if any attack gets detected, a suitable fast controller must bring back the system to X_R in minimum time. Note that, the choice of X_S can be exercised depending upon system description and safety criteria. The invariant set based preferable operating region X_R is chosen in practice as some *i-step invariant set* within X_S for which the controller guarantees satisfactory performance as well as safety.

Synthesis of Preferable Operating Region: We consider K as a Linear-Quadratic-Gaussian (LQG) controller [21] (though in general, any optimal control framework is applicable). Since, the state progression of a CPS depends on both actual state x and estimated state $\hat{x}$ (Eq. 1), the size of the preferable region X_R is influenced by both x and $\hat{x}$. Hence, we introduce an *augmented system* $X = [x \;\; \hat{x}]^T$ with discrete state progression defined as,

$$\begin{aligned} X_{k+1} &= \mathcal{A}X_k + \mathcal{B}U_k + w_k; \\ Y_k &= \mathcal{C}X_k + v_k;\; U_k = -\mathcal{K}X_k; \end{aligned} \tag{6}$$

for which the resulting matrices $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{K}$ can be designated as $\mathcal{A} = \begin{bmatrix} A & 0 \\ LC & A - LC \end{bmatrix}$, $\mathcal{B} = \begin{bmatrix} B \\ B; \end{bmatrix}$, $\mathcal{C} = \begin{bmatrix} C & 0 \end{bmatrix}$, and $\mathcal{K} = \begin{bmatrix} 0 & K \end{bmatrix}$. The LQG controller gain K is designed to stabilize the system states with minimized control cost. The presence of noise and external disturbances can potentially cause the state trajectory to pass through unsafe regions under over/under-shoots, which is unacceptable. So a preferable operating region X_R needs to be defined in such a way that

when the system states initialize inside X_R, under normal circumstances (i.e. in presence of noise, disturbances but no intentional FDI attack) the controlled state trajectory will always remain within the safety region X_S, irrespective of any overshoot or undershoot conditions. Eventually, due to the inherent stability characteristics of the controller gain, the controlled states will approach the desired reference point in state space through this preferable operating region. X_R is formally defined in Def. 1 and we pose the problem of synthesizing such a preferable operating region in terms of a robust optimization problem [22]. Given the optimal controller gain K and the augmented system representation, the optimization problem formulation is as follows.

$$\max_{X_c} \ d \tag{7}$$

$$\text{s.t.} \quad X_0 = zd + X_c, \quad ||z||_1 \leq 1 \tag{8}$$

$$U_k = -\mathcal{K}X_k, \quad |U_k| \leq \epsilon_u \ \forall k \in [0, N] \tag{9}$$

$$Y_k = \mathcal{C}X_k + DU_k + v_k, \ |Y_k| \leq \epsilon_y \ \forall k \in [0, N] \tag{10}$$

$$X_{k+1} = \mathcal{A}X_k + \mathcal{B}U_k + w_k, \ X_k \in X_S \ \forall k \in [0, N+1] \tag{11}$$

Here, the symbols ϵ_y and ϵ_u denote the allowable sensor range and actuation saturation limit respectively. Let us define a bounded region with center at $c \in \mathbb{R}^n$ and radius r as $\mathcal{P}_r(c) \triangleq \{x | x \in \mathbb{R}^n, ||x - c||_\infty \leq r\}$. As the solution of the above optimization problem, we get a point $X_c \in \mathbb{R}^n$ and a scalar d such that the bounded region $\mathcal{P}_d(X_c)$ is of maximum size. This resultant bounded region $\mathcal{P}_d(X_c)$ is our preferable operating region X_R. The constraints $(9) - (11)$ ensure legal state transition operations and the property of X_R as defined in Def. 1. Following the specification of X_S and synthesis of X_R, we next demonstrate a methodology for intelligent attack generation.

C. Intelligent Attack Generation

In the CPS context, the attacker's motive is to steer the system beyond the safe set X_S while trying to remain stealthy by reducing the TPR. Given the sensor measurement y^a_{k-1}, we present this attack estimation problem as the following optimization problem.

$$J_a = \max_{a^y_k, a^u_k} \quad -w_1 \times TPR_k + w_2 \times FAR_k + \tag{12}$$

$$\sum_{i=0}^{\infty} (|x^a_{i+1}| - |X_S|)^T W_3 (|x^a_{i+1}| - |X_S|) \tag{13}$$

$$\text{s.t.} \quad x^a_0, \ \hat{x}^a_0 \in X_R \tag{14}$$

$$u^a_k = -K\hat{x}^a_k, \ \tilde{u}^a_k = u^a_k + a^u_k, \ |u^a_k|, \ |\tilde{u}^a_k| \leq \epsilon_u \ \forall k \in [0, \ \infty] \tag{15}$$

$$y^a_k = Cx^a_k + D\tilde{u}^a_k + v_k + a^y_k, \tag{16}$$

$$|y^a_k| \leq \epsilon_y \ \forall k \in [0, \ \infty] \tag{17}$$

$$r^a_k = Cx^a_k - C\hat{x}^a_k, \ g^a_k \leq Th_k \ \forall k \in [0, \ \infty] \tag{18}$$

$$\hat{x}^a_{k+1} = A\hat{x}^a_k + Bu^a_k + L(Cx^a_k - C\hat{x}^a_k), \ \forall k \in [0, \ \infty] \tag{19}$$

$$x^a_{k+1} = Ax^a_k + B\tilde{u}^a_k, \ \forall k \in [0, \ \infty] \tag{20}$$

Here, w_1 and w_2 are weights that denote the relative priorities of the attack initiative similar to the optimal threshold cost function J_t (Eq. 5). Since the attack generation method will be used for experience learning of the thresholds tuning agent (discussed with further detail in Sec. III-E), it is imperative that J_a assumes knowledge about J_t and tries to negate its cost objective. While an attacker wants to decrease TPR_k and increase FAR_k simultaneously, detector's objective is to increase TPR_k and decrease FAR_k based on value of λ_k (Eq. 5). The last component of J_a accounts for deviation of the current system state from the safety boundary X_S using a quadratic weighted distance metric where W_3 is a diagonal matrix consisting of relative weights corresponding to criticality of each dimension. The constraints in (15) and (17) ensure that the adaptive threshold generation process is trained rigorously against optimal yet practical attacks. The forged control signal and sensor data must be within their respective bounds in a real attack scenario. Otherwise, the invalid sensor data and control signal will be skipped and won't be sent to controller and plant respectively, thus reducing the attack's effect. Since, an intelligent adversary would want to remain stealthy, it's intention would be bypassing the detector. This is taken into account using constraint (18) while estimating an optimal attack. The constraints (19) and (20) ensure system progression following Eq. 2.

D. Synthesis of Fast Controller

Once our proposed adaptive detector detects an intelligent attack effort while the system is still within $X_S \setminus X_R$, we assume that the system is switched to a secure mode until it is brought back to X_R. Since, the standard cryptographic security enforcement guarantees instant attack detection and an intelligent attacker would prefer to remain stealthy, it is presumed that during this secure mode no attack takes place. Also, we assume that due to real time scheduling constraints only a subset of control loops can operate through secure channel at a time. Therefore, we aim to recover the system state in minimum possible time. In this regard, we formally define the problem of synthesizing fast control inputs in terms of the augmented system presented in Eq. 6 as follows.

$$\exists \ U_0, U_1, \cdots, U_{t-1} \quad \forall X_0 \in X_S \setminus X_R, \ X_t \in X_R \tag{21}$$

$$\text{s.t.} \quad X_{i+1} = \mathcal{A}X_i + \mathcal{B}U_i \ \forall i \in [0, \ t-1] \tag{22}$$

$$X_i \notin X_R \ \forall i \in [0, t-1] \tag{23}$$

$$X_i \in X_S \ \forall i \in [0, t] \tag{24}$$

The constraint (23) implies that minimum t control loops are required to bring the system back to X_R from $X_S \setminus X_R$. Subsequently, as the solution of the problem, we get t control inputs that meet our objective. The major challenge behind synthesizing such control inputs is that the synthesized control inputs must fulfil the objective *for all* possible initial states (21). Since standard controller design frameworks (e.g. the performance focused LQR controller in our case) ensure performance in terms of stability, settling time, convergence,

cost, etc., but they may not ensure that while operating in secure mode, the system will never go beyond X_S due to overshoots before it is brought back to X_R. This may happen because of the uncertainties that have already been incurred due to an FDI attack before it is detected and secure mode is on. For guaranteeing that the system will never exceed X_S while coming back to X_R from $X_S \setminus X_R$, we map the control input synthesis problem to a Satisfiability Modulo Theory (SMT) instance, in lines of [23]. Instead of synthesizing control input for all the points in initial state space (which is $X_S \setminus X_R$ in our case), the authors in [23] divide the initial region into several non-overlapping sub-regions. From the center of each such sub-region, a sequence of open-loop control inputs are synthesized using SMT solvers to reach the target region X_R. The key idea in this kind of controller synthesis [23] methodology is to ensure that the control inputs derived for the center of a sub-region can be used for all the points in that sub-region (note that the region partitioning is also part of the synthesis process). This converts the $\forall\exists$ constraints to quantifier-free fragment of linear arithmetic, thus addressing the scalability issue. In similar lines, our methodology combines the LQR controller and the open-loop control inputs to ensure that irrespective of the initial point within a sub-region, the system can safely reach the target region in minimum number of control steps. Note that these fast recovery controller synthesis is accomplished in off-line mode, and thus, has no overhead on real time performance of the system.

In context of our augmented system description (with both controller and estimator in place) in Eq. 6, we define the state progression following [23] as given below.

$$\begin{aligned} U_k &= -\mathcal{K}(X_k - X_k^{ref}) + U_k^{ref};\ Y_k = \mathcal{C}X_k + DU_k + v_k \\ X_{k+1}^{ref} &= \mathcal{A}X_k^{ref} + \mathcal{B}U_k^{ref};\ X_{k+1} = \mathcal{A}X_k + \mathcal{B}U_k + w_k \end{aligned} \tag{25}$$

Consider X_0^{ref} be the center point of one of the sub-regions. The proposed method symbolically generates the open-loop control inputs U_k^{ref}s that drive the system to X_{k+1}^{ref}s. The optimal controller $\mathcal{K}$ and the open-loop control inputs U_k^{ref}s are generated independently. It has been shown in [23] that irrespective of X_0^{ref} and U_k^{ref}s, if a system is initiated from X_0 (a point in the initial sub-region such that $X_0 \neq X_0^{ref}$), it will reach a bounded region centered around X_t^{ref} at t-th iteration. Let us define the initial sub-region with radius r and centered at c as $\mathbb{B}_r(c) = \{X \mid \ ||X - c||_2 \leq r\}$. The open-loop control inputs U_k^{ref}s can be synthesized as the satisfying solution of the formula $\mathbb{A}\ \wedge \mathbb{A}_{goal}$ where,

$$\begin{aligned} \mathbb{A}\ \triangleq \exists\ U_0^{ref},\ U_1^{ref},\ \cdots\ U_{t-1}^{ref},\ X_0^{ref},\ X_1^{ref},\ \cdots\ X_t^{ref}, \\ \text{s.t. } \mathbb{A}_{control} \wedge \mathbb{A}_{execution} \wedge \mathbb{A}_{safe} \end{aligned} \tag{26}$$

$$\mathbb{A}_{control} \triangleq \bigwedge_{k=0}^{t-1} |U_k^{ref} \oplus (\mathcal{K} \otimes \mathcal{R}_{r_k v}(X_k^{ref}))| \ \leq \epsilon_u \tag{27}$$

$$\mathbb{A}_{execution}\ \triangleq (X_0^{ref} = X_0)\ \wedge \bigwedge_{k=0}^{t-1}\ X_{k+1}^{ref} = \mathcal{A}X_k^{ref} + \mathcal{B}U_k^{ref} \tag{28}$$

$$\mathbb{A}_{safe}\ \triangleq \bigwedge_{k=0}^{t} \mathcal{R}_{r_k v}(X_k^{ref}) \in X_S; \tag{29}$$

$$\mathbb{A}_{goal}\ \triangleq\ \mathcal{R}_{r_t v}(X_t^{ref}) \in X_R \tag{30}$$

Here, a hyper-rectangle centered at $c \in \mathbb{R}^n$ is represented as $\mathcal{R}_v(c) = \{x \mid \wedge_{i=1}^n\ c(i) - v(i) \leq x(i) \leq c(i) + v(i)\}$ where $v \in \mathbb{R}^n$. The notations $\oplus$ and $\otimes$ are defined as $a \oplus S \triangleq \{a + x | x \in S\}$ and $M \otimes S = \{Mx | x \in S\}$ where $a \in \mathbb{R}^n$, $S \subseteq \mathbb{R}^n$ and $M \in \mathbb{R}^{n \times n}$. The hyper-rectangles in $\mathbb{A}$ are computed using a sequence of optimization problems (refer to Sec. 3.3 and 3.4 of [23]). The constraint $\mathbb{A}_{control}$ in (27) imposes that the synthesized control inputs must be within the actuation limit. The control synthesis process must ensure that the reference trajectory of the system follows Eq. 25. This is ensured by the constraint $\mathbb{A}_{execution}$ in (28). Note that, in (28), the initial value X_0^{ref} is set to X_0 which is center of one initial sub-region for which we are computing the open-loop control inputs. The safety property of the system is presented in the constraints $\mathbb{A}_{safe}$ (29). And finally, the formula $\mathbb{A}_{goal}$ in (30) denotes that the final state X_t^{ref} and the hyper-rectangle centered at X_t^{ref} must be within the target region X_R. A satisfying solution of $\mathbb{A} \wedge \mathbb{A}_{goal}$ gives a sequence of control inputs U_0^{ref}, U_1^{ref}, $\cdots$, U_{t-1}^{ref} that can take every initial state from $\mathbb{B}_r(X_0^{ref})$ to X_R within t iterations without exceeding the safety region X_S.

In [23], the authors have considered a fixed number of iterations t (this is given as input to their proposed framework) and synthesized U_0^{ref}, U_1^{ref}, $\cdots$, U_{t-1}^{ref}. However, our motivation is to bring back the system within X_R in minimum time, and therefore, to minimize t. Moreover, considering the actuation limit, there may not exist a valid sequence of a fixed number of control inputs that safely takes the system from $X_S \setminus X_R$ to X_R. In light of this, we present the control input synthesis process in the context of our objective using the method proposed by [23] in Algo. 1. It takes as input the augmented system matrices, optimal controller gain, safety and preferable operating regions. It symbolically generates sub-regions $cover$ in the input space $X_S \setminus X_R$ and the corresponding sequence of control inputs $controllers$ that drives the system from the sub-regions to X_R.

The radius r of the initial sub-region is initialized as the diameter of the uncovered input space (line 7). At every iteration, the algorithm synthesizes the control inputs for the initial sub-region centered at $X_0 = center(X_{S \setminus R} \setminus cover)$ (line 7). This is required for generating the constraint in (28). In line 9, r is reduced to half (just a design choice). The safety constraints $\mathbb{A}$ are generated up to t iterations in line 10. Here, the GETCONSTRAINTS() function encodes progression of an LTI system with an estimator unit (similar to [9]). It is used to generate the constraints in equations (26)-(30) in SMT format. The SMT solver tries to find a satisfying solution of $\mathbb{A}$. If not, it keeps on reducing the radius r until $\mathbb{A}$ is satisfied

Algorithm 1 Fast control input synthesis

Require: Augmented system matrices $\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, D, \mathcal{K} \rangle$, safety region X_S, preferable operating region X_R
Ensure: Fast control inputs *controllers* for reach-avoid specifications and the corresponding initial sub-regions *cover*

1: **function** GETFASTCONTROLINPUT($\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, D, \mathcal{K} \rangle$,$X_S$,$X_R$)
2: $cover \leftarrow null$; $controllers \leftarrow null$; ▷ Initialization
3: $X_{S \setminus R} \leftarrow X_S \setminus X_R$; ▷ Initialization
4: **repeat**
5: $t = 0$;
6: **repeat**
7: $t \leftarrow t + 1$; $r \leftarrow diameter(X_{S \setminus R} \setminus cover)$; $X_0 \leftarrow center(X_{S \setminus R} \setminus cover)$;
8: **repeat**
9: $r \leftarrow r/2$;
10: $\mathbb{A} \leftarrow$GETCONSTRAINTS($\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, D, \mathcal{K} \rangle$, X_S, X_R, r, t, $cover$, X_0);
11: **until** $\mathbb{A}$ == SAT
12: $\mathbb{A}_{goal} \leftarrow$GETCONSTRAINTS($\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, D, \mathcal{K} \rangle$, X_S, X_R, r, t, $cover$, X_0);
13: **until** $\mathbb{A}_{goal}$ == SAT
14: r, X_0^{ref}, U_0^{ref}, U_1^{ref}, $\cdots$, $U_t^{ref} \leftarrow$MODEL($\mathbb{A}$, $\mathbb{A}_{goal}$);
15: $cover \leftarrow cover \cup \mathbb{B}(X_0^{ref})$; $controllers \leftarrow controllers \cup \{(\langle X_0^{ref}, U_0^{ref}, U_1^{ref}, \cdots, U_t^{ref} \rangle, \mathbb{B}(X_0^{ref}))\}$;
16: **until** $X_{S \setminus R} \nsubseteq cover$
17: **return** $controllers$, $cover$

(lines 8-11). Once we find a suitable radius r of the initial sub-region for which it is ensured that every state within that sub-region will reach a hyper-rectangle centered at X_t^{ref} avoiding any unsafe state, we check if the goal constraints $\mathbb{A}_{goal}$ are satisfied in lines 12-13. If not, t is incremented and lines 6-13 are executed until we find a minimum number of iterations that the synthesized control inputs will take to bring the system to X_R through safe states only. From the satisfying instances, we retrieve the radius r of the new sub-region, its center X_0^{ref}, and the associated control inputs U_0^{ref}, U_1^{ref}, $\cdots$, U_t^{ref} in line 14. The *cover* set is updated with the new sub-region and the *controllers* set is updated with the corresponding control inputs (line 15). The entire process is repeated until the entire input space $X_S \setminus X_R$ is covered (lines 4-16). The final set of control inputs is returned in line 17.

E. The Reinforcement Learning-Based Framework

In this section, we describe the multi-agent RL-based framework that is at the core of our methodology to build an adaptive threshold-based detection module to detect a stealthy FDI attack before it is successful in making the system unsafe. In Sec. III-A, we have established how changing the detection thresholds leveraging the non-centrality of the system residue can be useful to increase TPR and reduce FAR. We utilize that notion here. The detection and attacker modules implicitly learn how the system model behaves normally and under intelligent FDI attacks by analysing its outputs, states, residue etc. The smart attacker module would challenge the adaptive detection module by posing most stealthy yet effective FDI attacks depending on current system behaviour. On the other hand, the intelligent detector module learns the best possible FDI attack on the CPS by observing the non-centrality of the χ^2-distribution of system residue and adaptively change the threshold to expose the attacker with a promise of increased TPR and reduced FAR. Scarcity of system specific labeled falsified data makes Reinforcement Learning (RL) an obvious choice. The ability of RL algorithms to automatically update strategy by learning from the prior experience is useful in CPS context, something which we leverage in this work.

A plant-controller closed loop system equipped with a χ^2-based detector (as shown in Fig. 2) is the system under test here and false data effective on such a system is highly model-specific. Therefore, we design our environment by modeling such a closed loop system so that we can train our RL modules with simulated system data (both under attack and without attack situations). We build individual RL agents as part of our methodology to act as the FDI attacker and the adaptive threshold-based detector that run simultaneously in closed loop with the system environment (see Fig. 5). These agents (Λ) interact with the environment by observing certain states from the environment (obs), and learn how intelligent choice of action (act) values can influence the environment towards fulfilment of their objectives i.e. earning higher rewards (Rwd).

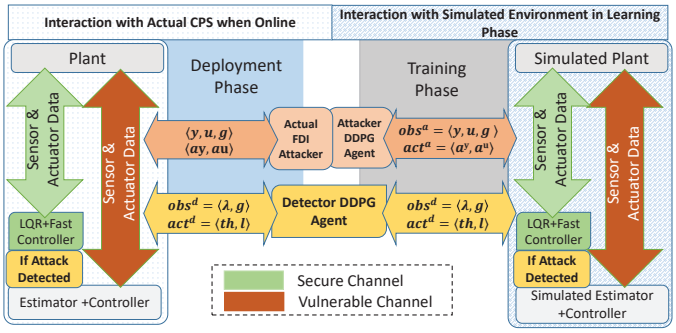


Fig. 5: The RL-based Methodology

RL Agent For Attacker: Following our intelligent attacker modeling in Sec. III-C, we assume the attacker has the information about the system characteristics (Eq. 1) and it can manipulate the sensor and actuator data communicated between the plant and controller side. We design an *Attacker RL Agent* Λ^a to intelligently inject false data into the system by observing the sensor data, actuator data and the χ^2-test result on the system residue. The false data injections are bounded by the sensor and actuator saturation limits (refer Eq. 13). We define the *actions* of Λ^a as $act^a = [a^y, a^u]$ and the *observations* as $obs^a = [y, u, g]$ (refer Fig. 5). Here y, u, a^y, a^u denote sensor and actuator data and false data injected in sensor and actuator respectively (refer Eq. 2). Here, g corresponds to the χ^2-test result on the system residue r as mentioned in Sec. III-A. Along with these system data, we also provide the last set of actions chosen by the agent Λ^a as an observation, i.e. $obs_k^a = [y_k, u_k, g_k, act_{k-1}^a]$ at k-th simulation instance. Considering these observations at every simulation instance, the attacker agent aims to choose proper act_k^a in order to make the system state unsafe without being detected. This

measurement of stealth and success of the chosen attack effort is captured in the *reward* function $Rwd^a_k(obs^a_k, act^a_k, obs^a_{k+1})$. The agent is *rewarded* against its choice of act^a_k at every k-th simulation instance following this reward function. The reward function for Λ^a is built following J_a (Eq. 13) i.e. $Rwd^a_k = -w_1 \times TPR_k + w_2 \times FAR_k + (|x^a_{k+1}| - |X_S|)^T W_3 (|x^a_{k+1}| - |X_S|)$ (the notations follow the same meaning as in Eq. 13 and are sub-scripted with k to denote their value at k-th simulation instance). As described earlier, the different components of $Rwd^a_k(obs^a_k, act^a_k, obs^a_{k+1})$ are contradictory and accounts for stealthiness ($-w_1 \times TPR_k + w_2 \times FAR_k$) and success ($(|x^a_{k+1}| - |X_S|)^T W_3 (|x^a_{k+1}| - |X_S|)$) of a chosen action act^a_k. Λ^a tries to optimize Rwd^a_k at every simulation instance by choosing an optimal action act^a_k for which the stealthiness and success of the false data injections are balanced.

RL Agent For Detector: Similar to the attacker agent, for the *Threshold-based Detector Agent* Λ^d, the objective is to choose an optimal attack detection threshold Th_k and a suitable χ^2-window l_k at k-th iteration by observing the χ^2 statistics g of the system residue, non-centrality λ of this χ^2 distribution and the action act^d_{k-1} chosen by the agent in the last iteration. The motivation behind choosing non-centrality of the χ^2 statistics of the system residue has already been discussed in Sec III-A with rigorous mathematical proof. The detector also needs to observe χ^2 statistics of the system residue since its objective is to track this value by choosing proper χ^2-window and threshold with an aim to maximize TPR and minimize FAR. Therefore, the action vector is chosen as $act^d_k = [Th_k, l_k]$ and observation vector is chosen as $obs^d_k = [g_k, \lambda_k, act^d_{k-1}]$ (refer Fig. 5). The reward function $Rwd^d_k(obs^d_k, act^d_k, obs^d_{k+1})$ is designed following J_t from Eq. 5, i.e. $Rwd^d_k(obs^d_k, act^d_k, obs^d_{k+1}) = TPR_k$ when $\lambda_k > \delta$ and $Rwd^d_k(obs^d_k, act^d_k, obs^d_{k+1}) = -FAR_k$ when $\lambda_k \leq \delta$ respectively (here the variables carry the similar meaning as Eq. 5 and are sub-scripted with k to denote their value at k-th iteration). Λ^d intelligently chooses act^d_k at every k-th simulation instance by optimizing the objective function Rwd^d_k so that the maximum TPR and minimum FAR can be approached.

Learning Technique: We employ Deep Deterministic Policy Gradient (DDPG) algorithm [24] to output deterministic actions over the continuous action space. Each DDPG agent consists of an actor neural network to deterministically choose an action (act) by observing the states (obs) from the environment and a Deep Q-Network to criticize the Q value for that action taking the observed states (obs) and actions (act) into consideration. Given a secure CPS model, we first train these model-free DDPG agents so that they can reprise their designated roles in the environment. The *learning process* for each of the DDPG agents (Λ^a, Λ^d) involves model specific customizations over the standard DDPG algorithm as in Algo. 1 in [24]. In a multi-agent RL environment both the agents are trained, simulated and updated by interacting with the environment parallelly. In each simulation instance/iteration of an episode, the actor chooses an action (act_k) by exploring the action space using a random process. From several iterations, transitions of observable states (from obs_k to obs_{k+1} due to the taken action act_k) are stored in the experience replay buffer along with the corresponding reward (Rwd_k) achieved during the transition. The DQN critic network picks a random batch from this replay buffer, calculates corresponding Q values in every iteration and updates itself by the mean square loss between the calculated Q values from consecutive iterations. On the other hand, the actor network policies are updated using the policy gradient over the expected Q value return. The training algorithm learns the highest expected return from the experiences, and updates the RL policy to output the optimal action for which it can earn this expected maximum return.

F. Overall Framework of The Proposed Learning Enable Secure CPS Model

Algo. 2 represents the overall methodology. It first solves the robust optimization problem in Eq. 7 (line 2) to obtain the preferable operating region X_R for the system under normal conditions, and initializes the system (the CPS model) by randomly choosing a state from X_R (line 3). Then, it trains the multi-agent RL framework following aforementioned steps (line 4) with an offline simulation of the target CPS model. Fast control inputs are formally synthesized for the CPS to bring its states back into the preferable operating region under no-attack situation using the call (line 7) to Algo. 1. Our methodology finally outputs a trained adaptive threshold-based detector agent Λ^d and the set of fast control inputs $controllers$ to act as an optimized defense mechanism that promises maximized TPR and quick recovery against FDI attacks with a minimized FAR. Fig. 5 provides a brief representation of the described framework.

Algorithm 2 RL Based Framework for Adaptive FDI Attack Monitoring with Fast Mitigation

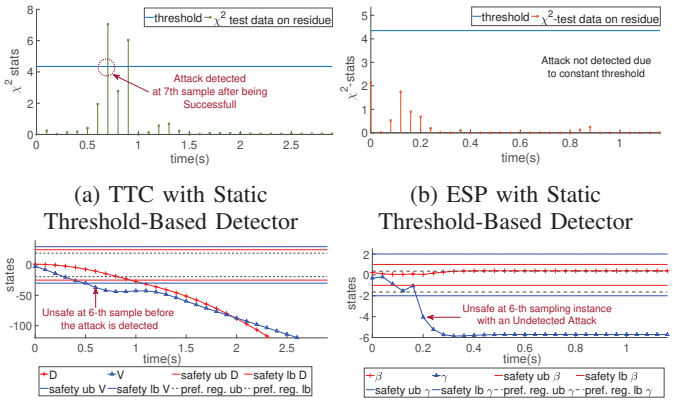
1: $system \leftarrow A, B, C, D, K, L$; $i \leftarrow$ settling time in number of samples ▷ Specifying discrete system model
2: $X_R \leftarrow$ GETPERFORMANCEREGION(system, safety region X_S,i) ▷ Compute the preferable operating region solving Eq. 7
3: $obs_{ini} \leftarrow rand(obs \in X_R)$ ▷ Initialize the environment/system with a random state from X_R
4: TRAINAGENTS(system, $[\Lambda^a, \Lambda^d]$,$training\ specs$) ▷ Competitive and Collaborative offline training of the multi-agent setup
5: $\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{K} \rangle \leftarrow$ GETAUGMENTEDSTATESPACE($system$) ▷ Derive augmented system matrices following Eq. 6
6: $controllers \leftarrow$ GETFASTCONTROLINPUT($\langle \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{K} \rangle, X_S, X_R$) ▷ Call to Algo. 1 to synthesis fast control inputs
7: Equip the system with trained Λ^d and $controllers$ and put the system online

IV. EXPERIMENTAL RESULTS

Automotive systems are collection of safety-critical CPSs. Heterogeneous communication protocols connect Electronic Control Units (ECUs) that implement safety-critical CPSs as real time control tasks. Vulnerability in any of those protocols (eg. Controller Area Network) [25] can grant an easy access to the attacker to manipulate majority of the

system communications. We consider two such CPSs, namely Trajectory Tracking Controller (TTC) and Electronic Stability Program or Control (ESP) as case studies in this work. As baseline, we consider these systems to be equipped with static threshold-based detectors and only the LQR controller used during attack mitigation.

Experimental Setup: As presented in the Algo. 2 we start by deriving the preferable operating region X_R for a system under normal circumstances (i.e. in presence of noise). We model the robust optimization problem formulated in Eq. 7 to derive the preferable operating region of a system using the popular optimization problem modeling tool YALMIP [26] along with Gurobi [27] solver which are available as MATLAB add ons. To synthesize the fast control inputs for all initial state space, we adopt the tool Realsyn proposed in [23] as demonstrated in Algo. 1. For solving the SMT constraints in Algo. 1, we have utilized z3py [28], a Python library for the SMT solver Z3 [29]. Our RL based multi-agent framework, is built using the MATLAB Reinforcement Learning Toolbox. We build the system model using Simulink that acts as the environment in our multi-agent RL framework. The training and problem solving are done in a 3.2 GHz Intel core i7-8700 system.



(a) TTC with Static Threshold-Based Detector

(b) ESP with Static Threshold-Based Detector

(c) TTC under Successful, stealthy Attack

(d) ESP under Successful, stealthy Attack

Fig. 6: TTC and ESP with Static Threshold-Based Detection. *(pref. reg. : preferable region; ub : upper bound; lb : lower bound)*

Case Study 1: Trajectory Tracking Control System: Trajectory Tracking Control (TTC) System regulates the deviation of a vehicle from a given trajectory (D) and a reference velocity (V) by applying proper acceleration [30]. The system matrices, safety region X_S, preferable operating region X_R of TTC are specified in Tab. I. The only measurable state of TTC is distance D. The acceleration acts as control input to TTC system. The range of distance sensor and the saturation limit of control signal i.e. acceleration are 30 m and 72 m/s^2 respectively. To simulate the behavior of TTC under both static threshold based detector and our proposed intelligent detection system, we consider an optimal FDI attack which is generated using the well-trained RL agent for the attacker. First, consider the scenario where the χ^2 detector uses a static threshold

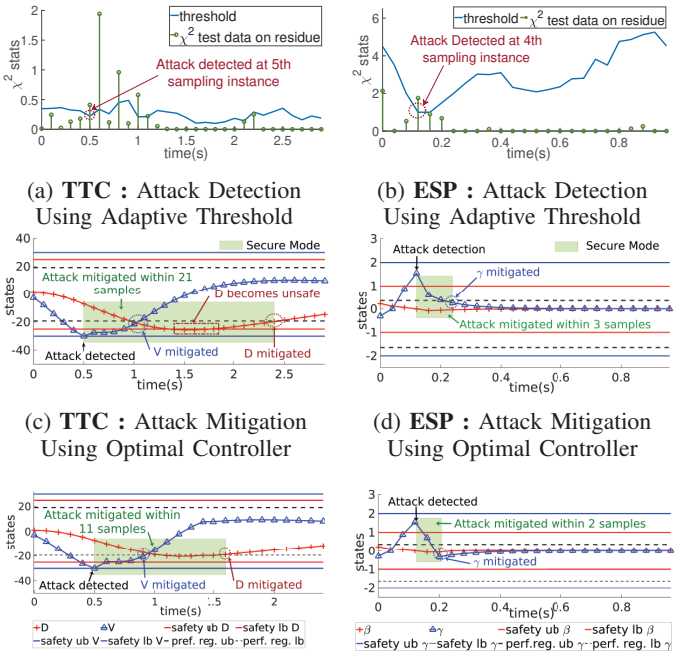
TABLE I: System Specification

System	Specification	X_S, X_R
TTC	x = [D; V]; A = [1.0000 0.1000; 0 1.0000]; B = [0.0050; 0.1000]; C = [1 0]; D = [0]; T_s = 0.1 s K = [0.9171 1.6356]; L = [0.8327; 2.5029];	D ∈ [-25, 25] V ∈ [-30, 30] X_c = [0 ; 0]; d = 19.5873;
ESP	x = [β; r]; A = [0.4450 -0.0458;1.2939 0.4402]; B = [0.0550; 4.5607]; C = [0 1]; D = [0]; T_s = 0.04 s K = [0.2826 0.0960]; L = [-0.0390; 0.4339];	$\beta \in [-1, 1]$; $r \in [-2, 2]$ X_c = [0 ; -0.6423]; d = 1;

to detect the FDI attack. Here, we have computed the static threshold as 4.35 using χ^2 table for a χ^2-test window length 1, to keep the FAR below 0.05. The behavior of TTC under the optimal FDI attack sequence with this static threshold is demonstrated in Fig. 6a and Fig. 6c. In Fig. 6c, the red and blue solid lines represent the safety boundaries of distance D and velocity V respectively. The region bounded by the black dotted lines represent the preferable operating region of both D and V. The $'+'$-marked and $'\triangle'$-marked plots are the respective state trajectories of D and V. In Fig. 6c, we can see the state V exceeds the safety bound at 6-th sampling instance. However, the static threshold based detector fails to detect the attack before 6-th sample (Fig. 6a). On the contrary, in Fig. 7a, we can see the well-trained RL agent for detection system explores and exploits different threshold values in attempt to detect the same attack vector. And, at 5-th sample it detects the attack before the system can go beyond the safety boundary. Once detected, we consider the system is switched to secure mode from 5-th sampling instance. From this point, first, we observe the performance of TTC when only the LQR controller is used to bring back the system within X_R in Fig. 7c. We can see from Fig. 7c, the LQR controller fails to ensure safety while trying to mitigate the effect of the attack i.e., the velocity goes beyond the safety bound. On the other hand, when we use our proposed fast control inputs along with LQR, we can see in Fig. 7e that safety is ensured throughout the mitigation process. Moreover, the fast controller takes 11 samples (Fig. 7e) to bring all the system states back to X_R, whereas the LQR controller takes 21 samples (Fig. 7c). Note that the plots in Fig. 7c and Fig. 7e carry the same meaning as in Fig. 6c. The proposed detector exhibits a TPR of 0.95 while detecting the optimal attack in TTC.

Case Study 2: Electronic Stability Control System: Electronic stability program (ESP) or electronic stability control adjusts the side slip β and yaw rate γ of a vehicle by controlling the steering wheel angle to maintain the vehicle's yaw stability [31]. The system matrices, X_S, X_R are specified in Tab. I. The range of yaw rate sensor and saturation limit of steering angle are 2.5 rad/s and 0.8125 rad respectively. We demonstrate ESP's behavior against an optimal FDI attack generated using the well-trained attacker agent. In case of ESP as well, we set the static threshold of the χ^2 detector as 4.35 considering FAR as 0.05 (from χ^2 table). With this static threshold based detector in place, the optimal attack

successfully steer the yaw rate γ beyond its safety bound (Fig. 6d), while the detector could not even detect the attack (Fig. 6b). In Fig. 6d, the red and blue solid lines represent the safety boundaries of side slip β and yaw rate γ respectively. The region bounded by the black dotted lines represent the preferable operating region of γ. The $'+'$-marked and $'\triangle'$-marked plots are the respective state trajectories of β and γ. We can see in Fig. 7b that our proposed adaptive detector flags the attacker at 4-th sample before the yaw rate can become unsafe at 6-th sample. From the point of detection, we can see the proposed fast controller can bring the system to X_R in 2 samples (Fig. 7f), while the LQR controller takes one sample extra to do so (Fig. 7d). Note that the plots in Fig. 7d and Fig. 7f carry the same meaning as in Fig. 6d. The proposed detector exhibits a TPR of 0.89 while detecting the optimal attack in ESP.



(a) **TTC :** Attack Detection Using Adaptive Threshold

(b) **ESP :** Attack Detection Using Adaptive Threshold

(c) **TTC :** Attack Mitigation Using Optimal Controller

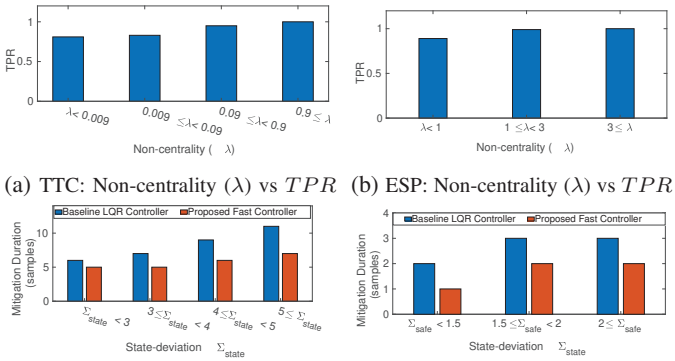
(d) **ESP :** Attack Mitigation Using Optimal Controller

(e) **TTC :** Attack Mitigation Using Fast Controller

(f) **ESP :** Attack Mitigation Using Fast Controller

Fig. 7: TTC and ESP with and without Proposed Mitigation Strategy *(pref. reg. : preferable region; ub : upper bound; lb : lower bound)*

Overall Real-time Performance Analysis: We provide a statistical performance analysis over a range of false data injection scenarios in order to evaluate the effectiveness of our RL-based detection and formal mitigation framework along with its timing overhead estimate. We have utilised the *non-centrality* parameter λ to evaluate the *stealth* of the attacker. The relation between the non-centrality of χ^2-statistics of the attacked system's residue and the detectability of an attack has already been established mathematically in Sec. III-A, i.e. higher the non-centrality value , more detectable is the FDI attack. We now introduce a *state-deviation* distance measure, Σ_{state} to evaluate the maximum deviation of system states from its preferable operating region, that is incurred by a stealthy FDI attack before it is detected. Let $\Sigma_{state} = \frac{\max|x_k - X_R|}{\max|X_R|}$, where x_k is the system states at k-th iteration, when the attack is detected, and X_R is the preferable safety region boundary of the system. Here, $X_R \in \mathcal{I}^n$ with $\mathcal{I}$ representing a bounded real interval (see Sec. III-B). $|x_k - X_R|$ denotes the absolute deviation of each component in x_k from the nearest boundary of the respective interval in X_R. Subsequently, $\max|x_k - X_R|$ denotes the maximum of these n deviations where n is the number of components in x_k and X_R. Also, $max\ X_R$ denotes maximum of the absolute boundary values among all the intervals in X_R. This parameter



(a) TTC: Non-centrality (λ) vs TPR

(b) ESP: Non-centrality (λ) vs TPR

(c) TTC: State Deviation (Σ_{state}) vs Mitigation Duration

(d) ESP: State Deviation (Σ_{state}) vs Mitigation Duration

Fig. 8: Statistical Analysis Data

helps us measure the *success-level* of an FDI attack sequence. A positive Σ_{state} value signifies that the FDI attack has maintained its stealth and corrupted x_k to be outside X_R. We measure the effectiveness of the proposed adaptive threshold-based detector with promised TPR during detection of an FDI attack. The effectiveness of the fast controller is measured by the number of secure iterations it requires to bring the attack-affected system states back to X_R ($mitigation\ duration$). We have simulated both TTC and ESP system models with the trained adaptive detector agent and attack-mitigating control inputs in the presence of the trained optimal attacker agent for 200 episodes (each spanning 80 sampling periods and starting from random initial value in the preferable operating region). We observe how well our proposed defense system works against different optimal FDI attack scenarios posed by the trained attacker agent. Fig. 8 (a) and (b) demonstrates average performance of the adaptive detection module against stealthy FDI attacks for TTC and ESP respectively. As we know, smaller the non-centrality (λ) value of χ^2 distribution of the system residue , harder it is to detect the attack. As the λ values obtained from the simulated attack scenarios decrease we can see the average TPR promised by our adaptive detector decreases as well, i.e. attacks are hard to be truly detected for both TTC and ESP. With suitable grouping of λ values we can see even for the FDI attacks which are most stealthy (i.e. $\lambda < 0.009$ for TTC and $\lambda < 1$ for ESP) our adaptive detection

module promises an average TPR of 0.81 and 0.89 for TTC and ESP. The proposed detection system exhibits an average FAR of $0.035 \sim 0.037$ and $0.12 \sim 0.15$ for TTC and ESP respectively. Other evaluation metrics are enlisted in Tab. II. To showcase an empirical real-time eligibility of the trained detector RL agent, we observe the run time of the detector agent during multiple set of simulations as mentioned earlier. The detector agent exhibits a maximum run time of 0.0086 seconds in each sampling period (refer to Tab. I) approximately in our 1.8 GHz octa-core i7 CPU, under a maximum usage of $\sim 23\%$ of one CPU core with average CPU load being around 0.5 for a 1 minute observation interval.

TABLE II: Evaluation Metrics

Evaluation Metrics	ESP	TTC
TPR	0.89	0.81
FAR	$0.12 \sim 0.15$	$0.035 \sim 0.037$
Precision	$0.86 \sim 0.88$	0.96
Recall	0.89	0.81
F1 Score	$0.87 \sim 0.89$	0.88
Accuracy	$0.87 \sim 0.89$	0.89

Fig. 8 (c) and (d) on the other hand demonstrate the average number of mitigation instances required to bring an attacked system state back to its preferable operating region for a range of state deviations Σ_{state}. Note that, for both the systems, higher the Σ_{state}, longer the average $mitigation\ duration$ required to mitigate the effect of FDI attack. With the blue bars showing average mitigation duration required for the LQR controller (that the systems use normally) and the orange bars showing the average mitigation duration required by the fast controller, we can see a significant improvement in mitigation. Even for the highest range of Σ_{state} values (i.e. $\Sigma_{state} \geq 5$ for TTC and $\Sigma_{state} \geq 2$ for ESP) introduced by the simulated optimal attacks, on average the fast controller performs almost 33% better than the LQR controller for both of the systems. Therefore, these test results help us visualize the performance improvement w.r.t. faster detection and mitigation while using the proposed adaptive threshold-based detection module and the fast control module under different attack scenarios.

V. Related Work

In this section, we discuss some of the literature related to the present problem definition. Specifically, we will focus on previous works on i) adaptive detectors that reduces FAR and enhances TPR, and ii) attack mitigation strategies. In line of our objective, authors of [9], [10] have also proposed anomaly detectors in context of CPS that varies the detection threshold. [9] presented two greedy algorithms based on formal methods to generate a set of monotonically decreasing thresholds in off-line mode. The detector uses these thresholds whenever the controller is triggered. They focus on detecting targeted performance degrading FDI attacks and reducing FAR. On the other hand, the authors of [10] presented an attacker-defender game to solve the adaptive threshold selection problem.

In the context of attack mitigation, we briefly discuss the methods proposed in [11], [15], [17], [16], [18]. The approach in [11] proposes an RL based robust control strategy for autonomous vehicles (AV) in the presence of an attacker who modifies the spacing information between vehicles. The method leverages the fact that measurement information (velocity of other vehicles) is drawn from multiple sensors and learns the optimal weights of these sensors that mitigate the attack's effect. [15] presented a secure state estimation problem which is further leveraged to compute attack-mitigating robust control inputs using RL. On the other hand, authors of [16] proposed a formal method based approach to design control inputs for safe system trajectory under FDI attacks. They showed their approach can successfully generate attack-resilient control inputs mostly for entire initial state space. In a different line of approaches, authors of [17] proposed a sporadic MAC based secure CPS design (like [3]) using formal method theory. A recent work [18] presented a online attack recovery method. They estimate the current system state from the most latest trusted data using the checkpoint method from [32] followed by which, they synthesize recovery control inputs using linear program (LP) and formal methods.

The idea of utilizing simplicity to control complexity, i.e. Simplex architecture was first proposed by Lui Sha et al. [13], [14]. In such systems, a high-assurance controller is deployed besides the default high-performance controller to recover the system from any fault or attack. Such architectures have been considered by many researchers in secure CPS domain [33], [34]. In [33], trusted hardware components are used as high-assurance unit to increase security of the system. As the decider unit, they proposed a side-channel analysis based intrusion detection system. In case of connected and automated vehicles as discussed in [34], on detection of an attack, the system is switched to adaptive cruise control from cooperative adaptive cruise control.

VI. Conclusion

The present work proposes an end-to-end secure CPS framework that intelligently learns and detects FDI attacks and performs performance recovery using a fast attack mitigation strategy. There exist other notable works that propose static security schemes ([3],[17]) which upper bound the attack tolerance of a CPS by an allowable estimation error under attacks. Such works, do not attempt attack detection, incur higher scheduling overhead and suffer from performance issues under attack. Combining our approach with such a method so that we always have recurring sporadic security after a pre-defined interval will amalgamate safety guarantee of such schemes with intelligent detection and performance recovery advantage of our scheme. This along with utilizing the model-based control-theoretic knowledge on *attackability, resilience* of CPS to enhance the learning process can be fruitful future endeavours.

Acknowledgment

This work has been done in the High-Performance Real-Time Computing (HiPRC) Lab of Computer Science and Engineering department at Indian Institute of Technology

Kharagpur. We wish to express our gratitude to Prof. Johan Löfberg from Linköping University, Sweden and Prof. Sayan Mitra from University of Illinois at Urbana-Champaign for their generous technical suggestions.

REFERENCES

[1] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—a survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.

[2] A. Munir and F. Koushanfar, "Design and analysis of secure and dependable automotive cps: A steer-by-wire case study," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[3] I. Jovanov and M. Pajic, "Relaxing integrity requirements for attack-resilient cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 4843–4858, 2019.

[4] V. Lesi, I. Jovanov, and M. Pajic, "Security-aware scheduling of embedded control tasks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–21, 2017.

[5] Y. Mo and B. Sinopoli, "False data injection attacks in cyber physical systems," in *SCS, Stockholm*, 2010.

[6] A. S. Willsky, J. J. Deyst, and B. S. Crawford, "Two self-test methods applied to an inertial system problem," *Journal of Spacecraft and Rockets*, vol. 12, no. 7, pp. 434–437, 1975.

[7] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[8] A. Teixeira *et al.*, "Secure control systems: A quantitative risk management approach," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 24–45, 2015.

[9] I. Koley, S. K. Ghosh, S. Dey, D. Mukhopadhyay, A. K. KN, S. K. Singh, L. Lokesh, J. N. Purakkal, and N. Sinha, "Formal synthesis of monitoring and detection systems for secure cps implementations," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 314–317.

[10] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for anomaly-based intrusion detection in dynamical environments," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9996 LNCS, pp. 415–434, 2016.

[11] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 307–312.

[12] Y. Wang, C. Huang, and Q. Zhu, "Energy-efficient control adaptation with safety guarantees for learning-enabled cyber-physical systems," *arXiv preprint arXiv:2008.06162*, 2020.

[13] L. Sha *et al.*, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[14] D. Seto, B. H. Krogh, L. Sha, and A. Chutinan, "Dynamic control system upgrade using the simplex architecture," *IEEE Control Systems Magazine*, vol. 18, no. 4, pp. 72–80, 1998.

[15] Y. Zhou, K. G. Vamvoudakis, W. M. Haddad, and Z.-P. Jiang, "A Secure Control Learning Framework for Cyber-Physical Systems under Sensor Attacks," in *2019 Am. Control Conf.* IEEE, jul 2019, pp. 4280–4285.

[16] Z. Huang, Y. Wang, S. Mitra, and G. Dullerud, "Controller synthesis for linear dynamical systems with adversaries," in *Proceedings of the Symposium and Bootcamp on the Science of Security*, 2016, pp. 53–62.

[17] S. Adhikary, I. Koley, S. K. Ghosh, S. Ghosh, S. Dey, and D. Mukhopadhyay, "Skip to secure: Securing cyber-physical control loops with intentionally skipped executions," in *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy*, 2020, pp. 81–86.

[18] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time attack-recovery for cyber-physical systems using linear approximations," in *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020, pp. 205–217.

[19] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72.

[20] A. F. Siegel, "The noncentral chi-squared distribution with zero degrees of freedom and testing for uniformity," *Biometrika*, vol. 66, no. 2, pp. 381–386, 1979.

[21] K. J. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[22] J. Löfberg, "Automatic robust convex programming," *Optimization methods and software*, vol. 27, no. 1, pp. 115–129, 2012.

[23] C. Fan, U. Mathur, S. Mitra, and M. Viswanathan, "Controller synthesis made real: reach-avoid specifications and linear dynamics," in *International Conference on Computer Aided Verification*. Springer, 2018, pp. 347–366.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[25] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[26] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 2004, pp. 284–289.

[27] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[28] N. Bjørner, L. de Moura, L. Nachmanson, and C. M. Wintersteiger, "Programming z3," in *International Summer School on Engineering Trustworthy Software Systems*. Springer, 2018, pp. 148–201.

[29] L. De Moura *et al.*, "Z3: An efficient smt solver," in *TACAS*. Springer, 2008.

[30] V. Lesi *et al.*, "Integrating security in resource-constrained cyber-physical systems," *ACM TCPS*, vol. 4, no. 3, pp. 1–27, 2020.

[31] S. Zheng *et al.*, "Controller design for vehicle stability enhancement," *Control Engineering Practice*, vol. 14, no. 12, pp. 1413–1421, 2006.

[32] F. Kong, M. Xu, J. Weimer, O. Sokolsky, and I. Lee, "Cyber-physical system checkpointing and recovery," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 22–31.

[33] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3a: Secure system simplex architecture for enhanced security and robustness of cyber-physical systems," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, 2013, pp. 65–74.

[34] C. Zhao, J. S. Gill, P. Pisu, and G. Comert, "Detection of false data injection attack in connected and automated vehicles via cloud-based sandboxing," *IEEE Transactions on Intelligent Transportation Systems*, 2021.