Fail-Safe: Securing Cyber-Physical Systems against Hidden Sensor Attacks

Mengyu Liu
Syracuse University
mliu71@syr.edu

Lin Zhang
Syracuse University
lzhan120@syr.edu

Pengyuan Lu
University of Pennsylvania
pelu@seas.upenn.edu

Kaustubh Sridhar
University of Pennsylvania
ksridhar@seas.upenn.edu

Fanxin Kong
Syracuse University
fkong03@syr.edu

Oleg Sokolsky
University of Pennsylvania
sokolsky@seas.upenn.edu

Insup Lee
University of Pennsylvania
lee@seas.upenn.edu

***Abstract*—In Cyber-Physical Systems (CPS), integrating new technologies that interact with and control physical systems raises new security risks beyond the classical cyber security domain. These risks motivated many attack detectors that focus on the binary outcome. However, one pressing risk in CPS is hidden sensor attacks that are well-designed by powerful attackers who gained full knowledge of our systems and detector. The hidden attacks inject such a small malicious signal into sensor measurement that they can stay undetected but eventually lead to a significant deviation. Thus, to secure the CPS, we propose a detection framework to identify these sensor attacks that can drive the system's physical states to an unsafe state within a given period, even if they are not detected. First, we solve optimization problems to find the optimal hidden sensor attack that leads to the minimal distance to a pre-defined unsafe state region within an observation window for a given system and detector. Then, based on this algorithm, we perform offline profiling to search for a conditionally safe region, where the system states are guaranteed to be safe within the observation window as long as the detector does not raise any alerts. Finally, the framework can online discover potential hidden sensor attacks that endanger the system by checking if the current system state moves out of the region and raising a yellow alert. The evaluation shows that the optimal hidden sensor attack results in the minimum distance to unsafe, within a given observation window among existing hidden sensor attacks. We implemented our method on four linear simulators to show the effectiveness of our method. Additionally, we provided a discussion on the challenges of applying the proposed method to non-linear systems.**

***Index Terms*—hidden sensor attack, cyber-physical systems, detection**

I. Introduction

Cyber-physical Systems (CPS) hybridizes software computational processes and physical components in our daily life. These systems convey life-critical functionalities such as smart grids, manufacturing, and driverless vehicles [1]–[3]. Therefore, unlike traditional software system malfunctions, there is a significant problem that failures in CPS can potentially lead to large financial loss, physical damages and can be life-critical. Targeting this problem, researchers have striven for solutions to secure CPS from faults and adversarial attacks [4]–[6].

CPS attacks target various components, including sensors, controllers, actuators and communication channels. Among these, sensor attacks maliciously modify sensory inputs and put the entire system at risk. One major reason that researchers focus on sensor attacks is that they can be injected without much expertise, via both non-transduction and transduction, i.e. with and without a computer, respectively [7], [8], methods that the traditional software security domain overlooks [9]–[11]. For instance, an attacker can manipulate the GPS signals of a yacht to drive it off a planned path [12] or cheat the wheel speed sensors to interrupt a vehicle's antilock brakes [13]. Upon an adversarially overwritten sensory input, the CPS controller can misbehave and cause catastrophic results [14], and these attacks will continue to co-evolve with the rise of CPS autonomy.

Such emerging threats have motivated novel proposals on sensor attack detection. On one hand, some researchers identify anomalies by leveraging the correlation of redundant sensors that measure the same physical variables [14]–[17]. The detection mechanism is therefore based on this correlation, which can be broken by sensor attacks. However, this type of detection method fails to find anomalous sensors uncorrelated to others. On the other hand, many researchers utilize prediction from machine learning, pre-known dynamics, or other techniques. These frameworks detect sensor attacks by comparing sensor measurements with predicted values [18]–[20] and raise a red flag upon a large residual between these two. The residual can be tested through stateless methods, such as Chi-Square, or stateful ones, such as cumulative sum (CUSUM) which leads to a smaller deviation from the expected states [21].

Despite the promising detectors such as CUSUM, **hidden sensor attacks** can still bypass them and stay undetected, because they are generated deliberately by the adversaries with full knowledge of the system and deployed detector. Thus, such sensor attacks have gained much attention from researchers, who study them against cumulative sum (CUSUM) or Chi-Square-based attack detectors in [22]–[25]. Note that there is a similar notion of sensor attack, called stealthy attacks, which is a superset of hidden attacks. According to the definition of stealthy attack from [26], during the stealthy attack, the squares of differences between the sensor measurements and their estimation are not significant.

Since the hidden sensor attacks are hidden from the detector, traditional metrics to evaluate detectors, such as true positive rate and F1-score, are invalid. Instead, one viable approach is to evaluate the worst impact caused by undetected attacks under a certain detector. An example impact metric [24] is defined as how much can the attacker drive variables of interest in the process towards its intended goal. To forge a worst case that maximizes impact metrics, researchers assume an omniscient adversary to which our system and detector parameters are visible, and then it operates one of the three widely used hidden attacks - surge, bias, and geometric [27]. These three attacks are not necessarily causing the worst case scenario for the system, based on their mathematical definitions. We are motivated by the existence of an **optimal hidden attack** for the adversary, which drives the system state closest to, or deepest into the unsafe region, regardless of its method.

Although many researchers have analyzed the impact of sensor attacks in CPS such as [28]–[31], a clear understanding of the following questions has not been addressed: (i) What is an optimal hidden sensor attack that causes the worst impact on state deviation? (ii) How can we defend against the optimal hidden attacks to secure a CPS, since they are hidden from the existing attack detector? We believe that understanding these questions is a crucial step to secure the CPS even if the sensor attacks are not detected.

In this paper, we investigate the two questions above. First, we check the performance of frequently used hidden attacks in literature, and observe that none of them can result in the minimal distance to a pre-defined unsafe region for all different systems with various attack detectors. Thus, we try to formulate the attack-detection process as an optimization problem to find the **optimal hidden sensor attack** that injects false sensory data and drives the system states closest to unsafe within an user-given time horizon on general dynamics, controls and detectors. Particularly, we demonstrate such an optimization problem can be solved efficiently as a linear programming or convex optimization problem, respectively for linear and convex dynamics, controls and detectors, such as CUSUM. Then, based on the optimal hidden sensor attack, we can conservatively analyze in which state region the system keeps safe within a given period. The above procedures are completed offline. At run time, if current states move outside of the region searched, our detector raises a **”yellow” alert**, which indicates a potential hidden attack that can drive the system to an unsafe state within the given period of time. This ”yellow” alert is auxiliary to the ”red” alert raised by an ordinary attack detector (such as CUSUM), with the latter indicating an already happened sensor attack but giving us less time to respond. Finally, we build four simulators to evaluate our approach. The results show the optimal hidden attacks perform no worse than other baselines, and the hidden attack detector can help identify potential hidden sensor attacks.

Our contributions are as follows.

- We present an optimization-based method to find the optimal hidden sensor attack within a given period. We show that this method is general to CPS, and can be efficient for linear and non-linear convex systems.
- We design an efficient search algorithm to find the conditionally safe region given the optimal hidden sensor attack. The region is a subset of state space, in which the system remains safe within the given period as long as the ordinary detector does not raise any alerts.
- We build a hidden attack detector based on the conditionally safe regions found and show its promising performance in experiments.

The rest of this paper is organized as follows. Section II presents the background and preliminaries. Section III describes the architecture overview of our attack detection framework. Section IV formulates optimization problems to find the optimal hidden sensor attack within a time period given an initial state. Section V searches the conditionally safe region for a system. Section VI validates the proposed framework with four CPS simulators. Section VII discusses the application of the proposed framework to non-linear systems.

II. Background and Preliminaries

A. Notations

Notation	Meaning
$\boldsymbol{x}$	The system’s ground-truth states
$\tilde{\boldsymbol{x}}$	State measurement by sensors
$\hat{\boldsymbol{x}}$	State estimation by computation
$\boldsymbol{u}$	The control inputs
T	Observation window size
$\boldsymbol{\tau}$	Detector threshold
$\boldsymbol{b}$	Detector drift
$\boldsymbol{r}$	Residual between $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$
$\boldsymbol{s}$	CUSUM detector score
U	Unsafe set
OHA	Optimal hidden sensor attack
CSR	Conditionally safe region

TABLE I: Notation table for the following sections

Table I lists the notations used in this paper. The bold text represents a multi-dimensional vector, such as system states and control inputs, and non-bold text represent a scalar value. Moreover, we use subscripts to denote timestamped variables, such as $\boldsymbol{x}_t$ means system’s ground-truth state at timestamp t. We also use square brackets to denote a specific dimension, such as $OHA[i]$ means the optimal hidden attack on dimension i. One remark is that, without loss of generality, we assume the system state is fully observable, i.e. $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are in the same state space.

B. System Model

The CPS model considered in this paper is a physical process, also called a plant, controlled by a computer program or controller. The controller operates at every constant time, called a control step. At the beginning of each step t, the controller reads sensor measurements and computes the state estimate of the plant, represented by the values of a set of real-valued variables $\tilde{\boldsymbol{x}}_t = \{\tilde{x}[1]_t, \dots, \tilde{x}[n]_t\}$, where n is the number of states in the system. Then, it generates the control inputs $\boldsymbol{u}_t = \{u[1]_t, \ldots, u[m]_t\}$ based on the control algorithm, where m represents the number of actuators. Generally, the control algorithms can be represented as a function. Then, actuators carry out the control inputs to drive the system to a reference or target state. For ease of presentation, we assume that the plant is fully observable to the sensors, i.e., all state estimates can be obtained from sensor measurements. Note that there are three types of states in the paper: (1) actual physical states, denoted as $\boldsymbol{x}$, are real states of the system, (2) measured states from sensors, denoted as $\tilde{\boldsymbol{x}}$, which might be attacked and do not have to follow dynamics f, and (3) estimated states, denoted as $\hat{\boldsymbol{x}}$, are computed from previous measurements through f. In short expressions, the actual state, expected state, and state estimate at time t_k are also expressed as $\boldsymbol{x}_k$, $\hat{\boldsymbol{x}}_k$, and $\tilde{\boldsymbol{x}}_k$, where $k \in \mathbb{N}_0$ means control step number.

C. Threat Model

We consider sensor attacks, which alter sensor measurements sent to the controller. That is, an attacker can manipulate the state estimates $\tilde{\boldsymbol{x}}$ computed by the controller. Generally, the attacker can affect all n state estimates, or part of them.

Moreover, to derive and analyze the optimal hidden sensor attack, we consider a strong adversarial model, which makes sure the sensor attacks can bypass or escape from detection, i.e., hidden sensor attacks, and deviate the system states farther from reference states. To be more specific, the attacker has full knowledge of

- the control program or control logic. The attacker can predict the control inputs $\boldsymbol{u}$ generated by the controller given current state estimates.
- the system model used by the detector. The attacker can compute the expected states $\hat{\boldsymbol{x}}$ predicted by the detector given control inputs.
- the attack detection method and its parameter settings. The attacker can forge malicious attack payload carefully so that no alert is raised by the detector.

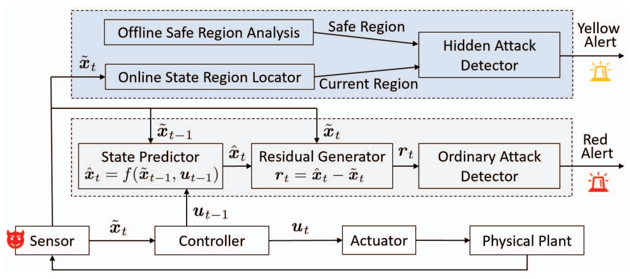
D. Ordinary Attack Detector

To distinguish the existing attack detector deployed on CPSs with our hidden attack detector, we call them *ordinary attack detectors*. Note that a powerful adversary with full knowledge of ordinary detectors can build a hidden sensor attacks. We divide these ordinary attack detectors into three categories according to the size of the detection window: long-memory, short-memory, and memory-less detectors. Long-memory detectors take all historical data into account, such as CUSUM statistic; short-memory detectors monitor data within a certain detection window; Memory-less detectors check only current system states or sensor measurements.

1) Long-Memory Detectors: The CUSUM (cumulative sum) statistic is a long-memory attack detector, which computes the CUSUM score $\boldsymbol{s}$ by comparing the state estimates with expected states over time.

We use $\boldsymbol{s}_t \in \mathbb{R}^n$ to denote the score at control step t. The update policy for the score is described as:

$$\boldsymbol{s}_{t+1} = (\boldsymbol{s}_t + |\boldsymbol{r}_t| - \boldsymbol{b})^+, \tag{1}$$



Offline Safe Region Analysis
Safe Region
Hidden Attack Detector
Yellow Alert
$\tilde{\boldsymbol{x}}_t$
Online State Region Locator
Current Region
$\tilde{\boldsymbol{x}}_{t-1}$
$\tilde{\boldsymbol{x}}_t$
State Predictor
$\hat{\boldsymbol{x}}_t = f(\tilde{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_{t-1})$
$\hat{\boldsymbol{x}}_t$
Residual Generator
$\boldsymbol{r}_t = \hat{\boldsymbol{x}}_t - \tilde{\boldsymbol{x}}_t$
$\boldsymbol{r}_t$
Ordinary Attack Detector
Red Alert
$\boldsymbol{u}_{t-1}$
Sensor
$\tilde{\boldsymbol{x}}_t$
Controller
$\boldsymbol{u}_t$
Actuator
Physical Plant

Fig. 1: The overview of system design

where a^+ means $max(0, a)$, $b_i > 0$ is a parameter representing the drift that can avoid the increasing of CUSUM score when there is no attack. That is, $\boldsymbol{b}$ is a selected drift term to eliminate the accumulated noise from raising false alarms [11], [24], [32]. Further, $\boldsymbol{r}_t$ represents the residual between state estimate $\tilde{\boldsymbol{x}}_t$ and expected state $\hat{\boldsymbol{x}}_t$,

$$\boldsymbol{r}_t = \hat{\boldsymbol{x}}_t - \tilde{\boldsymbol{x}}_t \tag{2}$$

The prediction can be done by using the system dynamics. Finally, under sensor attacks, an alarm will be raised whenever the score in any dimension goes over the threshold, i.e., $\boldsymbol{s}_t > \boldsymbol{\tau}$, and the score will be reset to 0.

2) Short-Memory Detectors: Short-memory detectors monitor residuals within a certain detection window with size w. The sum of the residual within the window is denoted as

$$\boldsymbol{r}_t^{sum} = \sum_{i=t-w}^{t} \|\boldsymbol{r}_t\| \tag{3}$$

Similarly, an alarm will be raised whenever the average residual exceeds the threshold, i.e., $\boldsymbol{r}_t^{sum} > \boldsymbol{\tau}$.

3) Memory-Less Detectors: Memory-less detectors only check the current residual $\boldsymbol{r}_t$, and raise an alert when the residual is larger than the threshold, i.e., $\boldsymbol{r}_t > \boldsymbol{\tau}$. In other words, the memory-less detectors are a special group of short-memory detectors where the window size is one.

III. Overview of Detection Framework

The detection system is divided into two parts, shown as Figure 1. The ordinary attack detectors (Section II-D) identify non-hidden attacks and raise a red alert, and this part has been thoroughly studied by previous works [33], [34]. Thus, this paper focuses on the second part (blue shaded box) to identify potential hidden sensor attacks and raise a yellow alert by checking the current system states. This part can find a safe region of states for a system offline by the following steps. First, it optimizes to find the minimum distance to unsafe at a certain control step given a system and an observing time window (Section IV). Then, it can find the optimal hidden attack in this system within the observation time by solving the optimization problems with all possible control steps when the attack achieve the largest deviation (Section IV-C). If the optimal hidden attack cannot drive the system to unsafe state within the observation window, then the current system state is conditionally safe. Finally, our framework heuristically searches for all conditionally safe states to obtain the conditionally safe region (Section V). When the system runs online, it checks the location of the current system states. The hidden attack detector will raise the yellow alert to hint a potential hidden sensor attack if the system states move out of the conditionally safe region.

IV. The Optimal Hidden Attack Against General Scenarios

In this section, we show that optimal hidden attack (OHA) is well-defined for general control systems and propose an optimization-based method to find the OHA for a cyber-physical system given the current state.

A. Assumptions and Problem

We have the following assumptions:

1) The sensor attack happens on a known control system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, which takes a finite change in every dimension of state per unit of time, i.e. $|\dot{\boldsymbol{x}}| \leq \Delta\overline{\boldsymbol{x}}$ where the $\leq$ is dimension-wise.
2) The system is equipped with a state estimator to predict state $\hat{\boldsymbol{x}}$ by roll-forwarding from a (not necessarily trustworthy) cached state on dynamics f.
3) The system is equipped with a detector $g(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}})$, with a state estimation and a physical measurement as input. An attack is detected at time t on $g(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t) \geq 0$.
4) Control signal $\boldsymbol{u}$ is bounded inside the interval $[\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}]$.
5) An unsafe state space is given in form of a Cartesian product of half spaces in every state dimension. That is, for n-dimensional state space, the unsafe set U takes the form of Equation (4).

$$U = U[1] \times U[2] \times \cdots \times U[n] \text{ where}$$
$$U[i] = (-\infty, xb[i]] \text{ or } U[i] = [xb[i], \infty) \text{ for } i = 1, \ldots, n \tag{4}$$

For arbitrary dimension i, we aim to find a conditional safe region (CSR) that is a subset of the state space, such that no matter what hidden sensor attack is applied, the system cannot reach the unsafe set within a time period T. If the CSR of every dimension can be computed offline, then during runtime, the system is able to query whether the current state is within the CSR - if not, we raise a yellow alarm. Our approach is to search for the worst-case scenario of each dimension i, where the hidden attack that drives the state closest to (or deepest in) $U[i]$ is computed. We denote this hidden attack as optimal hidden attack (OHA), from which the corresponding CSR can be found.

B. Optimal Hidden Attacks

To define an OHA, we start from a motivating example. Here, a vehicle is turning on a road, with a one-dimensional state x, representing its speed in meters per second, and a one-dimensional control variable u, representing the voltage difference between its two motors in volt. The system dynamics is given by $\dot{x} = -\frac{25}{3}x + 5u$. The vehicle will be in danger if its speed is too high, particularly with an unsafe set $U = \{x \mid x > 1.26\}$. Aiming for unsafe, a malicious attack is injected to the vehicle's sensor at $t = 8.00$. As shown in Figure 1, there is an attack (marked as optimal) that drives the system state closer to the unsafe region than others within time period $[8.00, 10.00]$. To secure a system from undetected attacks, we must consider the worst-case scenario, where the distance of state to unsafe is minimal. Therefore, we need to define optimal hidden attacks from this idea.

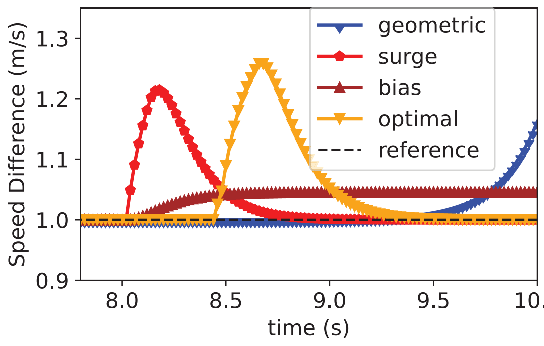


Fig. 2: Motivating example

A prerequisite to define OHAs is to formalize the distance metric from a state to an unsafe set U, which may span across multiple dimensions. A state within the unsafe set indicates something disastrous happens. For example, for an unmanned aerial vehicle (UAV), negative height values are unsafe, which means the UAV crashes into the ground; for a nuclear fuel element, the temperatures exceeding its melting point are unsafe, indicating a core meltdown accident.

Notice that in dynamics, each dimension can have very different physical meanings, such as length and mass, with different units, such as meters and kilograms. Therefore, it is difficult to minimize a cross-dimensional distance metric, e.g., L2-norm. We hence use a per-dimensional approach, finding OHAs with respect to each distinct dimension of a system as follows.

Definition IV.1 (Distance from a State to Unsafe)**.** *For an n-dimensional state $\boldsymbol{x} \in \mathbb{R}^n$ and an unsafe set $U \subset \mathbb{R}^n$, the i-th dimension of $\boldsymbol{x}$ is $x[i] \in \mathbb{R}$, while the i-th dimension of U an interval $U[i] = (-\infty, xb[i]$ or $[xb[i], \infty)$. We then define the distance from $\boldsymbol{x}$ to U in dimension i as*

$$dist^{(i)}(\boldsymbol{x}, U) = \begin{cases} |x[i] - xb[i]| & \text{if } x[i] \notin U[i] \\ -|x[i] - xb[i]| & \text{otherwise} \end{cases} \tag{5}$$

In other words, if the state is outside of the unsafe set, its distance is how far away it is to the boundary. Otherwise, if it is inside, the distance is the negation of depth.

From the per-dimensional definition of distance, we can then define an OHA at a given dimension for a given period of time T. Denote actual states as $\boldsymbol{x}$, observed states (under attack) as $\tilde{\boldsymbol{x}}$ and control signals as $\boldsymbol{u}$.

Definition IV.2 (Optimal Hidden Attack)**.** *Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ be a control system of n dimensions, with state space $\mathcal{X} \subseteq \mathbb{R}^n$. Assume we have the knowledge of an unsafe set $U \subset \mathcal{X}$. An optimal hidden attack $OHA[i]$ of dimension i at time t_a for a period T is a sensor attack injected at t_a that minimizes $dist^{(i)}(\boldsymbol{x}, U)$ during an observation period $[t_a, t_a + T]$. The attack is hidden, such that every sensor measurement during the attack period bypasses a detector g. That is, $g(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t) < 0$ for all $t \in [t_a, t_a + T]$. In other words, no other attacks starting at t_a can drive the system state's i-th dimension closer to unsafe than this attack and still remain hidden.*

Based on the definition, we can obtain the following theorem that states the existential property of OHA.

Theorem IV.1 (Existence of OHA on General Systems)**.** *We have a general control system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, with a dimension-wise upper-bounded as $|\dot{\boldsymbol{x}}| \leq \Delta\overline{\boldsymbol{x}}$. At any instance t_a of its runtime, if there exists a hidden attack on dimension i, then there exists an OHA on that dimension with respect to an unsafe set $U[i]$ and an observation period T.*

Proof. Without loss of generality, we let the time step be 1, i.e., $|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t| \leq \Delta\overline{\boldsymbol{x}}$. At time t_a, given the upper bound in dimension i, we have

$$|x_{a+1}[i] - x_a[i]| \leq \Delta\overline{x}[i].$$

This inequality also applies to any time step in $[t_a, t_a + T - 1]$, and by summing both sides of all these inequalities together throughout the period T, we obtain

$$\sum_{t=t_a}^{t_a+T-1} |x_{t+1}[i] - x_t[i]| \leq T\Delta\overline{x}[i].$$

Then, with triangle rule of distance metrics,

$$|x_{a+T}[i] - x_a[i]| \leq T\Delta\overline{x}[i].$$

Therefore, no matter the system is under what sensor attack, its dynamics only allow a finite distance away from $x_a[i]$ at dimension i after T steps.

Based on this fact, the metric $dist^{(i)}(\boldsymbol{x}_{a+T}, U)$ must be finite based on its definition, and therefore it must have a minimum. Denote this lower bound as $\underline{d}$. Consequently, given hidden attacks exist, there is at least one hidden attack that gives the smallest $dist^{(i)}(\boldsymbol{x}_{a+T}, U) \geq \underline{d}$, and this is the OHA. □

We can assume this upper bound in change of state per time step because most systems have inertia. For instance, the motivating example in Section IV-B shows a linear relationship between the acceleration $\dot{x}$ and the speed x. Since the law of physics upper bounds the speed, the acceleration is also upper-bounded. In a more general sense, there exist dynamical systems that do not abide by inertia, but these are not our focus.

The metrics to evaluate attack detectors, such as accuracy and recall rate, are not suitable for hidden attacks including OHA, because they remain undetected. We can use the maximal state deviation from reference (or target) states x_r within observing time T to evaluate the effectiveness of hidden attacks. For example, a vehicle running the cruise control task on road suffers from hidden sensor attacks. Although attacks are undetected, for given $T = 30$ seconds, an attack that caused more speed deviation from target speed in these 30 seconds is more powerful. Note that, the observing time T is user-given, and provide a cushion time for system to handle potential attacks.

C. Searching for Optimal Hidden Attacks

In order to understand what is the worst-case scenario of a system during a period of time, we need to compute the OHA efficiently. Therefore, we construct an algorithm to search for OHAs. For simplicity, we first assume the largest state deviation happens at a fixed time instance $t_c \in [t_a, t_a + T]$, or equivalently step c in the observation period. We formulate an optimization problem to find the corresponding measurement sequence $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \cdots \tilde{\boldsymbol{x}}_c$ desired by an attacker. The initial time is denoted as $t_0 = t_a$, and the observing period length is T. Thus, the objective is to minimize $dist^{(i)}(\boldsymbol{x}_c, U)$ at t_c, with the actual state $\boldsymbol{x}_c$ at time t_c and given U and i, under the following three constraints:

1) System dynamics. The state estimator takes in the measurement at a time step and predicts the next state based on system dynamics. Here, we use the equivalent discrete dynamics as constraint, i.e.

$$\hat{\boldsymbol{x}}_t = f(\tilde{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_{t-1}). \tag{6}$$

2) Hidden constraint. The detector g must output a value smaller than 0 starting from t_a. It makes sure the attack can not trigger an alert from the attack detector.

$$g(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t) < 0 \tag{7}$$

3) Control limits. It is usually determined by the physical properties of the actuator. For example, the maximum inflow of a water pipe is 0.05 meter^3/sec, so the inflow is within the range $[0, 0.05]$. Formally,

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}_t \leq \overline{\boldsymbol{u}} \tag{8}$$

We solve this optimization problem for an attacked measurement sequence affected by sensor attack. The sequence leads to the minimal $dist^{(i)}$ at time t_c.

Next, we lift the assumption that minimal $dist^{(i)}$ happens at a specific time t_c. Now we do not know at which step in $[t_a, t_a + T]$ the attack achieves the smallest distance to unsafe set. Thus, we need to solve multiple optimization problems as above, with t_c varying in $t_a, t_a + 1, \cdots, t_a + T$. The entire algorithm is shown in Algorithm 1. Note that at Line 3, we formulate an optimization problem m as Equation (9).

$$\begin{aligned} &\text{minimize } dist^{(i)}(\boldsymbol{x}_c, U) \\ &\text{subject to } \bigwedge_{t=0}^{c} \big((6) \wedge (7) \wedge (8)\big) \end{aligned} \tag{9}$$

with the corresponding three constraints (6), (7) and (8). Then, at Line 5-7, we update the solution to the one with the smallest objective $dist^{(i)}$. Based on this algorithm, if we obtain the same minimal objective at multiple t_c, we pick the earliest t_c. That is, the OHA needs to do its damage as early as possible.

Algorithm 1: OHA search for general system

```
Input: $\boldsymbol{x}_0$, $U$, $i$, $f$, $g$, $\underline{\boldsymbol{u}}$, $\overline{\boldsymbol{u}}$, $T$
/* $\boldsymbol{x}_0$: system state at time $t_0 = t_a$      */
/* $U$: unsafe set                                      */
/* $i$: dimension to optimize attack                    */
/* $f$: system dynamics                                 */
/* $g$: detector statistics function,
   $g \geq 0$ means attack is detected                  */
/* $\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}$: control limits   */
/* $T$: observation period length                       */
Output: $opt\_seq$, $opt\_d$
/* $opt\_seq$: the optimal attacked
   measurement sequence $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \cdots \tilde{\boldsymbol{x}}_c$   */
/* $opt\_d$: minimal $dist^{(i)}(\boldsymbol{x}_c, U)$      */
1 $opt\_seq \leftarrow \emptyset$, $opt\_d \leftarrow \infty$
2 for $t_c \leftarrow 1$ to $T$ do
3     Formulate problem $m$ as Equation (9)
4     $obj, seq \leftarrow m.solve()$
5     if $obj < opt\_d$ then
6         $opt\_d \leftarrow obj$ ;      // objective value
7         $opt\_seq \leftarrow seq$ ;        // opt. var.
```

D. OHAs on Linear and Convex Systems and Detector Statistics

Algorithm 1 provides a scheme to search for OHAs, assuming we have a fixed trustworthy initial state $\boldsymbol{x}_0 = \tilde{\boldsymbol{x}}_0$, and such that we are able to obtain ground-truth $\boldsymbol{x}_c$ by roll-forwarding on f for arbitrary $c \geq 0$. Nevertheless, each optimization problem m can be hard to formulate and solve in general, and the bottlenecks are the system dynamics constraint f and the hidden constraint g. Fortunately, we can leverage linear programming (LP) and convex optimization (CVXOPT) for linear and nonlinear but convex control dynamics and detector statistics. The relevant analysis is detailed in the proofs of Theorem IV.1 and IV.2. We note that whether an efficient search algorithm exists for other systems remains an open question.

Lemma IV.1 (OHA Search on Linear Systems)**.** *Searching for an OHA in a system with linear dynamics, control function and detector statistics can be formulated as a linear programming problem.*

Proof. Consider a discrete linear time-invariant (LTI) system as Equation (10).

$$f(\boldsymbol{x}_t, \boldsymbol{u}_t) := \boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{c} \tag{10}$$

where A and B are state and input matrix, representing how next state evolves with the current state and control input, and $\boldsymbol{c}$ is a drift of proper dimension. The system also has a linear function to compute control input from estimated states as

$$\boldsymbol{u}_t = D\tilde{\boldsymbol{x}}_t + \boldsymbol{e} \tag{11}$$

where D is a linear coefficient and $\boldsymbol{e}$ is an offset of proper dimension. So the system dynamics becomes a linear function of the actual and estimated states only, i.e.,

$$\hat{\boldsymbol{x}}_{t+1} = A\tilde{\boldsymbol{x}}_t + B(D\tilde{\boldsymbol{x}}_t + \boldsymbol{e}) + \boldsymbol{c} \tag{12}$$

Next, a linear detector function g means that the detector statistics constraint is in some linear form

$$g(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t) := F\tilde{\boldsymbol{x}}_t + G\hat{\boldsymbol{x}}_t - \boldsymbol{\tau} < 0 \tag{13}$$

where F and G are linear coefficients and $\boldsymbol{\tau}$ is the detector threshold of proper dimensions, usually a scalar. Combining everything together, we have the following optimization problem m:

$$\begin{aligned} &\text{minimize } dist^{(i)}(\boldsymbol{x}_c, U) \\ &\text{subject to } \bigwedge_{t=0}^{c} \big(\text{ (12)} \wedge \text{ (13)} \wedge \text{ (8)}\big) \end{aligned} \tag{14}$$

Notice that the objective function is linear because at any dimension i, either $dist^{(i)} = x[i]_c - xb[i]$ or $dist^{(i)} = xb[i] - x[i]_c$, and all the constraints are linear. Therefore this is a linear programming problem and there exist efficient solvers such as GLPK, MOSEK and CONELP [35]. □

The vehicle turning example in Section IV-B lies in this category. It has only one dimension, and the objective function for each problem is $xb - x_c$, with $xb = 2.7$, based on our definition of $dist^{(i)}$. Without loss of generality, we can consider an infinite-horizon, discrete-time linear–quadratic regulator (LQR) controller, i.e.

$$u_t = e - K(\tilde{x}_t - x^*). \tag{15}$$

where e is a constant reference control signal, x^* is a constant reference state, and K is the cost coefficient. Next, we can use CUSUM detector as in Section II-D1 with a threshold τ and a drift d. The attacker subtracts some value from sensor measurement to enlarge the actual system state. To inject more attack, the drift is smaller than the residual, and $S_t + |r_t| - b \leq 0$. The opposite situation can be analyzed in a similar way.

Consequently, in the vehicle turning system, we can call Algorithm 1 such that each optimization problem m is formulated as follows.

$$\begin{aligned} &\text{minimize } xb - x_c \\ &\text{subject to } \bigwedge_{t=0}^{c} \big(\hat{x}_{t+1} = (A' - B'K)\tilde{x}_t + B'e + B'Kx^*\big) \wedge \\ &\qquad \bigwedge_{t=0}^{c} \Big(\sum_{s=1}^{t} (\hat{x}_s - \tilde{x}_s - d) \leq \boldsymbol{\tau}\Big) \wedge \\ &\qquad \bigwedge_{t=0}^{c} \big(\underline{u} \leq e - K(\tilde{x}_t - x^*) \leq \overline{u}\big) \end{aligned}$$

where the three constraints correspond to system dynamics, hidden and control limit constraints.

We extend this conclusion for linear systems to convex systems in Theorem IV.2.

Lemma IV.2 (OHA Search on Convex Systems)**.** *Searching for an OHA in a system with convex dynamics, control function and detector statistics can be formulated as a convex optimization problem.*

Proof. The proof is similar as for Theorem IV.1, except for the system dynamics constraint and the hidden constraint are convex instead of linear. Each optimization problem m can be formulated as

$$\begin{aligned} &\text{minimize } (9) \\ &\text{subject to } \bigwedge_{t=0}^{c} \left(\hat{\boldsymbol{x}}_{t+1} = f_{cvx}(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t)\right) \wedge \\ &\qquad \bigwedge_{t=0}^{c} \left(g_{cvx}(\tilde{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_t) < 0 \wedge \right. \\ &\qquad \bigwedge_{t=0}^{c} (8) \end{aligned} \tag{16}$$

where f_{cvx} and g_{cvx} are convex functions. Notice that this is a convex optimization problem and there exists efficient solvers such as ECOS [36] and OSQP [37]. □

One final remark is that all these optimization problems are solved offline, before the system is running. Consequently, and thus, their solving time and needed computational resources can be considered not critical.

V. Conditionally Safe Region Searching

In Section IV, we find the OHA of a dimension i given the current state $\boldsymbol{x}_0$ for an observation period T. If this OHA cannot drive the system state to unsafe state set at that dimension within the period, we consider the current state is *conditionally safe*, i.e., the system will be safe within the observing time T if no alert is raised. This section presents our main contribution on how we efficiently find the conditionally safe region (CSR) with the knowledge of OHA only, regardless of the system dynamics or detector.

A. Conditionally Safe Region

We first formally define a CSR of a control system f at a dimension i within n-dimensional states.

Definition V.1 (Conditionally Safe Region (CSR))**.** *Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ be a control system of n-dimensional states. Its state space is $\mathcal{X} \subseteq \mathbb{R}^n$ and there is a pre-defined unsafe state set $U \subset \mathcal{X}$, the same way defined as in Equation (4). A conditionally safe region of dimension i is a set of states $CSR[i] \subseteq \mathcal{X}[i] - U[i]$, such that for all $x_0[i] \in CSR[i]$, there exists no $OHA[i]$ that can drive $x[i]$ into $U[i]$ in a period of time T.*

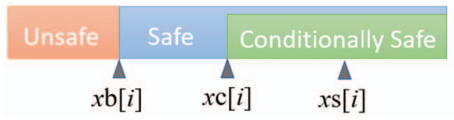


Fig. 3: Region demonstration for the i^{th} state. $xb[i]$ is the boundary between safe and unsafe states, $xs[i]$ is a known conditionally safe state, and $xc[i]$ is the boundary of conditionally safe region to be searched.

Fig. 3 demonstrates the relationship between unsafe states, safe states and CSR regions in the i^{th} dimension of the state space. Also, there is a clear boundary $\mathbf{xb}$ dividing states into unsafe region and safe region, and $xb[i]$ is the boundary for the i^{th} dimension of states. When we consider if the system remains safe within an observing window T in the near future, conditionally safe region comes in, and is marked as green. It is a subset of safe region, and there is also a clear boundary $xc[i]$ of conditionally safe state within T for the i^{th} dimension. For example, a car runs below 40 miles/hour cannot exceed a dangerous speed of 80 miles/hour within 2 seconds without breaking thrust limit.

B. Searching for Conditionally Safe Region

The next task is to search for CSR, and we conclude the following theorem.

Theorem V.1 (Efficient CSR Search)**.** *Given OHAs of each dimension i, starting at every discrete runtime t_a and lasting for period T of a control system (with known detector, control limits and unsafe set), we are able to efficiently compute CSR of that system, regardless of the system dynamics.*

Proof. We construct an algorithm that satisfies the requirements. The idea is to apply binary search on each state dimension to find the boundary of conditionally safe state region. The algorithm is shown as Algorithm 2. □

In summary, this algorithm searches the boundary of conditionally safe region $xc[i]$ given the boundary of unsafe state set $xb[i]$ and a known state in conditionally safe region $xs[i]$, which can be the i-th dimension of the reference state of a system. At Line 1, we initialize the xc using xs, so that the initial pivot is in the conditionally safe region. Then, we push the pivot towards the unsafe state region to find the boundary. Specifically, at Lines 1-3, we initialize the lower and upper bound of search range using $xs[i]$ and $xb[i]$. Then, at Lines 4-13, we search the boundary of conditionally safe region in a binary search manner. At Line 7, the function $isConditionallySafe$ requests the cached knowledge of computed from Algorithm 1 to check if the current pivot can be driven to unsafe set given an OHA. If not, the pivot is in the conditionally safe region and we continue to push the pivot towards the unsafe region.

Algorithm 2: Conditionally safe region searching

```
Input: $\boldsymbol{xb}, \boldsymbol{xs} \in \mathbb{R}^n$
/* $n$: number of state dimensions.      */
/* $i$: dimension to find boundary.      */
/* $xb[i]$: the boundary between safe and
   unsafe states of dimension $i$.       */
/* $xs[i]$: an existing state in
   conditionally safe region of
   dimension $i$.                        */
Output: $xc[i] \in \mathbb{R}$
/* $xc[i]$: the boundary of the
   conditionally safe region of
   dimension $i$                         */
$xc[i] \leftarrow xs[i]$          // init. $xc[i]$ using $xs[i]$
$lo \leftarrow xs[i]$, $hi \leftarrow xb[i]$          // init. range
$lcs \leftarrow xs[i]$     // last conditionally safe
do
    $r \leftarrow hi - lo$                // search range
    $xc[i] \leftarrow lo + r/2$                   // pivot
    if isConditionallySafe($xc[i]$) then
        $lo \leftarrow xc[i]$           // towards unsafe
        $lcs \leftarrow lo$      // keep last con. safe
    else
        $hi \leftarrow xc[i]$              // towards safe
    $xc[i] \leftarrow lcs$        // conditionally safe
while $r > v_{max}$           // term. conditions
```

Algorithm 2 outputs the CSR of dimension i from the boundary computed. For all dimensions, we call this algorithm and the final CSR is the Cartesian product

$$CSR = CSR[1] \times CSR[2] \times \cdots \times CSR[n] \tag{17}$$

C. Time Complexity

The proposed method has two phases: Algorithm 1&2 generates a CSR reference table offline, and the detector queries the table online. The offline phase solves $O(nTlog1/\epsilon)$ times of LP, where n, T, and ϵ are the number of dimensions of system state, user-given time horizon, and precision term in Algorithm 2, respectively, which is pseudo-polynomial.

VI. EVALUATION

In this section, we validate the theoretical analysis results above with four linear simulators of CPS and provide detailed experimental result analysis.

A. Simulation Setting

1) Experimental Setting: The experiments were implemented on a PC with 32GB memory and an Intel(R) Core(TM) i7-10700KF 3.80GHz CPU. All the optimization results were produced by the GLPK solver and CVXOPT library [38] in python.

2) Simulators: The proposed analysis is performed on vehicle turning, DC motor position, RLC circuit and quadrotor. We can obtain the linear ordinary difference equation (ODE) by system identification from a real system.

Vehicle Turning. The vehicle turning simulator models a vehicle's steering behaviour, which changes the voltage difference of two motors u to adjust their speed difference x. The ODE of this system can be found in [9]. **RLC circuit.** The RLC circuit formed by a resistor, an inductor, and a capacitor connected in series controlled by the voltage source u. $x[1]$ represents the voltage across the capacitor and $x[2]$ denotes the current in the circuit. The system dynamics can be found in [39]. **DC motor Position** DC motor position simulator models a motor shaft behaviour, which change the voltage of the motor u to keep the rotation angle of the motor shaft $x[1]$, we also have another state $x[2]$ which is the rotary angular velocity. The details of system model for DC motor position can be found in [40]. **Quadrotor** This simulator shows how the altitude is affected by the thrust force. This benchmark has a total of 12 states: (x, y, z) and (ϕ, θ, ψ) denote the linear and the angular positions, (u, v, w) and (p, q, r) are the linear and angular velocities. The ODE can be found in [41].

3) Attacks and detector settings: Attack happened at 8 seconds for vehicle turning, RLC and quadrotor, and each time step is 0.02 seconds, and for DC motor position it happens at 100 seconds and each time step is 0.2 seconds, the future horizon is 100 steps. The CUSUM attack threshold was set to 5, drift was set to 0. For geometric attack, the parameter α was set to 0.75, and the β parameter was set to 0.85 for vehicle turning. And the α and β was set to 0.7 and 0.8 for DC, RLC and quadrotor. The upper bound and lower bound of the control inputs for vehicle turning are $[-1, 2.35]$, and the upper bound of control inputs for RLC was set to 4.6, the upper bound of control inputs for DC was set to 1.66, the upper bound of the control inputs for quadrotor was set to 6.95. The setting of the LQR controller Q and R matrix is trivial, both of them are set to identity, for quadrotor benchmark, the gain matrix is obtained by decouple methods adapted from [41].

4) Baseline Hidden Sensor Attacks: We consider frequently used hidden attacks from [24], [27] as our baselines. The observing window is T, the initial attack time is t_1, and the final attack time should be $t_1 + T$.

Surge attack Surge attacks maximize the statistic score as soon as possible, and then keep the statistic score at the threshold level, i.e., $\boldsymbol{s} = \boldsymbol{\tau}$, after the statistic score reaches the threshold. A greedy surge attack is given in [24], and the attack is given as follow:

$$\tilde{\boldsymbol{x}}_t = \begin{cases} \boldsymbol{x}_t \pm (\boldsymbol{\tau} + \boldsymbol{b} - \boldsymbol{s}_{t-1}) & \text{if } t = t_1 \\ \boldsymbol{x}_t \pm \boldsymbol{b} & \text{otherwise} \end{cases}$$

The statistic score $\boldsymbol{S}$ reaches the threshold at the first step, and then offset the predicted state by a drift $\boldsymbol{b}$ at the following steps.

Bias attack A bias attack that try to falsify the system measurements discretely by adding small perturbations over a period of time. Without loss of generality, the perturbation was unified for each step and can be formed as follow [27]:

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t \pm (\boldsymbol{\tau}/T + \boldsymbol{b})$$

where T is the observing window size.

Geometric attack A geometric attack will attack the system smoothly at the beginning and surge the attack at the last several steps [27]. To be noticed, there are 2 hyper parameters α and β in the design of the geometric attack and we can have infinite number of combinations of α and β. The design of geometric attack is given as follow [27]:

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{x}_t \pm (\beta\alpha^{T-t} + \boldsymbol{b})$$

subject to

$$\sum_{t=1}^{T} \beta\alpha^{T-t} - n\boldsymbol{b} = \boldsymbol{\tau} \tag{18}$$

Any combination of α and β satisfied equation 18 was a feasible geometric attack.

B. Long-memory Detector Attack Effect

Fig. 4 showed the system states that have been attacked by the optimal attacks generated by Algorithm 1. There are three important observations to be noticed: First, the optimal attack always caused minimum distance to unsafe set compared to the three baseline hidden attacks on the four systems. It is obvious the maximum of the orange curve is greater than that of other curves. Secondly, the optimal hidden attack could have same distance to unsafe set as other hidden attacks. In other words, the optimal hidden attack generated by Algorithm 1 could be identical to the three baseline hidden attacks on some settings. We can see the optimal hidden attack has the same pattern as surge attack on DC Motor system. Thirdly, the trend of the attacked state by optimal hidden attack may not be monopoly. For example, the vehicle turning example showed a two-step trajectory before reaching the maximum, our optimization problem has no constraints about the monotonous property. Because the unsafe set for vehicle turning is a halfspace $(1.26, \infty)$, therefore, since the OHA causes the maximum deviation, it has the minimal distance to the unsafe region.

C. Short-Memory Detector Attack Effect

In this subsection, to evaluate Algorithm 1, we show the optimal hidden attacks for detectors with various detection window size. The observations are similar to that from long-memory detectors. Cross each column in Fig. 5, we can see the window size does not affect one fact: Our optimal hidden attack always has the maximum deviation. Another observation is the trajectory of system under short-memory detector protection attacked by hidden attack may have multiple peaks as shown in the vehicle turning column. We only care about the highest peak from all the peaks according to our problem statement in section IV. Another observation to be noticed is the window size does not affect the maximum deviation too much. Compared the different size detection window results of the four systems, we can see the maximum deviation of them are not far from each other. Furthermore, there might be some slight errors come from the solver and precision. For example, on the DC motor column, we see the surge attack trajectory is slightly over the optimal hidden attack trajectory of window size 50 and 75. The state reaches the minimal distance at the last step for the 4 systems with every listed window size.

D. Earliest Attack

In Algorithm 1, we compute the OHA in a loop of every step in the future. In other words, at each step, the distance from attacked state to the unsafe set is the objective of the optimization problem. Therefore, we can guarantee our OHA is the earliest one from the solutions found at each step in the loop. Fig.7 showed the results of OHA and sub-optimal attacks on the systems using long-memory detectors, there are two observations to be noticed. The plot for Vehicle Turning clearly demonstrates that the optimal attack reaches the maximum devaition before suboptimal attacks. It is the fastest attack. In the plot for RLC, the suboptimal attacks are on a tracjectory to the maximum devaition but will only reach them after the simulation time. Thus, here too, we see that the optimal attack which reaches the maximum deviation just at the end of the simulation is the fastest attack. In the DC motor and quadrotor, the optimal and suboptimal attacks behave similarly. First, there might be several sub-optimal attacks that can reach the maximum deviation, but the OHA has the earliest time to reach it compared to other sub-optimal attacks. We can see from the vehicle turning results, the OHA reach maximum deviation at the 22nd time step, and sub-optimal attacks reach it at 24th, 26th, and 28th time step. Secondly, it is possible that the OHA will reach maximum deviation at the last step, then other sub-optimal may not reach maximum deviation as showed in the RLC, DC motor and quadrotor results.

E. Conditionally Safe Region

Fig. 6 showed the conditionally safe region searching process. The dashed line showed the unsafe boundary for each system, for vehicle turning, the unsafe set is $x[1] \in (1.26, \infty)$. Similarly, the unsafe set of RLC circuit is $x[1] \in (4.17, \infty)$, the unsafe set for DC motor is $x[1] \in (4.56, \infty)$, $x[-1] \in (21.94, \infty)$the unsafe set for quadrotor is . And CSR for vehicle turning is $x[1] \in (-\infty, 1]$, CSR for RLC circuit is $x[1] \in (-\infty, 3]$, CSR for DC motor is $x[1] \in (-\infty, \frac{\pi}{2}]$, CSR for quadrotor is $x[-1] \in (-\infty, 1.5]$. The colored regions were the unsafe, safe and conditional safe region on the corresponding dimension of the four systems. And the colors are refer to that in Fig. 3. There are three observations to be noticed: First, if the start state is out of CSR, there exists at least one OHA which can drive the system to touch unsafe. This could be seen from the red curves in Fig. 6. Secondly, if the start state is in CSR, there is no attack that can drive the system to unsafe in the given future without noticed by the detector. Thirdly, OHA with different start states can reach the maximum deviation at the same time step, but the trajectories are not parallel due to the system dynamics. For example, all the 3 OHAs in vehicle turning example reach the maximum

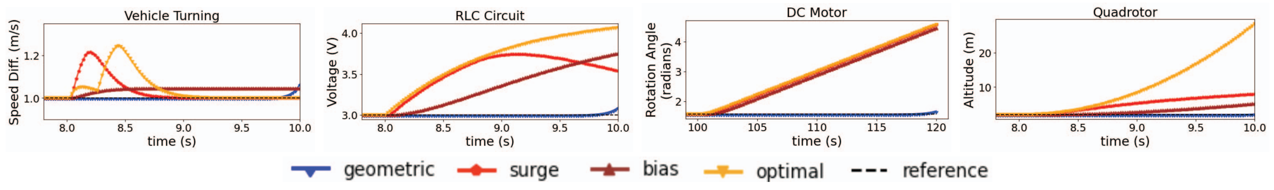


Fig. 4: Plot of attacked state against time for Vehicle Turning (left), RLC (second to the left), DC Motor (second to the right) and Quadrotor(right) systems using long memory detectors.

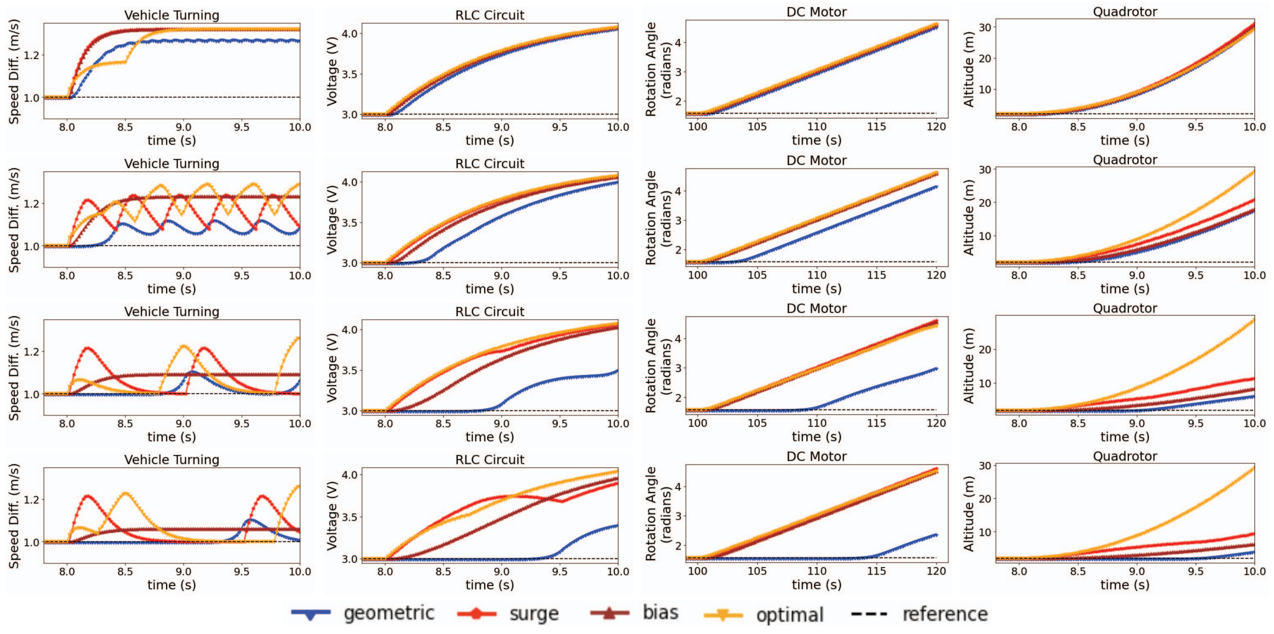


Fig. 5: Plot of attacked state against time for Vehicle Turning (left), RLC (second to the left), DC Motor (second to the right) and Quadrotor(right) systems using short memory detectors of window sizes 5 (first row), 20 (second row), 50 (third row), and 75 (last row).

deviation at 22nd time step. And the OHAs for RLC, DC motor and quadrotor reach it at the last time step. For example, when the vehicle turning system is running in real-time, when the speed difference is over 1, there will be a yellow alert raised, if the attacker inject OHA to our system, after 22 time steps, the system will be drive to unsafe. In other words, the system is notified 22 time steps ahead.

F. Scalability Analysis

In this subsection, we examine the scalability of our method in terms of three parameters: (1) the number of system states, (2) the precision and (3) the time horizon. The number of states of the 4 benchmarks are $1, 2, 3$ and 12, the horizon range is $[20, 200]$ and the precision term (lower means higher precision) range is $[0.009, 0.5]$, with lower value means higher precision. Long represents the long-memory detector, and the numbers $5, 20, 50, 75$ represent the window size of the short-memory detectors. It is possible to have a longer time horizon, but the current range is sufficient to show a trend .

Since querying the table takes $O(1)$ time which is trivial, we focus on the offline computation time in this subsection. We run the experiments on the 4 benchmarks on the same settings. The results are shown in Table II which contains 2 parts: The upper part is the running time to find a single OHA, and the bottom part shows the running time to find the CSR. For consistency, we set the time horizon as 100, which is on the same settings for the experiments in the section VI. There are 4 main observations from Table II:

1) The offline running time of our method is acceptable for large systems. The highest running time for each benchmark was highlighted. For example, for the quadrotor benchmark which has 12 states with 100-step horizon and 0.009 precision, the running time is around 80-90 seconds to find the CSR for one dimension which is reasonably short for offline computation. And the running time is around 6-9 seconds to find an OHA with the same setting.
2) The searching time for CSR increases with higher precision, since there might be more iterations in the

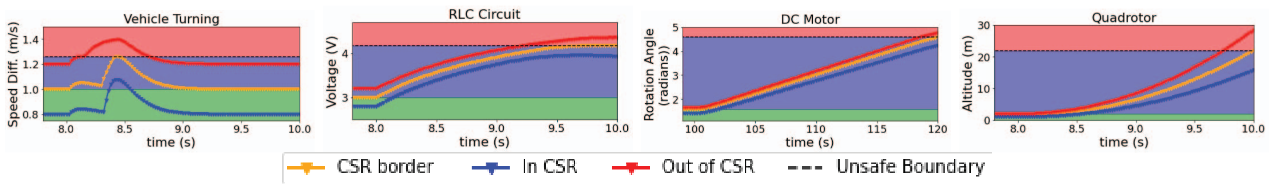


Fig. 6: Plot of CSR searching process for Vehicle Turning (left), RLC (second to the left), DC Motor (second to the right) and Quadrotor(right) systems using long memory detectors. The red region is the unsafe set on the shown dimension, the green region is the CSR on the shown dimension, the blue region is the safe region of the shown dimension

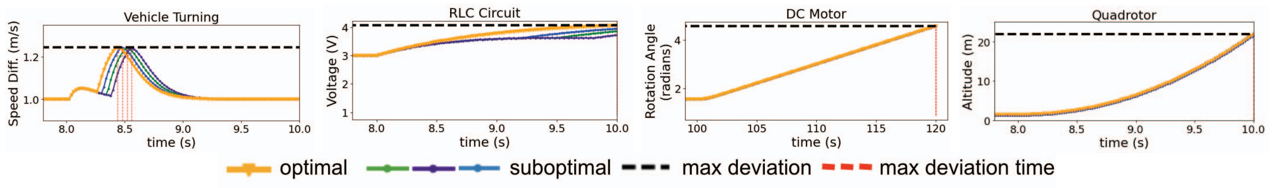


Fig. 7: Plot of optimal attack (also fastest attack) alongside suboptimal (slower) attacks against time for Vehicle Turning (left), RLC (second to the left), DC Motor (second to the right) and Quadrotor(right) systems using long memory detectors.

	VT					RLC					DC					QD				
Horizon	Long	5	20	50	75	Long	5	20	50	75	Long	5	20	50	75	Long	5	20	50	75
20	0.34	0.32	0.33	0.34	0.32	0.38	0.38	0.41	0.37	0.38	0.38	0.36	0.37	0.36	0.36	0.26	0.27	0.26	0.26	0.26
50	0.36	0.31	0.33	0.35	0.34	0.55	0.51	0.50	0.54	0.53	0.57	0.57	0.51	0.55	0.56	0.84	0.75	0.83	0.85	0.87
100	0.36	0.27	0.43	0.40	0.43	1.01	0.76	0.74	0.90	1.17	1.74	0.99	1.06	1.91	2.29	6.19	6.69	6.96	7.89	8.64
200	2.96	1.45	1.56	2.33	**3.31**	**12.8**	2.29	2.41	2.88	9.58	**23.0**	4.72	5.00	14.40	14.14	**65.52**	46.35	50.11	54.21	61.10
T=100	VT					RLC					DC					QD				
Precision	Long	5	20	50	75	Long	5	20	50	75	Long	5	20	50	75	Long	5	20	50	75
0.5	0.34	0.20	0.25	0.41	0.37	0.79	0.34	0.38	0.45	0.58	4.59	1.62	1.80	4.26	2.87	6.58	6.82	6.02	6.89	6.01
0.1	0.34	0.20	0.24	**0.44**	0.36	2.01	1.01	1.07	1.45	2.06	10.70	1.61	1.81	4.22	2.89	6.65	6.96	6.37	6.95	6.34
0.09	0.34	0.20	0.23	0.39	0.36	2.06	0.92	0.99	1.32	1.92	10.53	4.03	4.40	10.48	22.08	6.59	6.87	6.74	6.82	6.42
0.05	0.34	0.21	0.25	0.41	0.37	2.08	1.03	1.04	1.35	2.14	10.58	3.59	3.99	9.89	22.05	33.26	24.59	26.88	29.61	30.45
0.02	0.34	0.21	0.25	0.41	0.38	2.09	1.02	1.12	1.45	2.17	13.49	4.69	5.12	13.32	**22.10**	58.97	48.98	58.85	56.74	59.93
0.009	0.34	0.21	0.25	0.41	0.38	2.17	1.09	1.20	1.53	**2.19**	19.63	4.76	5.13	12.33	22.08	85.08	77.39	79.21	71.98	**86.48**

TABLE II: Offline Running Time analysis results

main loop of Algorithm 2. However, a higher precision does not necessarily mean a drop in time. For example, the vehicle turning benchmark shows that no additional iterations is required even with a change in precision, because it achieves a small error in the first iteration.

3) Finding an OHA costs more time with a larger number of system states. The time increment comes from the solving time of the each optimization problem since the problem size is increased. Comparing the single-dimension vehicle turning benchmark and 12-dimension quadrotor benchmark, the running time of finding an OHA with a long-memory detector and 200-step horizon, quadrotor benchmark's running time is about 22 times to that of the vehicle turning benchmark.
4) When the horizon is getting longer, finding an OHA costs more time. Referring to Algorithm 1, the time increment mainly comes from 2 parts: the number of optimization problems solved and solving time for each optimization problem. Finding an OHA will need to solve T number of optimization problems as Algorithm 1 line 2 shown. Also, each optimization problem takes more time to solve since the number of variables increases if the horizon is getting longer as Equation 9 shown. The size of the optimization problem does not change no matter what kind of detectors are applied, but the optimization solving time may vary.

The above observations from Table II point out that our method is scalable, it is possible to be deployed on larger system if there is a need. Furthermore, our method is capable to find CSR with different levels of precision for users' needs.

VII. Discussion

In this section, we will discuss the challenges when applying our method to non-linear systems. The discussion is from 3 perspectives: convexity, sparsity and the exponential explosion of the objective. In the end, a possible solution to handle these challenges will be discussed.

A. *Sparsity*

If the optimization problem is linear, the solver does not suffer from the sparsity since we can use simplex method to get the solutions efficiently [42]. Since we have three kinds of constraints in our optimization problem, a system dynamics constraint or a control limits constraint may only depend on a few variables which make the optimization problem sparse. Usually, interior-points are implemented in non-linear optimization solvers to solve large, sparse optimization problems [43]. However, there is a big problem: During the optimization, implementing the interior-points method requires the Newton 's method to find the solution. Since we have no guarantee for the convexity of the problem, this method may not produce an optimal attack.

B. Convexity

In section IV, we stated that the method is designed for convex problems. Unfortunately, it is hard to find a convex optimization problem for non-linear systems. For example, if there are trigonometric functions in the system dynamics, which is common if the system state is related to some angle or angular velocity etc., then the problem is non-convex since the second-order partial derivatives of the original function is still trigonometric and cannot be non-negative. If the system dynamics contains cubic or quadratic terms, the system is not guaranteed to be convex, either. For example, consider a non-linear system is as follows:

$$x_{n+1}[1] = x_n[2] + 0.1u$$
$$x_{n+1}[2] = -x_n[1] + \frac{1}{3}x_n[1]^3 + x_n[2] + 0.1u$$

There is a cubic term in the system model, so the system is not guaranteed to be convex. If the optimization problem is convex, the Hessian matrix should be positive semi-definite. If the problem is non-convex, then there is no guarantee that our method find the OHA since the solution might be a local optimum instead of global. Then there is no guarantee that the corresponding CSR is conditional safe.

C. Objective Explosion

The easiest non-linear optimization problem might be a quadratic optimization problem. However, it is impossible to have a quadratic optimization problem for a non-linear system if the proposed framework was applied. By definition, quadratic programming is trying to optimize the quadratic objective function subject to linear constraints. If the system is non-linear, the system dynamics constraints and hidden constraints are no longer linear. Additionally, according to Algorithm 1, the objective is searching for the minimum distance to the unsafe region in the observation time T, and the search space grows exponentially with number of time steps. For example, considering a trivial non-linear system that has a single state and single control input as follows.

$$x_{n+1} = x_n^2 + u$$

Therefore, when we are looking for the optimal solution at the last several steps in the observation period, the exponents of a few terms in the objective will become incredibly large since we have a square in the system dynamics. It is studied that deciding the non-negativity of a polynomial is NP-hard if the highest order is greater than 3. This is because the sum of squares property (SOS) is not satisfied, such that there is no guarantee that the optimal solution can be find in polynomial time [44]. Therefore, we are not able to find a solution efficiently.

D. Possible Solution

Though we face the challenges discussed above, there exist some possible solutions to apply our method to non-linear systems. An intuitive method is linearizing a non-linear system around its equilibrium points. Assuming a non-linear system have three equilibrium points, then it is possible to have three linearized the systems for the region around each equilibrium points. For example, the trigonometric functions in the system dynamics could be linearized when the system states are closed to the equilibrium points. Then the non-linear optimization problem was trivialized to three linear optimization subproblems. Therefore, the above three challenges are avoided. The approach seems promising, but raises several interesting research questions. In particular, for some system states between equilibrium points, different choices of the subproblem to solve may lead to different OHA results. We will explore this approach in our future works.

In summary, our proposed method is a general framework to secure CPS from hidden sensor attacks. We also identified several challenges that need to be solved before applying it to non-linear CPS.

VIII. Conclusion

In this paper, we proposed a framework for hidden attacks. Moreover, we proposed a definition of optimal hidden attacks and an algorithm to find OHA. Furthermore, based on the OHA, we proposed an algorithm to find CSR offline. We evaluated our methods on four linear systems with various window size for detectors. The experimental results show that the OHA can achieve minimal distance to a pre-defined unsafe set, and the CSR can notify the system in advance. We also discussed the challenges of applying our method to non-linear systems in terms of sparsity, convexity and objective explosion. For future work, there are plenty of extension of this paper. First, it is interesting to implement the linearization solution for various non-linear systems. Secondly, it is possible to solve more general problems if the method can be extended to more complex unsafe sets such as strips or zonotopes. Thirdly, another extension is to adapt our framework to actuator attacks, which are also common in CPS.

IX. Acknowledgement

We would like to thank the anonymous reviewers and the anonymous shepherd for constructive comments and being with us along the revision process. This research was supported in part by NSF 2143256, NSF 2143274 and ONR N00014-20-1-2744. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF), Office of Naval Research (ONR), the Department of Defense, or the United States Government.

REFERENCES

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference (DAC)*. IEEE, 2010, pp. 731–736.

[2] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[3] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.

[4] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *The 28th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2008, pp. 495–500.

[5] M. Wolf and D. Serpanos, "Safety and security in cyber-physical systems and internet-of-things systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 9–20, 2017.

[6] A. M. Wyglinski, X. Huang, T. Padir, L. Lai, T. R. Eisenbarth, and K. Venkatasubramanian, "Security of autonomous systems employing embedded computing and sensors," *IEEE micro*, 2013.

[7] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time recovery for cyber-physical systems using linear approximations," in *41st IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020.

[8] Y. Zhang and K. Rasmussen, "Detection of electromagnetic interference attacks on sensor systems," in *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[9] 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.

[10] F. Kong, O. Sokolsky, J. Weimer, and I. Lee, "State consistencies for cyber-physical system recovery," in *Workshop on Cyber-Physical Systems Security and Resilience (CPS-SR)*, 2019.

[11] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[12] A. H. Rutkin, "spoofers use fake gps signals to knock a yacht off course," MIT Technology Review, 2013, online; accessed May 2020.

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

[14] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," in *57th Design Automation Conference*. ACM, 2020.

[15] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 130–139.

[16] A. Ganesan, J. Rao, and K. Shin, "Exploiting consistency among heterogeneous sensors for vehicle anomaly detection," SAE Technical Paper, Tech. Rep., 2017.

[17] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 92–98.

[18] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[19] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Design & Test*, vol. 34, no. 4, pp. 7–17, 2017.

[20] 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.

[21] C. Murguia and J. Ruths, "Cusum and chi-squared attack detection of compromised sensors," in *2016 IEEE Conference on Control Applications (CCA)*, 2016, pp. 474–480.

[22] C. Kwon, W. Liu, and I. Hwang, "Security analysis for cyber-physical systems against stealthy deception attacks," in *2013 American control conference*. IEEE, 2013, pp. 3344–3349.

[23] C. Murguia and J. Ruths, "Cusum and chi-squared attack detection of compromised sensors," in *2016 IEEE Conference on Control Applications (CCA)*. IEEE, 2016, pp. 474–480.

[24] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1092–1105.

[25] R. Tunga, C. Murguia, and J. Ruths, "Tuning windowed chi-squared detectors for sensor attacks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1752–1757.

[26] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–33, 2011.

[27] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*, 2011, pp. 355–366.

[28] Y. Mo and B. Sinopoli, "On the performance degradation of cyber-physical systems under stealthy integrity attacks," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2618–2624, 2015.

[29] C. Murguia and J. Ruths, "On reachable sets of hidden cps sensor attacks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 178–184.

[30] N. Hashemi, C. Murguia, and J. Ruths, "A comparison of stealthy sensor attacks on control systems," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 973–979.

[31] N. Hashemi and J. Ruths, "Gain design via lmis to minimize the impact of stealthy attacks," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1274–1279.

[32] A. A. Cárdenas, S. Radosavac, and J. S. Baras, "Evaluation of detection algorithms for mac layer misbehavior: Theory and experiments," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 605–617, 2008.

[33] C. Murguia and J. Ruths, "Characterization of a cusum model-based sensor attack detector," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 1303–1309.

[34] P. Luo, T. A. DeVol, and J. L. Sharp, "Cusum analyses of time-interval data for online radiation monitoring," *Health physics*, vol. 102, no. 6, pp. 637–645, 2012.

[35] B. Meindl and M. Templ, "Analysis of commercial and free and open source solvers for the cell suppression problem." *Trans. Data Priv.*, vol. 6, no. 2, pp. 147–159, 2013.

[36] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An socp solver for embedded systems," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 3071–3076.

[37] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[38] M. Andersen, J. Dahl, Z. Liu, L. Vandenberghe, S. Sra, S. Nowozin, and S. Wright, "Interior-point methods for large-scale cone programming," *Optimization for machine learning*, vol. 5583, 2011.

[39] K. Astrom and R. Murray, "Feedback systems-an introduction for scientists and engineers, version v 2.10 c," 2010.

[40] K. Tan and Y. Li, "Performance-based control system design automation via evolutionary computing," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 4, pp. 473–486, 2001.

[41] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," Master's thesis, KTH Royal Institute of Technology, 2015.

[42] R. H. Bartels and G. H. Golub, "The simplex method of linear programming using lu decomposition," *Communications of the ACM*, vol. 12, no. 5, pp. 266–268, 1969.

[43] L. Lukšan, C. Matonoha, and J. Vlček, "Interior-point method for non-linear non-convex optimization," *Numerical linear algebra with applications*, vol. 11, no. 5-6, pp. 431–453, 2004.

[44] D. Bertsimas and I. Popescu, "Optimal inequalities in probability theory: A convex optimization approach," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 780–804, 2005.