Randomness-based Detection and Recovery for Resilient Autonomous Robot Operations

---

A

Doctoral Dissertation

Presented to

the Faculty of the School of Engineering and Applied Sciences

The University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Paul J Bonczek

2022

APPROVAL SHEET

This
Dissertation
is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Paul J Bonczek

This Dissertation has been read and approved by the examining committee:

Advisor:	Nicola Bezzo
Committee Member:	Zongli Lin
Committee Member:	Laura Barnes
Committee Member:	Lu Feng
Committee Member:	Carlo Pinciroli

Randomness-based Detection and Recovery for Resilient Autonomous Robot Operations

by

Paul J Bonczek

M.E., Electrical Engineering
The University of Virginia, 2021

B.S., Electrical Engineering and Applied Mathematics
(*double major)*
State University of New York Polytechnic Institute, 2016

Abstract

Today's autonomous systems, such as unmanned ground and aerial vehicles, are becoming integral in various aspects of our daily lives. With the help of technological advancements in sensing, actuation, communication, and computation, modern robots are capable of many civilian and military applications with minimal to no human supervision. However, with these increases in capabilities comes an increased risk of security vulnerabilities to cyber attacks or system faults that induce undesired system behavior. Thus, it is crucial to provide improved detection and recovery measures to ensure proper performance and safety. The objective of an attacker is typically to implement stealthy attack sequences to manipulate the system of interest, all while attempting to remain undetected. Although traditional attack detection techniques have been shown to be effective in mitigating attacks for resilient operation, they are susceptible to being fooled by intelligent attackers that are able to hide within the system noise profile. Furthermore, cyber attacks and/or faults that occur on a single vehicle within multi-agent systems can potentially compromise the performance of all vehicles, resulting in the failure of the multi-agent system from accomplishing an operation. To counteract these undesirable scenarios, the utilization of recovery frameworks in single- and multi-agent systems can ensure the safety of all agents while still being able to resiliently accomplish tasks.

This dissertation focuses on increasing the state-of-the-art attack detection capabilities on autonomous single- and multi-robot systems to discover previously undetectable stealthy attacks on the sensor and communication infrastructure. To this end, we present novel runtime randomness-based detection techniques to identify stealthy falsified sensor measurements and received information over communication channels that intentionally hide within system noise profiles, but leaves traces of non-random, inconsistent behavior. Additionally, we introduce an approach utilizing such randomness-based techniques to covertly pass safety-critical information within a robotic swarm through hidden signatures without the need to explicitly broadcast this information between robots. We also highlight two cooperative recovery frameworks within multi-robot systems to aid in re-localization of compromised robots that suffer from attacks or faults to critical positioning sensors within unknown or landmark-free environments. Lastly, we present a detection and recovery framework for individual autonomous systems that suffer from spoofed or faulty on-board controllers, which can drive the vehicle to undesired locations in the environment.

These techniques are validated through extensive simulations and proof-of-concept lab experiment case studies with single and cooperative unmanned ground vehicles. Current and future work includes the development of cooperative frameworks for teams of robots to defend safety-critical regions of the environment from malicious intruders.

"If everyone is thinking alike, then somebody isn't thinking."
– General George S. Patton

Acknowledgements

Not only has my research taught me much about on maintaining resiliency of autonomous systems, but also my PhD studies have taught me much about resilience and perseverance on a personal level. To develop these critical skills of resiliency and perseverance, I want to express my deepest gratitude to Nicola Bezzo as my advisor for his mentorship and guidance throughout my time at the University of Virginia. Nicola has helped with my extensive growth in numerous areas that will enable me to have a successful career beyond my academic studies. His passion for the field of robotics and commitment to always putting forth full effort to be better at his craft has provided me with a baseline to become the researcher and person that I am today. I will always be thankful for the five years that I have spent during his guidance.

I would like to thank my dissertation committee members: Zongli Lin, Laura Barnes, Lu Feng, and Carlo Pinciroli for insightful and constructive feedback about my research as well as their support during the final stages of my PhD studies. I also thank Gang Tao, Farzad Farnoud Hassanzadeh, and Ronald Williams for serving as a committee member for my qualification exam in 2018.

I appreciate the support from my former professors at SUNY Polytechnic Institute, namely Timothy Busch, Daniel Benincasa, and Michael Medley, who taught me the fundamentals of electrical engineering which propelled me to begin my graduate school studies. Furthermore, I am sincerely grateful to Bryant Wysocki at the Air Force Research Laboratory in Rome, New York for opening my eyes to graduate school. Bryant's words and actions about research and graduate school deeply influenced my decision to continue my education after finishing my undergraduate degree, where he taught me that PhD research can be a fun, engaging, and fulfilling experience that is not beyond my reach.

I would like express my gratitude to current and former members of the AMR Lab – Esen, Tony, Rahul, Shijie, Carmelo, Jacob, Phil, Lauren, Nick, Will, Patrick – for being excellent lab mates and friends over the years. We have had many wonderful conversations and times together, which I will always cherish.

To the wonderful doctors at the University of Virginia hospital who helped me heal from my medical ailments during my time at UVA. In particular, I want to thank Anne Tuskey and Sook Hoang for treating me in order to get back to normal again.

Last but not the least, I would like to thank my parents Michael and Linda and my brother Andrew for their persistent understanding, support, help, and love over the years. They have always been there for me during the highs and lows in life. Without them, I would not be the person that I have become.

To fund my PhD research, I acknowledge: the National Science Foundation (NSF) through grant #1816591, the Office of Naval Research (ONR) under agreement number N000141712012, and the Defense Advanced Research Projects Agency (DARPA) under Contract FA8750-18-C-0090.

Contents

List of Figures

List of Tables

List of Abbreviations

ARL	Average Run Length
CPS	Cyber-Physical System
CUSIGN	Cumulative Sign
CUSUM	Cumulative Sum
EKF	Extended Kalman Filter
FOV	Field of View
GG	Gabriel Graph
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KF	Kalman Filter
LOS	Line of Sight
LTI	Linear Time-Invariant
MAS	Multi-Agent System
MITM	Man-in-the-middle
MRS	Multi-Robot System
PF	Particle Filter
RF	Radio Frequency
RSSI	Received Signal Strength Indication
SCADA	Supervisory Control and Data Acquisition
UGV	Unmanned Ground Vehicle
VSDM	Virtual Damper-Spring Mesh

Notation

Vectors

$\boldsymbol{x}^\mathsf{T}$	Transpose of a vector $\boldsymbol{x}$
x_i	The ith element of a vector $\boldsymbol{x}$
$\boldsymbol{x}_{[i:j]}$	The elements $[x_i, x_{i+1}, \ldots, x_{j-1}, x_j]^\mathsf{T}$ of vector $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	Euclidean norm of a vector $\boldsymbol{x}$
$\mathbf{1}_n$	An n–dimensional vector with all elements equal to 1
$\mathbf{0}_n$	An n–dimensional vector with all elements equal to 0

Matrices

$\boldsymbol{A}^\mathsf{T}$	Transpose of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^\dagger$	Moore-Penrose pseudoinverse of a matrix $\boldsymbol{A}$
$\boldsymbol{A} > 0$	Positive definite matrix $\boldsymbol{A}$
$\boldsymbol{A} \geq 0$	Positive semi-definite matrix $\boldsymbol{A}$
$\boldsymbol{I}_n$	An $(n \times n)$–dimensional identity matrix
A_{ij}	The element in the ith row and jth column of a matrix $\boldsymbol{A}$
$\rho(\boldsymbol{A})$	Spectral radius of a matrix $\boldsymbol{A}$
$\mathbf{1}_{n\times m}$	An $(n \times m)$–dimensional matrix with all elements equal to 1
$\mathbf{0}_{n\times m}$	An $(n \times m)$–dimensional matrix with all elements equal to 0

Sets

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of real n–dimensional vectors
$\mathbb{R}^{n \times m}$	Set of real $(n \times m)$–dimensional matrices
$\mathbb{R}_{>a}$	Set of real numbers greater than $a \in \mathbb{R}$
$\mathbb{R}_{\geq a}$	Set of real numbers greater than or equal to $a \in \mathbb{R}$
$\mathbb{N}$	Set of natural numbers (i.e., positive integers)

Miscellaneous

$\mathbb{E}[x]$	Expected value of a random variable x
$\text{Var}[x]$	Variance of a random variable x
$\Pr[x]$	Probability of an event x occurring
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\chi^2(n)$	Chi-square distribution with n degrees of freedom
$\mathcal{VG}(a, b, c, d)$	Variance-gamma distribution with parameters: location, spread, asymmetry, shape
$\mathcal{B}(a, b)$	Binomial distribution with probability of success a and number of trials b
$\Phi(\cdot)$	Cumulative distribution of a Gaussian distribution
$\Phi^{-1}(\cdot)$	Inverse cumulative distribution of a Gaussian distribution

Chapter 1

Introduction

Today's autonomous cyber-physical systems (CPSs) like unmanned aerial and ground mobile robots have become integral in various aspects of our daily lives. With the help of technological advancements, modern robots are fitted with multiple on-board sensors and computers that make them capable of many civilian and military applications with minimal to no human supervision. Applications such as automated navigation [58], surveillance [104], search and rescue [96], and task oriented jobs [19] are becoming more common and ready for deployment in real world applications, especially in the automotive, industrial, and military domains. These various enhancements in autonomy are possible thanks to the tight interaction between computation, sensing, communications, and actuation that characterize CPSs. With these increases in capabilities comes however an increased risk of security vulnerabilities to cyber attacks with the intent to induce undesired system behavior. Thus, it is crucial to provide tighter security measures to ensure proper performance and safety.

A successful attacker is able to implement a stealthy cyber attack sequence to manipulate the system of interest, all while remaining undetected from attack detection schemes. The execution of such a stealthy attack allows the successful attacker to degrade system performance, and potentially cause damage, to the unknowingly compromised system. For example, in the motivational Figure 1.1, an autonomous unmanned ground vehicle (UGV) experiences a stealthy sensor attack that intentionally hides within system noise characteristics to slowly drive the UGV away from its intended path. Previously demonstrated attacks of this nature include cases like: the Global Positioning System (GPS) spoofing of a vessel [8], sensor and communication attacks on vehicle technologies [66], infiltration of the Ukrainian power grid [54], and the infamous Stuxnet attack on industrial SCADA systems [72].

In order to repel these stealthy attacks, detection algorithms are designed to find compromised system's components to maintain safe operation. Intelligent attackers, in turn, are forced to resort to new methods to hide and deceive on-board control systems and their anomaly detection

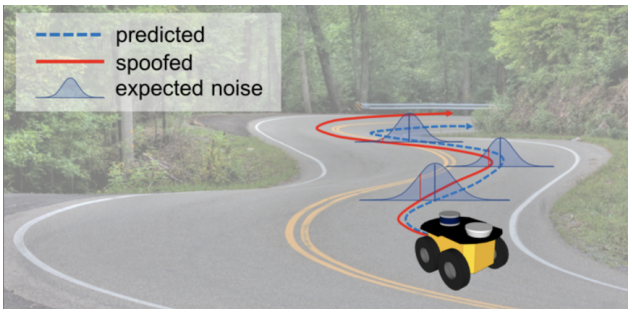


Figure 1.1: A pictorial representation where a cyber attack is able to hijack a vehicle to unsafe states while remaining hidden within the noise profile of its sensors.

counterparts. While such attack vectors are less effective since the attacker needs to maintain a low profile, performance degradation can be still accomplished if the attack is able to remain undetected.

Various traditional detection techniques [32] have been shown to be effective in mitigating these attacks to provide resiliency in the scenario of attacks to vulnerable sensors and communication channels on robotic systems. However, these traditional techniques monitor residual error magnitudes, which consequently are susceptible to being fooled by intelligent attackers that are able to construct attack sequences that hide within noise profiles. Typically, malicious attackers aim to compromise a system by hijacking its states to unsafe regions while hiding within the system's noise profile. Despite hiding within magnitude-based boundaries to remain undetected, non-random patterns arise that violate the expected behavior from normal system behavior. As an example, an attacker that is hijacking an autonomous system while intentionally manipulating sensor measurements in a stealthy manner to hide within system noise, will push the measurements toward one direction. In order for the attack to be effective, it must inherently create inconsistent, non-random behavior of the measurement residual. Randomness-based monitoring approaches for detection of anomalous behavior in measurements and received information sequences have not been considered, thus potentially leaving systems vulnerable to stealthy attacks.

While attack detection on-board a single robot is challenging in its own right, the problem becomes much more complex when dealing with multi-robot systems where the motion of each robot is dependent on other agents when using consensus mechanisms. As depicted in Fig. 1.2, a single robot within a multi-robot system may perform in a non-cooperative way which affects neighboring vehicles. If this issue is not eliminated from the system, then the effects of the misbehaving agent may be propagated through the entire robot network. However, within the field of robotics, coordination and swarming of multi-robot systems have long been studied and are gaining even greater attention thanks to the many technological advances. These classes of systems are typically used to perform coordinated tasks in a distributed fashion. This collaborative nature allows for numerous applications that would be more difficult or not possible to perform with just a single agent, such as: factory

and warehouse logistics [19], vehicle platooning [61], connected vehicle-to-vehicle operations [86, 41], surveillance [104], and disaster-relief [26].

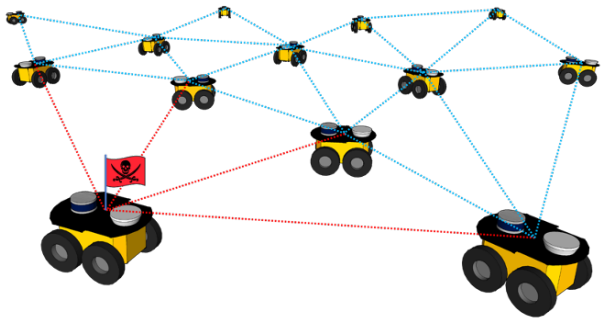


Figure 1.2: A multi-robot system which suffers from a compromised agent that impacts control performance of other vehicles in the entire swarm.

With such benefits in multi-robot systems, also comes the risk of cyber attacks. In fact, many of the aforementioned applications are typically designed without considering cyber-security issues, assuming that all the actors (i.e., other robots) in the multi-robot setting are cooperative. In the presence of a compromised robot in the network, liveness (i.e., the ability to perform and complete correctly a task) and safety (i.e., avoid collisions or reaching undesired states) properties can be violated. The presence of malicious actors in a network can potentially manipulate the entire multi-robot system, hijacking a mission and potentially leading the system toward undesired states. Such situations can be caused by: compromised communications which results in incorrect sharing of information between robots, or by manipulated sensor measurements, leading compromised robots to react to altered on-board signals that are also broadcast to surrounding neighbors. In a successful hijacking attempt, an attacker is able to implement a *stealthy* attack sequence to degrade system performance, all while remaining hidden from detection. It is imperative that stealthy cyber attacks are discovered such that any compromised agents are isolated and removed from the system to eliminate the malicious effects from the multi-robot system. Within this dissertation, we expand on our randomness-based detection techniques to also provide resiliency in multi-robot systems. These randomness-based methods, are able to recognize stealthy data inconsistencies and patterns within information exchanges between vehicles. This detection capability enables a system to isolate compromised robots in order to remove any undesirable effects to the multi-robot system's control performance while completing an operation.

Furthermore, during operations the individual robots communicate with each other for coordination in order to complete various tasks. Communication broadcasts containing safety-critical information (e.g., such as important tasks in an operation) that are not properly encrypted/protected can be intercepted, thus creating further security issues. The most secure option is to avoid ex-

changing data altogether; however, this is not a feasible option when safety-critical information needs to be passed along. If agreed upon before operations, cooperative multi-robot systems can leverage side-channel information that contain hidden signature data, which are unknown to malicious attackers. In this way, an agent can collect data and infer the safety-critical information from the received broadcasts of other agents without them explicitly broadcasting this information. Hidden signatures within information broadcasts can be leveraged to covertly pass and/or detect safety-critical information between robots in multi-agent systems to maintain safe operations. We depict this scenario in Fig. 1.3, where a robot passes safety-critical information to its neighboring vehicles when an object of interest has been found in the environment.

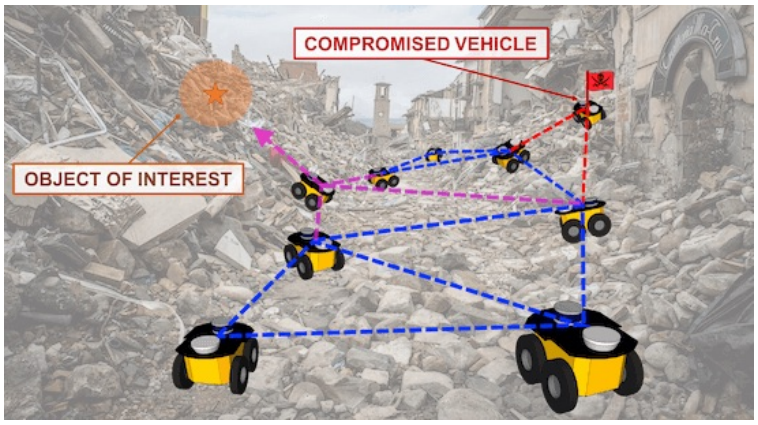


Figure 1.3: A multi-robot system cooperatively performs a task while inferring the objective of other teammates that are accomplishing mission-critical tasks.

Beyond attack detection and isolation/removal of compromised agents in robotic systems, is the essential capability of maintaining resilience for all robots during operation. A main aspect for a robot to sustain resiliency in undesirable situations is the ability to localize itself within an environment. The capacity to perform an accurate and robust localization allows unmanned systems to achieve truly autonomous operations. This can be accomplished in various ways, such as by relying on positioning sensors like global positioning systems (GPS), odometry, and IMU or through range sensors such as LiDAR, infrared (IR), and camera systems [21]. The sensing information can then be leveraged via implementation of localization methods such as Particle filters (PF) and Simultaneous Localization and Mapping (SLAM) techniques.

While many of the aforementioned localization techniques are measured on-board a single vehicle to aid in performing a desired task/goal; however, systems with multiple robots need to take extra measures to ensure all robots are performing properly. For example, consensus algorithms are typically considered in robotic swarms where agents share information (e.g., their states) to attain coordinated behaviors in a decentralized fashion to accomplish a desired goal. A variety of issues can cause undesirable information to be exchanged between robots, such as cyber attacks or

faults to on-board sensors or malicious man-in-the-middle attacks to communication broadcasts. If known landmarks/obstacles are present in the operating space, range sensors can be utilized for localization or to determine if the system is performing as expected. However, if agents within a multi-robot system are navigating in open spaces (e.g., in the middle of the ocean), landmarks may not be available, thus leaving compromised agents unable to reliably localize themselves. Resilient measures have been incorporated to multi-robot systems to ensure continued safe operations in the presence of uncooperative robots within a swarm [89, 93]. To accomplish this, most methods (e.g., [93, 88]) make the uncompromised robots typically "ignore" any misbehaving robot to remove undesirable effects that could compromise the MRS mission. Thus robots experiencing cyber attacks or faults are essentially discarded without a recovery method implemented. In turn, discarded agents can potentially enter undesirable/restricted regions within the environment and the asset will be more likely lost or damaged. However, instead of simply discarding any agent deemed compromised, a more resilient approach should consider recovering the compromised robots while continuing to perform the planned mission. Given these potentially important factors, designing recovery frameworks for both single- and multi-robot systems is a critical research topic to ensure safe autonomous operations.

With these considerations in mind, the objectives of this work are to solve the following challenges:

- How to detect attacks on sensors and communication broadcasts that intentionally hide within system noises that no longer follow random behavior.
- How to characterize the improvement in control performance in terms of state deviation under the effects of undetectable attacks to on-board sensors on a single system.
- How to detect sensor and communication attacks and recover/reconfigure a multi-robot system.
- How to inform neighboring vehicles in multi-robot systems about safety critical tasks without explicitly sending this information over communication broadcasts.
- How to recover compromised agents that lose localization capabilities due to cyber attacks or faults to on-board positioning sensors through cooperative recovery methods within multi-agent systems.
- How to detect and recover from compromised operating regions within an on-board controller caused by cyber attacks or faults.

In order to accomplish these objectives, this work proposes to leverage a randomness-based approach to the traditional residual-based attack detection scheme to find previously undetectable cyber attacks on single- and multi-agent systems which increases resiliency to autonomous operations. Furthermore, we utilize the randomness-based approach in a similar fashion, but this time for

discovering signals through hidden motion models and hidden signals in multi-robot systems for covert information broadcasts in adverse environments.

1.1 Related Literature

In this section, an overview of related literature is provided in residual-based anomaly detection schemes that are running on-board individual systems to discover inconsistent sensor measurements due to cyber attacks and sensor/system faults. Then, an overview of techniques to provide resilience from cyber attacks and faults in multi-agent systems to maintain desired global control performance, followed by approaches in cooperative recovery to aid in localization of faulty agents in multi-robot systems.

1.1.1 Residual-based Anomaly Detectors

Cyber-physical systems generally employ attack/fault detection mechanisms to provide for safer autonomous operations. Traditional attack detection techniques monitor for inconsistent behavior of the *residual* [32], defined as the difference between a measurement and the state prediction, which is typically provided by a Kalman filter (KF) [42, 100]. When sensor measurements are altered by malicious actors, the system at hand may be hijacked to undesirable states. In a successful hijacking attempt, an attacker is able to implement a *stealthy* attack sequence to degrade system performance, all while remaining hidden from detection. The term stealthy has been adopted in a wide-range of attack scenarios on stochastic systems, such as in zero-dynamics [79], replay [111], zero-alarm [13], and hidden [69] attack cases. In our context, the term *stealthy* indicates an attack sequence that mimics normal (attack-free) behavior of traditional detection schemes (i.e., a hidden attack [69]), where attackers leverage the noise characteristics within a system to evade detection during a hijacking attempt. Previous literature in the field of cyber-physical system security have considered deceptive cyber attacks to on-board sensors of a system by injecting false data to measurements while trying to remain undetected within system noise [67]. Other have analyzed the effects of malicious sensor attacks on individual systems that leverage a Kalman filter for state estimation [2]. Similarly, authors in [49] characterize how undetected attacks compromise closed-loop systems that utilize the Kalman filter in terms of state and system dynamic degradation. Frequently used residual-based techniques for attack detection include: the Generalized Likelihood Ratio test (GLRT) [24, 28], Sequential Probability Ratio testing (SPRT) [35, 50], Bad Data detection [68], and the Cumulative Sum (CUSUM) detector for change detection [70]. A common residual-based detection technique is Compound Scalar Testing (CST), commonly known as the chi-square detector, which reduces the residual vector to a scalar quadratic test measure for monitoring [68]. Various improvements that leverage the traditional chi-square detection scheme have utilized a tuning window [84], coding sensor

outputs [64], and the model-based CUSUM detector [71]. The use of these residual-based detection techniques have shown their effectiveness when implemented on a wide range of applications that include: attack detection on unmanned aerial systems [50], fault monitoring on mobile robots [102], and securing industrial systems such as smart grids [76].

1.1.2 Multi-agent System Resiliency and Security

The topic of resilience in multi-agent systems has received extensive consideration in the engineering and computer science communities [114]. Much attention has gone into resilience of these systems based on network connectivity, determined by the underlying graph topology of the network [93]. A widely used method for multi-agent resilience is through consensus protocols that leverage the Mean Subsequence Reduced (MSR) algorithms [119, 95, 120, 34, 16] in which all vehicles in a network come to an agreement on a global variable of interest (e.g., velocity, position, heading angle). Such consensus protocols are resilient to F number of compromised (e.g., non-cooperative) agents, which rely on network topologies that satisfy the robustness properties in which every agent in the network follows the strategy of diminishing the effect of potentially deceptive information due to cyber-attacks or faults by ignoring up to F agents with shared values that contrast the most from its own value of the global consensus variable. As noted by authors in [107], the purpose of MSR algorithms is not to detect misbehaving (i.e., compromised) agents in a network, but rather to simply leave out values consisting of the greatest difference in magnitude. The issue that arises in proximity-based formation control is the need for reliable received position information from neighbors to maintain a desired proximity from the other agents. With the inclusion of misbehaving agents in the system, the proximity-based formation control performance can be compromised.

An example of misbehaving agent detection in multi-agent networks was presented by authors in [18] that propose the Flag Raising Distributed Estimator ($\mathcal{FRDE}$) such that each agent in the network estimates an unknown parameter by an iterative algorithm that leverages both its own sensor measurement and its neighbor's estimate of the parameter to detect the presence of adversarial agents. As a neighbor's parameter estimate differs from an agent's own parameter beyond a chosen threshold, the neighbor is deemed adversarial, thus raising a flag. Authors in [117] utilize agents as mobile detectors that allow for isolation of any malicious agents that collude with each other in an attempt to take advantage of network connectivity constraints. Another example can be found in [113] where every uncompromised agent can detect and isolate misbehaving agents in leader-follower and leaderless consensus networked systems. Each agent employs a multi-phase reputation-based protocol by relying on local observations and adaptive consensus weight updates on neighbors to allow for resilient convergence of uncompromised agents in the formation. Taking a different approach to detection, authors in [44] propose a network-wide shared watermarking

signal that is applied to control inputs of each agent in multi-robot systems, then a residual-based anomaly detection scheme is used to find any misbehaving agents. In [53], the authors leverage the residual-based *Cumulative Sum* (CUSUM) anomaly detector, first characterized in [77], to discover spoofs to on-board navigation systems of robots in multi-robot systems, thus allowing the mobile robot team to arrive at its desired destination. Different from the aforementioned works, our proposed decentralized framework considers deceptive cyber-attacks that intentionally hide within the uncertainties to avoid detection from traditional residual-based detection procedures in multi-robot systems consisting of stochastic linear time-invariant (LTI) modeled agents.

1.1.3 Cooperative Recovery in Multi-agent Systems

The topic of resilient multi-agent system operations has been extensively researched recently in the robotics and computer science communities [115]. Within the cooperative localization literature, an early work leveraged other robots in the swarm as landmarks, where robots were split into two subgroups that differed in roles and motion from each other [47]. However, the use of a centralized controller limited the scalability and robustness of the framework. Improvements were made in [81] with a decentralized approach to alleviate scalability issues and then verified on swarms of holonomic robots equipped with range and bearing sensors. Authors in [92] utilized the joint estimate of robot poses computed by an EKF using both centralized and decentralized methods for robots to estimate their pose from the shared information between other robots. Scalability issues arise for both methods and an assumption of swarm size is required prior to operations. Other works, such as [15], have improved upon scalability issues by leveraging the Covariance Intersection Algorithm to perform belief updates of neighboring robots in a decentralized manner. Differing from [92], the framework in [15] enables each robot to compute only its own state covariance matrix for estimation updates in a decentralized manner instead of requiring to compute state covariances for neighboring robots. Authors in [82] maneuver a single leader robot consisting of a more capable localization sensor with improved state estimation toward the center of the swarm via potential functions to enhance localization capabilities of the remaining vehicles. A central position for the leader robot enhances localization capabilities of the remaining robots when information is exchanged between each other. In [83], the authors analyze optimal formation topologies for cooperative localization within multi-AUV systems to improve control performance in the presence of errors in measurement information.

We also find works that cover approaches for MRS resiliency from various types of attacks/faults on: sensors, actuators, communications, and physical damage [116]. For example, authors in [106] proposed a recovery framework on swarms to utilize LOS measurements when positioning sensors fail, with the assumption that neighboring agents are within visual range at all times. In [93], authors

proposed a resilient flocking framework that leverages a consensus algorithm along with a hybrid control policy that maintains connectivity of the mobile robot team when uncooperative robots share incorrect information. In [53], the Cumulative Sum detector was used to discover spoofs to navigation systems of individual robots within multi-robot systems to isolate and remove malicious agents. Different from these works, we assume that robots operate in unknown environments (i.e., lacking known landmarks) and are positioned beyond range sensing of other robots. When a cyber attack or fault occurs to positioning sensors, an impacted robot loses the ability to localize itself within the environment. In turn, it is necessary for a compromised robot to alert neighboring robots to aid in recovery for re-localization.

Another topic of multi-agent system resilience that has received significant attention recently has been the use of time- and radio frequency (RF)-based methods for cooperative localization/recovery. Various time-based measurement techniques used for localization have been proposed, such as Time of Arrival (ToA), Time Difference on Arrival (TDoA), Angle of Arrival (AoA), and Time of Flight (ToF) [57]. An RF-based technique used for aid in localization are from measuring the Received Signal Strength Indicator (RSSI). As the name suggests, RSSI-based techniques rely on measuring the strength of the received radio frequency signal over the communication channel.

Much effort has been placed on leveraging RSSI to aid in localization within an environment. In particular, numerous recent articles have leveraged anchor nodes with known positions placed throughout an environment to aid in localization within indoor environments [110]. An example of RSSI-based localization is found in [60], authors combined Weighted Distance Vector Hop (WDV-Hop) from RSS measurements with a hyperbolic weighting matrix to estimate the positions of any nearby agents in MASs while utilizing the known locations of reference nodes (i.e., anchors) in the environment. Authors in [39] proposed a method for robotic swarms deployed in indoor environments to effectively navigate through narrow passageways where robots are allocated specific roles to ensure localization accuracy. In [14], an approach was proposed to provide robustness in localization performance within swarms when nearby agents satisfy both Line-of-Sight and Non-Line-of-Sight conditions, which affect estimation performance. Authors in [40] presented a framework that utilizes the particle filter on the RSSI-based measurements from known anchor nodes and then fused the position estimate with the remaining system states for improved localization capabilities. In [31], the authors present theoretical results for an adaptive framework that positions a team of robotic agents acting as mobile routers to provide communication coverage to the remaining subset of client robots, when the knowledge of the clients' positions are unknown. The multi-robot system is able to position the robots behaving as routers to satisfy the client robots' demands, while adapting to changes in wireless signals (i.e., leveraging directionality of signal strength) and the dynamic (and potentially unmapped) environment. Authors in [43] leverage RSSI signals to estimate AoA of signal

sources (i.e., transmitters) and humans/robots for target tracking. Similar to our work, authors in [75] proposed an RSSI-based localization algorithm for multi-robot teams within anchor-less environments. Their approach combines a Kalman Filter and the Floyd-Warshall algorithm to compute smooth distance estimates between agents, then multidimensional scaling is utilized to estimate relative positions of nearby agents.

1.2 Overview of Research

The research presented in this dissertation consists of three successive Parts that include: I) sensor attack detection on a single robot, II) isolation and network reconfiguration for resilient multi-robot systems, and III) system recovery for both single- and multi-robot systems. A final Part IV summarizes what we have gained and learned from in the first three Parts. Figure 1.4 provides an overview of the research presented in this dissertation.

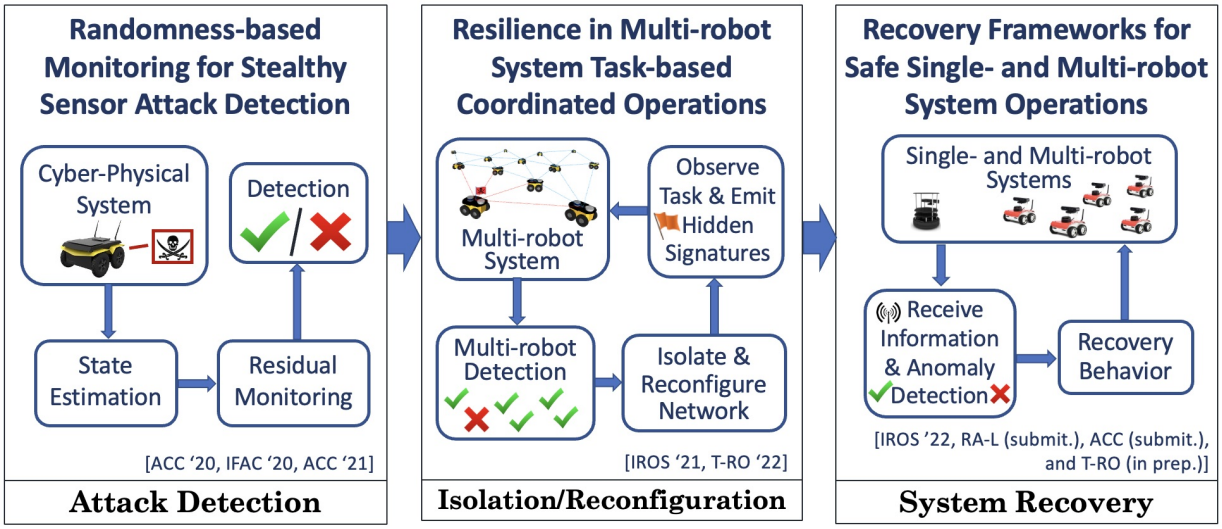


Figure 1.4: Overview of the presented research in this dissertation.

Beginning with **Part I**, we characterize various randomness-based detection techniques for identifying stealthy sensor attacks that hide within system noise profiles. In order to effectively design attack detectors to identify stealthy attacks, we mathematically model an autonomous system with stochastic uncertainties with a discrete-time linear time-invariant model. From this model, we are able to extract residual-based test measures which are used in monitoring for anomalies of the on-board sensor measurements. In **Part II**, we extend the mathematical frameworks and detection designs from Part I by providing resiliency to multi-robot system operations. In particular, we characterize how robots can decipher whether its neighboring robots are behaving in proper manner by analyzing information received from each of the vehicles. When neighboring robots are determined to be compromised by cyber attacks, the remaining robots isolate these agents then reconfigure

the network to remove any malicious effects that the compromised robots may pose. An additional feature that is included within this Part is the ability for the robots to communicate safety-critical information to each other without explicitly sending this data through communication broadcasts. This is especially beneficial when multi-robot systems navigate within adversarial environments where malicious attackers have the ability to intercept the safety-critical information, which could possibly lead to devastating consequences if the attackers are successful. **Part III** presents recovery frameworks for autonomous single- and multi-robot systems. We begin with a detection and recovery framework as well for individual mobile autonomous systems. This framework enables autonomous systems, such as mobile robots, to maintain desirable control signals during operations when on-board controllers are faulty due to specific triggering conditions that provide malicious control inputs. Our framework highlights a novel information compensator which alters data sent to the controller to avoid any malicious conditions while also computing desirable control inputs to the system. For the scenario of recovery in multi-robot systems, we highlight two frameworks where other nearby robots cooperatively recover (i.e., re-localize) any compromised robots that suffer from cyber attacks or faults to on-board positioning sensors, which deem is positioning estimate to be unreliable. Before concluding Part III, we also include our preliminary work on a novel cooperative multi-robot system framework. In this framework, a team of robots cooperatively defend a protected region of the environment from a malicious intruding agent, then cooperatively shepherd the intruder to a desired safe region. Throughout Parts I-III in this dissertation, we feature our contributions with simulation results using realistic system dynamical models and lab experiments on real unmanned ground vehicles to validate our frameworks. Finally, in **Part IV**, we conclude the dissertation by providing insight in what we have accomplished and learned, followed by a discussion on possible directions we could take in the future that we could take.

1.3 Dissertation Organization and Contributions

In this section, we present the composition of this dissertation by providing summaries of each chapter and specifying contributions within each chapter. As a summary for this dissertation, Part I consists of Chapter 2 which highlights the attack detection phase on individual systems, Part II is covered by Chapters 3 and 4 that focus on detection, isolation, and reconfiguration in multi-robot systems, and Part III is highlighted by Chapters 5-7 which feature recovery frameworks that have been developed for both single- and multi-robot systems. To end this dissertation, Part IV summarizes our results and a discussion in provided for possible future work. Now, let us outline each chapter and mention our contributions.

Chapter 2: Randomness-based Monitoring for Sensor Attack Detection

In this chapter, we introduce our novel randomness-based framework for detection of stealthy sensor attacks that intentionally hide within noise profiles on a system. These detection frameworks and their corresponding characteristics are utilized primarily throughout the remaining of this dissertation for various applications. We begin by characterizing the system and noise models, the on-board state estimator, and the threat model in the context of linear time-invariant discrete-time system. Unlike previous residual-based detection schemes that monitor for sensor attacks, our schemes are purposefully designed to discover stealthy attacks that hide within system noise profiles. We introduce novel randomness-based approaches to find previously undetectable cyber attacks to on-board sensors when using residual-based attack detection schemes on an individual system, which monitors for inconsistent and non-random behavior that contradict the known system and noise models. We cover three different frameworks in this chapter; one of which relies on a sliding window of past data, while the other two relax this windowed constraint when monitoring at runtime. Within these frameworks, we assume that an intelligent attacker can gain knowledge of critical aspects of the system, such as the dynamical model, noise models, estimation technique, and on-board controller to enable stealthy attack sequences. We compare our detection techniques to previously state-of-the-art detectors to highlight the effectiveness of our methods when monitoring for stealthy sensor attacks hidden within noise profiles. This chapter is based on the publications:

- P.J. Bonczek, S. Gao, and N. Bezzo, "Model-based Randomness Monitor for Stealthy Sensor Attacks," *in Proceedings of the IEEE American Control Conference (ACC), 2020.*
- P.J. Bonczek and N. Bezzo, "Memoryless Cumulative Sign Detector for Stealthy CPS Sensor Attacks," *in Proceedings of the International Federation of Automatic Control World Congress (IFAC), 2020.*
- P.J. Bonczek and N. Bezzo, "Detection of Hidden Attacks on Cyber-Physical Systems from Serial Magnitude and Sign Randomness Inconsistencies," *in Proceedings of the IEEE American Control Conference (ACC), 2021.*

Chapter 3: Multi-robot System Attack Detection and Network Reconfiguration

In this chapter, we extend our attack detection frameworks that were formalized in the previous chapter to provide resiliency in multi-robot systems. In particular, the multi-robot systems follow decentralized control consensus protocols to maintain desired proximity-based formations. We assume that the robots operate in adversarial environments where on-board sensors and communication broadcasts for information exchange between agents can be spoofed by intelligent attackers. A randomness-based attack detection scheme is leveraged to identify not only on-board sensor spoofs,

but also malicious information being received from neighboring agents. Upon detection of inconsistent information being received, our framework allows each robot to isolate and reconfigure the multi-robot network in a decentralized manner to allow for resilient navigation of all uncompromised robots. We validate our framework with simulation and experiments consisting of autonomous ground robots using both Gazebo and live experiments in a lab setting. This chapter is based on the following publication:

- P.J. Bonczek, R. Peddi, S. Gao, and N. Bezzo, "Detection of Non-random Sign-based Behavior for Resilient Coordination of Robotic Swarms," *IEEE Transactions on Robotics (T-RO) in the Special Issue for Resilience in Networked Robotic Systems, vol. 38, no. 1, pp. 92-109, 2022.*

Chapter 4: Detection and Inference of Randomness-based Behavior for Resilient Multi-robot Coordinated Operations

This chapter deals with maintaining information secrecy during multi-robot system operations. We tackle the problem of hiding safety-critical information that is passed between robots from potential interception by malicious attackers. Virtual spring-damper mesh physics models are leveraged to create detectable hidden motion signatures for nearby robots to detect. When important tasks are discovered within the environment by an agent, this agent then changes its control behavior to emit the hidden motion signature which is unknown to attackers to enable the transfer of safety-critical information. This allows for the remaining vehicles in the swarm to aid in accomplishing safety-critical tasks. We highlight our decentralized framework with both simulations and lab experiments to validate the cooperative behavior of multi-robot swarms as information is passed amongst the agents. This chapter is based on the following publication:

- P.J. Bonczek and N. Bezzo, "Detection and Inference of Randomness-based Behavior for Resilient Multi-vehicle Coordinated Operations," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*

Chapter 5: Recovery of Autonomous Systems Operating under On-board Controller Failures

Chapter 5 introduces a detection and recovery framework on single autonomous systems that experience cyber attacks or faults to its on-board controller during operations. When cyber attacks, failures, and implementation errors occur within the controller of an autonomous, nominal behavior is affected leading to unsafe states and degraded control performance. In this chapter, we focus on the specific problem when controller parameters are manipulated, such as control gains, which alter system behavior that are triggered when specific values of the state or tracking error is present within

the on-board controller. If these changes are not detected, they can lead to partial or complete loss of the system's control authority, resulting in a hijacking and leading the autonomous system towards unforeseen states. To deal with this problem, the system must detect which conditions within the controller trigger the anomalous behavior and then have a recovery plan to maintain a desirable control signal for resilient operation. Our novel recovery mechanism is designed as an information compensator, which alters the reference signal and state vector information fed to the controller to avoid compromised regions within the state or tracking error spaces that trigger anomalous behavior to maintain desirable control inputs to the autonomous system. Our detection and recovery framework is validated using both MATLAB simulations and lab experiments on an autonomous robot to show resilient go-to-goal operations are performed in the presence of malicious behavior to on-board controllers. This chapter is based on the following publication:

- P.J. Bonczek and N. Bezzo, "Resilient Detection and Recovery of Autonomous Systems Operating under On-board Controller Cyber Attacks," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.*

Chapter 6: Multi-robot System Cooperative Recovery from Loss of Localization in Unknown Environments

In this chapter, we introduce two frameworks for cooperative localization in multi-robot systems. Here, we assume that robots are susceptible to cyber attacks or faults to on-board positioning sensors such that compromised robots lose reliable positioning estimates, thus impacting performance of themselves and the remaining of the robotic swarm. Different from other research literature, we also assume that robots operate beyond sensing range from each other and also operate in unknown and/or landmark-free environments, such that leveraging known objects or features for localization is not possible. However, agents remain within communication range of each other, which is necessary for exchanging information (e.g., state, input, task information) to preserve desired proximity-based formations during operations. To overcome these undesirable circumstances, our first cooperative recovery framework allows compromised robots to notify nearby robots of its loss of localization capabilities. Then, the nearby agents switch control modes to perform a novel cooperative motion behavior to aid in re-localizing the compromised agent by coming within distance sensing range of the compromised robot. From the perspective of the compromised robots, these cooperative robots are then used as mobile landmarks such that localization capabilities are returned to any compromised robot with unreliable nominal positioning sensor. We verify this framework with MATLAB simulations and lab experiments using swarms of ground robots.

In our second cooperative recovery framework, an agent can extract a distance estimate to each nearby agent within communication range by measuring the received signal strength indication (RSSI)

from their information broadcasts. We present a novel decentralized cooperative recovery framework for individual agents to re-localize themselves within multi-agent systems by leveraging RSSI when on-board positioning sensors are unreliable. Compromised agents leverage sensor reconfiguration to replace their anomalous on-board position sensor with an RSSI-based position measurement to re-localize itself. Given that RSSI measurements are noisy, we provide an optimal solution to reduce uncertainty of the RSSI-based position measurements for improved state estimation performance. Furthermore, we present a novel approach to robustly estimate the RSSI-based measurement covariance to further improve state estimation accuracy, thus allowing for better control performance. A series of MATLAB simulations using multi-agent system formations that experience attacks and faults to positioning sensors highlight our approach. This chapter is based on the following works:

- P.J. Bonczek and N. Bezzo, "A Cooperative Recovery Framework for Resilient Multi-robot Operations in Unknown Environments," *submitted and under review to the IEEE Robotics and Automation Letters (RA-L).*
- P.J. Bonczek and N. Bezzo, "Resilient Multi-agent Formation Control via RSSI-based Localization," *submitted and under review to the 2023 IEEE American Control Conference (ACC).*

Chapter 7: Cooperative Robotic Teams for Defending Against Malicious Intruders

In this chapter, we highlight our current work where a team of autonomous robotic agents cooperatively defend a protected an object/region of interest within the environment from malicious intruders (i.e., a robot that is considered completely compromised). The objectives for each of the defensive robots is to: 1) intercept and impede the motion of an intruding agent from reaching a known protected region within the environment by constructing a robotic wall/barrier and 2) cooperatively shepherd the malicious intruder to a safe region in the environment. We validate our cooperative approach with MATLAB simulations using teams of defensive robots to protect against a malicious agent. This chapter is based on the following current work:

- P.J. Bonczek and N. Bezzo, "Cooperative Robotic Teams for Defending Against Malicious Intruders," *in preparation for submission to the IEEE Transactions on Robotics (T-RO).*

Chapter 8: Conclusions and Future Work

In this chapter, we conclude the dissertation by summarizing the results from all the aforementioned works and discuss potential future directions to build on.

1.4 Summary of Contributions

To summarize, the work presented in this dissertation will contribute to the existing state-of-the-art in autonomous robotic system resiliency and security by providing:

- Novel randomness-based approaches to find previously undetectable cyber-attacks in measurement residual-based attack detection schemes on an individual system, which monitor for inconsistent and non-random behavior that contradict the known system and noise models.
- A decentralized detection and isolation framework to discover stealthy attacks to both sensors and communication broadcasts that attempt to hijack the entire multi-agent systems, allowing for network reconfiguration by isolating any maliciously behaving agents.
- A novel decentralized framework to include hidden signals by utilizing hidden motion models to alert neighboring vehicles of specific tasks, which enables the passing of safety-critical information to other vehicles without explicitly broadcasting the information.
- A novel decentralized cooperative recovery framework to coordinate motion amongst neighboring robots to aid in re-localization of compromised agents that suffer from cyber attacks and faults to on-board positioning sensors.
- A decentralized cooperative recovery framework for individual agents to re-localize themselves within multi-agent systems by leveraging received signal strength indication when on-board positioning sensors are unreliable due to cyber attacks and faults.
- A novel recovery approach for autonomous systems to maintain desired control inputs during operations when attacks or faults occur within specific operating regions of the on-board controller.

Part I

Sensor Attack Detection

Chapter 2

Randomness-based Monitoring for Sensor Attack Detection

In this chapter, a novel framework is presented for randomness-based attack detectors within a measurement residual-based monitoring scheme. The randomness-based approach leverages signed inconsistencies of the measurement residual when monitoring for sensor attacks, whereas traditional monitoring schemes monitor for magnitude-based inconsistencies of the residual test measure. By employing the presented randomness-based detection schemes in this chapter, stealthy sensor attack sequences hidden within noise profiles that evade detection from traditional methods can now be discovered. These methods provide tighter security measures on autonomous systems for more resilient operations. The attack detection schemes are validated with MATLAB simulations and lab experiments using an unmanned ground vehicle. The material covered in this chapter was published in the following:

- P.J. Bonczek, S. Gao, and N. Bezzo, "Model-based Randomness Monitor for Stealthy Sensor Attacks," *in Proceedings of the IEEE American Control Conference (ACC), 2020.*
- P.J. Bonczek and N. Bezzo, "Memoryless Cumulative Sign Detector for Stealthy CPS Sensor Attacks," *in Proceedings of the International Federation of Automatic Control World Congress (IFAC), 2020.*
- P.J. Bonczek and N. Bezzo, "Detection of Hidden Attacks on Cyber-Physical Systems from Serial Magnitude and Sign Randomness Inconsistencies," *in Proceedings of the IEEE American Control Conference (ACC), 2021.*

2.1 Introduction

The foundation behind attack detection is to leverage an estimator to predict the evolution of a system's state and compare it with measured/received information. This monitored difference,

commonly referred to as the *residual*, is leveraged to detect anomalous behavior introduced by cyber attacks. Discrepancies between this comparison and an expected behavior imply that there may be an attack present. The detector is placed in the system feedback, as depicted in Figure 2.1, to provide a determination on the health of the system (i.e., whether the system is safe or if a spoof has been detected).

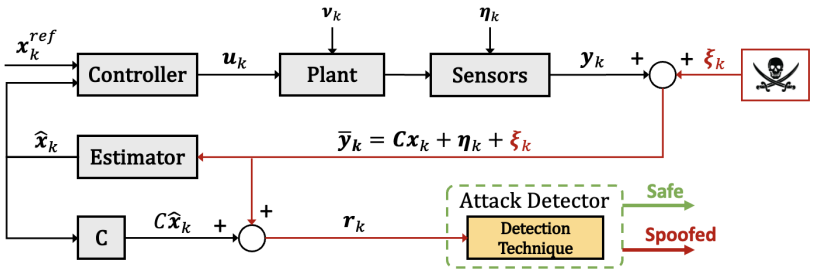


Figure 2.1: The overall residual-based monitoring architecture of a cyber-physical system.

Traditional attack detectors, namely the GLRT, Bad Data, and CUSUM detectors, leverage an alarm-based method when monitoring residual magnitudes for attacks. These detection procedures: 1) compute a test measure from the residual magnitude, 2) provide a tuned threshold for comparison with the test measure, and 3) trigger an alarm as the test measure passes the threshold. Under nominal conditions, the tuned threshold parameter will result in a desired (i.e., expected) false alarm rate. During an attack to the system's on-board sensors, the alarm rate deviates from the expectation which notifies the system of a potential attack. However, if a malicious attacker gains access to critical system information (e.g., such as the dynamical model, noise characteristics, state estimator), this offers the attack an opportunity to develop stealthy attack sequences which can evade detection by hiding within noise profiles.

2.2 Preliminary Modeling

This section introduces the dynamical, measurement, noise, attack, and estimation models used throughout this chapter.

2.2.1 System Dynamical and Noise Models

An autonomous system is considered to be modeled as discrete-time linear time-invariant (LTI) system dynamics. Systems are equipped with sensors capable of measuring system states to achieve full autonomy during operations without the need for humans.

The discrete-time LTI system is modeled in the following form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{\nu}_k, \tag{2.1}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{\eta}_k, \tag{2.2}$$

with $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ representing the state matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ the input matrix, and $\boldsymbol{C} \in \mathbb{R}^{N_s\times n}$ the output matrix with the state vector $\boldsymbol{x}_k \in \mathbb{R}^n$, control input vector $\boldsymbol{u}_k \in \mathbb{R}^m$, measurement output vector $\boldsymbol{y}_k \in \mathbb{R}^{N_s}$ providing measurements from $N_s \in \mathbb{N}$ sensors, and sampling time instants $k \in \mathbb{N}$. Process and measurement noises are assumed to be i.i.d. zero-mean Gaussian uncertainties $\boldsymbol{\nu} = \mathcal{N}(0, \boldsymbol{Q}) \in \mathbb{R}^n$ and $\boldsymbol{\eta} = \mathcal{N}(0, \boldsymbol{R}) \in \mathbb{R}^{N_s}$ with covariance matrices are described by $\boldsymbol{R} > 0$, $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{Q} > 0$, $\boldsymbol{Q} \in \mathbb{R}^{N_s\times N_s}$.

One critical assumption is that the system is controllable, or in other words, which describes the ability of the control inputs $\boldsymbol{u}_k$ to influence all elements of the state vector $\boldsymbol{x}_k$ of the system from an initial state to a desired state within a finite amount of time. We denote the controllability matrix $\mathscr{C} \in \mathbb{R}^{n\times nm}$ by:

$$\mathscr{C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} & \dots & \boldsymbol{A}^{n-1}\boldsymbol{B} \end{bmatrix} \tag{2.3}$$

and if $\text{rank}(\mathscr{C}) = n$ then the system is considered controllable.

It is also assumed that the system is observable, meaning that the system state $\boldsymbol{x}_k$ can be estimated by utilizing the information from the system outputs $\boldsymbol{y}_k$. The observability matrix $\mathscr{O} \in \mathbb{R}^{ns\times n}$ is represented by:

$$\mathscr{O} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C}\boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n-1} \end{bmatrix} \tag{2.4}$$

such that if $\text{rank}(\mathscr{O}) = n$ then the system is determined to be observable.

2.2.2 State Estimation Model

During operations, a linear Kalman Filter (KF) is implemented to provide a state estimate $\hat{\boldsymbol{x}}_k \in \mathbb{R}^n$. For simplicity, the asymptotic form of the Kalman Filter is utilized in the form:

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{A}\hat{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{L}(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k) \tag{2.5}$$

with the Kalman gain matrix $\boldsymbol{L} \in \mathbb{R}^{n \times N_s}$ computed by

$$\boldsymbol{L} = \boldsymbol{P}\boldsymbol{C}^{\mathsf{T}}(\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{R})^{-1} \tag{2.6}$$

where $\lim_{k\to\infty} \boldsymbol{P}_k = \boldsymbol{P}$ is the asymptotic estimation covariance found by solving the discrete time algebraic Riccati equation with the following:

$$\boldsymbol{P} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{A} - (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{B})(\boldsymbol{R} + \boldsymbol{B}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{B})^{-1}(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{A}) + \boldsymbol{Q}. \tag{2.7}$$

2.2.3 Threat Model

In this chapter, the assumption is that attackers are able to manipulate sensor measurements in order to influence an autonomous system to behave in an undesired manner. This malicious intent is captured in an attack vector $\boldsymbol{\xi}_k \in \mathbb{R}^{N_s}$, such that the updated measurement output vector considering sensor attacks is characterized as:

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k \tag{2.8}$$

when the attack vector satisfies $\boldsymbol{\xi}_k \neq 0$, where it is assumed that any sensor (i.e., any element within the attack vector $\boldsymbol{\xi}_k$) may be compromised.

2.2.4 Measurement Residual and the Chi-square Test Measure

The state estimation model is leveraged to aid in determining whether sensor measurements are behaving in an expected manner. More specifically, the *measurement residual* $\boldsymbol{r}_k \in \mathbb{R}^{N_s}$, which is defined as the difference between observed sensor measurements $\tilde{\boldsymbol{y}}_k$ and the expectation of the measurements using the state estimate $\hat{\boldsymbol{x}}_k$, and computed by:

$$\boldsymbol{r}_k = \tilde{\boldsymbol{y}} - \boldsymbol{C}\hat{\boldsymbol{x}}_k \tag{2.9}$$

is leveraged to monitor for anomalous sensor behavior. The measurement residual distribution is modeled to have an expected covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{N_s \times N_s}$ when sensor measurements are behaving in a nominal manner (i.e., $\boldsymbol{\xi}_k = 0$) with the following:

$$\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{r}_k \boldsymbol{r}_k^{\mathsf{T}}] = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{R}. \tag{2.10}$$

In the absence of sensor attacks, the measurement residual corresponding to an sth sensor $r_{k,s},\ s \in \{1, \ldots, N_s\}$ follows a Gaussian distribution $r_{k,s} \sim \mathcal{N}(0, \sigma_s^2)$ where σ_s^2 is the sth diagonal element of the residual covariance matrix $\boldsymbol{\Sigma}$ such that $\mathbb{E}[r_{k,s}] = 0$ and $\mathrm{Var}[r_{k,s}] = \sigma_s^2$.

Furthermore, the measurement residual can be reduced down to a scalar quadratic *test measure* $z_k \in \mathbb{R}_{\geq 0}$ that is then utilized to monitor the system for sensor attacks. This monitoring technique, also known as the chi-square detector, compute the test measure by:

$$z_k = \boldsymbol{r}_k^\mathsf{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_k. \tag{2.11}$$

In the absence of attacks, the measurement residual is an N_s-dimensional vector of normally distributed random variables $\boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma})$. The test measure z_k is then expected to be a random variable that follows a chi-square distribution with N_s degrees of freedom, i.e. $z_k \sim \chi^2(N_s)$, that follows:

$$\mathbb{E}[z_k] = N_s, \quad \mathrm{Var}[z_k] = 2N_s. \tag{2.12}$$

2.2.5 Hypothesis Testing

When monitoring the measurement residual for consistent behavior, we test two different hypotheses: The null hypothesis $\mathcal{H}_0$ for nominal (i.e., attack-free) conditions and the alternative hypothesis $\mathcal{H}_a$ where attacks are present. Formally, the hypotheses are written as:

$$\mathcal{H}_0 : \begin{cases} \mathbb{E}[\boldsymbol{r}_k] = \boldsymbol{0}, \\ \mathbb{E}[\boldsymbol{r}_k \boldsymbol{r}_k^\mathsf{T}] = \boldsymbol{\Sigma}, \end{cases} \qquad \mathcal{H}_a : \begin{cases} \mathbb{E}[\boldsymbol{r}_k] \neq \boldsymbol{0}, \text{and/or} \\ \mathbb{E}[\boldsymbol{r}_k \boldsymbol{r}_k^\mathsf{T}] \neq \boldsymbol{\Sigma}. \end{cases} \tag{2.13}$$

2.2.6 Assumptions

The assumption is that the given linear dynamical and noise models capture the true behavior of the autonomous system's dynamics and stochastic uncertainties. The system is assumed to be equipped with necessary on-board sensors to satisfy controllability and observability conditions. The sensors are assumed to not suffer performance degradation from issues that include wear and tear or reduction of accuracy due to environmental conditions.

2.3 Problem Formulation

This section formalizes the various problems that are addressed in this chapter. It is considered that stealthy sensor attacks are implemented by intelligent attackers in an attempt to evade detection while also hijacking the system. The attack signal $\boldsymbol{\xi}_k \neq 0$ contains malicious data that can disrupt randomness, thus causing measurement residuals to display non-random behavior. The first problem is formally defined as follows:

Problem 2.1 (Randomness Monitoring over Windowed Sequences) *An autonomous system has an objective to discover stealthy sensor attacks that are hidden within noise profiles. To*

aid in monitoring for anomalous measurement residual behavior, the on-board detection scheme for randomness monitoring leverages a sliding window length $T \in \mathbb{N}$ of saved residual vector information.

Definition 2.1 (Sensor Measurement Consistency) *While utilizing the sliding window T, an individual sensor measurement is considered to be behaving in a nominal random manner if:*

- *A sequence of measurement residuals over the time window occurs in an unpredictable, pattern-free manner.*
- *Measurement residual elements $s = \{1, \ldots, N_s\}$ have a proper distribution over $\mathbb{E}[r_{k,s}]$.*

Given each measurement residual element $r_{k,s}$ over the sliding window T from a time $k - \ell + 1$ to time k, find a policy to determine at runtime whether its corresponding sensor measurement is behaving in a random manner such that all conditions in Definition 2.1 hold.

The previous problem to monitor for non-random residual behavior due to stealthy attacks relies on storing a window of residual data in order to be effective. The requirement of storing historical residual information over a sliding window can be relaxed, while still being effective in discovering stealthy attacks. In the next problem, the specific focus is on monitoring for unexpected sign occurrences of a single chi-square test measure to detect sensor attacks, all while eliminating the need for a sliding window.

Problem 2.2 (Window-less Sign Randomness Monitoring) *Given the chi-square test measure z_k computed from the measurement residual $\boldsymbol{r}_k$ and its expected covariance matrix $\boldsymbol{\Sigma}$, find a policy to determine at runtime whether on-board sensor measurements are compromised (i.e., $\boldsymbol{\xi}_k \neq 0$) due to stealthy attacks that intentionally hide within noise profiles, without the need to store a window of residual data.*

If an intelligent attacker has knowledge of on-board detection schemes, hidden attack sequences can be generated in an attempt to evade detection. However, these sequences can create patterns, thereby causing the chi-square test measure to display non-random behavior.

Definition 2.2 (Serial Consistency) *Sensor measurements are determined to be behaving consistently if:*

- *The chi-square test measure is pattern-free and follows a chi-square distribution $z_k \sim \chi^2(N_s)$.*
- *The difference between consecutive test measures follows an expected distribution and also an expected sign switching rate.*

Problem 2.3 (Runtime Detection of Serial Inconsistencies) *Provided with the chi-square test measure z_k, design a policy to determine at runtime if sensor measurements satisfy serial consistencies as defined in Definition 2.2.*

2.4 Randomness Monitor Over Windowed Measurement Residual Sequences

A detection framework is introduced to monitor randomness of the measurement residual sequence through two tests and tuning bounds are provided for each to result in desired false alarm rates. In this section, two primary components of the approach are introduced: i) two statistical tests that are utilized in residual monitoring and ii) tuning bounds are provided for each test to result in desired behavior during attack-free conditions. The overall cyber-physical system architecture, including our Randomness Monitor, is placed in the control system feedback to monitor the measurement residual sequence.

2.4.1 Monitoring Window

A monitor is built to check if sequences of measurements within the residual $\boldsymbol{r}_k$ over a sliding monitoring window $T = (k - \ell + 1, l)$ for $\ell \in \mathbb{N}$ previous time instants is behaving consistently. The residual sequences over the window T are denoted as

$$\boldsymbol{r}_T = (\boldsymbol{r}_{T,1}, \ldots, \boldsymbol{r}_{T,s}, \ldots, \boldsymbol{r}_{T,N_s}) \tag{2.14}$$

where the residual sequence for an sth sensor is:

$$r_{T,s} = (r_{k-\ell+1,s}, \ldots, r_{k,s}). \tag{2.15}$$

The residual sequence for each sth sensor over the sliding window formalized in (2.15) is utilized to monitor for expected residual symmetry and serial run randomness behavior.

2.4.2 Residual Symmetry

Following $\mathcal{H}_0$, we would expect that the number of positive and negative values of $\boldsymbol{r}_k$ over the monitoring window are equal. Additionally, a symmetric distribution indicates that the expected absolute magnitude of positive and negative residuals over a given window of length ℓ are equal,

$$\mathbb{E}[|\boldsymbol{r}^+_{T,s}|] = \mathbb{E}[|\boldsymbol{r}^-_{T,s}|], \;\; s \in \{1, \ldots, N_s\} \tag{2.16}$$

where $\mathbb{E}[|\boldsymbol{r}^+_{T,s}|]$ and $\mathbb{E}[|\boldsymbol{r}^-_{T,s}|]$ denote the expected absolute magnitude for positive and negative values of the residual $r_{k,s}$ within the window T for any given sth sensor. In other words, we would expect the sum of absolute values from the residual to be equal for both the positive and negative values. The WSR test takes both the sign and magnitude of the residual into account to determine whether

conditions satisfy $\mathcal{H}_0$. Large differences in the residual signs or signed magnitudes imply non-similar distributions, causing the test to reject the no attack assumption and triggering an alarm.

To perform the WSR test at each time instant k, we first look at the ℓ number of residuals over the monitoring window T of a given sth sensor, ranking the *absolute values* of residuals $\boldsymbol{r}_{T,s}$, starting with $rank = 1$ for the smallest absolute value, $rank = 2$ for the second smallest, and so on until reaching the largest absolute value with $rank = \ell$. Ranks of absolute values for positive (i.e. $|r^+_{T,s}|$) and negative (i.e. $|r^-_{T,s}|$) residuals over the window T are placed into the sets $\mathcal{R}^+_{k,s}$ and $\mathcal{R}^-_{k,s}$ at every time instance k, respectively.

Remark 2.1 *For residuals equal to each other and not equal to* 0 *(tied for the same rank), an average of the ranks that would have been assigned to these residuals is given to each of the tied values. Furthermore, residuals equal to* 0 *are removed and* ℓ *is reduced accordingly.*

Following, we compute the sum of ranks for both the positive and negative valued residuals,

$$W^+_{k,s} = \sum \mathcal{R}^+_{k,s}, \quad W^-_{k,s} = \sum \mathcal{R}^-_{k,s}. \tag{2.17}$$

Residuals with symmetric distributions have similar valued sum of ranks, i.e. $W^+_{k,s} \sim W^-_{k,s}$, whereas the sum of ranks in non-symmetric distributions are not similar $W^+_{k,s} \nsim W^-_{k,s}$ resulting in a rejection of $\mathcal{H}_0$, which we will now discuss how to solve. Assuming a large window of size $\ell \geq 20$[1] [99], the Wilcoxon random variables $W^+_{k,s}$, $W^-_{k,s}$ converge to a Normal distribution (without attacks) as $\ell \to \infty$ and can be approximated to a standard normal distribution. The approximated expected value and variance of the two sum of ranks $W^+_{k,s}$ and $W^-_{k,s}$, denoted as $W^\pm_{k,s} = \{W^+_{k,s}, W^-_{k,s}\}$ is

$$\mathbb{E}[W^\pm_{k,s}] = \tfrac{\ell^2+\ell}{4}, \quad \mathrm{Var}[W^\pm_{k,s}] = \tfrac{(\ell^2+\ell)(2\ell+1)}{24}. \tag{2.18}$$

The z-score of (2.17) for a given sth sensor is computed by

$$Z^W_{k,s} = \frac{\min(W^\pm_{k,s}) - \mathbb{E}[W^\pm_{k,s}]}{\sqrt{\mathrm{Var}[W^\pm_{k,s}]}} = \frac{\min(W^\pm_{k,s}) - \frac{(\ell^2+\ell)}{4}}{\sqrt{\frac{(\ell^2+\ell)(2\ell+1)}{24}}} \tag{2.19}$$

and the p-value used to determine whether to reject the null hypothesis $\mathcal{H}_0$ (i.e., no attacks) is computed from (2.19) as:

$$p^W_{k,s} = \Phi(|Z^W_{k,s}|) = 2 \cdot \frac{1}{\sqrt{2\pi}} \int_{|Z^W_{k,s}|}^{\infty} \exp\left\{\frac{-\lambda^2}{2}\right\} d\lambda. \tag{2.20}$$

[1]For window length of smaller size, exact tables need to be used for probability distributions of the Wilcoxon Signed-Rank random variable [99].

When $p^W_{k,s}$ falls below the threshold $\tau^W_s = \alpha^{\text{des}}_s$, i.e., $p^W_{k,s} < \tau^W_s$, we reject $\mathcal{H}_0$ and an alarm $\psi^W_{k,s} = 1$ is triggered, otherwise $\psi^W_{k,s} = 0$. In the absence of attacks, the alarm rate α^W_s for an sth sensor should be approximately the same as the desired false alarm rate $\alpha^W_s \sim \alpha_s$des. Computation of α^W_s is over the sliding window $T^\alpha = (k - \ell^\alpha + 1, k)$ of length ℓ^α by $\alpha^W_s = \frac{1}{\ell^\alpha}\sum_{j=k-\ell^\alpha+1}^{k} \psi^W_{j,s}$. Conversely, an attack that affects the residual distribution symmetry, triggering the alarm $\psi^W_{k,s}$ more frequently, causing an elevation of alarm rate α^W_s. For alarm rates exceeding a user defined alarm rate threshold, i.e., $\alpha^W_s > \alpha^\tau_s$, the sth sensor is deemed compromised. In the following lemma we provide a proof for bounds of the WSR test variables (2.17) to satisfy a desired false alarm rate α^{des}_s.

Lemma 2.1 *Given the residual $r_{k,s}$ for an sth sensor over a monitoring window T consisting of ℓ residuals and desired false alarm rate α^{des}_s, an alarm is triggered by the WSR test when $\Omega^W_- \leq \{W^\pm_{k,s}\} \leq \Omega^W_+$ is not satisfied where*

$$\Omega^W_\pm = \pm|\Phi^{-1}(\alpha^{des}_s/2)|\sqrt{(\ell^2+\ell)(2\ell+1)/24} + (\ell^2+\ell)/4. \tag{2.21}$$

Proof: From the Wilcoxon test statistic equaling the sum of ranks in (2.17), we can rearrange (2.19) such that $\min(W^\pm_{k,s}) = Z^{W_{\text{crit}}}_{k,s}\sqrt{(\ell^2+\ell)(2\ell+1)/24} + (\ell^2+\ell)/4$ where $Z^{W_{\text{crit}}}_{k,s} = \Phi^{-1}(\alpha^{\text{des}}_s/2)$ is the critical value of $Z^W_{k,s}$ for $\min(W^\pm_{k,s})$ satisfying a desired alarm rate α^{des}_s to not reject $\mathcal{H}_0$. The lower bound of $\{W^-_{k,s}, W^+_{k,s}\}$ must satisfy

$$\Omega^W_- = \Phi^{-1}(\alpha^{\text{des}}_s/2)\sqrt{(\ell^2+\ell)(2\ell+1)/24} + (\ell^2+\ell)/4 \leq \min(W^-_{k,s}, W^+_{k,s}), \tag{2.22}$$

to not sound off an alarm $\psi^W_{k,s}$. Conversely, we want to show that if the lower bound $\Omega^W_- \leq \min(W^\pm_{k,s})$ in (2.22) holds then the upper bound Ω^W_+ holds as well. By again manipulating (2.19) such that we take the maximum $\max(W^\pm_{k,s}) = Z^{W_{\text{crit}}}_{k,s}\sqrt{(\ell^2+\ell)(2\ell+1)/24} + (\ell^2+\ell)/4$ where this time $Z^{W_{\text{crit}}}_{k,s} = \Phi^{-1}(1-\alpha^{\text{des}}_s/2)$ is the critical value of $Z^W_{k,s}$ for the upper bound $\max(W^\pm_{k,s})$ satisfying a desired alarm rate α^{des}_s to not reject $\mathcal{H}_0$, the upper bound is written as

$$\Omega^W_+ = \Phi^{-1}(1-\alpha^{\text{des}}_s/2)\sqrt{(\ell^2+\ell)(2\ell+1)/24} + (\ell^2+\ell)/4 \geq \max(W^-_{k,s}, W^+_{k,s}) \tag{2.23}$$

to not trigger the alarm $\psi^W_{k,s}$. In the calculation of the critical z-score value from the standard normal distribution $\mathcal{N}(0,1)$ to satisfy a given desired alarm rate α^{des}_s, it is easy to show that $|\Phi^{-1}(\alpha^{\text{des}}_s/2)| = \Phi^{-1}(1-\alpha^{\text{des}}_s/2)$ and $\Phi^{-1}(\alpha^{\text{des}}_s/2) = -|\Phi^{-1}(\alpha^{\text{des}}_s/2)|$ giving the final bounds of $\Omega^W_- \leq (W^\pm_{k,s} = \{W^-_{k,s}, W^+_{k,s}\}) \leq \Omega^W_+$ as

$$-|\Phi^{-1}(\alpha^{\text{des}}_s/2)|\sqrt{(\ell^2+\ell)(2\ell+1)/24}+(\ell^2+\ell)/4 \leq W^\pm_s \leq |\Phi^{-1}(\alpha^{\text{des}}_s/2)|\sqrt{(\ell^2+\ell)(2\ell+1)/24}+(\ell^2+\ell)/4$$

satisfying the bounds of $\Omega_{\pm}^{W}$ in (2.21). With this we conclude that if $\min(W_{k,s}^{\pm})$ does not satisfy (2.22) then $\Omega_{-}^{W} \leq \{W_{k,s}^{-}, W_{k,s}^{+}\} \leq \Omega_{+}^{W}$ is not satisfied, triggering alarm $\psi_{k,s}^{W}$ for a desired false alarm rate α_s^{des}, ending the proof. ■

2.4.3 Serial Run Randomness

The WSR test alone is not sufficient to test for randomness, since an attacker could manipulate measurements by creating specific patterns to avoid detection on the WSR test. To test further, we need to determine if the sequence of residuals are being received randomly by leveraging the Serial Independence runs (SIR) test [12]. The SIR test examines the number of runs that occur over the sequence, where a "run" is defined as one or more consecutive residuals that are greater or less than the previous value. A random sequence of residuals over a given window length should exhibit a specific expected number of runs: too many or too few number of runs would not satisfy random sequential behavior. A hypothesis test is formed with $\mathcal{H}_0$ for the absence of sensor attacks and $\mathcal{H}_a$ where attacks are present by

$$\mathcal{H}_0\text{: } N_R = \mathbb{E}[N_R], \quad \mathcal{H}_a\text{: } N_R \neq \mathbb{E}[N_R], \tag{2.24}$$

where N_R is the number of observed runs, to determine whether the number of runs satisfy a randomly behaving sequence. First, we take the difference of residuals at time instances k and $k-1$ over a window T'

$$\boldsymbol{r}'_{T',s} := r'_{k,s} = r_{k,s} - r_{k-1,s}\ ,\ k \in T', \tag{2.25}$$

where $T' = \{k-\ell+2, \ldots, k\} = T \setminus \{k-\ell+1\}$ is the monitor window T shortened by one by removing the oldest time instance. This in turn gives us $\ell' = \ell - 1$ residual differences.

Remark 2.2 *A residual difference $r'_{k,s} = 0$, $k \in T'$ from (2.25) is not considered in the test and the size of ℓ' is reduced accordingly, i.e., $\ell' = \ell' - 1$.*

From the sequence of residual differences (2.25), we take the sign of each residual within the window T'

$$\text{sign}(r'_{k,s}),\ k \in T' \tag{2.26}$$

forming a sequence of ℓ' positive and negative signs. The number of runs N_R are observed over the sequence of ℓ' residual differences. The expected mean and variance of runs [12] are computed by

$$\mathbb{E}[N_R] = \frac{2\ell'-1}{3}, \quad \text{Var}[N_R] = \frac{16\ell'-29}{90}. \tag{2.27}$$

Assuming large data sets (i.e. window length $\ell \geq 25$) [12], the distribution of N_R converges to a normal distribution as $\ell' \to \infty$ and can be approximated to a zero mean unit variance standard

normal distribution $N_R \sim \mathcal{N}(0,1)$. From the number of observed runs N_R and number of residual differences ℓ', we compute the z-score test statistic for Serial Independence from a standard normal distribution

$$Z^S_{k,s} = \frac{N_R - \mathbb{E}[N_R]}{\sqrt{\text{Var}[N_R]}} = \frac{N_R - \big(2\ell' - 1\big)/3}{\sqrt{\big(16\ell' - 29\big)/90}}. \tag{2.28}$$

Using the z-score from (2.28) we compute the p-value of the observed signed residual differences by

$$p^S_{k,s} = \Phi(|Z^S_{k,s}|) = 2 \cdot \frac{1}{\sqrt{2\pi}} \int_{|Z^S_{k,s}|}^{\infty} \exp\left\{\frac{-|\lambda|^2}{2}\right\} d\lambda. \tag{2.29}$$

When $p^S_{k,s} < \tau^S_s$ is satisfied where $\tau^S_s = \alpha^{\text{des}}_s$ denotes the threshold, we reject the null hypothesis $\mathcal{H}_0$ from (2.24) and an alarm $\psi^S_{k,s} = 1$ is triggered. In the absence of attacks, the alarm rate α^S_s is approximately the same as the desired false alarm rate $\alpha^S_s \sim \alpha^{\text{des}}_s$. Alarm rate α^S_s over the sliding window T^α is computed by $\alpha^S_s = \frac{1}{\ell^\alpha} \sum_{j=k-\ell^\alpha+1}^{k} \psi^S_{j,s}$. Alarm rates exceeding a user defined alarm rate threshold, i.e. $\alpha^S_s > \alpha^\tau_s$, signifies that the sth sensor is compromised.

Remark 2.3 *A special case of triggering alarm $\psi^S_{k,s} = 1$ is when Remark 2.2 is satisfied, when two consecutive residuals are equal. Since $r_{k,s} \sim \mathcal{N}(0, \sigma^2_s)$, the probability of having two residuals of the same value is equal to* 0.

The following lemma provides a proof for bounds of N_R in the SIR test to satisfy a desired false alarm rate α^{des}_s.

Lemma 2.2 *Given the residual differences $r'_{k,s} = r_{k,s} - r_{k-1,s}$ for an sth sensor over a window T' and desired false alarm rate α^{des}_s, an alarm is triggered by the SIR test when $\Omega^S_- \leq N_R \leq \Omega^S_+$ is not satisfied where*

$$\Omega^S_\pm = \pm|\Phi^{-1}(\alpha^{des}_s/2)|\sqrt{(16\ell' - 29)/90} + (2\ell' - 1)/3. \tag{2.30}$$

Proof: With an observed number of runs N_R within a window of ℓ' residual differences, we can rearrange (2.28) such that $N_R = |Z^S_{k,s}|\sqrt{(16\ell' - 29)/90} + (2\ell' - 1)/3$ where $|Z^S_{k,s}| = |\Phi^{-1}(\alpha^{\text{des}}_s/2)|$, we find the bounds of N_R to not reject (2.24) for a desired false alarm rate α^{des}_s are

$$\begin{aligned} -|\Phi^{-1}(\alpha^{\text{des}}_s/2)|\sqrt{(16\ell' - 29)/90} + (2\ell' - 1)/3 \leq N_R \\ \leq |\Phi^{-1}(\alpha^{\text{des}}_s/2)|\sqrt{(16\ell' - 29)/90} + (2\ell' - 1)/3. \end{aligned} \tag{2.31}$$

From (2.31), we obtain the bounds of $\Omega^S_\pm$ in (2.30) for alarm triggering at a desired false alarm rate α^{des}_s. ■

2.4.4 Comparable Detectors

To show that our framework can easily be augmented with any detector that provides magnitude boundaries, two different boundary detectors found in the CPS security literature are considered. Both boundary detectors discussed in this section leverage the absolute value of the residual (2.9) for attack detection. Consequently, in the absence of attacks (i.e., $\boldsymbol{\xi}_k = \mathbf{0}$), this leads to $|r_{k,s}|$ following a half-normal distribution [90] defined by

$$\mathbb{E}[|r_{k,s}|] = \sqrt{2/\pi}\sigma_s,\ \mathrm{Var}[|r_{k,s}|] = \sigma_s^2(1-2/\pi) \tag{2.32}$$

where σ_s^2 was defined as the sth diagonal element in $\boldsymbol{\Sigma}$.

The first detector considered is the *Bad-Data Detector* (BDD) [67], a benchmark attack detector to find anomalies in sensor measurements, alarming when the residual error goes beyond a threshold. Similar to our detection framework, the BDD can also be tuned for a desired false alarm rate α_s^{des}. Considering the residual $r_{k,s}$ in (2.9), the BDD procedure for each sth sensor is as follows:

Bad-Data Detector Procedure

$$\text{If } |r_{k,s}| > \tau_s^B,\ \text{then alarm } \psi_{k,s}^B = 1,\ s \in \{1,\dots,N_s\} \tag{2.33}$$

Assuming the system is without attacks, the tuned threshold τ_s^B for the BDD in (2.33) with $r_{k,s} \sim \mathcal{N}(0,\sigma_s^2)$ is solved by $\tau_s^B = \sqrt{2}\sigma_s \mathrm{erf}^{-1}(1-\alpha_s^{\mathrm{des}})$ where $\mathrm{erf}^{-1}(\cdot)$ is the *inverse error function*, resulting in $\alpha_s^B \sim \alpha_s^{\mathrm{des}}$.

The second well-known boundary detector that we consider is the *CUmulative SUM* (CUSUM), which has been shown to have tighter bounds on attack detection than the BD [70]. The CUSUM leverages the absolute value of the residual in the detection procedure and is solved by:

CUSUM Detector Procedure

$$\begin{cases} \textbf{Initialize } S_{1,s} = 0,\ s \in \{1,\dots,N_s\} & \\ S_{k,s} = \max(0, S_{k-1,s} + |r_{k,s}| - b_s), & \text{if } S_{k-1,s} \le \tau_s^C \\ S_{k,s} = 0 \text{ and Alarm } \psi_{k,s}^C = 1, & \text{if } S_{k-1,s} > \tau_s^C \end{cases} \tag{2.34}$$

The working principle of of this detector is to accumulate the residual sequence in $S_{k,s}$, triggering an alarm $\psi_{k,s}^C = 1$ when the test variable surpasses the threshold τ_s^C. A detailed explanation of how to tune threshold τ_s^C given a bias b_s for a desired false alarm rate α_s^{des} can be found in [70].

2.4.5 State Deviation Under Worst-case Stealthy Attacks

In this analysis, a reference tracking feedback controller is considered that is written by the following:

$$\boldsymbol{u}_k = \boldsymbol{K}\hat{\boldsymbol{x}}_k + \boldsymbol{k}_r \boldsymbol{x}_k^{\mathrm{ref}} \tag{2.35}$$

where $\boldsymbol{K} \in \mathbb{R}^{N_s \times n}$ is the state feedback control gain matrix, $\boldsymbol{k}_r \in \mathbb{R}^{m \times m}$ is a reference gain for output tracking, $\boldsymbol{x}_k^{\text{ref}}$ is the reference state, and $\hat{\boldsymbol{x}}_k$ is the state estimate computed from the Kalman Filter (i.e., state estimator). Choosing a suitable $\boldsymbol{K}$ such that $(\boldsymbol{A} + \boldsymbol{BK})$ is stable (i.e. $\rho[\boldsymbol{A} + \boldsymbol{BK}] < 1$, where $\rho[\cdot]$ is the spectral radius) and $(\boldsymbol{A}, \boldsymbol{C})$ is assumed stabilizable, the closed-loop system can be written within terms of the KF estimation error as:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= (\boldsymbol{A} + \boldsymbol{BK})\boldsymbol{x}_k + \boldsymbol{Bk}_r\boldsymbol{x}_k^{\text{ref}} - \boldsymbol{BKe}_k + \boldsymbol{\nu}_k, \\ \boldsymbol{e}_{k+1} &= (\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{e}_k - \boldsymbol{L}(\boldsymbol{\xi}_k + \boldsymbol{\eta}_k) + \boldsymbol{\nu}_k. \end{aligned} \tag{2.36}$$

As an attacker injects signals into the measurements (i.e. $\boldsymbol{\xi} \neq \boldsymbol{0}$), system dynamics are indirectly affected via the interconnected term $\boldsymbol{BKe}_k$ from the estimation error dynamics.

In the remaining of this section we describe the maximum damage that can occur due to worst-case scenario stealthy sensor attacks. We assume the attacker has perfect knowledge of system dynamics, detection procedures, and state estimation. The objective of an attacker is to cause maximum damage to the system state by injecting attack signals $\boldsymbol{\xi}_k$ to measurements while also remaining undetected. With only the BDD implemented, the effects of a worst-case scenario attack while not triggering an alarm can be derived from (2.9) and (2.33) with a sustained attack signal

$$\xi_{k,s} = -\boldsymbol{C}_s\boldsymbol{e}_k - \eta_{k,s} + \tau_s^B \tag{2.37}$$

causing the residual $|r_{k,s}| = \tau_s^B$ to not trigger the BDD alarm.

Now considering CUSUM as a stand-alone detector, an adversarial wants to avoid attack vectors such that the monitoring test variable exceeds threshold τ_s^C, thereby causing a reset $S_{k,s} = 0$, if $S_{k-1,s} > \tau_s^C$ in (2.34) by satisfying the CUSUM procedure sequence $S_{k,s} = \max(0, S_{k-1,s} + |\boldsymbol{C}_s\boldsymbol{e}_k + \eta_{k,s} + \xi_{k,s}| - b_s) \leq \tau_s^C$ if $S_{k-1,s} \leq \tau_s^C$. For maximum effect on state deviation, the attacker saturates the CUSUM test statistic such that $S_{k,s} = \tau_s^C$ to achieve no alarm sequences. Here we define a saturation as follows:

***Definition* 2.3** *Saturation of a boundary detector is defined as the maximum allowable attack signal to force the residual to, but without exceeding, the detector threshold.*

Beginning at a time k, an attacker immediately saturates $S_{k,s}$ with the attack signal

$$\xi_{k,s} = -\boldsymbol{C}_s\boldsymbol{e}_k - \eta_{k,s} + b_s - S_{k-1,s} + \tau_s^C \tag{2.38}$$

followed by

$$\xi_{k,s} = -\boldsymbol{C}_s\boldsymbol{e}_k - \eta_{k,s} + b_s \tag{2.39}$$

for all future time instances to hold $S_{k,s}$ at threshold τ_s^C.

With the Randomness Monitor augmented with either BDD or CUSUM, an attacker can no longer hold an attack sequence to one side as described in attacks (2.37)-(2.39). Rather, an attacker is forced to create an attack sequence such that $r_{k,s}$ alternates residual signs if it wants to avoid triggering alarms for both the WSR and SIR tests. The most effective attack for maximum state deviation with our augmented framework is to *saturate* the boundary detector as often as possible, while leaving the remaining attack signals with an opposite sign with respect to the saturating attacks to force the residual to be as close as possible to zero.

From the WSR test, given a monitoring window ℓ, the minimum number of *non-saturating* attack signals $\xi_{k,s}$ to not trigger an alarm $\psi^W_{k,s}$ is

$$\gamma^\ell_s = \min_{\ell^j} \left(\sum_{rank=1}^{\ell^j} rank \right) \Bigg| \sum_{rank=1}^{\ell^j} rank > \min(W^\pm_s), \tag{2.40}$$

in which $\ell^j \in \mathcal{L} = (1, \ldots, \ell)$ and $\mathcal{L}$ is the set of all *ranks*. From (2.40), we can then find the maximum number of *saturating* attack signals by $\beta^\ell_s = \ell - \gamma^\ell_s$.

Proposition 2.1 *The maximum allowable saturating attack signal converges to* $\lim_{\ell\to\infty} \frac{\beta^\ell_s}{\ell} = 1 - \frac{\sqrt{2}}{2} \approx .293$ *for any* α^{des}_s *as shown by the dashed black line in Fig. 2.2.*

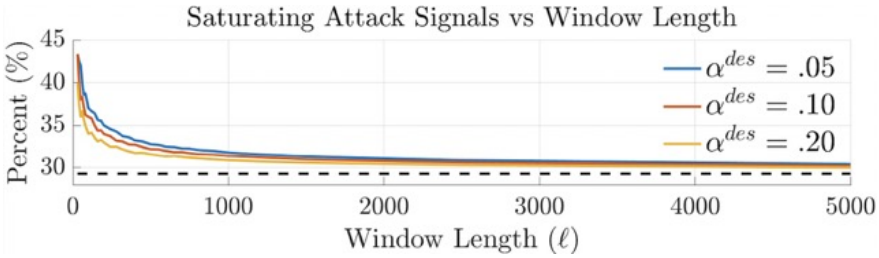


Figure 2.2: Allowable percentage of saturating attack signals of given windows lengths for different desired alarm rates α^{des}.

To this point, we have discussed worst-case scenario attack sequences causing saturation of the test variable (in this paper BDD and CUSUM) to maximize the effect of the attack. However, from Remark 2.3, a special case to satisfy requirements of the SIR test is when two consecutive residuals of same value triggers an alarm $\psi^S_{k,s} = 1$. To work around this issue, a stealthy attacker with perfect knowledge of the SIR test can include a small uniformly random number to the attack signal $\xi_{k,s}$ denoted by $\delta_{k,s} \sim \mathcal{U}(0, \epsilon)$ where $\epsilon \in \mathbb{R}^+$ is infinitesimally small and $\mathbb{E}[\delta_{k,s}] = \frac{\epsilon}{2} \approx 0$. Thus, the worst-case scenario with the Randomness Monitor augmented to the BDD follows

$$\begin{cases} \xi_{k,s} = -\boldsymbol{C}_s \boldsymbol{e}_k - \eta_{k,s} + \tau^B_s - \delta_{k,s}, & \textbf{if saturating,} \\ \xi_{k,s} = -\boldsymbol{C}_s \boldsymbol{e}_k - \eta_{k,s} - \delta_{k,s}, & \textbf{if non-saturating,} \end{cases} \tag{2.41}$$

in order to not trigger an alarm. Similarly, but with the CUSUM detector, an undetectable attack sequence follows

$$\begin{cases} \xi_{k,s} = -S_{k-1,s} - \boldsymbol{C}_s \boldsymbol{e}_k - \eta_{k,s} + b_s + \tau_s^C - \delta_{k,s}, & \textbf{if } \text{saturating}, \\ \xi_{k,s} = -\boldsymbol{C}_s \boldsymbol{e}_k - \eta_{k,s} + b_s - \delta_{k,s}, & \textbf{if } \text{non-saturating}. \end{cases} \tag{2.42}$$

Given the alternating signed sequence of residuals over the monitoring window, the expected value of $r_{k,s}$ under worst-case scenario stealthy attacks is denoted as

$$\begin{cases} \mathbb{E}[r_{k,s}^B] = \tau_s^B(\frac{\beta_s^\ell}{\ell} - \delta_{k,s}) \approx \tau_s^B \frac{\beta_s^\ell}{\ell}, & \textbf{for } \text{Bad-Data}, \\ \mathbb{E}[r_{k,s}^C] = \tau_s^C(\frac{\beta_s^\ell}{\ell} - \delta_{k,s}) \approx \tau_s^C \frac{\beta_s^\ell}{\ell}, & \textbf{for } \text{CUSUM}. \end{cases} \tag{2.43}$$

With our framework augmented to the BDD, the expected value of the residual sequence is described as $\mathbb{E}[\boldsymbol{r}_k^B] = (\mathbb{E}[r_{k,1}^B], \ldots, \mathbb{E}[r_{k,s}^B])^\mathsf{T}$ and the expectation of the closed-loop system (2.36) under attack (2.41) results in

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}_{k+1}] &= (\boldsymbol{A} + \boldsymbol{BK})\mathbb{E}[\boldsymbol{x}_k] - \boldsymbol{BK}\mathbb{E}[\boldsymbol{e}_k], \\ \mathbb{E}[\boldsymbol{e}_{k+1}] &= \boldsymbol{A}\mathbb{E}[\boldsymbol{e}_k] - \boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^B]. \end{aligned} \tag{2.44}$$

Note, in (2.44), the reference signal term $\boldsymbol{Bk}_r\boldsymbol{x}_k^{\text{ref}}$ from (2.36) has been removed as we are interested in the expected state deviation under an attack. It is clear that if $\rho[\boldsymbol{A}] > 1$ and $\mathbb{E}[\boldsymbol{r}_k^B] \neq \boldsymbol{0}$ then the expectation of the estimation error $\mathbb{E}[\boldsymbol{e}_k]$ for destabilized states diverge to infinity as $k \to \infty$ (depending on algebraic properties of $\boldsymbol{A}$), indirectly causing these states within $\mathbb{E}[\boldsymbol{x}_k]$ to also diverge unbounded.

Lemma 2.3 *Considering a closed-loop system from* (2.1) *and* (2.44)*, where* $\rho[\boldsymbol{A}] < 1$ *and attack sequence in* (2.41)*, the limit for expected state divergence* $\lim_{k\to\infty} \mathbb{E}[\boldsymbol{x}_k] = \Delta^B$ *is*

$$\Delta^B = (\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{BK})^{-1}\boldsymbol{BK}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^B]. \tag{2.45}$$

Proof: Assuming both $\rho[\boldsymbol{A}] < 1$ and $\rho[\boldsymbol{A} + \boldsymbol{BK}] < 1$ are satisfied, signifying the invertibility of $(\boldsymbol{I} - \boldsymbol{A})$ and $(\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{BK})$ in (2.45), an expected equilibrium is reached as $k \to \infty$ by $\mathbb{E}[\boldsymbol{x}_\infty] = (\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{BK})^{-1}\boldsymbol{BK}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^B]$ and $\mathbb{E}[\boldsymbol{e}_\infty] = (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^B]$ such that the evolution of (2.44) with the expected differences $\mathbb{E}[\boldsymbol{x}_k] - \mathbb{E}[\boldsymbol{x}_\infty]$ and $\mathbb{E}[\boldsymbol{e}_k] - \mathbb{E}[\boldsymbol{e}_\infty]$ is described by

$$\begin{aligned} \mathbb{E}[\boldsymbol{x}_{k+1}] - \mathbb{E}[\boldsymbol{x}_\infty] &= (\boldsymbol{A} + \boldsymbol{BK})(\mathbb{E}[\boldsymbol{x}_k] - \mathbb{E}[\boldsymbol{x}_\infty]) - \boldsymbol{BK}(\mathbb{E}[\boldsymbol{e}_k] - \mathbb{E}[\boldsymbol{e}_\infty]), \\ \mathbb{E}[\boldsymbol{e}_{k+1}] - \mathbb{E}[\boldsymbol{e}_\infty] &= \boldsymbol{A}\mathbb{E}[\boldsymbol{e}_k] - \mathbb{E}[\boldsymbol{e}_\infty], \end{aligned} \tag{2.46}$$

are asymptotically stable i.e., $\lim_{k\to\infty}(\mathbb{E}[\boldsymbol{x}_{k+1}] - \mathbb{E}[\boldsymbol{x}_\infty]) = \mathbf{0}$ and $\lim_{k\to\infty}(\mathbb{E}[\boldsymbol{e}_{k+1}] - \mathbb{E}[\boldsymbol{e}_\infty]) = \mathbf{0}$, therefore concluding the proof. ■

Similarly, with the Randomness Monitor augmented to CUSUM, the expected closed-loop system evolution under attack sequence (2.42) is described by

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}_{k+1}] &= (\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})\mathbb{E}[\boldsymbol{x}_k] - \boldsymbol{B}\boldsymbol{K}\mathbb{E}[\boldsymbol{e}_k],\\ \mathbb{E}[\boldsymbol{e}_{k+1}] &= \boldsymbol{A}\mathbb{E}[\boldsymbol{e}_k] - \boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^C].\end{aligned} \tag{2.47}$$

where $\mathbb{E}[\boldsymbol{r}_k^C] = (\mathbb{E}[r_{k,1}^C], \ldots, \mathbb{E}[r_{k,N_s}^C])^\mathsf{T}$ is the expected value of the residual sequence vector for CUSUM in (2.43).

Lemma 2.4 *Considering a closed-loop system from* (2.1) *and* (2.47)*, where* $\rho[\boldsymbol{A}] < 1$ *and attack sequence in* (2.42)*, the limit for expected state divergence* $\lim_{k\to\infty}\mathbb{E}[\boldsymbol{x}_k] = \Delta^C$ *is*

$$\Delta^C = (\boldsymbol{I}-\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})^{-1}\boldsymbol{B}\boldsymbol{K}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{L}\mathbb{E}[\boldsymbol{r}_k^C]. \tag{2.48}$$

Proof: The proof is omitted here due to space constraints but follows the proof for Lemma 2.3. ■

2.4.6 Simulation Results

The proposed randomness monitoring framework was validated in simulation and experiments, then compared to state-of-the-art detection techniques. The case study presented in this paper is an autonomous waypoint navigation of a skid-steering differential-drive UGV with a linearized model:

$$\begin{aligned}\dot{v} &= \frac{1}{m}(F_l + F_r - B_r v),\\ \dot{\omega} &= \frac{1}{I_z}\Big(\frac{w}{2}(F_l - F_r) - B_l\omega\Big),\ \dot{\theta} = \omega,\end{aligned} \tag{2.49}$$

where v is the velocity, θ is the vehicle's heading angle, and ω its angular velocity, forming the state vector $\boldsymbol{x} = [v, \theta, \omega]^\mathsf{T}$. The continuous-time model (2.49) is discretized to satisfy the system model described in (2.1).

In both simulation and experiment, we perform two different attack sequences: *Attack #1* where a stealthy attack sequence concentrates the residual distribution with a non-zero mean and smaller variance whereas *Attack #2* creates a signed pattern sequence {+, +, +, -} of residual differences $r'_{k,s}$. Both attacks are chosen to not increase the boundary detector (i.e., a traditional magnitude-based detector) alarm rate.

In our simulation case study, we show the effect of stealthy attacks on the velocity sensor with a monitoring window length $\ell = 100$. Table 2.1 gives the resulting alarm rates when our framework is

augmented to boundary detectors (BDD and CUSUM) with all detectors tuned for desired false alarm rates $\alpha^{\text{des}} \in \{.05, .20\}$. Under *Attack #1*, alarm rates for only the WSR increase to higher values and similarly the *Attack #2* pattern gives an increased alarm rate to only the SIR test. Figure 2.3 demonstrates attacks where our detectors are tuned for $\alpha_1^{\text{des}} = 0.10$ and compared with the CUSUM boundary detector. *Attack #1* occurs between $(50, 125)$s triggering the WSR test, *Attack #2* between $(175, 250)$s triggering the SIR test, and from 300s a third attack satisfying bounds for both tests but violating the CUSUM test is presented. Velocity is reduced as expected while experiencing the effects of each attack.

Table 2.1: Simulated Alarm Rates

State x_1 with $\ell = 100$		Randomness Monitor		Boundary Detectors	
$b_1 = 1.10$ *for CUSUM*		WSR	SIR	BDD	CUSUM
$\alpha^{\text{des}} = .20$	α_1 (No attack)	0.1809	0.2133	0.1941	0.1771
	α_1 (Attack 1)	0.9945	0.2089	0.1555	0.1569
	α_1 (Attack 2)	0.1874	1.0000	0.1800	0.2000
$\alpha^{\text{des}} = .05$	α_1 (No attack)	0.0392	0.0492	0.0519	0.0340
	α_1 (Attack 1)	0.9982	0.0513	0.0393	0.0361
	α_1 (Attack 2)	0.0377	1.0000	0.0500	0.0500

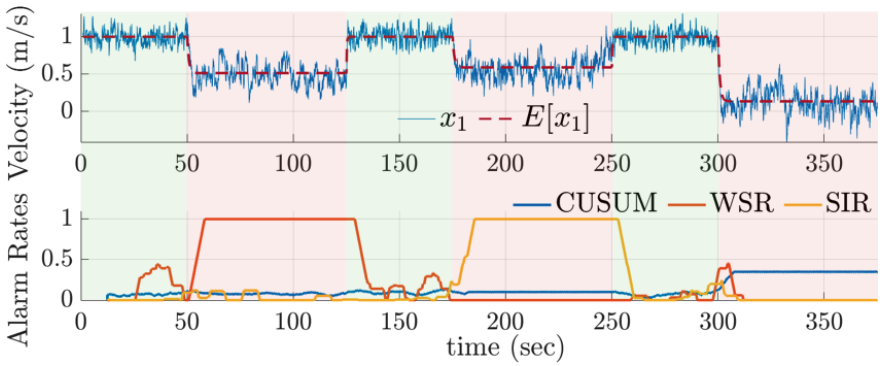


Figure 2.3: State deviation and alarm rates under various attacks over a moving window of length 100.

2.4.7 Experiment Results

In this section we present a case study for a UGV performing way-point navigation under stealthy sensor attacks. For our case, the UGV travels to a series of goal positions while avoiding a restricted area with a desired cruise velocity $v^{\text{ref}} = 0.15$m/s while experiencing the same class of attacks as in Section [Simulation]. This time the IMU sensor that measures angle θ is spoofed while our Randomness Monitor is augmented with the BDD. Fig. 2.4 shows the UGV position while traveling to the four goal points. For both attacks the vehicle enters the restricted area (marked by red tape)

while the boundary detector (BDD) does not see the attack in each case. The alarm rate for the WSR test increases for the case under *Attack #1* (solid line) and the SIR test alarm rate increases during the case for *Attack #2* (dashed line), as expected.

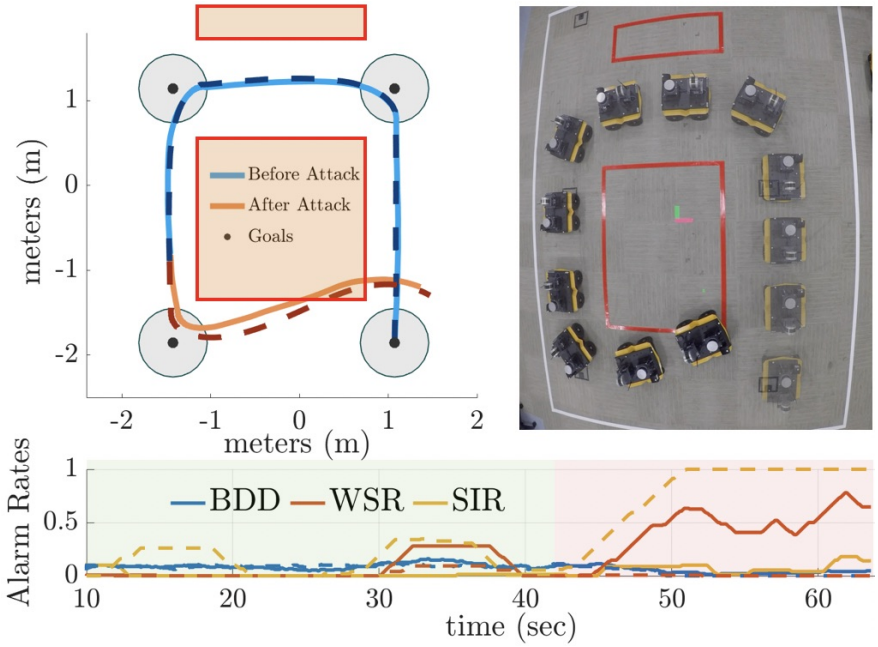


Figure 2.4: UGV position under *Attack #1* (solid line) and *Attack #2* (dashed line) with accompanying resulting alarm rates.

2.5 Cumulative Sign Attack Detector

The overall cyber-physical system architecture including the CUSIGN detector is summarized in Fig. 2.5. CUSIGN, which can be augmented to any boundary detector providing magnitude bounds, is placed in the system feedback to monitor the relationship between measurement and state prediction. We focus on stealthy sensor attacks where an attacker may inject an attack signal at any point between the sensors and the state estimator, in an attempt to affect system behavior.

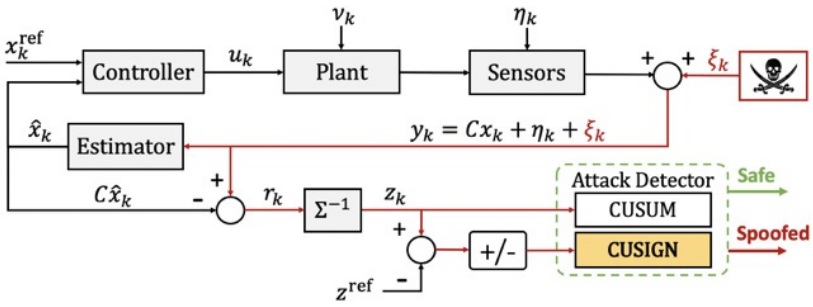


Figure 2.5: The detection architecture of a CPS to monitor for sensor attacks with the CUSIGN detector.

2.5.1 Test Measure Reference Point

We develop a Cumulative Sign (CUSIGN) detector that analyzes the sign of the given test measure z_k relative to a reference point and determines whether there is non-random behavior occurring. The model-based CUSIGN detector monitors the chi-square test measure z_k and outputs an alarm when the CUSIGN test variable reaches a user defined threshold. For a given user defined threshold, an expected alarm rate can be found that is independent from the model of the system (2.1).

In normal conditions, i.e., without attacks or sensor malfunctions, the test measure z_k has a specific probability of being higher or lower than a given user defined reference point $z^{\text{ref}} \in \mathbb{R}_{>0}$ within its known distribution. We formalize these probabilities of z_k being higher or lower than the reference point by

$$\begin{aligned} \Pr\left(z_k < z^{\text{ref}}\right) &= \gamma\Big(\frac{s}{2}, \frac{z^{\text{ref}}}{2}\Big), \\ \Pr\left(z_k > z^{\text{ref}}\right) &= 1 - \gamma\Big(\frac{s}{2}, \frac{z^{\text{ref}}}{2}\Big), \end{aligned} \tag{2.50}$$

where $\gamma(\cdot,\cdot)$ is the *regularized lower incomplete gamma function* [91]. The sign of z_k with respect to the reference z^{ref} is computed by the following

$$\operatorname{sgn}(z_k - z^{\text{ref}}) := \begin{cases} -1, & \textbf{if } z_k - z^{\text{ref}} < 0, \\ 0, & \textbf{if } z_k - z^{\text{ref}} = 0, \\ 1, & \textbf{if } z_k - z^{\text{ref}} > 0, \end{cases} \tag{2.51}$$

where the probability of each scenario occurring is

$$\begin{aligned} \Pr\big(\operatorname{sgn}(z_k - z^{\text{ref}}) = -1\big) &= p_-, \\ \Pr\big(\operatorname{sgn}(z_k - z^{\text{ref}}) = 0\big) &= 0, \\ \Pr\big(\operatorname{sgn}(z_k - z^{\text{ref}}) = 1\big) &= p_+. \end{aligned} \tag{2.52}$$

An example of (2.52) is shown in Fig. 2.6, where the probabilities p_+ and p_- determine whether z_k will be higher or lower than z^{ref} given $z_k \sim \chi^2$.

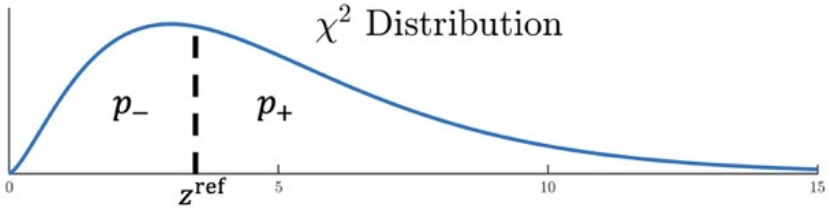


Figure 2.6: Probabilities p_+ and p_- determined by z^{ref}.

2.5.2 Detector Design

The procedure of CUSIGN is an accumulation of signed values, denoted by the CUSIGN test variables S_k^+ and S_k^-. Each variable is a monitor checking for a change in the probability of the signed value $\operatorname{sgn}(z_k - z^{\text{ref}})$, one for *positive* and the other for *negative* changes. The following procedure summarizes the CUSIGN detection in both the positive and negative cases:

CUSIGN Detector Procedure

$$\begin{aligned} &\textbf{Initialize } S_0^+ = 0, \\ &S_k^+ = \max\big(0, S_{k-1}^+ + \operatorname{sgn}(z_k - z^{\text{ref}})\big), \\ &S_k^+ = 0 \text{ and Alarm } \zeta_k^+ = 1, \qquad \textbf{if } S_{k-1}^+ = \tau, \\ &\textbf{Initialize } S_0^- = 0, \\ &S_k^- = \min\big(0, S_{k-1}^- + \operatorname{sgn}(z_k - z^{\text{ref}})\big), \\ &S_k^- = 0 \text{ and Alarm } \zeta_k^- = 1, \qquad \textbf{if } S_{k-1}^- = -\tau. \end{aligned} \tag{2.53}$$

The design of the test variable sequences S_k^+ and S_k^- are to accumulate the signed value $\text{sgn}(z_k - z^{\text{ref}}) \in \{-1, 0, 1\}$ and triggering an alarm $\zeta_k^{\pm} = \{\zeta_k^+, \zeta_k^-\} \in \{0, 1\}$ when the test variables reach the threshold values $\tau \in \mathbb{N}$. When either of the test variables are equal to their corresponding thresholds, the given test variable is reset to 0. An example of the CUSIGN test variable is shown in Fig. 2.7 where three consecutive iterations $z_k > z^{\text{ref}}$ are satisfied at $k = 1, 2, 3$ (transitioning S_k^+ in the direction of p_+). At $k = 3$, the CUSIGN test variable S_k^+ reaches the threshold value $\tau = 3$ causing a reset such that $S_k^+ \to 0$.

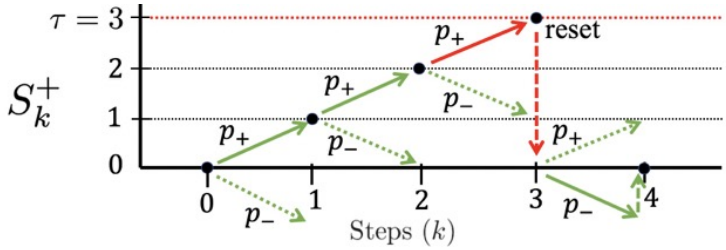


Figure 2.7: Transitions of the CUSIGN test variable S_k^+ with threshold $\tau = 3$.

Choosing a specific threshold τ results in expected alarm rates $\mathbb{E}[\alpha^+]$ and $\mathbb{E}[\alpha^-]$ for both the positive and negative cases of the CUSIGN procedure (2.53). In the case that $z^{\text{ref}} = \mathbb{E}[\text{median}(z_k)]$ such that $p_+ = p_-$, the resulting expected alarm rates are equal $\mathbb{E}[\alpha^+] = \mathbb{E}[\alpha^-]$.

Similar to the implementation in [71], the transition of the CUSIGN test sequences $S_k^{\pm}$ can be constructed as a Markov chain with a transition matrix modeled from the probabilities of $\text{sgn}(z_k - z^{\text{ref}})$. With a user defined threshold τ to trigger an alarm and causing a reset condition of the CUSIGN test variable to 0, we show the transitions of $S_k^{\pm}$ with a Markov chain diagram, as follows in Fig. 2.8.

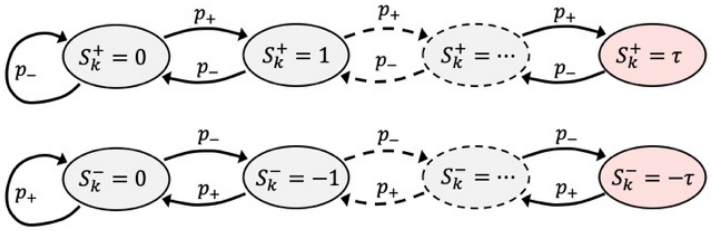


Figure 2.8: Markov chain for both CUSIGN test sequences with threshold τ.

Given a chosen threshold value $\tau \in \mathbb{N}$ as a value that triggers an alarm when $|S_k^{\pm}| = \tau$, we describe the Markov chain in Fig. 2.8 in the form of a Markov transition matrix $\mathcal{T}^{\pm} \in \mathbb{R}^{(\tau+1)\times(\tau+1)}$. The CUSIGN Markov Chain, occurring in a discrete manner, contains $\tau + 1$ states denoted as $\mathcal{M} = \{M_0, M_1, \ldots, M_\tau\}$ where M_τ is an absorbing state that is equal to the threshold, causing the CUSIGN test sequence $S_k^{\pm}$ to reset to M_0 (i.e., $S_k^{\pm} = 0$). The CUSIGN Markov transition matrix $\mathcal{T}^{\pm}$ for both positive $\mathcal{T}^+$ and negative $\mathcal{T}^-$ cases with a probability distribution of $\text{sgn}(z_k - z^{\text{ref}})$ are

written by

$$\mathcal{T}^{\pm} = \begin{bmatrix} p_{\mp} & p_{\pm} & 0 & 0 & \dots & 0 \\ p_{\mp} & 0 & p_{\pm} & 0 & \dots & 0 \\ 0 & p_{\mp} & 0 & p_{\pm} & & 0 \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \dots & 0 & p_{\mp} & 0 & p_{\pm} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.54}$$

The transition matrix $\mathcal{T}^{\pm}$ structure remains the same on any system, where the matrix size depends only on the value of the threshold τ. Transition probabilities for transient states in $\mathcal{T}^{\pm}$ adhere to the following:

$$\mathcal{T}^{\pm} = \begin{cases} \Pr(M_j \to M_{j+1}) = p_{\pm}, & \text{for } j = \{0, \dots, \tau - 1\}, \\ \Pr(M_j \to M_{j-1}) = p_{\mp}, & \text{for } j = \{1, \dots, \tau - 1\}, \\ \Pr(M_0 \to M_0) = p_{\mp}, \end{cases} \tag{2.55}$$

and the final row represents an absorbing state containing elements equal to 0 besides the last element equaling 1.

We define $\mathcal{R}^{\pm} \in \mathbb{R}^{\tau \times \tau}$ as a matrix obtained from $\mathcal{T}^{\pm}$ with its last row and column removed (i.e., the absorbing state at threshold τ is removed), representing the transition probabilities to and from the transient states, also known as the fundamental matrix. Elements of $\mathcal{R}^{\pm}$ are all non-negative and row sums are equal to or less than one, while the eigenvalues satisfy $\rho[\mathcal{R}^{\pm}] < 1$ such that $(\mathcal{R}^{\pm})^i \to 0$ as $i \to \infty$ and $\sum_{i=0}^{\infty} (\mathcal{R}^{\pm})^i = (\boldsymbol{I}_\tau - \mathcal{R}^{\pm})^{-1}$, where $\rho[\cdot]$ is the spectral radius and $\boldsymbol{I}_\tau$ is the identity matrix of size τ.

Lemma 2.5 *Given a system with a CUSIGN detector* (2.53) *with a chosen threshold* $\tau \in \mathbb{N}$ *and reference point* $z^{ref} \in \mathbb{R}_{>0}$ *that is not affected by sensor attacks such that the residual sequence satisfies* $\boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}) \in \mathbb{R}^{N_s}$ *and* $z_k = \boldsymbol{r}_k^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_k \sim \chi^2$ *with* N_s *degrees of freedom, then the inverse of the first element of the following vector*

$$\boldsymbol{\mu}^{\pm} = (\boldsymbol{I}_\tau - \mathcal{R}^{\pm})^{-1} \mathbf{1}_{\tau \times 1} = (\mu_1^{\pm}, \dots, \mu_\tau^{\pm})^{\mathsf{T}} \tag{2.56}$$

is the expected alarm rate, i.e., $\mathbb{E}[\alpha^{\pm}] = (\mu_1^{\pm})^{-1}$.

Proof: Given the Markov chain containing $\tau + 1$ states denoted by $\mathcal{M} = \{M_0, M_1, \dots, M_\tau\}$, a fundamental matrix $\mathcal{R}^{\pm}$ is taken from a designed Markov transition matrix (2.54) to satisfy the transition probabilities (2.55). Leveraging the theory of average run length (ARL) in CUSUM [10], the ARL is defined as the average length of time for the test sequence to reach the threshold τ to trigger an alarm, determined by the fundamental matrix $\mathcal{R}^{\pm}$ containing the transient states within

$\mathcal{T}^{\pm}$. By definition, the inverse of the ARL to observe an alarm results in the average frequency of obtaining an alarm, known as the alarm rate. The ARL can be found by computing (2.56), then by inverting the first element of $\boldsymbol{\mu}^{\pm}$, i.e., $(\mu_1^{\pm})^{-1}$, we obtain the expected alarm rate $\mathbb{E}[\alpha^{\pm}]$. ■

2.5.3 Window-less Alarm Rate Estimation

In the design of CUSIGN, we trigger an alarm when a test variable reaches a chosen threshold τ. Given a system not experiencing sensor attacks, we have an expectation of the alarm rates. Typically, to find an alarm rate, the number of triggered alarms are tallied over a given period of time. Here, we want to create a "memoryless" procedure to find an alarm rate.

The conventional method of finding an average $\bar{x}$ of a stochastic variable is $\bar{x}_n = \frac{1}{n}\left[\sum_{i=1}^{n} x_i\right]$ where n is the size the data set. This procedure requires storage of the complete data set, where computation becomes less efficient as n grows. A memoryless online algorithm known as Welford's online algorithm for computing a mean incrementally was developed in [108] by transforming the conventional method into an online update by the following form

$$\begin{aligned} \bar{x}_n &= \frac{1}{n}\left[x_n + \sum_{i=1}^{n-1} x_i\right] = \frac{1}{n}\left[x_n + (n-1)\bar{x}_{n-1}\right] \\ &= \frac{1}{n}\left[x_n + n\bar{x}_{n-1} - \bar{x}_{n-1}\right] = \bar{x}_{n-1} + \frac{\left[x_n - \bar{x}_{n-1}\right]}{n}. \end{aligned} \tag{2.57}$$

It can be seen in (2.57) that n grows indefinitely, equal to the number of data points. We set a maximum value for n such that $\max(n) = \ell \in \mathbb{N}$ to create a "pseudo-window" for a rolling sequential estimation of an expected mean. We name this modified version of Welford's online algorithm utilizing a pseudo-window ℓ as a Memoryless Run-time Estimator (MRE). The behavior of MRE when computing the mean similarly imitates the conventional method of calculating the mean consisting of ℓ data points, but without the need to store the entire sequence.

For the case of attack detection using alarm rates for CUSIGN, we leverage MRE in (2.57) to find an online estimation of an expected alarm rate $\mathrm{E}[\alpha]$ (we omit $\pm$ for $\alpha^{\pm}$ in this section as the MRE applies to both the positive and negative cases). Leveraging the pseudo-window of length ℓ and replacing the counter n from (2.57) with k for sampling time instances, we attain the equation

$$\hat{\alpha}_k = \hat{\alpha}_{k-1} + \frac{\left[\zeta_k - \hat{\alpha}_{k-1}\right]}{\ell}, \tag{2.58}$$

where ζ_k is the triggered alarm for CUSIGN, $\hat{\alpha}_k$ is an estimate of the alarm rate at time instance k, and $\hat{\alpha}_0 = 0$ initially at $k = 0$.

Proposition 2.2 *Assuming the system is not experiencing sensor attacks and the test measure follows $z_k \sim \chi^2$ for time instances $k \geq 0$, we empirically find that the alarm rate is a Normal distribution as follows*

$$\hat{\alpha} \sim \mathcal{N}\bigg(\mathrm{E}[\alpha], \frac{\theta \mathrm{E}[\alpha](1-\mathrm{E}[\alpha])}{\ell}\bigg), \tag{2.59}$$

where ℓ is the user defined pseudo-window length, $\theta \in \mathbb{R}^{>0}$ is an empirically found scaling value, and $\mathrm{E}[\alpha]$ is the expected alarm rate, i.e., the probability that the test variable S_k reach the threshold, triggering an alarm $\zeta_k = 1$.

Given the distribution of $\hat{\alpha}$ in Proposition 2.2, the expectation of the estimated alarm rate follows

$$\mathrm{E}[\hat{\alpha}] = \mathrm{E}[\alpha], \quad \mathrm{Var}[\hat{\alpha}] = \frac{\theta \mathrm{E}[\alpha](1-\mathrm{E}[\alpha])}{\ell}. \tag{2.60}$$

Values of θ are found to be dependent on the chosen threshold τ. Observed approximates of θ are presented in Table 2.2 for thresholds $\tau = 1, 2, 3, 4$ and $\ell \geq 10$.

Table 2.2: Empirical values for the scaling value θ given thresholds $\tau = 1, 2, 3, 4$.

Thresholds	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$
θ	$\frac{\ell}{2\ell-1}$	$\frac{.74\ell}{2\ell-1}$	$\frac{.7\ell}{2\ell-1}$	$\frac{.69\ell}{2\ell-1}$

Remark 2.4 *For the CUSIGN detector, we empirically find that $\hat{\alpha}$ follows* (2.59) *when $p_+ \approx p_-$ (i.e., z^{ref} is chosen to be at $\mathbb{E}[median(z_k)]$ such that $p_- = p_+ = 0.5$). For a reference point z^{ref} not placed near the expected distribution median, i.e., $p_+ \not\approx p_-$, we found that the distribution of $\hat{\alpha}$ loses properties of the Normal distribution in* (2.59). *Empirical results for observed $\hat{\alpha}$ and $\mathrm{Var}[\hat{\alpha}]$ considering the case when $p_+ \not\approx p_-$ can be found in Section 2.5.6.*

By leveraging the distribution of the estimated alarm rate in (2.59), bounds of the alarm rate can be made.

Lemma 2.6 *Assuming an uncompromised system with a CUSIGN detector* (2.53) *with a reference point z^{ref} and threshold $\tau \in \mathbb{N}$, detection of sensor attacks occurs when $\tau_-^\alpha \leq \hat{\alpha} \leq \tau_+^\alpha$ where*

$$\tau_\pm^\alpha = \mathbb{E}[\alpha] \pm Z\sqrt{\frac{\theta \mathbb{E}[\alpha](1-\mathbb{E}[\alpha])}{\ell}}. \tag{2.61}$$

Proof: Given a CUSIGN detector with threshold $\tau \in \mathbb{N}$ and reference point $z^{\mathrm{ref}} \in \mathbb{R}_{>0}$ that determine transition probabilities p_- and p_+, an expected alarm rate $\mathbb{E}[\alpha]$ can be computed by inverting the first element in (2.56). With $\mathbb{E}[\alpha]$ and leveraging the Memoryless Run-time Estimator

with a pseudo-window of length ℓ, the distribution of the estimated alarm rate follows $\hat{\alpha} \sim \mathcal{N}(\cdot, \cdot)$ with properties from (2.60). Detection bounds $\tau_{\pm}^{\alpha}$ of a specific confidence level determined by Z of a Normally distributed random variable with properties from (2.60) follow:

$$\mathbb{E}[\alpha] - Z\sqrt{\frac{\theta\mathbb{E}[\alpha](1-\mathbb{E}[\alpha])}{\ell}} \leq \hat{\alpha}_{\pm} \leq \mathbb{E}[\alpha] + Z\sqrt{\frac{\theta\mathbb{E}[\alpha](1-\mathbb{E}[\alpha])}{\ell}}, \tag{2.62}$$

satisfying (2.61), concluding the proof. ■

Detection of sensor attacks occur when an estimated alarm rate $\hat{\alpha}$ goes beyond a threshold from $\tau_{\pm}^{\alpha} = \{\tau_{-}^{\alpha}, \tau_{+}^{\alpha}\}$. Lower bounds resulting in $\tau_{-}^{\alpha} < 0$ are omitted as $\hat{\alpha} \in [0, \frac{1}{\tau}]$.

2.5.4 CUSUM Detector for Comparison

The CUSIGN detector alone may not be sufficient as an attacker can change the magnitude of a measurement, but still maintain random signed behavior of the test measure z_k. The non-parametric quality of CUSIGN results in the inability to monitor the magnitude of the test measure. A well-known dynamic detector, the *CUmulative SUM* (CUSUM) detector, leverages the magnitude of the test measure sequence z_k to look for changes in the mean from an expectation. Formalized into a model-based attack detector by [71] that outputs an alarm, the CUSUM attack detection procedure follows

CUSUM Detector Procedure

$$\begin{cases} \textbf{Initialize } C_0 = 0, & \\ C_k = \max(0, C_{k-1} + z_k - b), & \textbf{if } C_{k-1} \leq \tau^C, \\ C_k = 0 \text{ and Alarm } \zeta_k^C = 1, & \textbf{if } C_{k-1} > \tau^C. \end{cases} \tag{2.63}$$

The working principle of this detector is to accumulate the test measure z_k in C_k, triggering an alarm $\zeta_k^C = 1$ when the test variable surpasses the threshold τ^C. The test variable C_k resets to zero either when the threshold τ^C is surpassed or when C_k goes negative. A bias b is selected based on properties of $\mathbf{\Sigma}$ such that C_k does not grow unbounded. A detailed explanation of how to construct a transition matrix for the probability distribution $z_k - b$ for the model-based CUSUM can be found in [71]. The authors provide a method for tuning the threshold τ^C given a bias b for a desired alarm rate $\mathbb{E}[\alpha^C]$ with an assumption that the system is free of sensor attacks, where the residual follows $\boldsymbol{r}_k \sim \mathcal{N}(0, \mathbf{\Sigma})$, hence a shifted χ^2 distribution $z_k - b = \boldsymbol{r}_k^{\mathsf{T}}\mathbf{\Sigma}^{-1}\boldsymbol{r}_k - b$.

Considering CUSUM as a stand-alone detector, an adversarial wants to avoid attacks such that the test variable C_k exceeds threshold τ^C at a higher rate, thereby causing a reset $C_k = 0$ in (2.34) by satisfying the CUSUM procedure sequence $C_k = \max(0, C_{k-1} + z_k - b) \leq \tau^C$ to trigger alarms more often, resulting in a higher alarm rate α^C. Subsequently, an attacker can design an attack such that it remains within bounds of CUSUM to not trigger alarms more than expected. To include

this attack vector, we can rewrite the CUSUM procedure such that

$$C_k = \max(0, C_{k-1} + \left(\|\boldsymbol{\Sigma}^{-\frac{1}{2}}(\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k)\|^2\right) - b). \tag{2.64}$$

Assuming that a malicious attacker can have access to the sensor measurements $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{\eta}_k$ and has perfect knowledge of the state estimator, it will be able to find the estimator output $\boldsymbol{C}\hat{\boldsymbol{x}}_k$. With this information, the attacker can solve for $\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k = \boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k$ to achieve the ability of manipulating elements of $\boldsymbol{\xi}_k$ by

$$\boldsymbol{\xi}_k = -\boldsymbol{C}\boldsymbol{e}_k - \boldsymbol{\eta}_k + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}_k^{\tau^C} \tag{2.65}$$

such that $\max(0, C_{k-1} + (\boldsymbol{\xi}_k^{\tau^C})^\mathsf{T}\boldsymbol{\xi}_k^{\tau^C} - b) \leq \tau^C$ can maintain the test variable C_k within the detection threshold τ^C.

2.5.5 Simulation Results

The proposed CUSIGN detector was validated in simulation and augmented with the CUSUM detector. The case study presented is an autonomous way-point navigation of a skid-steering differential-drive UGV with the following linearized model [73]:

$$\begin{aligned} \dot{v} &= \frac{1}{m}(F_l + F_r - B_r v), \\ \dot{\omega} &= \frac{1}{I_z}\Big(\frac{w}{2}(F_l - F_r) - B_l\omega\Big),\ \dot{\theta}_h = \omega, \end{aligned} \tag{2.66}$$

where v, θ_h, and ω denotes the velocity, heading angle, and angular velocity, forming the state vector $\boldsymbol{x} = [v, \theta_h, \omega]^\mathsf{T}$. F_l and F_r describe the left and right input forces from the wheels, w is the vehicle width, while B_r and B_l are resistances due to the wheels rolling and turning. The continuous-time model (2.66) is discretized with a sampling rate $t_s = 0.01$ to satisfy the system model described in (2.1). The UGV is tasked to continuously navigate to four goal-points along a square trajectory with side lengths of 5m maintaining a velocity $v = 0.5$m/s for 200s.

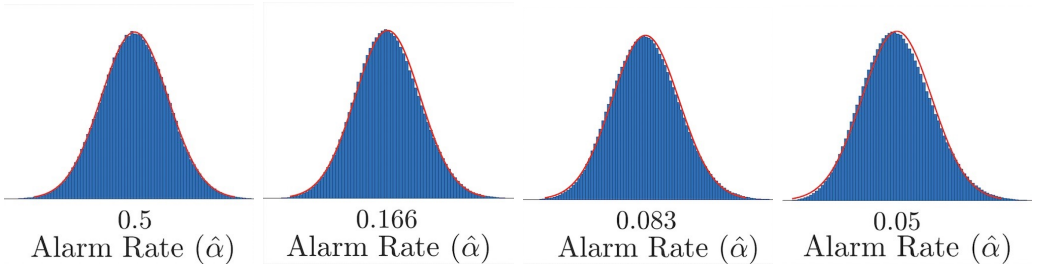
In the simulation, we perform two different attack sequences on the velocity sensor on-board the vehicle: 1) a persistent attack and 2) an alternating pattern attack. Both stealthy attack sequences are designed to be undetectable by CUSUM, but are detected by CUSIGN due to the creation of non-random patterns.

A system is first considered under nominal conditions where $\boldsymbol{\xi}_k = 0$. In Table 2.3 we show the alarm rate of the system over 5 million data samples and compare the results to the expected alarm rate $\mathbb{E}[\alpha^\pm]$ computed from (2.56) in the case where $p_+ = p_- = 0.5$ for thresholds $\tau = 1, 2, 3, 4$. Next,

in Fig. 2.9 we show the distribution of the alarm rate estimate $\hat{\alpha}$ from the four cases in Table 2.3 overlaid with the expected distributed curve (in red) according to (2.60).

Table 2.3: $\mathbb{E}[\alpha^{\pm}]$ when $p_+ = p_- = 0.5$.

Thresholds	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$
$\mathbb{E}[\alpha^{\pm}]$	0.5	$0.1\bar{6}$	$0.08\bar{3}$	0.05
$\alpha^{\pm}$ (sim.)	.50006	.16692	.083291	.050012



(a) (b) (c) (d)

Figure 2.9: Resulting distributions of $\hat{\alpha}$ when $p_+ = p_- = 0.5$ for (a) $\tau = 1$, (b) $\tau = 2$, (c) $\tau = 3$, (d) $\tau = 4$.

Now, considering the UGV (2.66) case study in the presence of hidden attacks on the velocity sensor on state $x_1 = v$, we show the detection capabilities of CUSIGN. The CUSIGN is designed with $z^{\text{ref}} = \mathbb{E}[\text{median}(z_k)] \approx s\big(1 - \frac{2}{9s}\big)^3$ where $s = 3$ such that the transition probabilities satisfy $p_{\pm} = 0.5$ and threshold $\tau = 2$. The expected alarm rate $\mathbb{E}[\alpha] = 0.1\bar{6}$ and the Memoryless Run-time Estimator (2.58) with pseudo-window length $\ell = 100$ has detection bounds (2.61) at $\tau_-^{\alpha} = 0.0987$ and $\tau_+^{\alpha} = 0.2347$ where $Z = 3$ for a 99.7% confidence. The design of CUSUM contains a bias $b = 1.1s = 3.3$ with a threshold $\tau^C = 2.3226$ to satisfy an expected alarm rate $\mathbb{E}[\alpha^C] = 0.15$ (see [71] for tuning details), where the alarm rate is computed by a conventional method of length ℓ by $\frac{1}{\ell}\sum_{k-\ell+1}^{k} \zeta_k^C$. Fig. 2.10 shows the results of a persistent attack (2.64), (2.65) beginning at $k = 10,000$ with a noiseless magnitude of $0.1\tau^C$. The alarm rate $\hat{\alpha}^C$ for CUSUM is unaffected while CUSIGN discovers the attack and alarm rates $\hat{\alpha}^{\pm}$ both go beyond the detection bounds $\tau_{\pm}^{\alpha}$ (red dashed lines). A second attack shown in Fig. 2.11 is attempted with an alternating noiseless pattern of $\{0.1\tau^C, -0.1\tau^C\}$ to show that CUSIGN can detect patterns. Again, alarm rates for CUSIGN find the non-random patterns and go beyond the detection bounds $\tau_{\pm}^{\alpha}$ while CUSUM is not able to detect the non-random behavior.

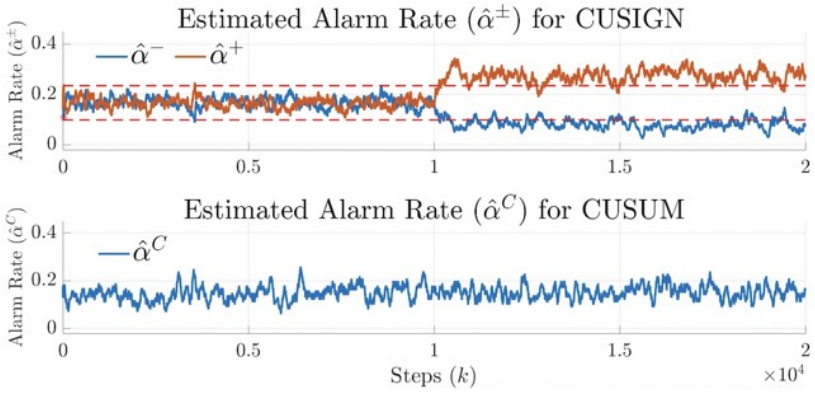


Figure 2.10: Alarm rates $\hat{\alpha}^{\pm}$ and $\hat{\alpha}^C$ for both CUSIGN and CUSUM with a hidden persistent sensor attack.

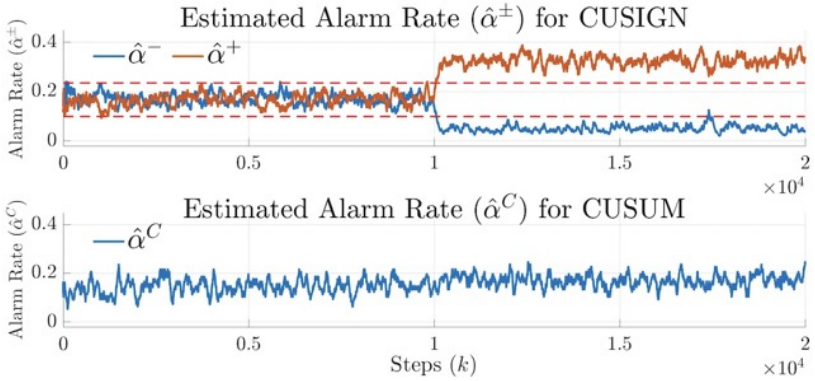
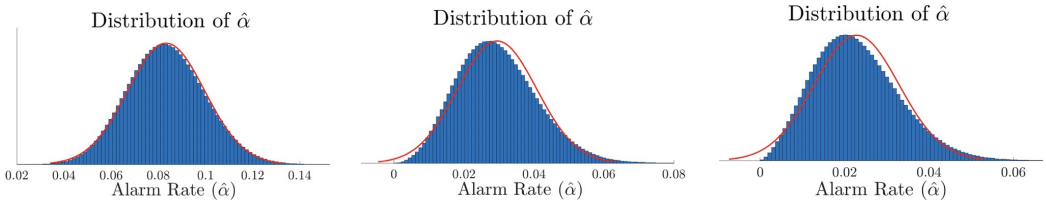


Figure 2.11: Alarm rates $\hat{\alpha}^{\pm}$ and $\hat{\alpha}^C$ for both CUSIGN and CUSUM with a hidden alternating sensor attack.

2.5.6 Empirical Results

From Remark 2.4, we show in Fig. 2.12 the gradual divergence from the normal approximation as p_+ and p_- are no longer similar as the distribution of the estimated alarm rate estimate $\hat{\alpha}$ becomes skewed. The empirical results provided throughout this section are results from 5 million samples, thus giving an accurate representation of the resulting distributions.



(a) (b) (c)

Figure 2.12: Resulting distributions of $\hat{\alpha}$ when (a) $p_+ = p_- = 0.5$, (b) $p_- = 0.37$, (c) $p_- = 0.3$.

Furthermore, Table 2.4 provides the expected $\mathbb{E}[\hat{\alpha}]$ and simulated alarm rates, while Table 2.5

provides the square root of the expected and simulated variance $\sqrt{\text{Var}[\hat{\alpha}]}$ (i.e., standard deviation). It can be seen that as $p_{\pm} \approx 0.5$, the simulated mean of the alarm rate estimates remain approximately equal to the expectation (i.e., $\bar{\hat{\alpha}} \approx \mathbb{E}[\alpha]$), but the simulation results for standard deviation diverge from the expected variance as $p_+ \not\approx p_-$.

Table 2.4: Results of $\mathbb{E}[\hat{\alpha}]$ for $\ell = 100$.

$p_{\pm}$	.4	.5	.6
$\mathbb{E}[\hat{\alpha}]$/sim (τ=1)	.400/4.01	.500/.500	.600/6.01
$\mathbb{E}[\hat{\alpha}]$/sim (τ=2)	.1143/.1142	.166$\bar{6}$/.1665	.2250/.2251
$\mathbb{E}[\hat{\alpha}]$/sim (τ=3)	.0484/.0483	.083$\bar{3}$/.0832	.1256/.1258
$\mathbb{E}[\hat{\alpha}]$/sim (τ=4)	.0244/.0239	.0500/.0500	.0835/.0833

Table 2.5: Results of std$[\hat{\alpha}]$ for $\ell = 100$.

$p_{\pm}$	.4	.5	.6
std$[\hat{\alpha}]$/sim (τ=1)	.0346/.0347	.0354/.0355	.0346/.0347
std$[\hat{\alpha}]$/sim (τ=2)	.0194/.0204	.0227/.0226	.0254/.0238
std$[\hat{\alpha}]$/sim (τ=3)	.0127/.0138	.0163/.0163	.0196/.0185
std$[\hat{\alpha}]$/sim (τ=4)	.0091/.0099	.0128/.0128	.0163/.0153

2.6 Serial Randomness Attack Detector

Thus far in this chapter, both a window-based and a window-less monitoring frameworks have been introduced to detect non-random measurement residual behavior in the presence of cyber attacks to sensor measurements. However, if intelligent attackers have gained enough knowledge about the system at hand such as the dynamical model, state estimator, noise model, and on-board detection schemes, the attacker can create specific attack vector sequences to remain hidden from intrusion detectors. With these attack vector sequences, non-random patterns arise within the chi-square test measure that can then be exploited by defensive mechanisms to raise a flag when these inconsistencies are present.

2.6.1 Background on Undetectable Hidden Attacks

A successful attacker is capable of modeling an attack sequence to achieve a desirable effect while remaining undetectable to any on-board fault detection mechanisms. In order to accomplish such stealthy behavior, it is necessary to attain information about critical aspects of the system, such as: acquiring knowledge to the modeled dynamics, sensor measurements, state estimator, and detection procedure(s). To intentionally avoid detection, an intelligent attacker will carefully construct an attack sequence to evade raising any flags. Below we describe a sequence an attacker may take with respect to the Bad-Data detector [67] leveraging the chi-square test measure procedure. However, this may be extended to satisfy any detection procedure using a similar concept to avoid detection.

Zero-alarm attacks are sequences designed by an attacker that maintains the test measure from exceeding the defined threshold value ($z_k \leq \tau_z$). This class of attack does not trigger an alarm throughout the attack sequence, as the test measure never passes the threshold. In order to satisfy such requirements, an attacker can construct the attack vector by

$$\boldsymbol{\xi}_k = -\boldsymbol{C}\boldsymbol{e}_k - \boldsymbol{\eta}_k + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\delta}_k \tag{2.67}$$

where $\boldsymbol{\delta}_k \in \mathbb{R}^{N_s}$ is a vector that satisfies $\boldsymbol{\delta}_k^{\mathsf{T}}\boldsymbol{\delta}_k \leq \tau_z$. With this attack vector designed at a time k, the test measure z_k results in

$$\begin{aligned} z_k &= (\tilde{\boldsymbol{y}}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{y}}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k) \\ &= (\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k) \leq \tau_z \end{aligned} \tag{2.68}$$

that remains within the threshold value to not trigger an alarm. While generating an attack sequence that does not trigger alarms may seem like a favorable attack design, it is necessary to recall that alarms are triggered in a system operating in normal conditions without attacks. If alarms are

no longer being triggered as designed for in an attack-free case, then these conditions may raise suspicions of a possible attack. To avoid these alarm rate discrepancies, an attacker would want to design an attack sequence that is undetectable to emulate normal (attack-free) conditions. This class of attack brings us to develop a sequence that exploits the system uncertainties to execute such a malicious attack.

Hidden attacks can be defined as designed attack sequences such that alarms are triggered at the same rate as the desired false alarm rate during nominal, attack-free operation. As shown in Fig. 2.13, during a hidden attack, a smart attacker can design a sequence where the test measure z_k exceeds the threshold τ_z at the same rate as nominal conditions. More specifically, in the top graph of Fig. 2.13 is the chi-square distribution of z_k with $N_s = 4$ degrees of freedom (dof) under nominal conditions. A correctly chosen threshold value τ_z results in the test measure z_k exceeding the threshold at a desired rate of α in the attack-free case. Similarly, during a hidden attack (in the bottom graph), an attacker can design an attack sequence such that the alarm rate matches the desired alarm rate α, all while altering the distribution of z_k and remaining hidden from detection.

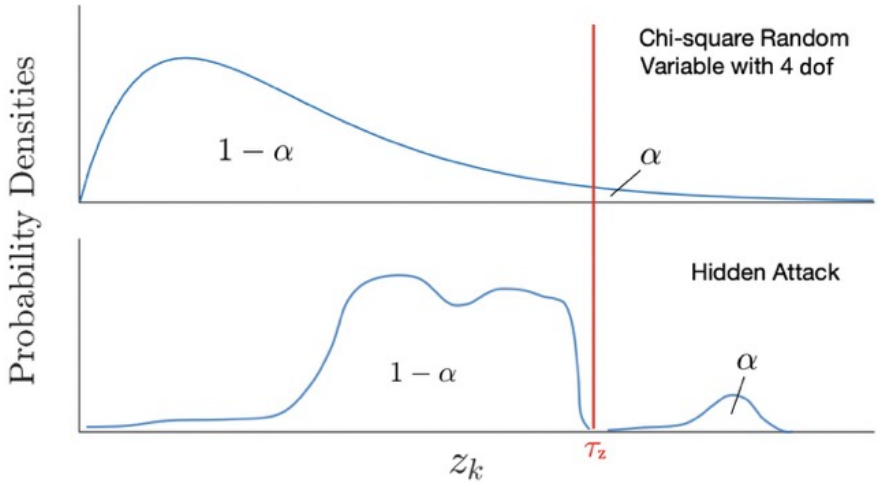


Figure 2.13: An example of a hidden attack within the chi-square detection scheme.

To tune for a desired alarm rate α (in the attack-free scenario) for Bad-Data detection while leveraging the chi-square procedure, the specific threshold τ_z is found by

$$\tau_z = 2\gamma^{-1}\Big(1-\alpha, \frac{N_s}{2}\Big) \tag{2.69}$$

to achieve a desired alarm rate, where $\gamma^{-1}(\cdot,\cdot)$ is the *inverse regularized lower incomplete gamma function* [91]. The vector $\boldsymbol{\delta}_k$ from (2.67) is designed such that

$$\Pr(z_k > \tau_z) = \Pr(\boldsymbol{\delta}_k^{\mathsf{T}}\boldsymbol{\delta}_k > \tau_z) \approx \alpha \tag{2.70}$$

to remain hidden from detection.

2.6.2 Magnitude-based Detection

The design of the Serial Detector is to find inconsistent behavior of chi-square test measures within its expected distribution due to stealthy attacks to on-board sensor measurements. An attacker deliberately attempting to fool test measure-based detection algorithms may leave traces of inconsistencies within the serial sequence. We propose the Serial Detector that analyzes consecutive chi-square test measures at time instances k and $k-1$, called the *test measure difference*, that is described as:

$$\begin{aligned} d_k &= z_k - z_{k-1} \\ &= \boldsymbol{r}_k^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_k - \boldsymbol{r}_{k-1}^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_{k-1} \in \mathbb{R}. \end{aligned} \tag{2.71}$$

Proposition 2.3 *A system that is free from sensor attacks, where we assume consecutive test measures are independent random variables that follow chi-square distributions $z_k, z_{k-1} \sim \chi^2(N_s)$ with s degrees of freedom, has the following expectations of the test measure difference d_k:*

$$\begin{aligned} \mathbb{E}[d_k] &= \mathbb{E}[z_k] - \mathbb{E}[z_{k-1}] = 0, \\ \mathrm{Var}[d_k] &= \mathrm{Var}[z_k] + \mathrm{Var}[z_{k-1}] = 4N_s. \end{aligned} \tag{2.72}$$

Given an attack-free system that follows the expectation (2.72) in Proposition 2.3, the test measure difference $d_k \in \mathbb{R}$ follows

$$d_k \sim \mathcal{VG}\Big(\mathbb{E}[d_k], \sqrt{\mathrm{Var}[d_k]}, 0, \frac{2}{N_s}\Big) \tag{2.73}$$

where $\mathcal{VG}(\cdot,\cdot,\cdot,\cdot)$ denotes the *variance-gamma distribution* [94], which is a mixed distribution of the normal distribution and gamma distribution. As the chi-square distribution is a special case of the gamma distribution, the difference of two gamma random variables (i.e. chi-square random variables) results in the variance-gamma distribution [45]. The parameters within the variance-gamma distribution that describe the difference of two chi-square random variables, generalized in [27], are the location $c = \mathbb{E}[d_k]$, spread $\bar{\sigma} = \sqrt{\mathrm{Var}[d_k]}$, asymmetry $\vartheta = 0$, and shape $\lambda = \frac{2}{N_s}$. The probability density function (PDF) of the variance-gamma distribution follows

$$f(d_k; c, \bar{\sigma}, \vartheta, \lambda) = \frac{2e^{(\vartheta(x-c)/\bar{\sigma}^2)}|x-c|^{\frac{1}{\lambda}-\frac{1}{2}}}{\bar{\sigma}\sqrt{2\pi}\lambda^{\frac{1}{\lambda}}\Gamma\big(\frac{1}{\lambda}\big)} \Bigg(\frac{1}{\sqrt{2\bar{\sigma}^2/\lambda + \vartheta^2}}\Bigg)^{\frac{1}{\lambda}-\frac{1}{2}} \times K_{\frac{1}{\lambda}-\frac{1}{2}}\Bigg(\frac{|x-c|\sqrt{2\bar{\sigma}^2/\lambda+\vartheta^2}}{\bar{\sigma}^2}\Bigg) \tag{2.74}$$

where K_λ is the *modified Bessel function of the third kind* of order λ and $\Gamma(\cdot)$ is the *gamma function* [91]. During nominal conditions, the test measure difference d_k is a symmetric zero-mean distribution (i.e., parameters $c = \vartheta = 0$).

False Alarms: Similar to other detection algorithms in literature [67, 71], we leverage an alarm rate to diagnose the health of the system from sensor attacks. The magnitude-based detection scheme compares the test measure difference d_k to a threshold $\tau_d \in \mathbb{R}_{>0}$ by:

$$\begin{cases} |d_k| > \tau_d & \longrightarrow \quad \textit{alarm: } \zeta_k^M = 1, \\ |d_k| \leq \tau_d & \longrightarrow \quad \textit{no alarm: } \zeta_k^M = 0, \end{cases} \tag{2.75}$$

where the chosen threshold τ_d is dependent on the expected test measure difference distribution described in (2.73). In Fig. 2.14 we show the resulting distribution of the test measure difference $d_k = z_k - z_{k-1}$ (the difference of two chi-square random variables), which is affected by the number of sensors N_s. The test measure difference d_k distribution follows a variance-gamma distribution in an attack-free scenario, where we show the effects on the distribution for the number of sensor $N_s = \{2, 3, 4, 5, 6, 7\}$ (i.e., dof).

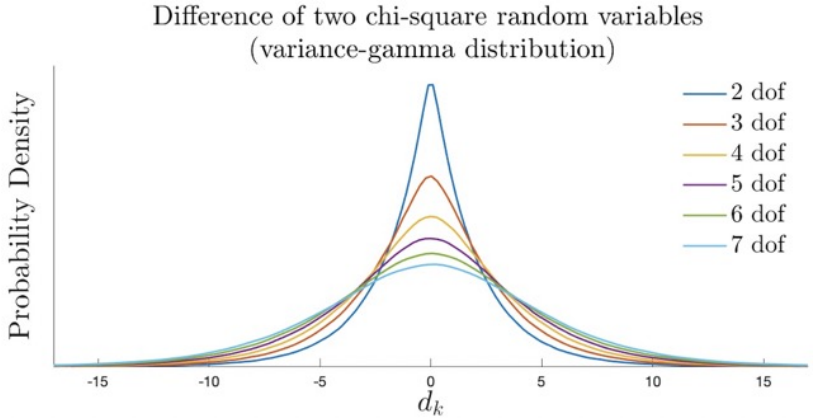


Figure 2.14: The resulting variance-gamma distribution of the test measure difference.

Under nominal circumstances, i.e. in the absence of attacks, an alarm is triggered at a desired rate $\psi_{des}^M \in (0, 1)$ given the chosen threshold value. The following lemma provides a method to choose a threshold to satisfy a user-defined desired alarm rate.

Lemma 2.7 *Assuming that the system is attack-free (i.e. $\boldsymbol{\xi}_k = \mathbf{0}$) and considering the procedure in* (2.75) *to trigger an alarm, a specific threshold value τ_d is chosen by*

$$\tau_d = \{\tau_d \in \mathbb{R}_{>0} : \Pr(\zeta_k^M = 1) = \psi_{des}^M\}, \tag{2.76}$$

such that the result is a desired alarm rate ψ_{des}^M.

Proof: Let $F_{d_k}(x; c, \bar{\sigma}, \vartheta, \lambda)$ denote the cumulative distribution function (CDF) of the random variable d_k from the PDF in (2.74). We compute the inverse CDF for a given desired false alarm rate ψ^M_{des} to find the threshold value

$$\tau_d = F^{-1}_{d_k}\Big(1 - \frac{\psi^M_{des}}{2}; c, \bar{\sigma}, \vartheta, \lambda\Big) \tag{2.77}$$

such that $\Pr(\zeta^M_k = 1) = \Pr(|d_k| > \tau_d) = \psi^M_{des}$ to achieve a desired false alarm rate, thus concluding the proof. ■

Alarm Rate Estimation: We employ a runtime method of estimating the alarm rate such that we are able to eliminate the need to store a sequence of values. A Memoryless Runtime Estimator (MRE) is leveraged to eliminate the need to use a windowed method to compute an alarm rate estimation $\hat{\psi}^M_k \in [0, 1]$. The MRE algorithm is updated by following

$$\hat{\psi}^M_k = \hat{\psi}^M_{k-1} + \frac{\zeta^M_k - \hat{\psi}^M_{k-1}}{\ell}, \tag{2.78}$$

where ℓ is a user-defined "pseudo-window" length. The resulting distribution while leveraging MRE can be approximated to a normal distribution for pseudo-window lengths $\ell \geq 10$ consisting of a variance that follows that of a exponential moving average (EMA) [30].

Lemma 2.8 *Given the test measure difference d_k defined in* (2.71) *for a system that is assumed to be attack-free and tuned for a desired false alarm rate ψ^M_{des}, the estimate alarm rate follows a Normal distribution described by*

$$\hat{\psi}^M_k \sim \mathcal{N}\bigg(\psi^M_{des}, \frac{\psi^M_{des}(1 - \psi^M_{des})}{2\ell - 1}\bigg). \tag{2.79}$$

Proof: We first characterize the magnitude-based detector tuned for a desired false alarm rate ψ^M_{des} as a Binomial distribution $\mathcal{B}(\cdot, \cdot)$ where ψ^M_{des} is a probability for a "success" during a specified number of "trials" (Refer to [91] for further explanations). By way of the binomial approximation for larger pseudo-window size $\ell \geq 10$, a normal distribution can be used to approximate the alarm rate while leveraging MRE (2.78) for estimation that results in

$$\mathbb{E}[\psi^M] = \psi^M_{des}, \;\; \mathrm{Var}[\psi^M] = \frac{\psi^M_{des}(1 - \psi^M_{des})}{2\ell - 1}. \tag{2.80}$$

From (2.80) we obtain the distribution in (2.79) to characterize the estimated alarm rate for magnitude-based detection. ■

With the known expected false alarm rate distribution described in (2.79), we want to find bounds on the estimated alarm rate $\hat{\psi}^M_k$, $\forall k$ to determine if an attack has occurred. The following

corollary provides detection bounds with a specific level of confidence $1-\beta$, where $\beta \in [0,1]$ is a user defined level of significance[2].

Corollary 2.1 *Assuming a system with s sensors that employs the chi-square detection scheme that is monitoring the test measure difference* (2.71) *with a level of significance* β *while leveraging MRE* (2.78)*, detection of sensor attacks occurs when* $\Omega_- \leq \hat{\psi}_k^M \leq \Omega_+$ *is not satisfied where*

$$\Omega_\pm = \mathbb{E}[\psi^M] \pm Z\sqrt{\frac{\mathbb{E}[\psi^M](1-\mathbb{E}[\psi^M])}{2\ell-1}}. \tag{2.81}$$

Proof: We construct confidence intervals for a normally distributed variable of a specific confidence level, determined by z-score $Z = \left|\Phi^{-1}\left(\frac{\beta}{2}\right)\right|$, that provide detection bounds by

$$\mathbb{E}[\psi^M] - \left|\Phi^{-1}\Big(\frac{\beta}{2}\Big)\right|\sqrt{\frac{\mathbb{E}[\psi^M](1-\mathbb{E}[\psi^M])}{2\ell-1}} \leq \hat{\psi}_k^M \leq \mathbb{E}[\psi^M] + \left|\Phi^{-1}\Big(\frac{\beta}{2}\Big)\right|\sqrt{\frac{\mathbb{E}[\psi^M](1-\mathbb{E}[\psi^M])}{2\ell-1}} \tag{2.82}$$

which satisfy (2.81), concluding the proof. ■

Detection of sensor attacks occur when an estimated alarm rate $\hat{\psi}_k^M$ travels beyond the thresholds from $\Omega_\pm = [\Omega_-, \Omega_+]$.

2.6.3 Sign-based Randomness

To further strengthen detection of inconsistencies within the test measure, we monitor the "runs" behavior of the test measure difference sequence. While a smart attacker may be able to fool the magnitude-based monitor as discussed in Section 2.6.2, an attacker may leave traces of non-random behavior on the signed test measure difference. The test we use to monitor for signed randomness is influenced by the Serial Independence Runs (SIR) Test [12]. An example of the SIR test is shown in Fig. 2.15, where it monitors a sequence of data by first computing the difference between the current and previous data values and taking the sign of the difference to create a two-valued data sequence (i.e., positive and negative values). Then, the number of observed runs N_R, defined as consecutive values of the same sign, are counted over the sequence length. In Fig. 2.15 we see that over the sequence length $W = 14$ there are $N_R = 12$ runs, which are highlighted by the red and blue lines.

A drawback of the SIR test is the requirement to store the W length sequence of test measure differences d_k and then count the number of observed runs N_R over this sequence. Alternatively, we would like to use a window-less method to eliminate the need for storing an entire sequence to

[2]Reducing the value of β causes the detection bounds to move farther from the expected alarm rate, thus reducing the frequency of falsely "detecting" under nominal (i.e., no attack) conditions while consequently giving an attacker more freedom to design an attack without being detected, while the opposite is true when increasing β.

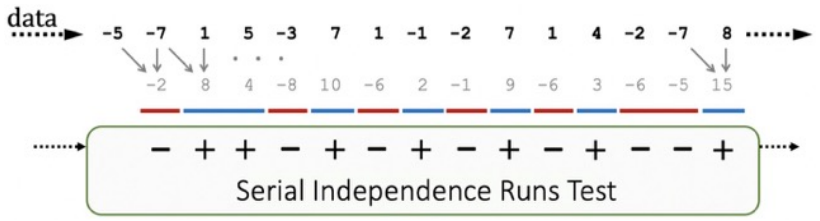


Figure 2.15: A sequence of data, from left to right, converted to sequence of signed values while leveraging the Serial Independence Runs Test.

determine whether the signed test measure difference is behaving randomly. To this end, we propose to observe sign switches at runtime by triggering an alarm at a time k when the the present test measure difference is of the opposite sign from the previous test measure difference at time $k-1$.

We first compute the sign of the test measure difference by the following:

$$\operatorname{sgn}(d_k) := \begin{cases} -1, & \text{if } d_k < 0, \\ 0, & \text{if } d_k = 0, \\ 1, & \text{if } d_k > 0, \end{cases} \tag{2.83}$$

and given that the distribution of the test measure difference d_k is symmetric (assuming nominal conditions) over the expected value $\mathbb{E}[d_k] = 0$, the probability of observing the signed values of the test measure difference are

$$\begin{aligned} \Pr\big(\operatorname{sgn}(d_k) = -1\big) &= 0.5, \\ \Pr\big(\operatorname{sgn}(d_k) = 0\big) &= 0, \\ \Pr\big(\operatorname{sgn}(d_k) = 1\big) &= 0.5. \end{aligned} \tag{2.84}$$

As sensor measurements are received at every kth time instance, the next test measure difference d_k is computed from (2.9), (2.11), and (2.71). A switch of the test measure difference sign signifies the end of a run and an alarm $\zeta_k^S \in \{0, 1\}$ is triggered such that $\zeta_k^S = 1$ at a time instance k, otherwise $\zeta_k^S = 0$. The procedure to trigger a test measure difference alarm follows:

$$\zeta_k^S := \begin{cases} 1, & \text{if } \operatorname{sgn}(d_k) = -\operatorname{sgn}(d_{k-1}), \\ 0, & \text{otherwise.} \end{cases} \tag{2.85}$$

The alarm $\zeta_k^S \in \{0, 1\}$ in (2.85) is then sent into the MRE to provide an updated runtime estimate of the test measure difference alarm rate $\hat{\psi}_k^S$ at time instance k.

Lemma 2.9 *Given a system that is not experiencing attacks, the test measure difference alarm rate while leveraging MRE* (2.78) *for estimation is described as a Normally distributed random variable*

by the following:

$$\hat{\psi}_k^S \sim \mathcal{N}\big(\mathbb{E}[\psi^S], \mathrm{Var}[\psi^S]\big). \tag{2.86}$$

Proof: We want to convert the distribution of expected runs $\mathbb{E}[N_R]$ of test measure differences over a window-based sequence of length W described in [12] by

$$N_R \sim \mathcal{N}\bigg(\frac{2W-1}{3}, \frac{16W-29}{90}\bigg) \tag{2.87}$$

to a runtime rate of expected test measure difference sign switching $\mathbb{E}[\psi^S]$. By first obtaining the asymptotic distribution and then transforming the expected observed runs to an expected rate of observed alarms $\mathbb{E}[\psi^S] = \frac{\mathbb{E}[N_R]}{W}$, we arrive to the expected sign switching alarm rate distribution

$$\mathbb{E}[\psi^S] = \frac{2}{3}, \quad \mathrm{Var}[\psi^S] = \frac{16}{90(2\ell-1)}, \tag{2.88}$$

while leveraging MRE for window-less estimation. ■

The following corollary provides a proof for detection bounds of $\hat{\psi}_k^S$ to satisfy an expected alarm rate $\mathbb{E}[\psi^S]$.

Corollary 2.2 *Given the test measure differences $d_k = z_k - z_{k-1}$, detection occurs by the test measure difference alarm rate when $\Psi_- \leq \hat{\psi}_k^S \leq \Psi_+$ is not satisfied where*

$$\Psi_\pm = \pm\Big|\Phi^{-1}\Big(\frac{\beta}{2}\Big)\Big|\sqrt{\frac{16}{90(2\ell-1)}} + \frac{2}{3}. \tag{2.89}$$

Proof: For a desired level of significance β we find the bounds of $\hat{\psi}_k^S$ for an expected alarm rate $\mathbb{E}[\psi^S]$ are

$$-Z\sqrt{\frac{16}{90(2\ell-1)}} + \frac{2}{3} \leq \hat{\psi}_k^S \leq Z\sqrt{\frac{16}{90(2\ell-1)}} + \frac{2}{3}, \tag{2.90}$$

where z-score is $Z = \big|\Phi^{-1}\big(\frac{\beta}{2}\big)\big|$. From (2.90) we can finally obtain the detection bounds of $\Psi_\pm$ in (2.89) for alarm triggering at an expected alarm rate $\mathbb{E}[\psi^S]$. ■

It is noted that while we describe a runtime method for detecting anomalous signed behavior within the serial sequence of a chi-square random variable z_k, this technique may be used on any randomly distribution variable. As this method is non-parametric, the signed behavior is independent from its underlying distribution [12, 99].

2.6.4 Undetectable Attacks

This section analyzes the attack sequence that an attacker must make in order to remain undetected from our serial randomness-based detector. Continuing with assumptions previously made in Section

2.6.1, a worst-case scenario is assumed where a smart attacker has access to the system model, noise characteristics, control inputs, and state estimator to fool the introduced serial detection technique. In particular, we focus on the attack sequences of $\boldsymbol{\xi}_k$ that can disrupt nominal closed-loop system behavior while remaining hidden.

Magnitude-based Detection

We begin by considering an attack sequence that does not allow the magnitude-based alarm rate $\hat{\psi}_k^M$ to travel beyond detection bounds described in (2.81). If we recall the test measure difference d_k in (2.71), but written in terms of the sensor attack vector $\boldsymbol{\xi}_k$, we have

$$\begin{aligned} d_k &= \boldsymbol{r}_k^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{r}_k - \boldsymbol{r}_{k-1}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{r}_{k-1} \\ &= (\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k) - (\boldsymbol{C}\boldsymbol{e}_{k-1} + \boldsymbol{\eta}_{k-1} + \boldsymbol{\xi}_{k-1})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{C}\boldsymbol{e}_{k-1} + \boldsymbol{\eta}_{k-1} + \boldsymbol{\xi}_{k-1}) \\ &= (\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{C}\boldsymbol{e}_k + \boldsymbol{\eta}_k + \boldsymbol{\xi}_k) - z_{k-1}. \end{aligned} \tag{2.91}$$

In order for an attacker to not trigger the alarm $\zeta_k^M = 1$ at time k, i.e. a zero-alarm attack, the sensor attack vector must maintain the test measure difference to satisfy $|d_k| \leq \tau_d$. For an attack vector sequence and the designed variance-gamma distribution threshold τ_d, we define a suitable vector

$$\boldsymbol{\delta}_k = \{\boldsymbol{\delta}_k \in \mathbb{R}^{N_s} : |\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}| \leq \tau_d\}, \tag{2.92}$$

that leads to the test measure difference d_k not triggering an alarm. Therefore, for any time k, the attack vector follows

$$\boldsymbol{\xi}_k = -\boldsymbol{C}\boldsymbol{e}_k - \boldsymbol{\eta}_k + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\delta}_k, \tag{2.93}$$

where $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is the symmetric square root of the measurement residual covariance matrix $\boldsymbol{\Sigma}$, such that

$$|d_k| = |z_k - z_{k-1}| \leq \tau_d, \tag{2.94}$$

is satisfied. To remain hidden from detection, an attacker must trigger alarms at a rate which the system is expecting. For the case of hidden attacks to evade detection of our serial monitor for magnitude-based detection, a suitable vector $\boldsymbol{\delta}_k$ in (2.92) is constructed as

$$\Pr(|d_k| > \tau_d) = \Pr(|\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}| > \tau_d) \approx \psi_{des}^M, \tag{2.95}$$

to emulate the alarm rate that would be seen during nominal conditions. More specifically, an observed estimated alarm rate computed in (2.78) must remain within detection bounds found in (2.81) to remain undetected. To ensure detection does not occur for the magnitude-based alarm rate, an attacker must design the attack vector such that the alarm rate remains within detection

bounds, $\hat{\psi}_k^M \in [\Omega_-, \Omega_+]$. To remain below the upper detection bound, the vector $\boldsymbol{\delta}_k$ follows

$$|\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}| \leq \tau_d \,:\, \left(\Omega_+ - \hat{\psi}_{k-1}^M - \frac{1-\hat{\psi}_{k-1}^M}{\ell}\right) < 0, \tag{2.96}$$

to guarantee $\hat{\psi}_k^M \leq \Omega_+$. Additionally, a requirement to remain above the lower detection bound adheres to

$$|\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}| > \tau_d \,:\, \left(\Omega_- - \hat{\psi}_{k-1}^M + \frac{\hat{\psi}_{k-1}^M}{\ell}\right) > 0. \tag{2.97}$$

Sign-based Detection

We continue with a scenario for an attacker to evade detection from the Serial Detector in which the attack design is required to satisfy signed randomness throughout the sequence. Similar to the magnitude-based detection, the attack sequence must result in alarm rates that emulate attack-free conditions to remain hidden from detection. To achieve this, the sign-based alarm rate satisfies

$$\Pr\left(\text{sgn}(d_k) = -\text{sgn}(d_{k-1})\right) = \Pr\left(\text{sgn}(z_k - z_{k-1}) = -\text{sgn}(z_{k-1} - z_{k-2})\right) \approx \mathbb{E}[\psi^S] \tag{2.98}$$

in order to behave similarly to nominal conditions. In order to not cause a sign switching condition, i.e. signed-based alarm $\zeta_k^S = 0$, the sign of d_k must consist of the same sign as d_{k-1}. In terms of the vector $\boldsymbol{\delta}_k$ while leveraging (2.67), the following inequality

$$\begin{cases} \boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k > z_{k-1}, & \text{if } d_{k-1} > 0, \\ \boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k < z_{k-1}, & \text{if } d_{k-1} < 0, \end{cases} \tag{2.99}$$

must be satisfied to not cause a sign change, thus not triggering an alarm. If the signed component alarm rate $\psi_k^S,\ \forall k$ approaches the upper detection bound, the following equation guarantees $\hat{\psi}_k^S \leq \Psi_+$, where

$$\text{sgn}(\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}) = \text{sgn}(d_{k-1}) \,:\, \left(\Psi_+ - \hat{\psi}_{k-1}^S - \frac{1-\hat{\psi}_{k-1}^S}{\ell}\right) < 0, \tag{2.100}$$

thus maintaining the alarm rate within bounds. Similarly, the requirement to not cross below the lower bound adheres to

$$\text{sgn}(\boldsymbol{\delta}_k^\mathsf{T}\boldsymbol{\delta}_k - z_{k-1}) = -\text{sgn}(d_{k-1}) \,:\, \left(\Psi_- - \hat{\psi}_{k-1}^S + \frac{\hat{\psi}_{k-1}^S}{\ell}\right) > 0, \tag{2.101}$$

to remain undetectable from the Serial Detector.

2.6.5 Simulation Results

The proposed Serial Detector was validated in simulation and compared to state-of-the-art detection techniques: Bad-Data (BD) [67], Cumulative Sum (CUSUM) [71], and Cumulative Sign (CUSIGN)

detectors. The case study presented in this paper is an autonomous differential-drive UGV with the linearized model in (2.66).

Two different attack sequences are performed: *Bias Attack* where the attack sequence concentrates the test measure distribution such that the magnitude detectors (BD and CUSUM) trigger alarms at a desired rate while signed behavior monitored by CUSIGN remains consistent whereas a *Pattern Attack* creates patterned concentrations on the chi-square test measure difference d_k. In Fig. 2.16, the resulting distributions for each case are shown that include: (a) the *No Attack* case where $z_k \sim \chi^2(N_s = 2)$, (b) *Bias Attack*, and (c) *Pattern Attack*. Both the Bad-Data and CUSUM detectors are tuned for a desired alarm rate of $\alpha = 0.20$ (see [67, 71]) and the CUSIGN detector has an expected alarm rate of 0.0833. The magnitude component of our proposed Serial Detector is tuned for an expected alarm rate $\mathbb{E}[\psi^M] = 0.20$ and the expected alarm rate for the signed component is $\mathbb{E}[\psi^S] = \frac{2}{3}$. All detectors employ detection bounds that are 3 standard deviations from their expectation.

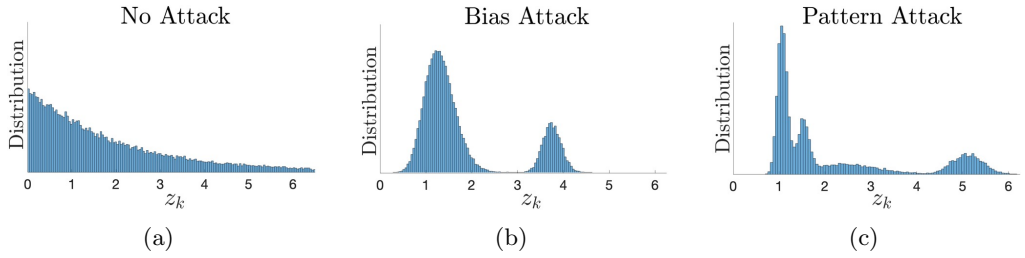


Figure 2.16: The test measure z_k distributions when (a) *No attack*, (b) *Bias Attack*, and (c) *Pattern Attack* occur in the simulation case study.

Next, a simulation showing the detector alarm rates is included during *No Attack* at times $k < 20000$, *Bias Attack* at $20000 \leq k < 40000$ and *Pattern Attack* beginning at time step $k \geq 40000$. During the *Bias Attack* in Fig. 2.17, the attack fools the BD, CUSUM, and CUSIGN detectors, but the magnitude component of the serial monitor notices the change in the test measure sequence due to the attack. The sign component does not detect the attack, as a bias attack does not disrupt the change of signed behavior of the test measure difference d_k. The *Pattern Attack*, while preserving expected test measure difference magnitude behavior, interferes with the expected sign switching rate of the test measure difference. As expected, in the absence of sensor attacks where $k < 20000$, alarm rates for all detection procedures have distributions centered at their expectations. While these modeled attack sequences are primitively designed examples that can fool comparative detectors (e.g., BD, CUSUM, and CUSIGN detectors), the Serial Detector is able to exploit hidden behaviors to strengthen detection capabilities.

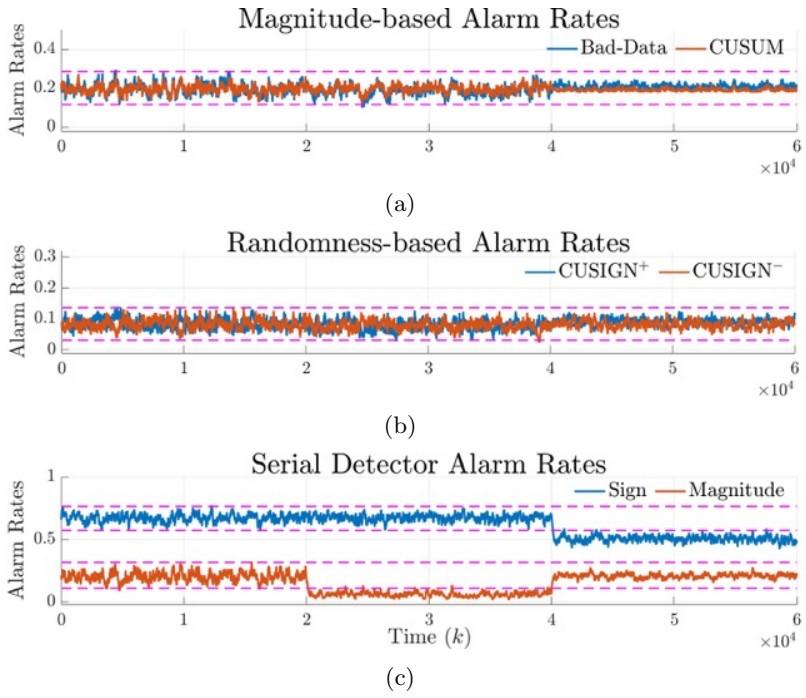


Figure 2.17: Resulting alarm rates during the *No Attack*, *Pattern Attack*, and *Bias Attack*. During both the attack scenarios, the comparable detectors (BD, CUSUM, and CUSIGN) are fooled, while the magnitude and sign components of the Serial Detector discover the *Bias* and *Pattern* attacks. Dashed magenta lines represent 3σ detection bounds for each detector.

2.7 Discussions

In this section, we presented three different monitoring frameworks to detect stealthy cyber attacks to on-board sensors by leveraging novel randomness-based methods. Our attack detection techniques monitored for non-random measurement residual behavior, which would indicate that a sensor measurements are being manipulated in an attempt to stealthily hijack a system toward an undesirable state. We began by introducing the discrete-time linear dynamical model of our system, noise and state estimation models, attack model, and the measurement residual test measure used in monitoring for non-random behavior. Our first framework leverages the Wilcoxon Signed-Rank test and Serial Independence Runs test over a sliding monitor window to detect stealthy attacks when augmented to state-of-the-art boundary detectors. Among the key results of this chapter we provide: bounds for desired false alarm rate for each test which are leveraged to detect attacks, bounds on state deviation under worst case attack scenario, demonstrating that the proposed framework outperform detectors that solely use boundary tests. Our second framework, called the Cumulative Sign (CUSIGN) detector improves upon the sliding window-based method by relaxing the window condition. In particular, we have constructed a Markov chain of the CUSIGN test sequence to

model a resulting expected alarm rate. We have formalized a run-time method for computing an alarm rate estimate using a modified version of Welford's online algorithm. We empirically found the resulting estimated alarm rate distribution and leveraged it to provide detection bounds given a specific level of confidence. Then, we characterized attack sequences that remain undetected to the CUSUM dynamic attack detector, that leave trails of non-random behavior for CUSIGN to detect the attack. Our third detector introduced is the Serial Detector to discover inconsistent test measure behavior due to hidden cyber attacks while employing a chi-square detection procedure. Our detection approach monitors the magnitude and signed sequence of the chi-squared test measure differences to detect inconsistent behavior. We characterized the expected alarm rates for both magnitude and sign, which are dependent on the system model. Furthermore, we provide bounds on detection while also providing an analysis of the detection bounds of our scheme. While our proposed Serial Detector can not replace traditional chi-squared test measure-based detection schemes, however, it can provide another layer of security to detect hidden attacks that are deceptive to these state-of-the-art detectors. The proposed approaches were validated through MATLAB simulations on UGV case studies.

The three detection frameworks introduced in this chapter are modeled for monitoring on-board sensors of an individual autonomous system. To continue these frameworks, our objective is to extend these frameworks to enable resilient operations on single-robot systems when impacted by attacks vectors at differing entry points and also for attack detection within multi-robot systems. We encourage the reader to refer to Chapter 5 for our framework in detecting and recovering from cyber attacks and faults to on-board controllers which compromised control inputs to the system. In this detection and recovery framework, we leverage randomness-based concepts that we have introduced in this chapter. However, in the next chapter, we leverage the CUSIGN attack detector to discover stealthy attacks within multi-robot systems where cyber attacks may occur to on-board sensors and also information broadcasts between robots.

Part II

Isolation and Reconfiguration for Resilient Multi-robot System Operations

Chapter 3

Multi-robot System Attack Detection and Network Reconfiguration

In this chapter, a decentralized framework for detection and network reconfiguration for recovery is introduced that provides resilient navigation for formations of robots in the presence of cyber attacks. The Cumulative Sign (CUSIGN) attack detector presented in Chapter 2 is utilized by each robot to monitor for on-board sensor attacks and also monitoring for misbehaving neighboring robots within the multi-robot system. To maintain resilient multi-robot operations, misbehaving agents are isolated from the remaining robots and the network is then reconfigured to mitigate the risk of undesired control performance of the system. The resilient multi-robot system framework is validated with MATLAB and Gazebo simulations and also with lab experiments using swarms of UGVs. This chapter is based on the following publication:

- P.J. Bonczek, R. Peddi, S. Gao, and N. Bezzo, "Detection of Non-random Sign-based Behavior for Resilient Coordination of Robotic Swarms," *IEEE Transactions on Robotics (T-RO) in the Special Issue for Resilience in Networked Robotic Systems, 2022.*

3.1 Introduction

Many advancements in sensing, control, planning, mobility, and networking have enhanced mobile robotic systems allowing precise and robust autonomous operations that were unthinkable until only recently. Within robotics, multi-agent system coordination and swarming have long been studied and are gaining back attention thanks to the many technological advances, but this also brings upon security issues. Multi-agent systems are typically used to perform coordinated tasks in a distributed fashion. This collaborative nature allows for numerous applications that would be more difficult or not possible to perform with just a single agent, such as: factory and warehouse logistics [19], vehicle platooning [61], connected vehicle-to-vehicle operations [86, 41], surveillance [104], disaster-relief [26], and exploration missions [58].

With such benefits in multi-robot systems, however comes the risk of cyber attacks. In fact, all the aforementioned applications are typically designed without considering cyber-security issues, assuming that all the actors (i.e., other robots) in the multi-robot settings are cooperative. In the presence of a compromised robot in the network, liveness (i.e., the ability to perform and complete correctly a task) and safety (i.e., avoid collisions or reaching undesired states) properties can be violated. The presence of malicious actors in a network can potentially manipulate the entire multi-robot system, hijacking a mission and potentially leading the system toward undesired states, as pictorially represented in Fig. 3.1.

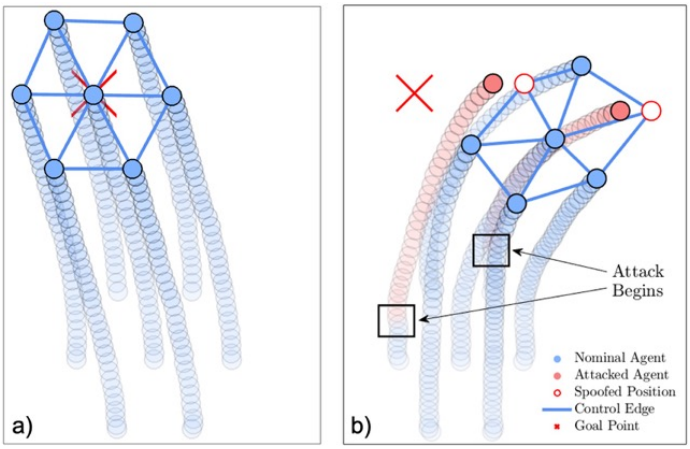


Figure 3.1: A pictorial motivation of the problem investigated in this paper: in nominal conditions a), i.e., with no attack, a multi-agent system can reach the desired goal (red 'X') whereas in the presence of an attack (red disks in b)) the system is hijacked away.

Such situations can be caused by: compromised communications which results in incorrect sharing of information between robots, or by manipulated sensor measurements, leading compromised robots to react to altered on-board signals that are also broadcast to surrounding neighbors. In a successful hijacking attempt, an attacker is able to implement a *stealthy* attack sequence to degrade system performance, all while remaining hidden from detection. The term stealthy has been adopted in a wide-range of attack scenarios on stochastic systems, such as in zero-dynamics [79], replay [111], zero-alarm [13], and hidden [69] attack cases. In this chapter, the term *stealthy* indicates an attack sequence that mimics normal (attack-free) behavior of traditional detection schemes (i.e., a hidden attack [69]), where attackers leverage the noise characteristics within a multi-robot system to evade detection during a hijacking attempt. To discover such attacks, the key principle that we leverage is that an attacker attempting to hijack one or more robots within a multi-robot system via stealthy sensor and/or communication attacks will inherently exhibit non-random/inconsistent behaviors in order to be effective, contradicting an expected behavior of the system model. Specifically, in this chapter we monitor the *residual* — which is defined as the difference between a measured/received

value and the predicted/expected value — in order to discover inconsistent behavior due to these hijacking attempts. Our proposed monitoring scheme — which we name the Cumulative Sign (CUSIGN) detector — differs from other residual-based detectors [67, 70, 71, 51, 49, 65, 25, 38, 85] as its purpose is to monitor for inconsistencies in signed behavior (i.e., non-randomness) of the residual in multi-robot systems. Once an attack is discovered, we propose a framework to: 1) isolate the compromised robots and 2) reconfigure the network to continue the desired task.

This paper has the following contributions: We propose a novel residual-based attack detection scheme for multi-robot systems to find non-random residual behaviors due to stealthy communication and sensor attacks that are undetectable by current state-of-the-art residual-based methods. We then present a decentralized framework in which each robotic agent acts independently by leveraging local information received from nearby robots while employing the proposed detection scheme to enable resilient control of the multi-robot system during stealthy attacks and reconfigure the network to maintain connectivity once one or more compromised robots have been isolated from the network. While we present the proposed framework in a general sense, as a case study, we consider cooperative autonomous multi-robot applications that leverage virtual spring-damper mesh physics for decentralized formation control [97, 6, 7, 17, 87]. Our proposed framework, however, can be used in any proximity-based consensus formation control (e.g., nearest neighbors [55]).

3.2 Preliminaries

This section introduces the multi-robot dynamical model, noise characteristics, and attack models used throughout this chapter.

3.2.1 Multi-robot System Model

Let us consider a multi-robot system with N mobile robots that maintains a proximity-based formation during a mission. Such system can be described using a *directed* graph, where each directed edge represents the control influence on a robot due to the proximity of a neighboring robot in the system. The directed graph describing the multi-robot system is modeled as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the set of vertices $\mathcal{V} = \{1, 2, \ldots, N\}$ denote the mobile robots and the set of edges $\mathcal{E} = \{(i, j) \,|\, i, j \in \mathcal{V}\}$ are control links between robots. An edge $(i, j) \in \mathcal{E}$ means that the control input of robot i is affected by the state of robot j within the proximity-based formation.

Each of the robots are modeled as LTI dynamical agents as follows:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i + \boldsymbol{\nu}_i, \quad i = 1, 2, \ldots, N, \tag{3.1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the state vector, control input $\boldsymbol{u}_i \in \mathbb{R}^m$, state and input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with appropriate dimensions, and zero-mean Gaussian process uncertainty $\boldsymbol{\nu}_i \in \mathbb{R}^n$. The robots successfully achieve tasks by performing a proximity-based consensus protocol $\psi(\cdot)$ in which all robots $i \in \mathcal{V}$ agree on a decentralized control input $\boldsymbol{u}_i \in \mathbb{R}^m$ that follows:

$$\boldsymbol{u}_i = \psi\big(\boldsymbol{x}_i, \boldsymbol{x}_j, \mathcal{O}_i\big), \quad i = 1, 2, \ldots, N, \tag{3.2}$$

where $\boldsymbol{x}_i$ is the state of robot i, $\boldsymbol{x}_j$ represents the states of the neighboring robots j, $j \neq i$, and the set $\mathcal{O}_i$ denotes any nearby obstacles of robot i that are utilized for obstacle avoidance.

To enable the robot network to satisfy the consensus-based control protocol in order to accomplish tasks, the robots exchange necessary information (e.g., state vector) with each other. The set $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N\}$ describes the information broadcast within the multi-robot system that is available to any robots within communication range $\delta_c > 0$. When all robots are cooperative, the mobile team is able to complete the desired task at hand, where inputs are computed based on information received from nearby robots.

***Definition* 3.1 (Communication Graph)** *Given the N robots in set $\mathcal{V}$ with a communication range δ_c, we define the graph $\mathcal{G}_\mathcal{C} = (\mathcal{V}, \mathcal{E}_\mathcal{C})$ with the edge set represented by,*

$$\mathcal{E}_\mathcal{C} = \big\{(i,j) \;\big|\; \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \leq \delta_c,\ i,j \in \mathcal{V}\big\}, \tag{3.3}$$

as the communication graph of the robot set $\mathcal{V}$, where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are position coordinates (within the state vector) of robots $i, j \in \mathcal{V}$, $i \neq j$.

The set of all neighboring robots within communication range of a robot i, as defined by the communication graph, is represented by,

$$\mathcal{C}_i = \big\{j \in \mathcal{V} \;\big|\; (i,j) \in \mathcal{E}_\mathcal{C}\big\}. \tag{3.4}$$

***Definition* 3.2 (Control Graph)** *Each robot $i \in \mathcal{V}$ leverages the received information to form a neighbor set $\mathcal{S}_i \subseteq \mathcal{C}_i$ for consensus control purposes to maintain a desired proximity from other robots. We define the graph $\mathcal{G}_\mathcal{U} = (\mathcal{V}, \mathcal{E}_\mathcal{U})$ with the edge set represented by,*

$$\mathcal{E}_\mathcal{U} = \big\{(i,j) \;\big|\; j \in \mathcal{S}_i, \forall i \in \mathcal{V}\big\}, \tag{3.5}$$

as the control graph of the robot set $\mathcal{V}$.

3.2.2 Measurement Model

Given that the robot states are not directly available, each robot i is equipped with N_s on-board sensors that provide sampled state measurements every $t_s \in \mathbb{R}_+$ seconds as indicated by the output vector given by,

$$\boldsymbol{y}_i^{(k)} = \boldsymbol{C}\boldsymbol{x}_i^{(k)} + \boldsymbol{\eta}_i^{(k)} \in \mathbb{R}^{N_s}, \tag{3.6}$$

with the output matrix $\boldsymbol{C}$ and measurement uncertainty vector $\boldsymbol{\eta}_i^{(k)}$ at every time instant $k \in \mathbb{N}$. The process and measurement uncertainties of all robots are described in discrete-time as multivariate zero-mean Gaussian distributed noise with covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. A Kalman Filter, with gain matrix $\boldsymbol{K}_i^{(k)} \in \mathbb{R}^{n \times N_s}$, is implemented on-board each robot i to provide discrete-time state estimates $\hat{\boldsymbol{x}}_i^{(k|k)} \in \mathbb{R}^n$ using a discretized dynamical model of (3.1).

3.2.3 Attack Model

Summarized in Fig. 3.2 are the cyber attacks considered in this chapter, which are a combination of on-board sensor and/or communication spoofs that can maliciously affect any robot within the multi-robot system. Next, we provide a brief description for each of the considered cyber-attack scenarios.

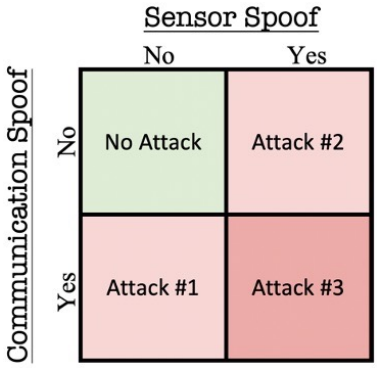


Figure 3.2: The classes of attacks considered.

(A1) – ***Communication Attack***: in which an attacker intercepts and replaces broadcast data such that the receiver and sender data are different (e.g., a man-in-the-middle attack [20]). We assume that an attacker is able to intercept communication broadcasts replacing the message with modified, yet plausible information. As an example, the sender of a communication broadcast that is being attacked may not be aware of the attack in which a receiver is obtaining falsified data. For the case studies investigated in this chapter, the exchanged information $\mathcal{I}^{(k)}$ at each time instant k between robots are assumed to be state estimates, inputs, and measurements. We will indicate the

spoofed broadcast information $\mathcal{I}_i^{(k)} \rightarrow \widetilde{\mathcal{I}}_i^{(k)}$ from a robot i as,

$$\widetilde{\mathcal{I}}_i^{(k)} = \{\hat{\boldsymbol{x}}_i^{(k|k)} + \boldsymbol{\xi}_{i,x}^{(k)},\, \boldsymbol{u}_i^{(k)} + \boldsymbol{\xi}_{i,u}^{(k)},\, \boldsymbol{y}_i^{(k)} + \boldsymbol{\xi}_{i,y}^{(k)}\}, \tag{3.7}$$

where in the presence of an attack, at least one of the following conditions is true: $\boldsymbol{\xi}_{i,x}^{(k)} \neq \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{\xi}_{i,u}^{(k)} \neq \mathbf{0} \in \mathbb{R}^m$, $\boldsymbol{\xi}_{i,y}^{(k)} \neq \mathbf{0} \in \mathbb{R}^{N_s}$, resulting in $\mathcal{I}_i^{(k)} \neq \widetilde{\mathcal{I}}_i^{(k)}$.

(A2) – ***Sensor Spoofing***: The second attack that we consider is sensor spoofing in which an adversary manipulates on-board sensor measurements as follows,

$$\widetilde{\boldsymbol{y}}_i^{(k)} = \boldsymbol{y}_i^{(k)} + \boldsymbol{\xi}_{i,y}^{(k)}, \tag{3.8}$$

where $\boldsymbol{\xi}_{i,y}^{(k)} \in \mathbb{R}^{N_s}$ is the attack vector that describes false data injections to sensor measurements. An attacker manipulating on-board sensor will be able to to drive the state estimate of the robot away from its true state, leading to unreliable on-board control decisions and, consequently, diverting neighboring robots whose control actions are based on inaccurate position information received from the compromised robot.

(A3) – ***Coordinated Attack***: This is a combination of the previous two cases in which attacks hide within the expected system behavior acting and hiding in a coordinated way on both sensing **(A1)** and communication **(A2)** constraints. The compromised robot in this case is able to perform a completely different operation while reporting plausible data to neighbor robots.

For each of the attack vectors $\left(\boldsymbol{\xi}_{i,x}^{(k)}, \boldsymbol{\xi}_{i,u}^{(k)}, \text{ and } \boldsymbol{\xi}_{i,y}^{(k)}\right)$, an attacker is assumed to be capable of leveraging both process and measurement uncertainties $\boldsymbol{Q}$ and $\boldsymbol{R}$, to construct attacks that emulate the expected behavior of measurements and communication broadcasts that can fool traditional residual-based attack detection techniques.

3.3 Problem Formulation

We consider a typical scenario in which robots in a multi-robot system coordinate their motion in a decentralized fashion to maintain a desired formation while navigating toward a given goal. The challenge is to provide a resilient approach for the multi-robot system to continue these operations in the presence of cyber attacks that are intentionally hiding within system noises while attempting to hijack the multi-robot system.

Problem 3.1 (Detection of Inconsistencies in Multi-robot Systems) *Consider a set of N homogeneous robots $\mathcal{V}$ in a multi-robot system. Design a decentralized policy for each robot $i \in \mathcal{V}$ to detect at runtime inconsistencies from any neighbor $j \neq i$ due to cyber attacks on: 1) sensor*

measurements, i.e, if the following holds:

$$\mathbb{E}[\boldsymbol{y}_j - \hat{\boldsymbol{y}}_{ij}] \neq 0, \tag{3.9}$$

or 2) on the communication channel when the received state from j is different from the predicted state computed by i:

$$\mathbb{E}[\boldsymbol{x}_j - \hat{\boldsymbol{x}}_{ij}] \neq 0. \tag{3.10}$$

where $\hat{\boldsymbol{y}}_{ij}$ and $\hat{\boldsymbol{x}}_{ij}$ are measurement and state predictions of j made by robot i.

To detect inconsistent behavior of neighboring robots, we employ an attack detection scheme that monitors inter-robot residuals (i.e., the comparison between received and predicted information) for unexpected behavior within the robot network. Upon detection, the system needs to isolate and reconfigure to continue its planned operation. Formally:

Problem 3.2 (Multi-robot System Recovery) *Find a decentralized policy for each robot $i \in \mathcal{V}$ to isolate and remove any maliciously attacked robot j from its neighbor set for control $\mathcal{S}_i$ that presents inconsistent behavior flagged by solving Problem 3.1, i.e., to obtain,*

$$\mathcal{S}_i' = \mathcal{S}_i \setminus \{j\}. \tag{3.11}$$

With the malicious robot j removed from any neighbor set $\mathcal{S}_i'$, the robot j is no longer able to influence the control of i.

3.3.1 Residual Characteristics in Multi-robot Systems

In this section, both the on-board and inter-robot residuals are characterized that are monitored for non-random (i.e., inconsistent) behavior due to cyber attacks within multi-robot systems. We then formalize the detection procedure that searches for non-random behavior in the residual sequences, before describing an attack sequence that an intelligent attacker must take to avoid detection.

In this proposed detection framework within multi-robot systems, each robot $i \in \mathcal{V}$ monitors its on-board measurement residual for discovery of sensor attacks as well as two types of inter-robot residuals to identify inconsistent behavior of communication broadcasts or sensor information that are received from neighboring robots j within the control graph, i.e. $(i, j) \in \mathcal{E}_{\mathcal{U}}$. Let us define the *on-board measurement residual* vector $\boldsymbol{r}_i^{(k)}$ on a robot i as,

$$\boldsymbol{r}_i^{(k)} = \boldsymbol{y}_i^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_i^{(k|k-1)} \in \mathbb{R}^{N_s}, \tag{3.12}$$

to monitor for on-board sensor attacks, which has an expected covariance matrix $\boldsymbol{\Sigma}_i^{(k)} = \mathbb{E}[\boldsymbol{r}_i^{(k)}(\boldsymbol{r}_i^{(k)})^\mathsf{T}] = \boldsymbol{C}\boldsymbol{P}_i^{(k|k-1)}\boldsymbol{C}^\mathsf{T} + \boldsymbol{R}$ during attack-free conditions, with $\boldsymbol{P}_i^{(k|k-1)}$ denoting the prediction error covariance. Each sth on-board measurement residual element is Normally distributed as follows,

$$\mathbb{E}[r_{i,s}^{(k)}] = 0, \quad \text{Var}[r_{i,s}^{(k)}] = \left(\sigma_{i,s}^{(k)}\right)^2, \tag{3.13}$$

where $\left(\sigma_{i,s}^{(k)}\right)^2$ is the sth diagonal element of the on-board measurement residual covariance matrix $\boldsymbol{\Sigma}_i^{(k)}$.

In our proposed multi-robot monitoring framework, each robot $i \in \mathcal{V}$ monitors its neighbors for consistent behavior by computing state predictions of each neighbor $j \in \mathcal{S}_i$ using their received state $\hat{\boldsymbol{x}}_j^{(k|k)}$ and input $\boldsymbol{u}_j^{(k)}$ information by,

$$\hat{\boldsymbol{x}}_{ij}^{(k+1|k)} = \boldsymbol{A}_d\hat{\boldsymbol{x}}_j^{(k|k)} + \boldsymbol{B}_d\boldsymbol{u}_j^{(k)} \in \mathbb{R}^n \tag{3.14}$$

where $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are discrete-time equivalents of the known robot dynamical model in (3.1). A robot i leverages these state predictions by comparing them to the received state and measurement information from neighboring robots. Let us define the *inter-robot state residual* $\check{\boldsymbol{r}}_{ij}^{(k)} \in \mathbb{R}^n$ by the following,

$$\check{\boldsymbol{r}}_{ij}^{(k)} = \hat{\boldsymbol{x}}_j^{(k|k)} - \hat{\boldsymbol{x}}_{ij}^{(k|k-1)} \tag{3.15}$$

which enables a robot i to monitor for consistent state and input information from a robot j. Each qth element $q \in \{1, \ldots, n\}$ of the inter-robot state residual vector (3.15) is Normally distributed as follows,

$$\mathbb{E}[\check{r}_{ij,q}^{(k)}] = 0, \quad \text{Var}[\check{r}_{ij,q}^{(k)}] = \sum_{s=1}^{N_s}\left(K_{j,(q,s)}^{(k)}\sigma_{j,s}^{(k)}\right)^2, \tag{3.16}$$

with $K_{j,(q,s)}^{(k)}$ representing the element at the qth row and sth column of the Kalman gain at time k on robot j. Additionally, robots compute the *inter-robot measurement residual*,

$$\boldsymbol{r}_{ij}^{(k)} = \boldsymbol{y}_j^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_{ij}^{(k|k-1)} \in \mathbb{R}^{N_s}, \tag{3.17}$$

to discover sensor attacks that may be occurring on the neighboring robot. The inter-robot measurement residual shares the expected zero-mean Normally distributed characteristics of the on-board measurement residual in (3.13). Note that in order for a robot i to compute inter-robot residuals of a robot j at a time k in (3.15) and (3.17), a state prediction (3.14) must be made at the previous time $k-1$.

A robot that is operating in normal conditions will have an expected occurrence of signed residual characteristics over time. With these considerations in mind, we propose a detector to

analyze the sign of incoming residuals within multi-robot systems to determine whether the residual behavior follows the expected random behavior. This technique, which we name the Cumulative Sign (CUSIGN) detector, is unique to previous state-of-the-art residual-based detectors [67, 70, 71, 51, 49, 65, 25, 38, 85] in that instead of monitoring for magnitude changes, it relies on the sign of a residual variable within an expected distribution in order to discover stealthy cyber attacks that may remain hidden within noisy systems. Since the magnitude of a residual variable is overlooked, the CUSIGN detector (characterized in Chapter 2.5) is non-parametric in nature and can be used on any known distribution. Next, we briefly introduce the technique used for alarm rate estimation before characterizing our alarm-based attack detector.

3.3.2 The Signed Residual

For ease of notation throughout the remaining of this subsection, the inter-robot state and measurement residuals are written with the variable $\boldsymbol{r}_{ij}^{(k)}$ and refer to both as simply the "inter-robot residual". Since all inter-robot residuals in the chapter are modeled as zero-mean Normally distributed random vectors during nominal operation, each inter-robot residual element is simply written as $r_{ij,q}^{(k)}$ with a generic indexing of $q \in \{1, \ldots, N_q\}$.

In normal operating conditions, i.e., in the absence of attacks defined in **(A1)**–**(A3)**, the signed value of inter-robot residuals have an expected probability of being higher or lower than their expected values $\mathbb{E}[r_{ij,q}^{(k)}] = 0$. The signed inter-robot residual element probabilities $\Pr(\cdot)$ are computed based on the expected residual distributions characterized in Section 3.3.1 by the following:

$$\begin{aligned} \Pr\left(r_{ij,q}^{(k)} < \mathbb{E}[r_{ij,q}^{(k)}]\right) &= \Phi\big(\mathbb{E}[r_{ij,q}^{(k)}]\big), \\ \Pr\left(r_{ij,q}^{(k)} > \mathbb{E}[r_{ij,q}^{(k)}]\right) &= 1 - \Phi\big(\mathbb{E}[r_{ij,q}^{(k)}]\big), \end{aligned} \tag{3.18}$$

where $\Phi(\cdot)$ is the *cumulative distribution function* of the standard normal distribution [91]. The sign of $r_{ij,q}^{(k)}$ with respect to the reference $\mathbb{E}[r_{ij,q}^{(k)}]$ follows:

$$\operatorname{sgn}(r_{ij,q}^{(k)}) = \begin{cases} 1, & \textbf{if } r_{ij,q}^{(k)} > \mathbb{E}[r_{ij,q}^{(k)}], \\ 0, & \textbf{if } r_{ij,q}^{(k)} = \mathbb{E}[r_{ij,q}^{(k)}], \\ -1, & \textbf{if } r_{ij,q}^{(k)} < \mathbb{E}[r_{ij,q}^{(k)}], \end{cases} \tag{3.19}$$

such that the probability of each scenario occurring is:

$$\begin{aligned} \Pr\left(\operatorname{sgn}(r_{ij,q}^{(k)}) = 1\right) &= p_+, \\ \Pr\left(\operatorname{sgn}(r_{ij,q}^{(k)}) = 0\right) &= 0, \\ \Pr\left(\operatorname{sgn}(r_{ij,q}^{(k)}) = -1\right) &= p_- = 1 - p_+, \end{aligned} \tag{3.20}$$

where $p_+ = p_- = \frac{1}{2}$ for a zero-mean Normally distributed residual from (3.13) and (3.16), as the mean and median are equal. The CUSIGN detector leverages the expected probabilities $\Pr\left(r_{ij,q}^{(k)} > \mathbb{E}[r_{ij,q}^{(k)}]\right) = p_+$ and $\Pr\left(r_{ij,q}^{(k)} < \mathbb{E}[r_{ij,q}^{(k)}]\right) = p_-$ in determining non-random behavior in the presence of attacks.

Alarm Rate Estimation: In the design of the non-randomness detector, alarms are triggered during operation to aid in determining if a system is behaving normally. In our case of a multi-robot network, the robots leverage this alarm-based method for self detection and to monitor the residual sequence of their neighbors for inconsistent behaviors. Given a robot that is not under attack, the frequency at which these alarms are triggered should follow an expected alarm rate. We employ a window-less method, which we name *Memoryless Runtime Estimator* (MRE), for computing the alarm rate estimate utilizing a "pseudo-window" length ℓ. The runtime update equation of MRE for alarm rate estimation follows:

$$\hat{A}_{ij,q}^{(k)} = \hat{A}_{ij,q}^{(k-1)} + \frac{\left[\zeta_{ij,q}^{(k)} - \hat{A}_{ij,q}^{(k-1)}\right]}{\ell}, \tag{3.21}$$

where $\zeta_{ij,q}^{(k)} \in \{0, 1\}$ is the alarm, $\hat{A}_{ij,q}^{(k)} \in [0, 1]$ is an estimated alarm rate at every time instant k, and $\hat{A}_{ij,q}^{(0)} = \mathbb{E}[A]$ initially at $k = 0$, where $\mathbb{E}[A] \in [0, 1]$ is the expected alarm rate. The resulting alarm rate estimate can be approximated to a Normal distribution when $\ell \geq 10$ as demonstrated in Chapter 2.5 with a resulting variance that shares properties of the exponential moving average [30].

3.3.3 CUSIGN Detection in Multi-robot Systems

To detect information inconsistencies (i.e., non-randomness) in multi-robot systems due to cyber attacks, we leverage the *Cumulative Sign* (CUSIGN) attack detector that analyzes residuals to determine whether non-random behavior is occurring. The CUSIGN detector monitors the residual over the sequence of time and outputs an alarm when a threshold is reached, which is then sent to the MRE to provide an updated alarm rate estimate. For any given user-defined threshold, an expected alarm rate can be found that is independent of the system model.

The CUSIGN procedure is an accumulation of signed residual values by two CUSIGN test variables $S_{ij,q}^{(k),+}$ and $S_{ij,q}^{(k),-}$, where each signifies a test variable at time instant k. Each test variable checks for changes in the probability for the signed residual value, one for *positive* and the other for *negative* changes. The following procedure summarizes the CUSIGN detector for both the positive and negative cases:

CUSIGN Detector Procedure in Multi-robot Systems

$$
\begin{aligned}
&S_{ij,q}^{(k),+} = \max\left(0, S_{ij,q}^{(k-1),+} + \operatorname{sgn}(r_{ij,q}^{(k)})\right), \\
&S_{ij,q}^{(k),+} = 0 \text{ and Alarm } \zeta_{ij,q}^{(k),+} = 1, \qquad \textbf{if } S_{ij,q}^{(k),+} = \tau \\
&S_{ij,q}^{(k),-} = \min\left(0, S_{ij,q}^{(k-1),-} + \operatorname{sgn}(r_{ij,q}^{(k)})\right), \\
&S_{ij,q}^{(k),-} = 0 \text{ and Alarm } \zeta_{ij,q}^{(k),-} = 1, \qquad \textbf{if } S_{ij,q}^{(k),-} = -\tau
\end{aligned}
\tag{3.22}
$$

The working principle of CUSIGN test variable sequences are to accumulate the signed residual value $\operatorname{sgn}(r_{ij,q}^{(k)}) \in \{-1, 0, 1\}$ and trigger an alarm $\zeta_{ij,q}^{(k),+}, \zeta_{ij,q}^{(k),-} \in \{0, 1\}$ when the test variables reach their corresponding threshold values $\tau \in \mathbb{N}$. As either of the test variables reach their respective thresholds, then the test variable is reset to zero. An example of the CUSIGN detection procedure (3.22) is shown in Fig. 3.3 where an incoming data sequence of residuals transition the positive and negative CUSIGN test variables $S_{ij,q}^{(k),+}$ and $S_{ij,q}^{(k),-}$. When either test variable reaches the threshold, for this example $\tau = 2$, an alarm is triggered (indicated by the red circles) and a reset to zero condition occurs. The CUSIGN detector monitors the occurrence of triggered alarms as the CUSIGN test variables reach their respective thresholds, where irregular occurrences indicated an attack may be happening.

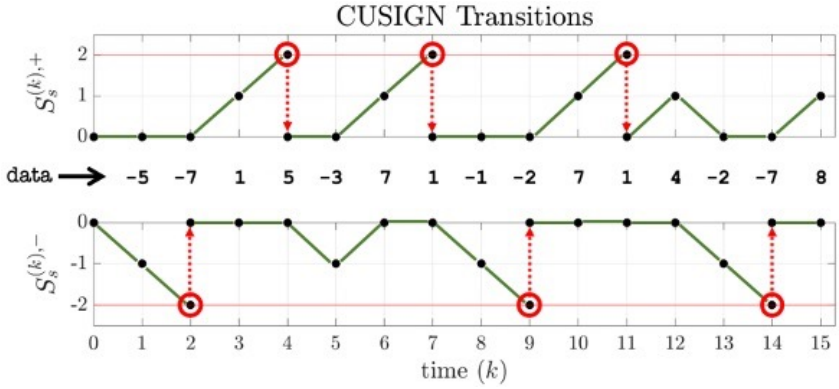


Figure 3.3: An example of transitions for the CUSIGN test variable $S_{ij,q}^{(k),\pm}$ with a threshold $\tau = 2$ given a sequence of data.

Similar to the implementation in [70], the transition of the CUSIGN test variable sequences can be constructed as a Markov chain with a transition matrix modeled from the probabilities p_+ and p_- computed in (3.20). Consisting of a user-defined threshold τ to trigger an alarm, we show the transitions of $S_{ij,q}^{(k),\pm}$ with a Markov chain diagram in Fig. 3.4.

Given a chosen threshold value $\tau \in \mathbb{N}_+$ as a value that triggers an alarm when $|S_{ij,q}^{(k),\pm}| = \tau$, we describe the Markov chain in Fig. 3.4 in the form of a Markov transition matrix $\mathcal{T}^{\pm} \in \mathbb{R}^{(\tau+1)\times(\tau+1)}$, denoted for both the positive and negative transition matrices, $\mathcal{T}^{+}$ and $\mathcal{T}^{-}$. The CUSIGN Markov

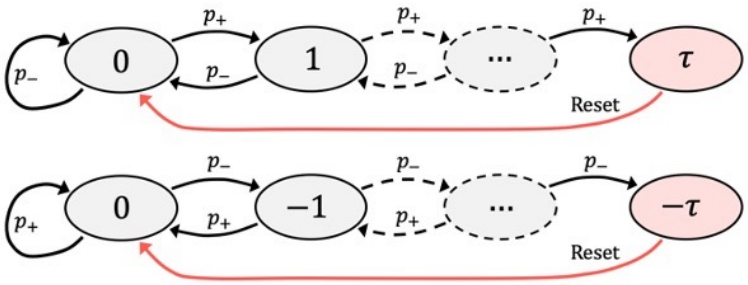


Figure 3.4: A Markov chain for both positive (top) and negative (bottom) cases of the CUSIGN test variable sequence with triggering threshold states in red.

Chain, occurring in a discrete manner, contains $\tau + 1$ states denoted as $\mathcal{M} = \{M_0, M_1, \ldots, M_\tau\}$ where M_τ is an absorbing state that is equal to the threshold, causing the CUSIGN test sequence $S_{ij,q}^{(k),\pm}$ to reset to M_0. The CUSIGN Markov transition matrix for the positive $\mathcal{T}^+$ with a probability distribution of $\text{sgn}(r_{ij,q}^{(k)})$ is written as:

$$\mathcal{T}^+ = \begin{bmatrix} p_- & p_+ & 0 & 0 & \ldots & 0 \\ p_- & 0 & p_+ & 0 & \ldots & 0 \\ 0 & p_- & 0 & p_+ & & 0 \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \ldots & 0 & p_- & 0 & p_+ \\ 0 & \ldots & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathcal{Q}^+ & * \\ \mathbf{0}_{1\times\tau} & 1 \end{bmatrix}. \tag{3.23}$$

The transition matrix $\mathcal{T}^+$ structure remains the same on any system, where the matrix size depends only on the value of the threshold τ. Transition probabilities for transient states in $\mathcal{T}^+$ adhere to the following:

$$\begin{cases} \Pr(M_j \to M_{j+1}) = p_+, & \text{for } j = \{0, \ldots, \tau - 1\}, \\ \Pr(M_j \to M_{j-1}) = p_-, & \text{for } j = \{1, \ldots, \tau - 1\}, \\ \Pr(M_0 \to M_0) = p_-, \end{cases} \tag{3.24}$$

and the final row represents an absorbing (i.e., triggering) state containing elements equal to 0, besides the last element equaling 1.

We define $\mathcal{Q}^+ \in \mathbb{R}^{\tau\times\tau}$ as the fundamental matrix obtained from $\mathcal{T}^+$ with its last row and column removed (i.e., the absorbing state at threshold τ is removed), representing the transition probabilities to and from the transient states. Elements of $\mathcal{Q}^+$ are all non-negative and row sums are equal to or less than one, while the eigenvalues satisfy $\rho[\mathcal{Q}^+] < 1$ such that $(\mathcal{Q}^+)^k \to 0$ as $k \to \infty$ and $\sum_{k=0}^{\infty}(\mathcal{Q}^+)^k = (\boldsymbol{I}_\tau - \mathcal{Q}^+)^{-1}$, where $\rho[\cdot]$ is the spectral radius and $\boldsymbol{I}_\tau$ is the identity matrix of size τ. Leveraging the fundamental matrix $\mathcal{Q}^+$, we can compute an expected alarm rate

as indicated in the following lemma.

Lemma 3.1 *Given a system with a CUSIGN detector* (3.22) *with a user-defined threshold* $\tau \in \mathbb{N}$ *that is not affected by cyber attacks such that the residual sequence satisfies* $r_{ij,q}^{(k)} \sim \mathcal{N}(0, \sigma_{ij,q}^{(k)})$*, then the inverse of the first element of the following vector,*

$$\boldsymbol{\mu}^{+} = (\boldsymbol{I}_{\tau} - \mathcal{Q}^{+})^{-1}\mathbf{1}_{\tau\times 1} = (\mu_1^{+}, \ldots, \mu_{\tau}^{+})^{\mathsf{T}}, \tag{3.25}$$

is the expected alarm rate, i.e., $\mathbb{E}[A^{+}] = (\mu_1^{+})^{-1}$.

Proof: Given the Markov chain containing $\tau + 1$ states denoted by $\mathcal{M} = \{M_0, M_1, \ldots, M_{\tau}\}$, a fundamental matrix $\mathcal{Q}^{+}$ is taken from a designed Markov transition matrix (3.23) to satisfy the transition probabilities (3.24). Leveraging the theory of average run length (ARL) introduced in [10], the ARL is defined as the average length of time for the test sequence to reach the threshold τ to trigger an alarm, determined by the fundamental matrix $\mathcal{Q}^{+}$ containing the transient states within $\mathcal{T}^{+}$. By definition, the inverse of the ARL to observe an alarm results in the average frequency of obtaining an alarm, known as the alarm rate. The ARL can be found by computing (3.25), then by inverting the first element of $\boldsymbol{\mu}^{+}$, i.e., $(\mu_1^{+})^{-1}$, finally obtain the expected alarm rate $\mathbb{E}[A^{+}] = (\mu_1^{+})^{-1}$. ■

Remark 3.1 *The design of transition matrix* $\mathcal{T}^{-}$ *with subsequent fundamental matrix* $\mathcal{Q}^{-}$ *and expected alarm rate* $\mathbb{E}[A^{-}] = (\mu_1^{-})^{-1}$ *for the negative case is computed by* (3.23)-(3.25) *with transition probability (*p_{+} *and* p_{-}*) signs inverted.*

The expected variance of estimated alarm rates $A_{ij,q}^{(k),\pm}$ using MRE for runtime estimation have been found through empirical results in Chapter 2.5.6. A scaling factor $\theta \in \mathbb{R}_{>0}$ is found to be dependent on the chosen threshold τ. The observed MRE scaling factor approximates of θ are presented in Table 3.1 for thresholds $\tau = 1, 2, 3, 4$ and $\ell \geq 10$.

Table 3.1: Empirical values for the scaling value θ given $\tau = 1, 2, 3, 4$.

Threshold τ	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$
θ	1	0.74	0.7	0.69

Proposition 3.1 *Assuming a residual is not affected by a cyber attack while using* (3.21) *for alarm rate estimation, the alarm rate is Normally distributed by the following:*

$$\hat{A}_{ij,q}^{(k),\pm} \sim \mathcal{N}\left(\mathbb{E}[A^{\pm}], \frac{\theta\mathbb{E}[A^{\pm}](1 - \mathbb{E}[A^{\pm}])}{2\ell - 1}\right). \tag{3.26}$$

By leveraging the expected distribution of the estimated alarm rate in (3.26), bounds of the alarm rate can be made. The following corollary provides alarm rate detection bounds for the CUSIGN detector.

Corollary 3.1 *Given a residual $r_{ij,q}^{(k)}$ monitored by the CUSIGN detector* (3.22) *consisting of a threshold $\tau \in \mathbb{N}$, detection of cyber attacks occur for a given level of significance $\alpha \in (0,1)$ when $\Omega_- \leq \hat{A}_{ij,q}^{(k),\pm} \leq \Omega_+$ is no longer satisfied.*

Proof: With the CUSIGN detector consisting of a threshold τ, an expected alarm rate $\mathbb{E}[A_{ij,q}^{\pm}]$ found in (3.25), and leveraging (3.21) with a pseudo-window of length ℓ, the distribution of the estimated alarm rate follows the Normally distributed properties from (3.26). Detection bounds $\Omega_\pm = [\Omega_-, \Omega_+]$ of a user-defined level of significance $\alpha \in (0,1)$ (i.e., the probability that a false detection occurs in nominal conditions) follows,

$$\mathbb{E}[A^{\pm}] - \left|\Phi^{-1}\left(\frac{\alpha}{2}\right)\right|\sqrt{\frac{\theta\mathbb{E}[A^{\pm}](1-\mathbb{E}[A^{\pm}])}{2\ell-1}} \leq \hat{A}_{ij,q}^{(k),\pm} \leq \mathbb{E}[A^{\pm}] + \left|\Phi^{-1}\left(\frac{\alpha}{2}\right)\right|\sqrt{\frac{\theta\mathbb{E}[A^{\pm}](1-\mathbb{E}[A^{\pm}])}{2\ell-1}} \tag{3.27}$$

where $\Phi^{-1}(\cdot)$ is the *inverse cumulative distribution function* of a standard normal distribution [91], thus satisfying Corollary 3.1 and concluding the proof. ■

In summary, with the CUSIGN detection procedure, we can monitor and detect non-random behavior in residual data. Under a worst-case scenario (i.e., assuming an attacker has full knowledge of the system model and detection procedure), an intelligent attacker could remain hidden by triggering alarms at rates that do not travel beyond detection bounds while maintaining an attack vector. However, the CUSIGN detector's attack deterring effects will be limited, and one could implement multiple detectors in parallel with different threshold values τ to further impair an attacker's ability to remain hidden.

For a more detailed discussion about undetectable attacks, the reader can follow the next section.

3.4 Examples of Undetectable Attacks

In this subsection, we discuss attack sequences that an attacker can take to remain hidden from detection from the CUSIGN detection scheme for both sensor and communication attacks. In order to evade detection from CUSIGN, an attacker must be mindful of both the positive and negative test variables $S_{ij,q}^{(k),+}$ and $S_{ij,q}^{(k),-}$ with their respective alarm rates $\hat{A}_{ij,q}^{(k),+}$ and $\hat{A}_{ij,q}^{(k),-}$. To maximize damage in a hijacking attack, a smart attacker would want to manipulate a variable of choice (e.g., sensor measurement) to push the system in a specific direction with maximum effect, without passing alarm rate detection bounds $\Omega_\pm$ defined in (3.27). As a result of maximizing the effects of

an attack, one alarm rate is driven toward the maximum alarm rate threshold Ω_+ and the other alarm rate is pushed toward the minimum threshold Ω_-.

Assumption 3.1 *Under a worst-case scenario, an attacker has knowledge of the robot dynamical model* (3.1)*, the network model (e.g., proximity-based consensus protocol), and the state estimation procedure (e.g., Kalman filter). Furthermore, a malicious attacker has the ability to manipulate any* s*th on-board sensor measurement* $y_{i,s}^{(k)}$ *and/or any information within* $\mathcal{I}_i^{(k)}$ *on a robot* $i \in \mathcal{V}$ *through communication broadcasts.*

3.4.1 On-board Sensor Attack

The first case considered is in the event that an attacker can inject false data to sensor measurements on-board a robot i where $\boldsymbol{\xi}_{i,y}^{(k)} \neq 0$. Utilizing the spoofed output vector in (2.8) combined with the on-board measurement residual defined in (2.9), we can rewrite the on-board measurement residual vector on an ith robot as:

$$\begin{aligned} \boldsymbol{r}_i^{(k)} &= \widetilde{\boldsymbol{y}}_i^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_i^{(k|k-1)} \\ &= \boldsymbol{C}\boldsymbol{x}_i^{(k)} + \boldsymbol{\eta}_i^{(k)} + \boldsymbol{\xi}_{i,y}^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_i^{(k|k-1)} \\ &= \boldsymbol{C}\boldsymbol{e}_i^{(k|k-1)} + \boldsymbol{\eta}_i^{(k)} + \boldsymbol{\xi}_{i,y}^{(k)}, \end{aligned} \tag{3.28}$$

where $\boldsymbol{e}_i^{(k|k-1)} = \boldsymbol{x}_i^{(k)} - \hat{\boldsymbol{x}}_i^{(k|k-1)} \in \mathbb{R}^n$ is the state prediction error. Each sth on-board measurement residual element, $s \in \{1, \ldots, N_s\}$, is defined as:

$$r_{i,s}^{(k)} = \boldsymbol{C}_s\boldsymbol{e}_i^{(k|k-1)} + \eta_{i,s}^{(k)} + \xi_{i,y,(s)}^{(k)} \in \mathbb{R}, \tag{3.29}$$

where $\boldsymbol{C}_s$ is the sth row of the output matrix $\boldsymbol{C}$ and $\xi_{i,y,(s)}^{(k)} \in \mathbb{R}$ is the sth element of the sensor measurement attack vector. An intelligent attacker can manipulate the measurement residual sign by constructing a suitable attack signal to create an attack sequence that avoids the CUSIGN detection bounds. An attacker can manipulate the residual sign by choosing an attack vector element s of the sensor measurement to satisfy:

$$\operatorname{sgn}\big(r_{i,s}^{(k)}\big) = \begin{cases} 1, & \text{if } \xi_{i,y,(s)}^{(k)} > -\boldsymbol{C}_s\boldsymbol{e}_i^{(k|k-1)} - \eta_{i,s}^{(k)}, \\ -1, & \text{if } \xi_{i,y,(s)}^{(k)} < -\boldsymbol{C}_s\boldsymbol{e}_i^{(k|k-1)} - \eta_{i,s}^{(k)}. \end{cases} \tag{3.30}$$

We first examine the scenario when either of the estimated alarm rates monitoring the positive $\hat{A}_{i,s}^{(k),+}$ or negative $\hat{A}_{i,s}^{(k),-}$ residual sign occurrences approach the maximum detection boundary threshold Ω_+ on a robot i. The objective of the attacker is to drive the alarm rate of the desired sign as close to the maximum threshold without crossing it. The following equation is a restriction

on the attack signal $\xi^{(k)}_{i,y,(s)}$ for a sensor s on-board a robot i, for both alarm rates denoted as $\hat{A}^{(k),\pm}_{i,s}$, such that neither cross the maximum threshold:

$$\xi^{(k)}_{i,y,(s)} = \left\{ \xi^{(k)}_{i,y,(s)} \underset{-}{\overset{+}{\lessgtr}} -\boldsymbol{C}_s \boldsymbol{e}^{(k)}_i - \eta^{(k)}_{i,s} \,\middle|\, \left(\Omega_+ - \hat{A}^{(k-1),\pm}_{i,s} - \frac{1 - \hat{A}^{(k-1),\pm}_{i,s}}{\ell} \right) < 0 \right\}. \tag{3.31}$$

The constraint in (3.31) determines if the detection threshold will be broken if an alarm is triggered at the time instant k. This forces an attack signal $\xi^{(k)}_{i,y,(s)}$ to result in a desired residual element sign, such that an alarm is not triggered.

A similar restriction for both alarm rates is necessary as either one (i.e., alarm rate for the opposite sign that approaches the maximum bound) nears the minimum threshold bound, Ω_-. An attacker must ensure that an alarm is triggered before the given alarm rate for an sth residual element falls below the minimum detection bound, such that the sth attack signal element satisfies:

$$\xi^{(k)}_{i,y,(s)} = \left\{ \xi^{(k)}_{i,y,(s)} \underset{-}{\overset{+}{\lessgtr}} -\boldsymbol{C}_s \boldsymbol{e}^{(k)}_i - \eta^{(k)}_{i,s} \,\middle|\, \left(\Omega_- - \hat{A}^{(k-1+\left|\pm\tau - S^{(k-1),\pm}_{i,s}\right|),\pm}_{i,s} \right) > 0 \right\} \tag{3.32}$$

such that,

$$\hat{A}^{(k'),\pm}_{i,s} = \hat{A}^{(k'-1),\pm}_{i,s} - \frac{\hat{A}^{(k'-1),\pm}_{i,s}}{\ell} \tag{3.33}$$

where $\forall k' = k, \ldots, k + \left(\left| \pm\tau - S^{(k'-1),\pm}_{i,s} \right| - 1 \right)$ denotes the number of time instants needed for the CUSIGN test variable $S^{(k)}_{i,s}$ to reach the CUSIGN threshold $\pm\tau$ in order to trigger an alarm.

3.4.2 Inter-robot Residual for Communication Attacks

We assume that an attacker can manipulate any information $\mathcal{I}^{(k)}_i$ sent from communication broadcasts from a robot $i \in \mathcal{V}$, which contains the robot's state estimate, input, and measurements. For the case of a communication attack, we provide a worst-case scenario when the broadcast state estimate information from a robot i is altered by a malicious attacker (i.e., $\boldsymbol{\xi}^{(k)}_{i,x} \neq \mathbf{0}$). The neighboring robots $j \in \mathcal{C}_i$ monitor for inconsistent information received from robot i, as it would be unaware of an attacker maliciously altering its information via communication broadcasts. The objective for an attacker is to avoid detection from the neighbors that are monitoring robot i.

The state prediction on-board a neighboring robot j monitoring a robot i is a function of the information $\mathcal{I}^{(k-1)}_i$ sent at the previous time instant $k-1$:

$$\begin{aligned} \hat{\boldsymbol{x}}^{(k|k-1)}_{ji} &= f\big(\hat{\boldsymbol{x}}^{(k-1|k-1)}_i, \boldsymbol{u}^{(k-1|k-1)}_i, \boldsymbol{\xi}^{(k-1)}_{i,x}, \boldsymbol{\xi}^{(k-1)}_{i,u}\big) \\ &= \boldsymbol{A}_d\big(\hat{\boldsymbol{x}}^{(k-1|k-1)}_i + \boldsymbol{\xi}^{(k-1)}_{i,x}\big) + \boldsymbol{B}_d\big(\boldsymbol{u}^{(k-1|k-1)}_i + \boldsymbol{\xi}^{(k-1)}_{i,u}\big) \end{aligned} \tag{3.34}$$

where $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are discrete-time equivalents of $\boldsymbol{A}$ and $\boldsymbol{B}$ in (3.1) such that the inter-robot residual on robot j to monitor robot i follows:

$$\begin{aligned}\breve{\boldsymbol{r}}_{ji}^{(k)} &= \hat{\boldsymbol{x}}_i^{(k|k)} + \boldsymbol{\xi}_{i,x}^{(k)} - \boldsymbol{A}_d\big(\hat{\boldsymbol{x}}_i^{(k-1|k-1)} + \boldsymbol{\xi}_{i,x}^{(k-1)}\big) - \boldsymbol{B}_d\big(\boldsymbol{u}_i^{(k-1|k-1)} + \boldsymbol{\xi}_{i,u}^{(k-1)}\big) \\ &= \big(\hat{\boldsymbol{x}}_i^{(k|k)} + \boldsymbol{\xi}_{i,x}^{(k-1)}\big) - \hat{\boldsymbol{x}}_{ji}^{(k|k-1)} \in \mathbb{R}.\end{aligned} \tag{3.35}$$

An attacker can manipulate the qth inter-robot residual element sign by choosing an attack vector element of the broadcast state estimate to satisfy:

$$\operatorname{sgn}\big(\breve{r}_{ji,q}^{(k)}\big) = \begin{cases} 1, & \text{if } \xi_{i,x,(q)}^{(k)} > \hat{x}_{ji,q}^{(k|k-1)} - \hat{x}_{i,q}^{(k|k)}, \\ -1, & \text{if } \xi_{i,x,(q)}^{(k)} < \hat{x}_{ji,q}^{(k|k-1)} - \hat{x}_{i,q}^{(k|k)}. \end{cases} \tag{3.36}$$

Similar to equations (3.31) and (3.32), an attack can manipulate the qth element of the sent state estimate signal $\xi_{i,x,(q)}^{(k)}$ in order to maximize the alarm rates for inter-robot residuals by

$$\xi_{i,x,(q)}^{(k)} = \left\{ \xi_{i,x,(q)}^{(k)} \overset{+}{\underset{-}{\lessgtr}} \hat{x}_{i,q}^{(k|k)} - \hat{x}_{ji,q}^{(k|k)} \,\middle|\, \left(\Omega_+ - \hat{A}_{ji,q}^{(k-1),\pm} - \frac{1 - \hat{A}_{ji,q}^{(k-1),\pm}}{\ell} \right) < 0 \right\} \tag{3.37}$$

and, similarly, to ensure the alarm rate never reaches the lower bound, the attack signal needs to satisfy

$$\xi_{i,x,(q)}^{(k)} = \left\{ \xi_{i,x,(q)}^{(k)} \overset{+}{\underset{-}{\lessgtr}} \hat{x}_{i,q}^{(k|k)} - \hat{x}_{ji,q}^{(k|k)} \,\middle|\, \left(\Omega_- - \hat{A}_{ji,q}^{(k-1+\left|\pm\tau - S_{ji,q}^{(k-1),\pm}\right|),\pm} \right) > 0 \right\} \tag{3.38}$$

where

$$\hat{A}_{ji,q}^{(k'),\pm} = \hat{A}_{ji,q}^{(k'-1),\pm} - \frac{\hat{A}_{ji,q}^{(k'-1),\pm}}{\ell} \tag{3.39}$$

and $\forall k' = k, \ldots, k + \big(\left| \pm\tau - S_{ji,q}^{(k'-1),\pm} \right| - 1\big)$.

We note, since the CUSIGN attack detector monitors only the signed values of the residual elements (i.e., magnitude is overlooked), it is not possible to quantify the worst-case effects of the cyber attack in terms of true system state deviation with CUSIGN operating as the lone on-board detector. However, when augmented in parallel with a traditional magnitude-based detector [67, 70, 71, 51, 49, 65, 25, 38, 85], the impact on state deviation due to a cyber attack may be quantified.

3.5 Multi-robot System Attack Detection and Recovery

In this section, we show how to deploy the proposed CUSIGN technique on a multi-robot system to detect non-random (i.e., inconsistent) residual behavior due to cyber attacks and to recover/reconfigure the system. Our scheme is leveraged to monitor and detect if neighboring robots are compromised, as well as to perform self-monitoring for discovering inconsistencies to on-board sensor measurements

due to cyber attacks. Fig. 3.5 summarizes the high-level procedure in a block diagram that is executed by each robot in the network to monitor for cyber attacks and locally recover the system when attacks have been detected. As a running case study for the remainder of this paper, we employ a virtual spring-damper mesh (VSDM) with Gabriel Graph rule for proximity-based formation control [5] to demonstrate our detection and recovery approach on multi-robot system formations. However, our approach is valid for any cooperative proximity-based formation control for multi-robot systems (e.g., nearest neighbor [55]).

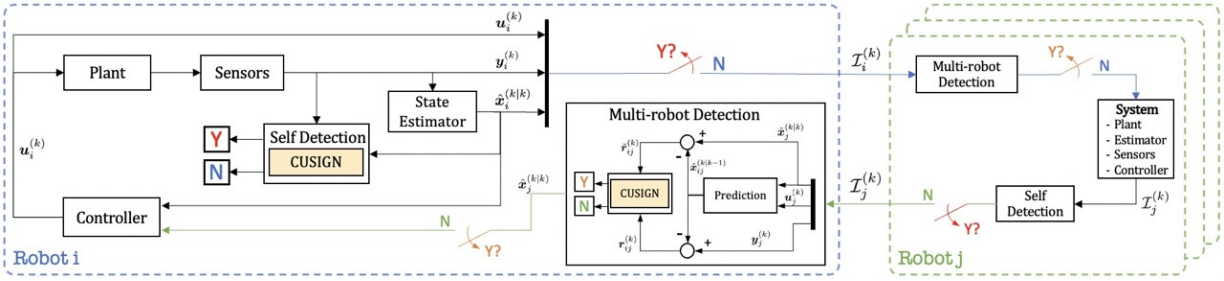


Figure 3.5: The overall architecture describing our resilient robotic framework executed by each robot $i \in \mathcal{V}$ in the network. Both the multi-robot detection on its neighbors $j \in \mathcal{S}_i$ and self detection on itself are performed to find stealthy attacks that exhibit non-random residual behavior.

3.5.1 Virtual Spring-Damper Meshes

As a working example, we consider virtual spring-damper meshes alongside the use of the Gabriel Graph (GG) rule [29, 4] for proximity-based control of multi-robot systems performing coordinated operations. Given that all agents are cooperative, this method allows for a decentralized algorithm where agents are required to leverage local (proximity-based) interactions that result in a desired global behavior of the system. Furthermore, systems that leverage VSDMs incorporate favorable characteristics that include: scalability, efficiency, and known stability properties such that all decentralized agents converge to a global consensus [97, 98]. Numerous works have leveraged VSDMs for various applications [6, 7, 17, 22, 5, 56, 87]; where all of these works did not consider security issues. A single compromised agent affected by cyber attacks can hijack the entire multi-robot system to an undesirable state due to the control interconnections that are propagated throughout the robot network, as demonstrated in Fig. 3.1. In a decentralized manner, our framework allows the agents to identify and remove nearby agents from the network (i.e., the control graph) that could potentially cause undesired behavior.

The objective of the multi-robot system is to navigate to a goal location while maintaining a desired distance between neighboring robots. Furthermore, the multi-robot system must be resilient to stealthy communication and sensor cyber attacks. In addition, we make the assumption that the robot network is navigating in an unknown cluttered environment: to this end all robots are fitted

with a range sensor providing 360 degree field of view (FOV) (e.g., a LiDAR) with limited range $\delta_r < \delta_c$ for obstacle/collision avoidance.

Each robot i is controlled to follow the VSDM network dynamics by leveraging a virtual spring-damper physics model:

$$\boldsymbol{u}_i = \ddot{\boldsymbol{p}}_i = \left[\sum_{j \in \mathcal{S}_i} \kappa_{ij}(l_{ij} - l_r^0)\vec{\boldsymbol{d}}_{ij} + \sum_{o \in \mathcal{O}_i} \kappa_{io}(l_{io} - l_o^0)\vec{\boldsymbol{d}}_{io} + \kappa_{ig} l_{ig} \vec{\boldsymbol{d}}_{ig} \right] - \gamma_i \dot{\boldsymbol{p}}_i \in \mathbb{R}^m, \tag{3.40}$$

where $\mathcal{S}_i$ is the neighbor set of robot $i \in \mathcal{V}$ in the control graph $\mathcal{G}_{\mathcal{U}}$ and $\mathcal{O}_i$ is the set of obstacles within the FOV of robot i. Given the use of a virtual spring model, l denotes the spring length between a robot i and neighboring robots j (l_{ij}), obstacles (l_{io}), and the goal g (l_{ig}), while l^0 are the desired virtual spring rest lengths, κ represents the spring constants, and $\vec{\boldsymbol{d}}$ is a unit vector. Given damping coefficients that satisfy $\gamma_i > 0$, the multi-robot system leveraging the VDSM emulates a true spring-damper mesh where dissipating forces act against the velocities, leading to an equilibrium state of zero velocity in the absence of other external forces.

Given the set of robots $\mathcal{V}$, each robot $i \in \mathcal{V}$ computes its neighbor set for control $\mathcal{S}_i \subset \mathcal{C}_i$ from the received information of nearby robots (3.4) in the communication graph $\mathcal{G}_{\mathcal{C}}$ by following Gabriel Graph rule [29, 4]. A Gabriel Graph is constructed in the following way: a robot j belongs to the neighbor set $\mathcal{S}_i$ of a robot i (i.e., a directed control edge is formed between i and j) if and only if there are no other robots $h \in \mathcal{C}_i$ within the circle of diameter $\overline{ij}$ [11] by the following,

A Gabriel Graph is constructed in the following way: a robot j belongs to the neighbor set $\mathcal{S}_i$ of a robot i (i.e., a directed control edge is formed between i and j) if and only if there are no other robots $h \in \mathcal{C}_i$ within the circle of diameter $\bar{ij}$ [11] by the following,

$$\mathcal{S}_i = \left\{ j \in \mathcal{C}_i \setminus \mathcal{V}_i^C \;\middle|\; \widehat{ihj} \leq \pi/2 \right\}, \tag{3.41}$$

where $\widehat{ihj}$, $i \neq j \neq h$ is the interior angle of the three robot positions configuration from the on-board position estimate $\hat{\boldsymbol{p}}_i^{(k)}$ of robot i and position estimates $\hat{\boldsymbol{p}}_j^{(k)}$ and $\hat{\boldsymbol{p}}_h^{(k)}$ from received information of nearby robots j and h. The set $\mathcal{V}_i^C \subset \mathcal{V}$ in (3.41) denotes robots that are deemed compromised by robot i, such that control edges are not constructed to remove compromised robots from the network for resilient control. The determination of the compromised set $\mathcal{V}_i^C$ is discussed in Section 3.5.2. As a side note, the utilization of GG rule allows for connected graphs with no crossing edges and hence an increased and uniform coverage as opposed to other graph techniques [9].

In Fig. 3.6 we show a sequence of snapshots for a simulation of a swarm of 15 robots deployed using the virtual spring model with GG in (3.40) and (3.41) to navigate towards a desired goal region while avoiding any obstacles in the environment.

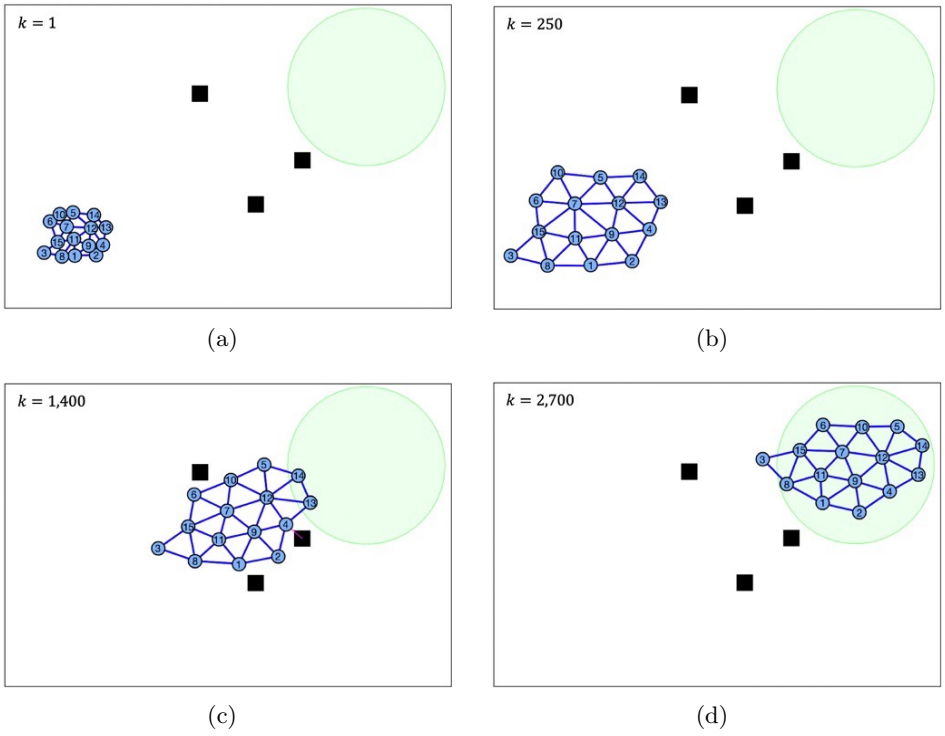


Figure 3.6: A sequence of snapshots with a robotic swarm consisting of $N = 15$ robots navigating toward a goal region (in green) using the VSDM network model in the absence of cyber attacks.

3.5.2 Multi-robot Attack Detection and Recovery

Robots within the multi-robot system monitor for inconsistent behavior of their neighboring robots to avoid stealthy attacks from hijacking uncompromised robots and, potentially, the entire robot network. Each robot $i \in \mathcal{V}$ leverages received information $\mathcal{I}_j^{(k)}$ from any neighboring robot $j \in \mathcal{C}_i$ to perform attack detection by monitoring elements within the inter-robot measurement (3.17) and inter-robot state (3.15) residual vectors, as characterized in Section 3.3.1.

To indicate that a robot $i \in \mathcal{V}$ is monitoring an sth inter-robot measurement residual element and qth inter-robot state residual element on a robot $j \in \mathcal{V}$, we denote the alarm rates as $\hat{A}_{ij,s}^{(k),\pm} = \{\hat{A}_{ij,s}^{(k),+}, \hat{A}_{ij,s}^{(k),-}\}$ and $\hat{A}_{ij,q}^{(k),\pm} = \{\hat{A}_{ij,q}^{(k),+}, \hat{A}_{ij,q}^{(k),-}\}$, respectively. If an alarm rate no longer satisfies detection bounds (i.e., suggesting inconsistent behavior), a robot i deems the monitored robot j compromised. Once inconsistent behavior is detected, the robot i then isolates and removes the compromised robot j by placing it in its compromised set $\mathcal{V}_i^C \subset \mathcal{V}$. By placing robot j in its compromised set, robot i performs a local reconfiguration of the network topology using the GG rule on the communication graph presented in (3.41), hence forming a new control neighbor set $\mathcal{S}_i' = \mathcal{S}_i \setminus \{j\}$. A previously found compromised robot j is allowed re-entry into the robot network, and the control graph, in the event that the attack disappears and j behaves as expected again (i.e.,

the residuals follows the expected distribution). In this case a local reconfiguration is again invoked using GG to compute $\mathcal{S}_i$.

Fig. 3.7 shows a pictorial example of the scheme in which compromised robots 4 and 5 are broadcasting spoofed position information (i.e., communication attack (**A1**)) to the robot network: the empty disks represent the spoofed broadcast position coordinates of the true positions of the compromised robots (red disks). The uncompromised robots (blue disks) detect non-random (i.e., inconsistent) behavior occurring from received information of robots 4 and 5, resulting in removal of any control edge connections that could affect the multi-robot system performance, where $(i, j) \notin \mathcal{E}_{\mathcal{U}}$, $i \in \mathcal{V}$, and $j = \{4, 5\}$. After removing the malicious nodes, the remaining seven nodes reconfigure using the formation rules presented in Section 3.5.1.

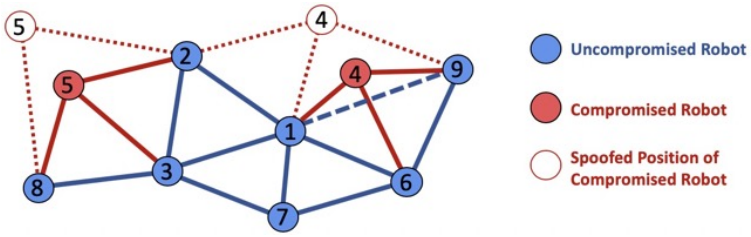


Figure 3.7: An example of a network reconfiguration where uncompromised robots isolate and remove compromised robots that are sending spoofed position broadcasts during a communication attack.

Fig. 3.8 shows, as an example, the effect of a stealthy on-board sensor attack (**A2** on an unprotected swarm with the same task in Fig. 3.6. The attack begins at time step $k = 400$ on four robots, dragging the entire multi-robot system away from the desired goal. In this example, the empty red disks represent the unreliable on-board state estimates of the compromised robots that are used for on-board control and are also broadcast to nearby agents in the robot network, whereas the red disks denote the true positions of the compromised robots. The unreliable states that are broadcast to nearby robot are then leveraged by the uncompromised robots (denoted by blue disks), which propagates the attack effects throughout the entire robot network affecting the overall mission.

3.5.3 Self Detection

Similarly, each robot $i \in \mathcal{V}$ performs self monitoring by leveraging the on-board measurement residual to search for stealthy on-board attacks on its sensors. Shown in Fig. 3.5, the CUSIGN detector is placed in the feedback of the traditional control loop to monitor the on-board measurement residual for potential attacks. As a sensor's measurements no longer satisfy an expected random behavior (i.e., alarm rates travel outside detection bounds), a robot i places itself into its compromised set $i \in \mathcal{V}_i^C \subset \mathcal{V}$.

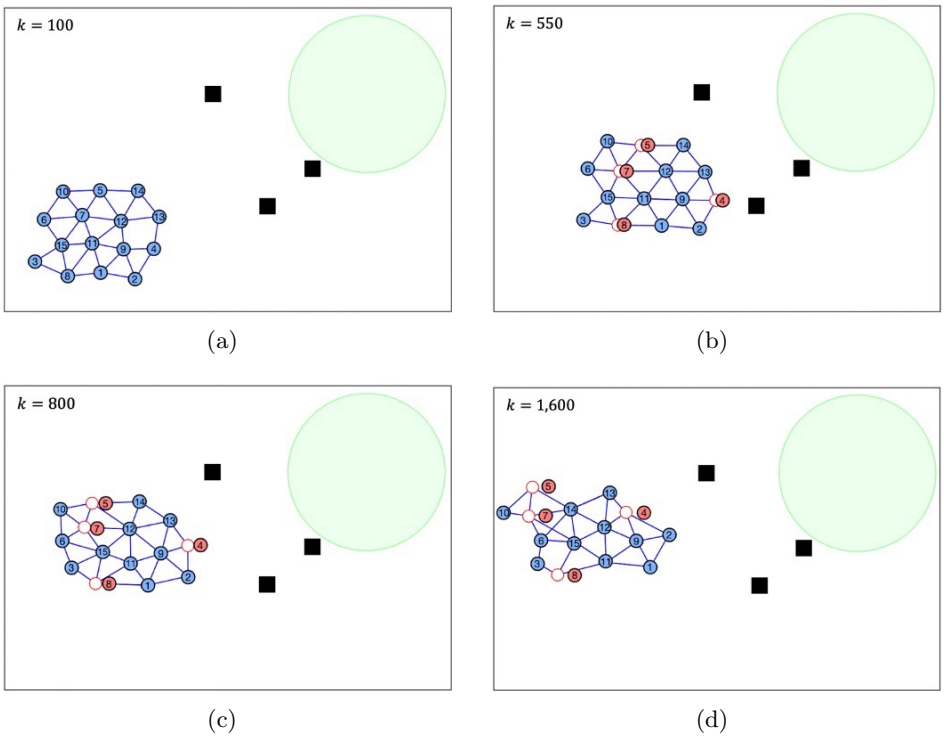


Figure 3.8: A swarm with four robots (red) that are experiencing malicious sensor attacks, causing the their state estimates (white disks) to diverge from their true state. In turn, the network is dragged away from its intended goal (green).

In this framework, the self detected compromised robot isolates itself from the rest of the network by cutting any communication broadcasts to the network (i.e., $(i,j) \notin \mathcal{E}_{\mathcal{U}},\ \forall j \in \mathcal{V}$) and also stops moving toward the goal; formally it will remove the first and third terms from (3.40), leaving any control effort only towards obstacle and other robot avoidance. While we decided to implement such a law for ease, different behaviors can be considered as we will discuss in more details in Section 3.9.

3.5.4 System Stability

The multi-robot system that leverages a virtual spring-damper mesh with Gabriel Graph rule for formation control, together with the attack detection scheme presented in Section 3.3.3, create a switching hybrid system in which edges construct and deconstruct as the robots move through the environment. Past works have proved the static (i.e., fixed topology) and dynamic (i.e., switching topology) stability of this time varying switching system by using Lyapunov theory [97, 7].

Here we extend some of these results and provide a stability proof also considering the cyber-security detection and isolation procedures described in the previous sections. As compromised robots are subject to cyber attacks that present detectable non-random behavior, certain directed edges (i.e., virtual springs used for control) from compromised robots are eliminated while others

between the remaining uncompromised robots may appear for network reconfiguration. Assuming cyber attacks are detected using the proposed CUSIGN method, the multi-robot system is guaranteed to re-converge to a new equilibrium after network reconfiguration occurs due to compromised robots being removed from the system, as formally described in the following theorem.

Theorem 3.1 *The hybrid system in* (3.40) *with switching dynamics imposed by the Gabriel Graph rule* (3.41)*, the CUSIGN detector and the network reconfiguration scheme as discussed in Sections 3.5.2 and 3.5.3 is stable (i.e., an equilibrium rest state can be reached).*

Proof: To prove Theorem 3.1, we use a similar argument as in [98, 7]. We first derive the potential energy of the system considering the removal of nodes due to detection of cyber attacks and then show that the energy of the system after detection converges to a rest state. The stored potential energy of each robot i in the network $\mathcal{V}$ is described as

$$U_i^- = \sum_{j\in\mathcal{V}} \left[\sum_{h\in\Delta h} \kappa_{jh}(l_{jh} - l_r^0)^2 \right], \;\; i \neq j \neq h. \tag{3.42}$$

where $\Delta h \subset \mathcal{S}_j \setminus \{i\}$ represents the change in neighboring robots for a robot j contained in the set $\mathcal{S}_j$ due to the removal of robot i. The assumption in (3.42) is that, because of the Gabriel graph rule, by removing a robot i new connections may appear between the remaining uncompromised robots. The total system potential energy $\boldsymbol{U}$ is then the sum of the stored potential energy of each robot $i \in \mathcal{V}$:

$$\boldsymbol{U} = \sum_{i\in\mathcal{V}} U_i^-, \tag{3.43}$$

Similarly, if a robot i is included into the system, the stored potential energy is described as

$$U_i^+ = \sum_{j\in\mathcal{V}} \left[\sum_{h\in\Delta h} \kappa_{jh}(l_{jh} - l_r^0)^2 \right], \;\; i \neq j \neq h, \tag{3.44}$$

Let $\mathcal{V}^{\mathcal{A}} \subset \mathcal{V}$ be the set of detected compromised robots. Given an instant of time when a robot i is removed due to an attack or introduced into the network, any uncompromised robots $j \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}$ re-converge to a new network equilibrium due to the changed number of uncompromised robots in the network, denoted by $|\mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}|$. In the case of the removal of robot i, the remaining uncompromised robots $j \in \mathcal{V} \setminus \{i\}$ converge to the new network equilibrium by first removing edges to robot i, i.e. $(j, i) \notin \mathcal{E}_{\mathcal{U}}$. Thereafter, the robots $j \in \mathcal{V} \setminus \{i\}$ construct edges to the new neighbors $h \in \Delta h \subset \mathcal{S}_j$, such that $(j, h) \in \mathcal{E}_{\mathcal{U}}$, to dissipate any stored energy U_i^- that belonged to robot i after it is removed. Conversely, when i is introduced to the network, the robots $j \in \mathcal{V}$ update their

virtual spring edges by (3.41) considering that robot i is now joining the system, i.e. $\mathcal{V} \cup \{i\}$, to converge to a new equilibrium.

As edge switching occurs due to network reconfiguration, the uncompromised robots $j \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}$ dissipate the stored potential energy U_i^- (or U_i^+) of robot $i \in \mathcal{V}^{\mathcal{A}}$ in order to converge to an equilibrium (i.e. rest state). Next we prove stability of the system assuming that a reconfiguration of the system has happened (i.e., by removing or adding a node to the network).

Static Stability. Let us consider the scenario in which the network topology is not switching after the removal of a compromised robot in $\mathcal{V}^{\mathcal{A}}$ or introducing a robot as described in (3.42) and (3.44). We let the total energy function of the system, including any remaining available potential energy from the removal or introduction of a robot be described as

$$V = \sum_{i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}} \frac{1}{2} \Bigg[\sum_{j \in \mathcal{S}_i} \kappa_{ij} (l_{ij} - l_r^0)^2 + \sum_{o \in \mathcal{O}_i} \kappa_{io} (l_{io} - l_o^0)^2 + \kappa_{ig} l_{ig}^2 + (\dot{\boldsymbol{p}}_i)^{\mathsf{T}} \dot{\boldsymbol{p}}_i \Bigg], \tag{3.45}$$

By taking the first order derivative of the total energy in (3.45), the time derivative becomes

$$\frac{dV}{dt} = - \sum_{i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}} \Big(\gamma_i (\dot{\boldsymbol{p}}_i)^{\mathsf{T}} \dot{\boldsymbol{p}}_i \Big). \tag{3.46}$$

in which because $\gamma_i > 0,\ \forall i \in \mathcal{V}$, we obtain that the total energy dissipation is negative semi-definite. Taking the second derivative of the total system energy, we obtain $\frac{d^2V}{dt^2} = -2 \sum_{i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}} \left(\gamma_i (\dot{\boldsymbol{p}}_i)^{\mathsf{T}} \ddot{\boldsymbol{p}}_i \right)$ which is bounded and finite if robot velocities and the differences $(l_{ij} - l_r^0)$ and $(l_{io} - l_o^0)$, $\forall i, j \in \mathcal{V}$ are finite.

Dynamic Stability. For the purpose of proving dynamic stability, we follow similar techniques to authors in [97, 7] that introduce an *energy reserve* variable ΔE that cancels switching effects of the network topology. Included in the switching topology of this proof, are effects from robots removing or introducing other robots to the network as described in (3.42) and (3.44). Given an interval of time Δt such that a switch occurs to create a different topology for uncompromised robots without network reconfiguration, the energy functions rate of variation is

$$\frac{\Delta V}{\Delta t} = \sum_{i \in \mathcal{V} \setminus \mathcal{V}^{\mathcal{A}}} \left[\frac{1}{2} \sum_{h \in \Delta h} L_U + \frac{1}{2} \sum_{j \in \Delta \mathcal{S}_i} L_r + \frac{1}{2} \sum_{o \in \Delta \mathcal{O}_i} L_o + \frac{1}{2} \kappa_{ig} l_{ig}^2 - \gamma_i (\Delta_t^{\boldsymbol{p}_i})^{\mathsf{T}} \Delta_t^{\boldsymbol{p}_i} \right]. \tag{3.47}$$

where $L_U = \kappa_{ih}(l_{ih} - l_r^0)^2$, $L_r = \kappa_{ij}(l_{ij} - l_r^0)^2$, $L_o = \kappa_{io}(l_{io} - l_o^0)^2$, $\Delta \mathcal{S}_i$ and $\Delta \mathcal{O}_i$ are switches in network topology (i.e., construction or deconstruction of edge connections) and $\Delta_t^{\boldsymbol{p}_i} = \frac{\Delta \boldsymbol{p}_i}{\Delta t}$.

Next, we build a modified potential function $V' = V + E$ where E is the *global energy reserve* by the following

$$\frac{\Delta E}{\Delta t} = \frac{1}{2} \sum_{i \in \mathcal{V} \setminus \mathcal{V}^A} \left[- \sum_{h \in \Delta h} L_U - \sum_{j \in \Delta \mathcal{S}_i} L_r - \sum_{o \in \Delta \mathcal{O}_i} L_o - \kappa_{ig} l_{ig}^2 + \gamma_i (\Delta_t^{\boldsymbol{p}_i})^{\mathsf{T}} \Delta_t^{\boldsymbol{p}_i} \right], \tag{3.48}$$

that is dependent on changes to $\mathcal{S}_i$ and $\mathcal{O}_i$, $\forall i \in \mathcal{V}$. Including the expressions (3.47) and (3.48) with the first derivative of the modified potential function $\dot{V}' = \dot{V} + \dot{E}$, we obtain the following negative semi-definite expression

$$\frac{dV'}{dt} = \frac{dV}{dt} + \frac{dE}{dt} = -\frac{1}{2} \sum_{i \in \mathcal{V} \setminus \mathcal{V}^A} \left(\gamma_i \big(\dot{\boldsymbol{p}}_i\big)^{\mathsf{T}} \dot{\boldsymbol{p}}_i \right). \tag{3.49}$$

Again, by taking the second derivative of V' we obtain $\frac{d^2V'}{dt^2} = -\sum_{i \in \mathcal{V} \setminus \mathcal{V}^A} \left(\gamma_i \big(\dot{\boldsymbol{p}}_i\big)^{\mathsf{T}} \ddot{\boldsymbol{p}}_i \right)$ which is bounded and finite. The modified energy function contains the following properties: V' is positive definite, $\dot{V}'$ is negative semi-definite, and $\ddot{V}'$ is bounded and finite. Therefore, by Barbalat's lemma we can conclude that the virtual spring network considering the modified energy function has stable dynamics. ■

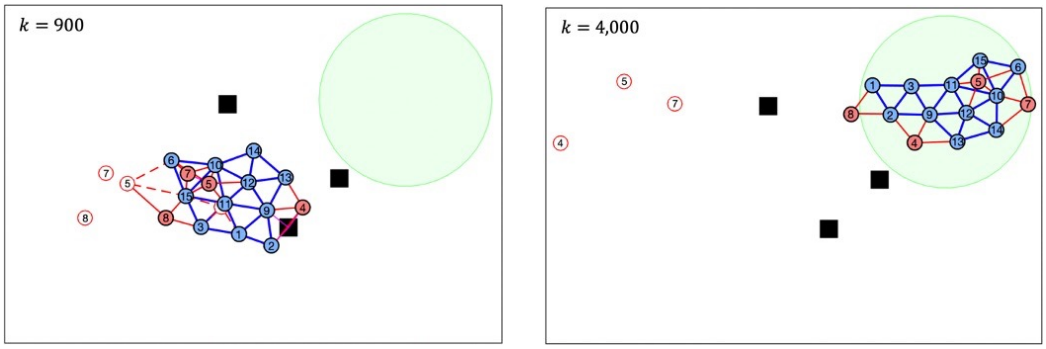
3.6 MATLAB Simulation Results

For the Matlab simulations, double-integrator point mass dynamics are considered for $N = 15$ robots in the swarm represented with the virtual spring model in (3.40) with each robot $i \in \mathcal{V}$ having a state vector $\boldsymbol{x}_i = [p_i^x, p_i^y, v_i^x, v_i^y]^{\mathsf{T}} \in \mathbb{R}^n$ consisting of positions and velocities in the x-y plane. Throughout all simulations, the set of point mass robots $\mathcal{V}$ share a maximum communication range $\delta_c = 15$m, maximum range sensing distance $\delta_r = 3$m, virtual spring rest lengths $l_r^0 = 4$m and $l_o^0 = 3$m, damping constant $\gamma_i = 3$, and spring constants $\kappa_{ij} = 15$, $\kappa_{io} = 40$, and $\kappa_{ig} = 5$. A pseudo-window length $\ell = 50$ for MRE alarm rate estimation (3.21) is used by all detectors. Additionally, the CUSIGN test variable threshold is chosen to be $\tau = 2$ such that the expected CUSIGN alarm rates are $\mathbb{E}[A^{\pm}] = \frac{1}{6}$. Each robot $i \in \mathcal{V}$ measures the x and y positions with a sampling time $t_s = 0.05$s, with measurement and process noise covariances $\boldsymbol{R} = \text{diag}(0.05, 0.05)$ and $\boldsymbol{Q} = \text{diag}(1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-5}, 1\text{e}{-5})$. During all simulations, the CUSIGN detector is compared to the BD and CUSUM detectors which monitor both the measurement and state residuals of the position for all robots $i \in \mathcal{V}$, which have thresholds tuned for an alarm rate of $A^{des} = 0.15$. Additionally, the CUSUM detector uses a bias of $b_s = 1.1\bar{b}_s$ (see [70] for further details on tuning of the BD and CUSUM detectors).

Two case studies are presented next: 1) a Man-in-the-middle communication attack and 2) a sensor spoofing attack. In both cases, we consider the multi-robot operation presented previously in Fig. 3.6 with robots $i = \{4, 5, 7, 8\}$ under attack from time instant $k = 500$.

3.6.1 Communication Attack

Our first case study involves a stealthy man-in-the-middle communication attack **(A1)**, as discussed in Section 3.2.3, in which position measurement data from compromised robots are intercepted and replaced with incorrect data before broadcasting to the rest of the swarm while slowly ramping the position in the $(-x)$-direction. Fig. 3.9 shows the behavior of the swarm once we deploy our framework, in which the compromised robots are detected in this case through the CUSIGN (3.22) detector and isolated by their neighbors.



(a) (b)

Figure 3.9: A robotic swarm navigating towards a goal point (in green) while protected from stealthy ramp attacks on communication broadcasts.

Fig. 3.10 shows the evolution of the alarm rate from the perspective of robot 2 monitoring robots $j = \{2, 3, 4, 5, 7, 8, 9, 11\}$ that belong to its neighbor set $\mathcal{S}_2$ at some point in time $k > 0$ during the stealthy communication attack case study presented in Fig. 3.9. For multi-robot detection, robot 2 monitors both the inter-robot measurement and state residuals of its neighboring robots $j \in \mathcal{S}_2$. As shown in Fig. 3.10(a), the CUSIGN detector of robot 2 that monitor the measurement residual $\boldsymbol{r}_{2j}^{(k)}$ of its neighboring robots $j \in \mathcal{S}_2$ do not detect the attack, while in Fig. 3.10(b) the detectors that are monitoring the inter-robot state residual $\breve{\boldsymbol{r}}_{2j}^{(k)}$ find the inconsistent behavior as the attacker is pushing the state estimate slowly to one side.

3.6.2 Sensor Attack

The second case study involves stealthy on-board sensor measurement attacks **(A2)** described in (2.8), attempting to hijack compromised robots to an undesired state. Similar to our simulation

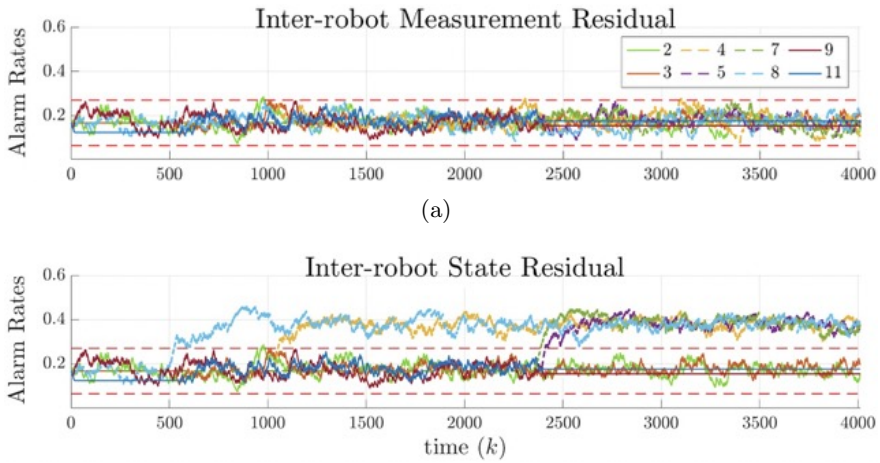
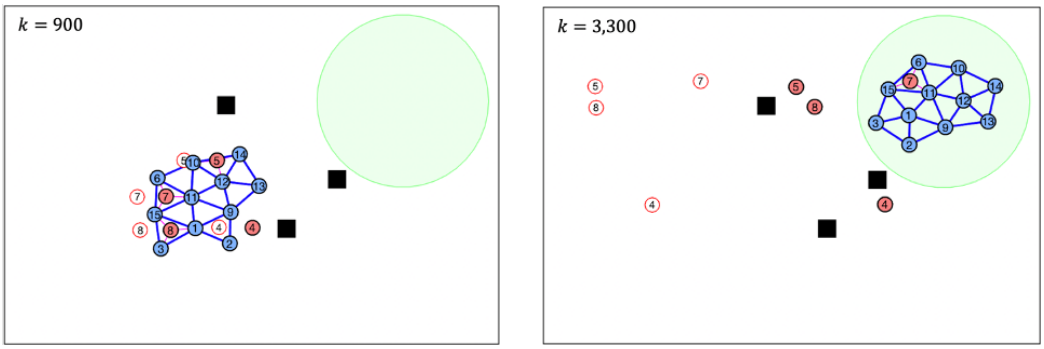


Figure 3.10: Resulting alarm rates of robot 2 performing multi-robot detection on any neighboring robots $j \in \mathcal{S}_2$ from the case depicted in Fig. 3.9 for: inter-robot (a) measurement and (b) state residuals. The state residual is able to detect stealthy communication attacks that are not detected by the measurement residual from robots $j = \{4, 13\}$.

case of a communication attack, an attack is slowly ramping the position measurement in the $(-x)$-direction, while remaining hidden from previously state-of-the-art detection schemes. Fig. 3.11 displays the detection results against stealthy sensor attacks, where uncompromised robots isolate and remove malicious robots from the network while maintaining the desired task of navigating to the goal point. The sensor spoof considered deliberately hides within the noise to evade detection from the CUSUM and Bad-Data detectors as shown in Fig. 3.12(b) and 3.12(c), but the attacker leaves trails of non-random residual behavior which is detected by the CUSIGN detector (Fig. 3.12(a)).



(a) (b)

Figure 3.11: A robotic swarm navigating towards a goal point (in green) while protected from stealthy attacks to on-board sensor measurements.

3.7 Gazebo Simulation Results

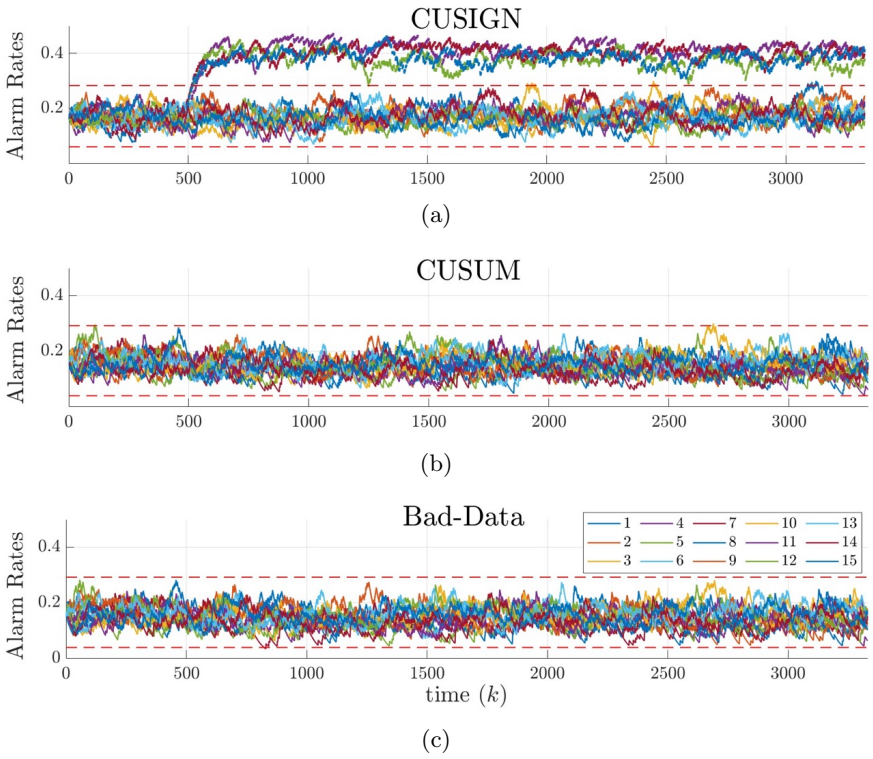


Figure 3.12: Alarm rates comparison between CUSIGN, CUSUM and Bad-Data for the case study shown in Fig. 3.11 for self detection while monitoring the on-board measurement residual for position in the x-direction as stealthy cyber attacks affect sensors on-board robots $i = \{4, 5, 7, 8\}$.

To further reinforce these results, a case study on sensor spoofing was demonstrated with an experiment in Gazebo with $N = 10$ Clearpath Jackal Robots performing a go-to-goal operation in a larger environment with more obstacles, as demonstrated in Figs. 3.14 and 3.13. We leverage Gazebo because it allows us to run longer experiments with more robots, larger spaces, and considering even stealthier attacks than experiments in our lab space. Also in this case study, we decided to use the Jackal robots to show the flexibility of our framework to deal with different dynamical models.

In the case of sensor attacks, the objective of an attacker is to slowly push a sensor measurement (e.g., positions) to one side, resulting in hijacking of the true state of the robot that diverges from the on-board state estimate. With this in mind, a larger environment is needed to perform a truly and effective stealthy attack. The robots share a maximum communication and sensing range of $\delta_c = 15$m and $\delta_r = 3$m, with virtual spring rest lengths $l_r^0 = 4$m and $l_o^0 = 3$m. Sensor measurement noise covariance follows $\boldsymbol{R} = \text{diag}(0.05, 0.05, 0.002, 0.0004)$ on the N_s sensors receiving measurements of the robot position, velocity, and heading angle. A pseudo-window length $\ell = 40$ for MRE alarm rate estimation (3.21) are used for all detectors. Additionally, the BD detector is tuned for an expected alarm rate $A^{des} = 0.15$, while CUSUM is tuned for $A^{des} = 0.1$ (with bias $b_s = 1.05\bar{b}_s$).

In Fig. 3.14 we show the sequence of snapshots for the robot network while experiencing stealthy

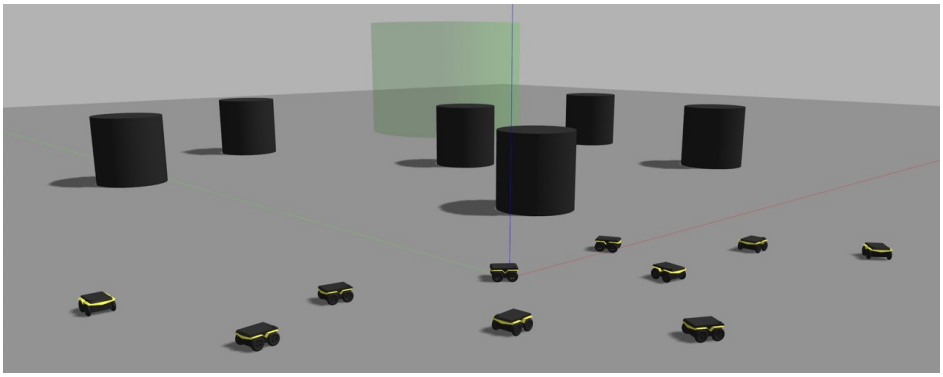


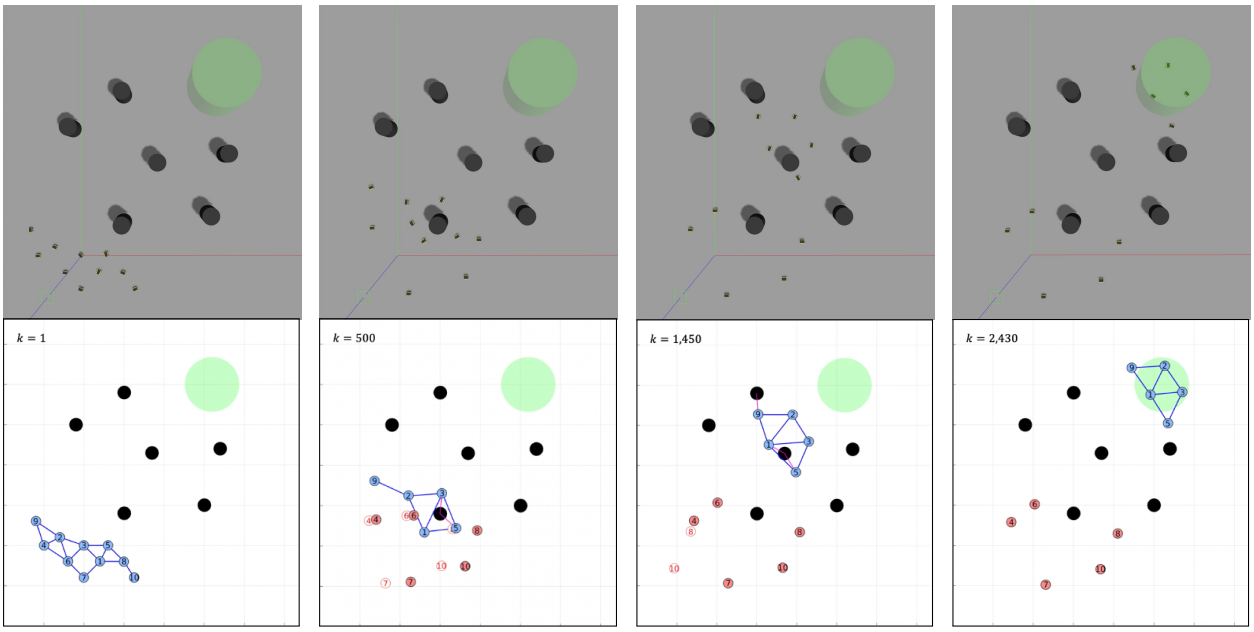
Figure 3.13: Initial positions of the $N = 10$ Clearpath Jackal UGVs within a cluttered environment for the experiments using Gazebo.

on-board sensor attacks on robots $\{7, 8, 10\}$ beginning at $k = 200$ and robots $\{4, 6\}$ beginning at $k = 400$. During the attack, compromised robots have their position measurements slowly ramped away in the $(-x)$-direction with the intention of driving the swarm away from the desired goal. Avoidance actions are required from nearby robots that leverage their on-board range sensors to prevent collisions. A comparison between detectors —CUSIGN, Bad-Data, and CUSUM— during the stealthy sensor spoof from Fig. 3.14 is shown in Fig. 3.15, with their on-board alarm rates displayed over the entire length of the case study. The CUSUM and Bad-Data detectors on-board the robots fail to detect the stealthy sensor attacks, while the CUSIGN detector is able to identify that the compromised robots are presenting inconsistent information, which allows the compromised robots to safely remove themselves from the formation.

3.8 Experiment Results

Experimental validations are performed on $N = 5$ TurtleBot2 differential-drive robots performing a go-to-goal operation within a lab environment. The hardware used is a Lenovo P51 Workstation equipped with an Intel Core i7-6820HQ processor at 2.7GHz running Linux with ROS enabled. The controller for each robot and the attacks are implemented in Matlab interfaced with ROS through the Robotic Systems Toolbox, and the operation is executed at 100Hz. In this experiment case study, the network of UGVs is tasked to navigate to a goal region (in green) while resiliently maintaining a desired network topology that satisfy edges by the Gabriel Graph rule (3.41).

Two different cases are implemented: 1) communication attack without detection and 2) communication attack with detection, with robots $i = \{3, 5\} \in \mathcal{V}$ subject to attacks. For both cases, we use the following system parameters; $\delta_c = 3\text{m}$, $\delta_r = 0.6\text{m}$, $l_r^0 = 0.7\text{m}$, $l_o^0 = 0.5\text{m}$, and $\gamma_i = 0.5$. Measurement noise covariance follows $\boldsymbol{R} = \text{diag}(0.01, 0.01, 0.002, 0.0004)$ on positions, velocity, and heading angle states, while a pseudo-window length $\ell = 40$ for MRE alarm rate estimation (2.78)



(a) (b) (c) (d)

Figure 3.14: A robotic swarm attempting to navigate towards a goal region (in green) while protected from stealthy attacks on sensor measurements. False data injections to the robot position measurements are discovered and the swarm is able to resiliently isolate and remove any robots under attack to reach the goal.

is used for all detectors. We begin with the case where no detection occurs in Fig. 3.16, showing how a stealthy communication attack is able to drive the network of UGVs to an undesirable state, away from the intended goal region. Fig. 3.17 shows the case where we have the CUSIGN detector monitoring the inter-robot state residual from information received from neighboring robots. The communication attacks on robots $\{3, 5\}$ are discovered by the remaining uncompromised robots, resulting in a network reconfiguration to remove the attacked robots. Fig. 3.18 displays the detector results from the perspective of robot $i = 1$ from Fig. 3.17, where in Fig. 3.18(a) the stealthy communication attack is not detectable on the inter-robot measurement residual, but in Fig. 3.18(b) it leaves traces of non-random behavior in the state residual $\breve{r}^{(k)}_{1j,q}$ for the position in the x-direction $\forall j \in \mathcal{V} \setminus \{1\}$.

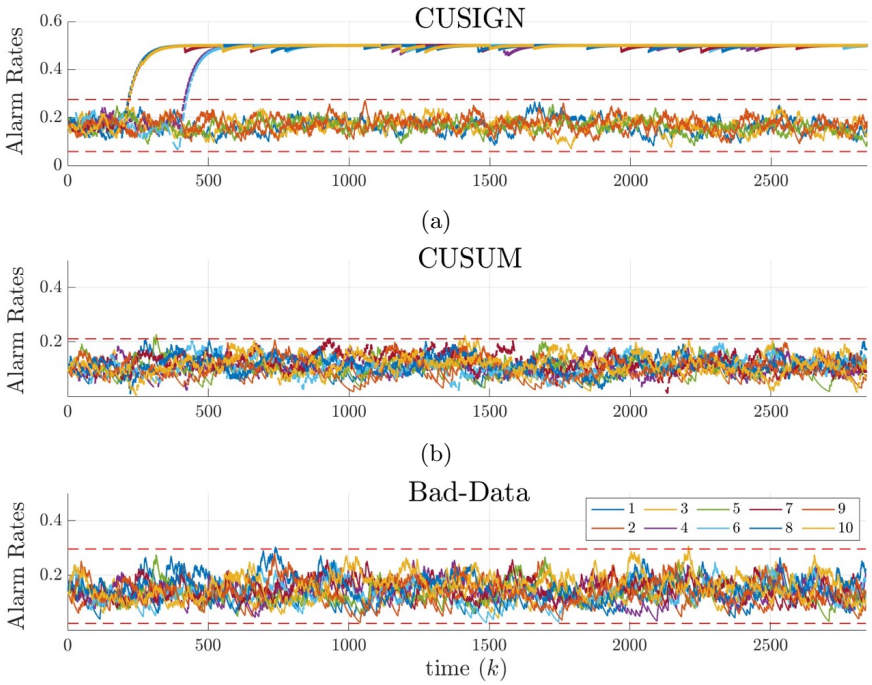
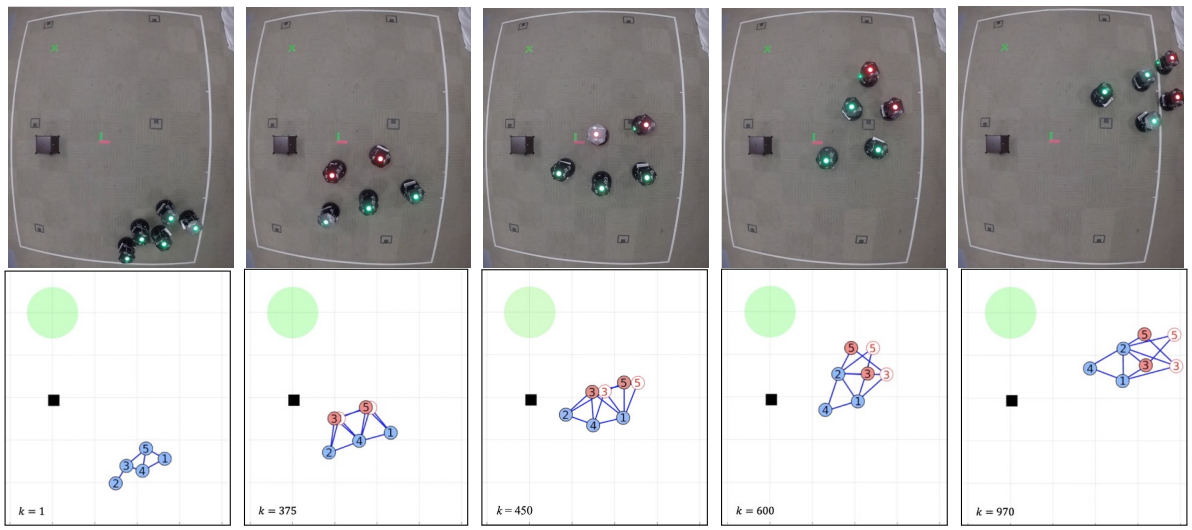
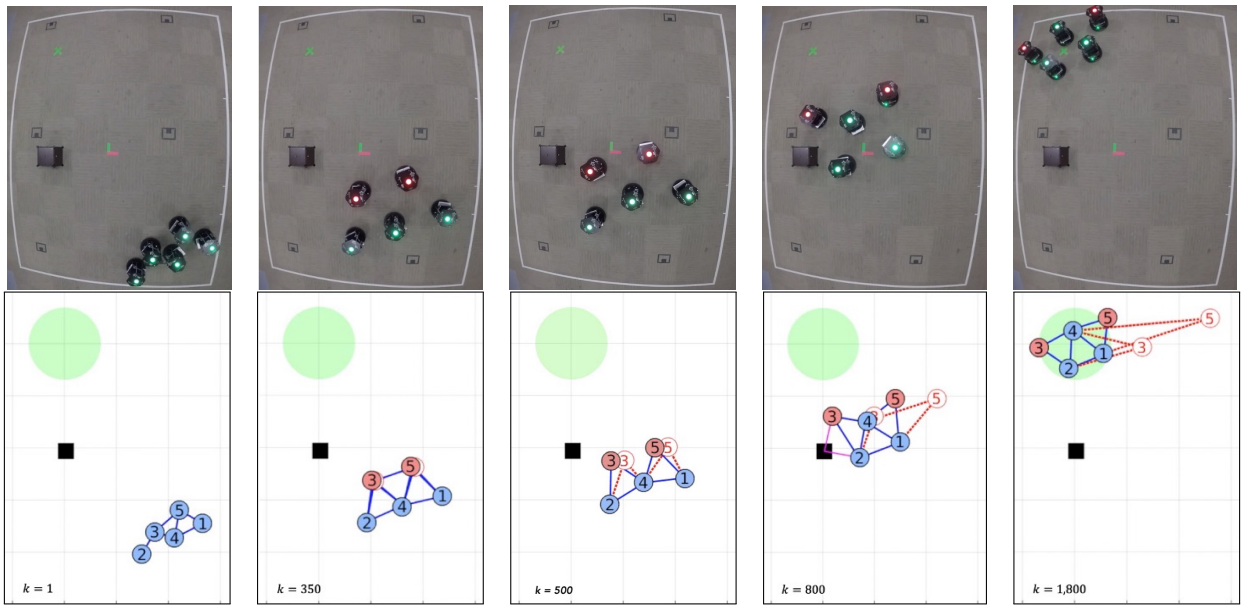


Figure 3.15: Alarm rate results from the experiment in Fig. 3.14 for robots $i \in \mathcal{V}$ performing self detection while monitoring the on-board measurement residuals as stealthy cyber attacks affect sensors on-board robots $i = \{4, 6, 7, 8, 10\}$. The CUSIGN detector detects non-random behavior of the sth measurement residual (affecting the x position) as alarm rates travel outside of detection bounds. The CUSUM and Bad-Data Detectors do not recognize the stealthy attacks.



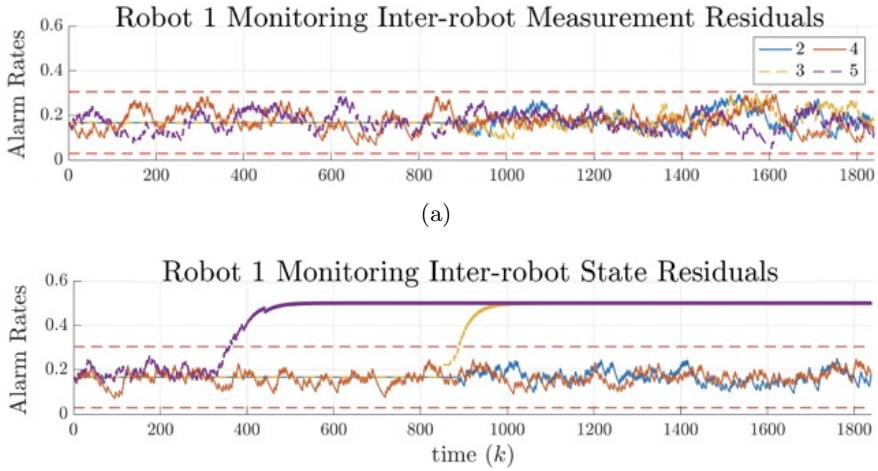
(a) (b) (c) (d) (e)

Figure 3.16: A robotic swarm attempting to navigate towards a goal region (in green) while unprotected from stealthy attacks on communication broadcasts.



(a) (b) (c) (d) (e)

Figure 3.17: A robotic swarm attempting to navigate towards a goal region (in green) while protected from stealthy attacks on communication broadcasts. False broadcast information of the robot positions are discovered and the swarm is able to isolate and remove any robots with spoofed communication broadcasts.



(a)

(b)

Figure 3.18: Resulting alarm rates from the perspective of robot 1 for the experiment in Fig. 3.17 while monitoring the inter-robot residuals.

3.9 Discussion

This chapter has presented a resilient approach to detect and defend against stealthy sensor and communication attacks that cause non-random behavior within homogeneous multi-robot systems.

The Cumulative Sign (CUSIGN) detector is introduced to counteract these stealthy attacks by monitoring alarm rates triggered by residual changes over time. Upon detection, the multi-robot system reconfigures to isolate the malicious robots in a decentralized fashion. The proposed scheme is scalable since each robot only relies on the local information received from its neighbors to assess security issues. Finally, in our extensive simulations and experiments we show how our framework can outperform well-known residual-based detection schemes like Bad-Data and CUSUM detectors. Assembling together these magnitude-based detection schemes with our proposed approach would increase the overall resilience of the system.

In the simulation and experiment demonstrations we have considered double integrator, differential drive, and skid-steering dynamics to show the generality and flexibility of our framework. The main assumption for our framework is to have *a priori* knowledge about the vehicle dynamics and the noise models. Currently we have assumed that communication within the network is ideal, such that synchronization errors, time delays, and communication failures are negligible. Future efforts expanding on this framework could include and leverage more accurate communication models with uncertainties as introduced in [74] and [78] to further increase resilience, for example, by using the dependencies between communication quality and distance between two communicating agents (i.e., as a side-channel detection scheme). Expanding this framework to heterogeneous robotic systems with different classes of vehicles and sensing capabilities is also another aspect that could be investigated in the future.

From a recovery/reconfiguration perspective, we believe that an important direction forward would be on how to deal with the robots that are found compromised. In this chapter, compromised robots were isolated and removed from the network, to avoid their malicious effect on the coordination of the rest of the uncompromised robots. However, more complicated approaches can be considered to stop the malicious robots; such as surrounding or dragging them toward a safe state. In order to enable such behaviors, it is necessary to predict the state of the compromised vehicles. One possibility here is to research checkpointing and recovery methods, inspired by traditional software engineering, by leveraging saved past reliable states and control inputs of the compromised system and rolling forward to predict its reconstructed state after it was compromised, similar to literature as in [46].

Predicting the intention of an attacker is also in our agenda since this will further increase resilience to better recover a system. The inclusion of learning enabled components like regression and classification techniques could further improve the on-board computation for detection. Furthermore, investigating the effects of worst-case attack sequences that an attacker can perform while evading detection from the CUSIGN detector to characterize the maximum damage in terms of the resulting true state divergence from the on-board state estimate.

Finally, another direction that we could head in is providing a framework for resilient task-based multi-robot operations in the presence of cyber attacks. Like most multi-robot system operations, and unlike the trivial go-to-goal operations implemented in this chapter, the robot swarms are typically tasked to complete one or more objectives. However, if malicious actors are actively attempting to intercept this critical tasked-based information that is communicated between robots, an attacker can gain knowledge of the task/objective and intentionally seek to compromise the operation. In the next chapter, we tackle this problem by designing a framework for the mobile robot team to share these safety-critical tasks without explicitly sending the information through communication broadcasts that can potentially be intercepted by malicious agents.

Chapter 4

Detection and Inference of Randomness-based Behavior for Resilient Multi-robot Coordinated Operations

In this chapter, a decentralized framework for coordinated multi-robot systems is introduced to allow for resilient task-based operations in the presence of malicious communication broadcast information attacks. This framework is an extension of the decentralized framework introduced in Chapter 3, while considering the scenario of malicious eavesdroppers potentially intercepting safety-critical information being passed between robots. In the context of this chapter, safety-critical information is considered to be important tasks that are necessary for the robots within the multi-robot system to cooperatively complete. Robots infer the safety-critical tasks via hidden motion signatures emitted amongst each other to protect the information from any malicious attackers. The cooperative multi-robot system framework is highlighted with MATLAB simulations and lab experiments using swarms of TurtleBot2 UGVs. This chapter is based on the following publication:

- P.J. Bonczek, N. Bezzo, "Detection and Inference of Randomness-based Behavior for Resilient Multi-vehicle Coordinated Operations," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*

4.1 Introduction

The use of coordinated multi-robot systems to perform various tasks has been extensively explored for many years [96, 104, 61, 58]. By leveraging multiple robots instead of only one, it is possible to perform more operations, and complete a task faster and more efficiently. Examples of such operations that can benefit from the use of multi-robot systems are search and rescue operations [96] depicted in Fig. 4.1, surveillance [104], military convoying/platooning [61], and exploration missions [58]. Generally, approaches that leverage multi-robot systems assume that all robots are cooperative while performing the desired operations to maintain swarming formations and can exchange all necessary information to achieve the desired goal. However, these robots are susceptible

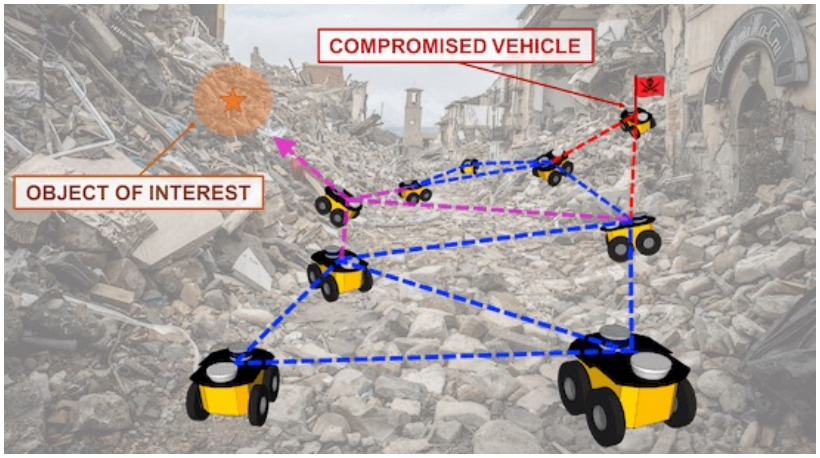


Figure 4.1: Pictorial motivation of the problem in this paper in which a multi-robot system cooperatively performs a task while inferring the objective of other teammates and detecting if they are compromised by cyber attacks.

to malicious external attacks, especially on their communication infrastructure, which can affect the entire network performance. For example, with a Man-In-The-Middle (MITM) attack [20], an attacker intercepts a communication broadcast and replaces it with altered data which are then received by neighboring robots. Successful attackers are able to purposefully block important information from being received by nearby robots in the formation or control the entire multi-robot network to an undesired location.

Safety-critical information, if not properly encrypted, can also be intercepted creating further security issues. Although encryption techniques can be deployed, there exists attacks that are capable of discovering encryption keys to extract data. The most secure option is to avoid exchanging data altogether. In this chapter, we propose to leverage side-channel information that contains hidden data, which is unknown to malicious attackers. For example, if a robot discovers an object of interest whose identity needs to be kept secret, as depicted in Fig. 4.1, it could perform a certain signature motion to indicate to neighboring robots of the discovered object. This motion triggers the surrounding robots to infer its position and switch tasks to get attracted to the same object. In this way, a robot can collect data and infer the behavior of other robots without explicitly broadcasting important information. With such premises, we focus on applications for cooperative autonomous robot networks in the presence of adversaries. We expand on literature that leverage virtual springs for decentralized formation control [97, 17, 6, 5] while introducing a monitoring approach to detect inconsistent behaviors between expected and received data to provide: 1) resiliency to cyber attacks on communication broadcasts and 2) discovery of a hidden signature via side-channel states.

4.2 Preliminaries

4.2.1 Multi-robot System Model

Let us consider a multi-robot network of N homogeneous mobile robots modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where we denote $\mathcal{V} = \{1, \ldots, N\}$ as the robot set and the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, such that an edge $(i, j) \in \mathcal{E}$ indicates a connection from robot $i \in \mathcal{V}$ to robot $j \in \mathcal{V}$. All robots are considered to have second order dynamics that can be represented in a linear time-invariant (LTI) state space form in (3.1) that leverage virtual spring-damper meshes for decentralized formation control from (3.40).

4.2.2 Attack Model

We assume the multi-robot network is navigating within an adversarial environment, such that individual robots may be subject to malicious communication attacks (e.g., MITM attacks [20]). In the case of an attack on an unprotected proximity-based formation, a single compromised robot can affect the entire network of N robots as the effects of the attack are propagated throughout the network. During a persistent communication attack, we assume that an attacker can continuously intercept and modify broadcast data with stealthy (i.e., hidden within the system noise profile) information in an attempt to intentionally fool (i.e., hijack) the robot network. Each robot i exchanges state estimates, nearby obstacle positions, and neighbor set information at every time instance k such that nearby robots have knowledge of its intended motion by construction of the network model in (3.40). We indicate the spoofed broadcast information from a robot $i \in \mathcal{V}$ that is received by other robots as:

$$\begin{aligned} \hat{\boldsymbol{x}}_i^{(k)} + \boldsymbol{\xi}_i^x &\longrightarrow \tilde{\hat{\boldsymbol{x}}}_i^{(k)}, \\ \boldsymbol{p}_o + \boldsymbol{\xi}_i^o &\longrightarrow \tilde{\boldsymbol{p}}_o, \ \forall o \in \mathcal{O}_i, \\ \{\mathcal{S}_i \setminus \mathcal{S}_i^{\xi^-}\} \cup \mathcal{S}_i^{\xi^+} &\longrightarrow \tilde{\mathcal{S}}_i, \end{aligned} \tag{4.1}$$

where $\boldsymbol{\xi}_i^x \in \mathbb{R}^n$ and $\boldsymbol{\xi}_i^o \in \mathbb{R}^2$ denote the attack vectors on state and obstacle positions, whereas the sets $\mathcal{S}_i^{\xi^-} \subset \mathcal{V}$ and $\mathcal{S}_i^{\xi^+} \subset \mathcal{V}$, $\{\mathcal{S}_i^{\xi^-} \cap \mathcal{S}_i^{\xi^+}\} = \emptyset$ are robot identifications that are removed from and added to the original neighbor set $\mathcal{S}_i$, respectively. For any attack vector $\boldsymbol{\xi}_i^x \neq 0$, $\boldsymbol{\xi}_i^o \neq 0$, $\forall o$, or sets satisfying $|\mathcal{S}_i^{\xi^+}|, |\mathcal{S}_i^{\xi^-}| > 0$, an attacker is replacing the original message such that the received information by any nearby neighbors will differ from the intended broadcast.

4.3 Problem Formulation

Given the network described by the virtual spring model (3.40) and the *control graph* $\mathcal{G}_{\mathcal{U}}(\mathcal{V}, \mathcal{E}_{\mathcal{U}})$, we are interested in solving the following problems:

Problem 4.1 (Robot Inconsistency Detection) *Create a decentralized detection policy $\mathcal{P}_d$ such that a robot $j \in \mathcal{V}$ that is experiencing inconsistent behavior can be discovered and isolated by any robot $i \in \mathcal{V}$ such that,*

$$(i, j) \notin \mathcal{E}_{\mathcal{U}},\ i \neq j, \tag{4.2}$$

to prevent undesirable effects to the multi-robot network.

A second problem that we explore is to enable indirect exchange of information by leveraging signature mobility behaviors of the agents of the swarm. While navigating through an adversarial environment, robots that come into sensing range of an object of interest desire to notify the remaining robots in the network of their discovery without revealing explicitly the identification and position of the object to maintain secrecy from adversaries.

Problem 4.2 (Hidden Signature Detection) *Given a robot $i \in \mathcal{V}$ that has found an object of interest while navigating within an environment, find a control policy $\mathcal{P}_u$ to covertly provide an identifiable hidden signature $\boldsymbol{u}_i^{\mathcal{H}} \in \mathbb{R}^m$ for any nearby robots $j \in \mathcal{C}_i \subset \mathcal{V}$ to detect without explicitly sending information of the discovered object through communication broadcasts.*

Upon recognizing a signature behavior, neighbors of the robot will estimate the position of the object based on the same signature and switch toward that object.

4.4 Approach

In this section, the decentralized monitoring framework is described for detection and isolation of inconsistently behaving robots in the network, while allowing each robot to provide a hidden signature for nearby robots. The diagram in Fig. 4.2 summarizes our proposed scheme in which each robot follows the primary or hidden control model, as well as detects whether neighboring robots have expected behavior according to the primary or hidden models.

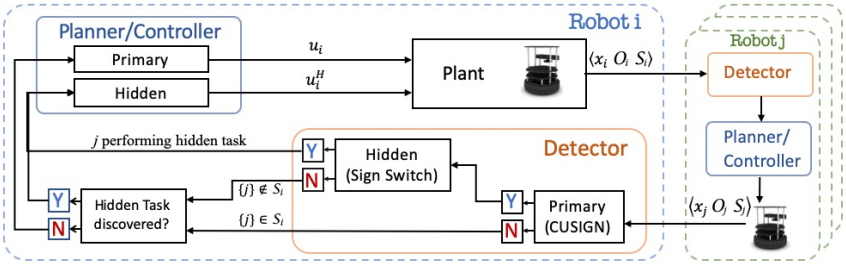


Figure 4.2: Overall framework architecture followed by each robot $i \in \mathcal{V}$.

4.4.1 Multi-robot Monitoring

During operations, each robot monitors nearby robots for consistent behavior according to the virtual spring-damper primary network model described in (3.40). Additionally, each robot i receives broadcasts containing state estimate vector information $\hat{\boldsymbol{x}}_j$ and control neighbor sets $\mathcal{S}_i$ from any nearby robot $j \in \mathcal{C}_i$ within communication range as represented in (3.4). Differing from multi-robot monitoring in the previous chapter, each robot i infers the motion of each nearby robot j based on their received information and the primary network model (3.40) where edges $\mathcal{E}$ are constructed via Gabriel Graph rule [29, 4]. This robot i is able to make state evolution predictions of a nearby robot $j \in \mathcal{C}_i$ such that the neighbor set of robot j satisfies $\mathcal{S}_j \subset \mathcal{C}_i$. The inclusion of the neighbor set $\mathcal{S}_j \subset \mathcal{C}_i$ is needed in order for robot i to predict the future state of the system using (3.40). The inter-robot state prediction $\bar{\boldsymbol{x}}_{ij}^{(k+1)} \in \mathbb{R}^n$ of a robot j computed by a robot i is computed as

$$\bar{\boldsymbol{x}}_{ij}^{(k+1)} = \boldsymbol{A}\hat{\boldsymbol{x}}_j^{(k)} + \boldsymbol{B}\boldsymbol{u}_{ij}^{(k)} \tag{4.3}$$

where $\boldsymbol{u}_{ij}^{(k)} \in \mathbb{R}^m$ is the estimated input for robot j that is computed by robot i which follows the primary network model (3.40). At every kth time iteration, a robot i compares the *inter-robot residual* $\boldsymbol{r}_{ij}^{(k)}$, defined as the difference between the received state information $\hat{\boldsymbol{x}}_j^{(k)}$ and the computed state prediction of a robot j, by the following:

$$\boldsymbol{r}_{ij}^{(k)} = \hat{\boldsymbol{x}}_j^{(k)} - \bar{\boldsymbol{x}}_{ij}^{(k)} \in \mathbb{R}^n. \tag{4.4}$$

If a robot j is attack-free and is following the primary network model while being monitored by a robot i, each element $q \in \{1, \ldots, n\}$ of the inter-robot residual vector is normally distributed $r_{ij,q}^{(k)} \sim \mathcal{N}\big(0, \sigma_{r,q}^2\big)$ characterized in the same manner as in (3.16). Similarly, each qth inter-robot vector element of $\boldsymbol{r}_{ij}^{(k)}$ is a zero-mean normally distributed variable follows (2.52) during nominal (i.e., no attack) conditions. To monitor whether the incoming information from nearby robots is behaving in an expected random manner with respect to the primary network model (3.40), we employ the Cumulative Sign (CUSIGN) detector to check for randomness with the procedure formalized in (3.22). The multi-robot detection procedure on a robot i accumulates the signed values of the inter-robot residual in the CUSIGN test variables for a robot j and triggers an alarm $\zeta_{ij,q}^{(k),\pm} = 1$ when a user-defined threshold $\tau \in \mathbb{N}$ is reached, otherwise $\zeta_{ij,q}^{(k),\pm} = 0$. As either of the test variables reach their respective thresholds, the test variable is then reset back to zero. The alarms for each qth element are then sent to the run-time update (2.78) for alarm rates $\hat{A}_{ij,q}^{(k),-}$ and $\hat{A}_{ij,q}^{(k),+}$, for simplicity denoted as $\hat{A}_{ij,q}^{(k),\pm}$, at each time instant k.

4.4.2 Hidden Signature Generation

During operations, the mobile robots are tasked to converge to observed objects of interest while navigating through the environment. As an ith robot comes within sensing distance δ_r of the on-board range sensor with respect to an object,

$$l_{ip} = \|\boldsymbol{p}_i - \boldsymbol{p}_p\| \leq \delta_r, \tag{4.5}$$

where $\boldsymbol{p}_p$ is the position of an object of interest, the robot will notify neighboring robots by creating a detectable hidden signature. To achieve this, the robot switches to a *hidden* virtual spring-damper model described by,

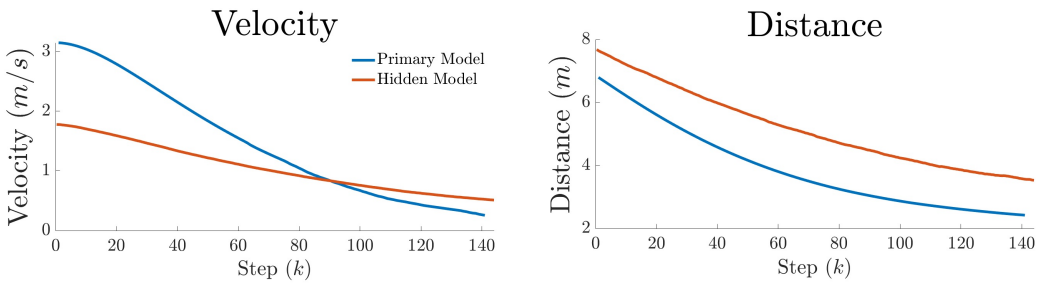
$$\boldsymbol{u}_i^{\mathcal{H}} = \ddot{\boldsymbol{p}}_i = \left[\kappa_h(l_{ip} - l_h^0)\vec{\boldsymbol{d}}_{ip} - \gamma_h \dot{\boldsymbol{p}}_i\right] \in \mathbb{R}^m, \tag{4.6}$$

where the virtual spring-damper parameters $\kappa_h \neq \kappa_v$ and/or $\gamma_h \neq \gamma_v$ are distinct from the primary network model in (3.40) to enable an identifiable dynamical signature. A robot i that follows the hidden model (4.6) removes all virtual spring interactions to neighboring robots and the goal from the primary network model that affect its control input. To maintain secrecy from attackers (with regards to the observance of the object of interest), a robot i will continue to broadcast state, observed obstacle positions, and its neighbor set information to nearby robots as it would in nominal conditions. In this way, a malicious agent who is listening will continue to see the same type of information as before. Any manipulation of such information that does not conform with the new hidden model (4.6) or with the primary model in (3.40) will be considered a cyber attack.

The challenge that arises is that the object position $\boldsymbol{p}_p$ remains unknown to the other robots in the network. In comparison to the primary model (3.40), nearby robots do not receive all necessary information when a robot i follows the hidden model (4.6) to monitor for consistency. This is due to constraints set in Problem 4.2, that information regarding a discovered object of interest can not be explicitly shared with the network to protect from interception by attackers.

4.4.3 Hidden Signature Detection

Given that the hidden model (4.6), robot dynamics, and maximum sensing range δ_r are known by all robots, an expected robot velocity behavior can be leveraged as a robot converges toward an object (i.e., a decaying velocity magnitude). More specifically, any robot i can recognize the hidden signature by monitoring the received velocity estimate $\hat{\boldsymbol{v}}_j^{(k)}$ behavior from a robot j and compare it to the expected velocity decay behavior from the hidden model. Shown in Fig. 4.3(a) is an example of the differing expected velocity behavior between springs of the primary and hidden models with the corresponding distances to the object.



(a) (b)

Figure 4.3: Differing expected behavior of the (a) velocity decay, and (b) distance to the object for the two different virtual spring-damper models.

Given that each robot i is making state predictions of any neighboring robot j according to the primary network model (3.40), an alternative action by this robot (i.e., utilizing the hidden model) would result in an unexpected behavior. Alarm rates from CUSIGN on-board a robot i that is monitoring robot j, in turn, go beyond detection bounds due to the unexpected behavior and robot j is placed in the compromised robot set $\mathcal{R}_i \subset \mathcal{V}$. Next, robot i would begin to monitor the received velocity information of robot j to determine if its behavior follows the hidden model (4.6). A velocity prediction of robot j by a robot i given $j \in \mathcal{R}_i$ is made from the received velocity estimate $\|\hat{\boldsymbol{v}}_j^{(k)}\|$ by,

$$\bar{v}_{ij}^{(k+1)} = h\big(\|\hat{\boldsymbol{v}}_j^{(k)}\|\big), \tag{4.7}$$

where the function $h(\cdot)$ represents the expected velocity behavior according to the hidden spring model (4.6), as shown in Fig. 4.3(a). At each time iteration k, the *hidden velocity residual* $\breve{r}_{ij}^{(k)}$ — the difference between received velocity magnitudes and velocity predictions using the hidden model — of robot j is computed by the following,

$$\breve{r}_{ij}^{(k)} = \|\hat{\boldsymbol{v}}_j^{(k)}\| - \bar{v}_{ij}^{(k)} \in \mathbb{R}, \tag{4.8}$$

to monitor whether robot j is following the hidden model in (4.6). We leverage the known zero-mean Normally distributed velocity estimate provided by robot j when characterizing the hidden velocity residual. An assumption can be made such that the received velocity estimate information from robot j is also approximately zero-mean Normally distributed around the expected velocity decay behavior in $h\big(\|\hat{\boldsymbol{v}}_j^{(k)}\|\big)$, *only* if robot j is following the hidden model. In this scenario, the hidden velocity residual (4.8) is expressed as a random variable that presents the following characteristics:

$$\begin{aligned} \Pr\left(\breve{r}_{ij}^{(k)} < 0\right) &= \breve{p}_- = 0.5, \\ \Pr\left(\breve{r}_{ij}^{(k)} > 0\right) &= \breve{p}_+ = 0.5, \end{aligned} \tag{4.9}$$

where the probability of the hidden velocity residual being greater or less than zero is equal. A random variable with characteristics that follow (4.9) should present an expected sign switching rate behavior (i.e. how frequently $\breve{r}_{ij}^{(k)}$ changes signs) in accordance to the probabilities in (4.9). To capture the rate of sign switching, we leverage an alarm that is triggered (i.e., $\psi_{ij}^{(k)} = 1$) when a sign switch occurs at a time k. The procedure to trigger a sign switching alarm follows:

$$\psi_{ij}^{(k)} = \begin{cases} 1, & \text{if } \operatorname{sgn}\big(\breve{r}_{ij}^{(k)}\big) = -\operatorname{sgn}\big(\breve{r}_{ij}^{(k-1)}\big), \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

The sign switching alarm $\psi_{ij}^{(k)} \in \{0, 1\}$ is then sent into the MRE algorithm (2.78) to provide an updated run-time estimate of the hidden signature sign switching alarm rate $\hat{H}_{ij}^{(k)} \in [0, 1]$ at time instance k.

Lemma 4.1 *Given a robot j that is following the hidden model* (4.6) *while being monitored by robot i, the expected sign switching rate to signify random behavior is* $\mathbb{E}[H] = \frac{1}{2}$.

Proof: We first examine the asymptotic distribution of the expected number of observed runs $\mathbb{E}[U]$ from the Wald-Wolfowitz runs test [105]. Then, we convert $\mathbb{E}[U]$ over a defined sequence length to a rate described by how frequently runs should occur (i.e., how often sign switching occurs) by leveraging the known characteristics of the probabilities $\breve{p}_+$ and $\breve{p}_-$, such that the random variable follows $\mathbb{E}[H] = 2\breve{p}_+\breve{p}_- = \frac{1}{2}$, thus concluding the proof. ■

Lemma 4.2 *The expected variance of the sign switching rate $\hat{H}_{ij}^{(k)}$ for a robot j that follows the hidden model* (4.6) *while monitored by robot $i \in \mathcal{V}$ is* $\operatorname{Var}[H] = \frac{1}{4(2\ell-1)}$.

Proof: Let the expectation of a sign switch be modeled by a Binomial distribution where the probability of success (i.e., sign switch) is $\mathbb{E}[H]$. By normal approximation and utilizing MRE (2.78) for sign switching rate estimation, the random variable follows a normal distribution with variance $\operatorname{Var}[H] = \frac{\mathbb{E}[H](1-\mathbb{E}[H])}{2\ell-1} = \frac{1}{4(2\ell-1)}$, concluding the proof. ■

The following corollary provides bounds of $\hat{H}_{ij}^{(k)}$ to satisfy an expected behavior to detect the hidden model signature.

Corollary 4.1 *Given the sequence of hidden velocity residuals $\breve{r}_{ij}^{(k)}$, hidden signature detection occurs by the sign switching alarm rate when $\Psi_- \leq \hat{H}_{ij}^{(k)} \leq \Psi_+$ is satisfied.*

Proof: A proof can be obtained by leveraging confidence intervals within a Normal Distribution. Due to page limitations, we omit the proof. ■

To summarize Corollary 4.1, when the sign switching alarm rate for the detection of the hidden signature satisfies,

$$\hat{H}_{ij}^{(k)} \in [\Psi_-, \Psi_+] \longrightarrow \textit{Signature Detection}, \tag{4.11}$$

robot i detects a hidden signature behavior in j.

4.4.4 Object of Interest Position Estimation

Robot i reacts by estimating the position of the object by leveraging the training set (i.e., expected hidden model behavior) mapping $f : \mathbb{R}^2 \to \mathbb{R}$ in Fig. 4.3 that maps the received velocity estimate of robot j to its distance to the object by,

$$\hat{d}_{p,ij}^{(k)} = f(\|\hat{\boldsymbol{v}}_j^{(k)}\|). \tag{4.12}$$

The position of the object $\boldsymbol{p}_p$ is then estimated by robot i from the received position information of robot j by,

$$\hat{\boldsymbol{p}}_{p,ij}^{(k)} = \hat{\boldsymbol{p}}_j^{(k)} + \hat{d}_{p,ij}^{(k)} \vec{\boldsymbol{d}}(\|\hat{\boldsymbol{v}}_j^{(k)}\|), \tag{4.13}$$

where $\vec{\boldsymbol{d}}(\cdot)$ is a unit vector indicating velocity direction of robot j. Once the object position coordinates $\hat{\boldsymbol{p}}_{p,ij}^{(k)}$ have been estimated, robot i then detaches virtual springs from its neighbors and goal to converge to the object of interest by also following the hidden model (4.11). Robots within the network will continue to converge toward the point of interest until its task is completed. Upon completion, all robots involved with the hidden task return to their normal control behavior with the primary network model in (3.40).

4.5 Simulations Results

For the simulation case study, we consider $N = 10$ mobile robots treated as double integrator point-masses navigating in an x-y plane. A sequence of snapshots are presented in Fig. 4.4 showing the network of robots resiliently navigating through an obstacle filled environment. From Fig. 4.4(c)-(h), robots $\{4, 8\}$ (red circles) are subject to MITM attacks that falsify position information with the intention of hijacking the network.

In Fig. 4.4(e) robot 1 discovers an object and then switches to the hidden model (4.6) to covertly notify neighboring robots about the discovery. In turn, neighboring robots detect this hidden signature (4.11), estimate the position of the object (4.13), then also switch to the hidden model to converge to the object. CUSIGN and sign switching alarm rates from the perspective of robot $i = 7$ are provided in Fig. 4.5 while monitoring neighboring robots for primary and hidden model behaviors. Alarm rates for CUSIGN (3.22) monitoring robots $\{1, 2, 4, 8, 10\}$ travel beyond detection

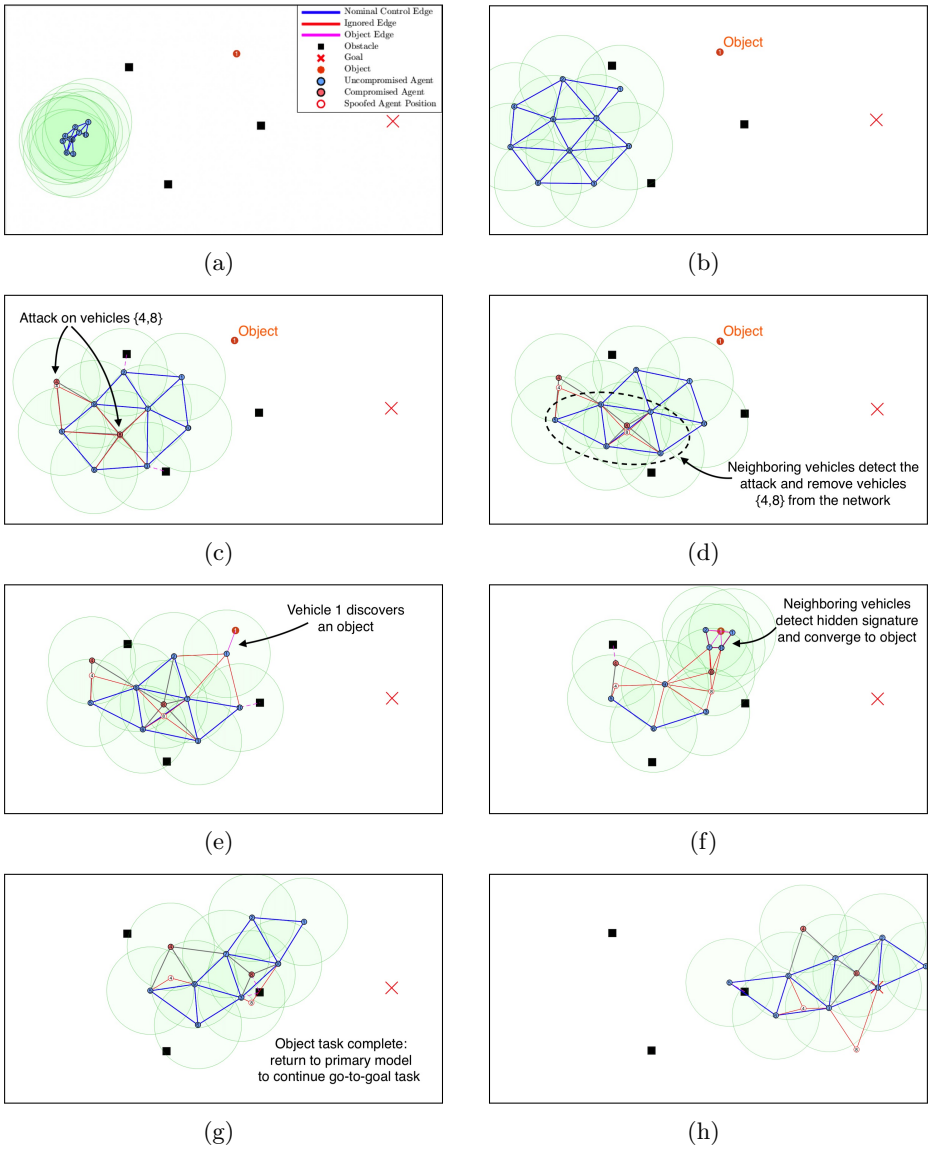
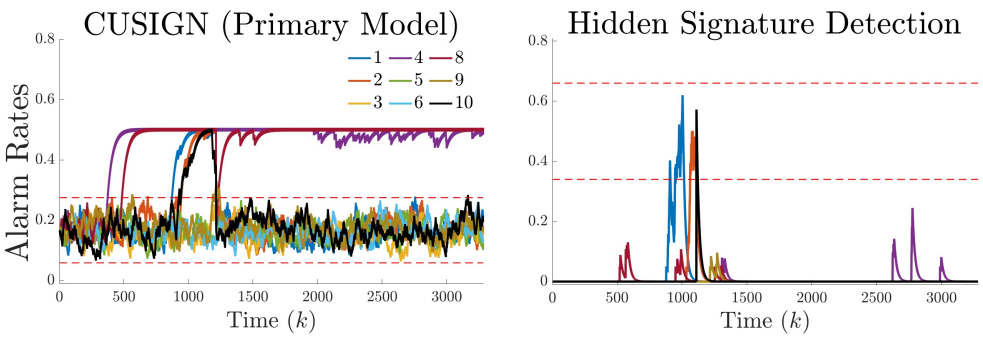


Figure 4.4: A network of $N = 10$ robots resiliently navigate through an obstacle-filled (black squares) environment to a desired goal (red 'X'). Robots converge to an object (orange disk) as it comes within their viewing range (green disks) or if they detect the hidden signature from neighboring robots.

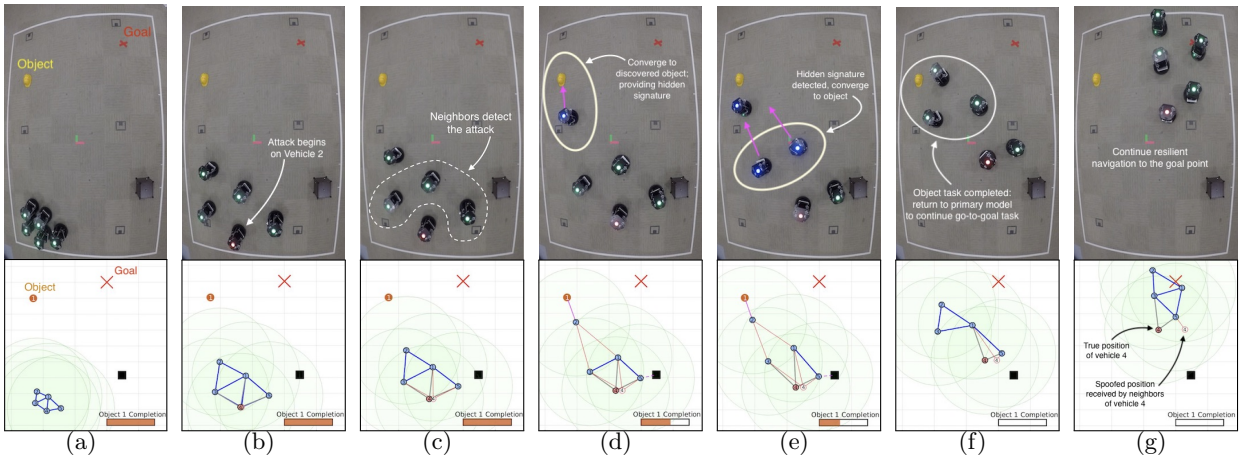
bounds indicated by red dashed lines. However, shown in Fig. 4.5(b), the hidden signature alarm rates (4.11) for robots $\{1, 2, 10\}$ satisfy the hidden model detection bounds to signify consistent behavior is occurring with respect to the hidden model (4.6), thus deeming these robots trustworthy. Alternatively, robots $\{4, 8\} \in \mathcal{R}_7$ are treated compromised as a hidden signature was not detected from their motion.



(a) (b)

Figure 4.5: (a) CUSIGN detection alarm rates from the perspective of robot $i = 7$ and (b) sign switching alarm rates for hidden signature detection.

4.6 Experiment Results

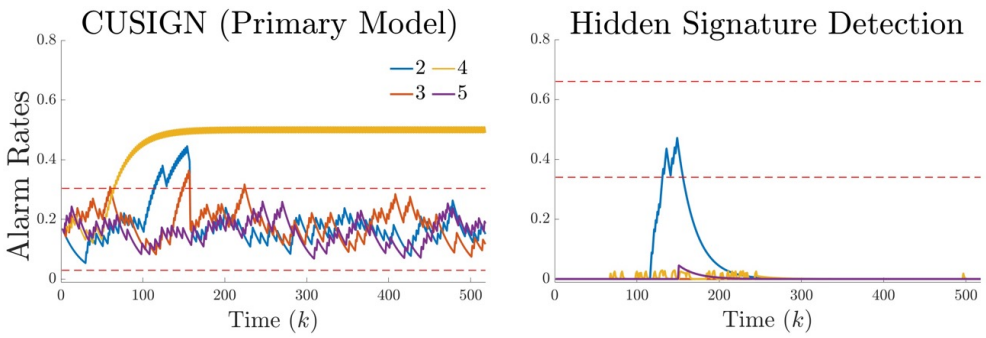


(a) (b) (c) (d) (e) (f) (g)

Figure 4.6: An experiment showing a network of $N = 5$ TurtleBot2 robots resiliently navigating to a goal (red 'X'). Robot 2 discovers an object (yellow helmet) as comes within its sensing range (depicted by the green translucent circle), then provides a hidden signature behavior for nearby robots to recognize as it converges to the object. Neighboring robots also converge to the object of interest upon detection of this hidden signature.

Experimental validations are performed on $N = 5$ TurtleBot 2 differential-drive robots performing a go-to-goal operation within a lab environment. Snapshots of this experiment are presented in Fig. 4.6 capturing the following sequence of events; the initial robot positions (Fig. 4.6(a)), robot 2 discovering the object (Fig. 4.6(d)), neighboring robots converge toward the object after detecting the hidden signature from robot 2 (Fig. 4.6(e)), and the network continuing to the goal once the object "task" has been completed (Fig. 4.6(f)-(g)). During the simulation, communication broadcasts from robot $j = 4$ are corrupted with false position data that attempt to drive the system

to an undesirable location, but the CUSIGN detector finds these stealthy attacks, allowing the network to resiliently perform the operation. In Fig. 4.7, alarm rates that are monitoring the primary (3.40) and hidden (4.6) models throughout the experiment show robot 1 detecting the compromised robot 4, as well as detecting the hidden signature from robot 2.



(a) (b)

Figure 4.7: (a) CUSIGN and (b) sign switching alarm rates from the perspective of robot $i = 1$. Detection bounds are indicated by red dashed lines.

4.7 Discussion

In this chapter, we have presented a decentralized framework for a network of homogeneous robots to resiliently perform desired task-based operations. The robots are able to distinguish between received inconsistent information from neighboring robots due to man-in-the-middle attacks to communication broadcasts and hidden model behaviors that provide a detectable signature to implicitly pass safety-critical information. To detect stealthy attacks and the hidden signature, we leverage randomness-based detection techniques —Cumulative Sign (CUSIGN) and sign switching rate— to identify whether robots are following a primary or hidden network model. Future extensions to the framework in this chapter could include: i) investigating the effects of different attack classes/models and ii) develop an adaptive approach for the virtual spring parameters to conform to changing network or environmental conditions.

To summarize Part II, consisting of Chapters 3 and 4, we introduced attack detection methods to discovery stealthy cyber attacks within multi-robot systems to enable for resilient task-based operations. Upon detection of misbehaving robots, our frameworks allow the remaining robots to isolate the compromised robot then reconfigure the network topology to remove any malicious effects that the compromise robots may induce. However, if the compromised robots suffer from cyber attacks (or faults) to on-board sensors which compromise localization capabilities and control behavior, the robots are discarded from the network without safety implications or a recovery method

in mind. More specifically, these robots may continue navigating in an undesirable manner, thus potentially leading them to undesirable states within the environment that could damage/destroy themselves or critical infrastructure. In the next Part, named System Recovery, we explore recovery methods for both single- and multi-robot autonomous systems such that the vehicles have a recovery framework in place due to various undesirable conditions that impact localization and control performance on-board affected robots. In particular, we invite the reader to continue on to Chapter 6 where we introduce cooperative recovery frameworks that utilize much of the theory and applications discussed in Part II.

Part III

System Recovery

Chapter 5

Recovery of Autonomous Systems Operating under On-board Controller Failures

5.1 Introduction

Present-day autonomous robotic systems possess increased complexities to support an expanded array of computers and sensors to assist in advanced capabilities such as navigation, warehouse logistics, and industrial operations, towards truly unmanned operations. With such complexity, however, comes higher risks of malicious cyber attacks due to their unsupervised, autonomous applications and the numerous entry points to implement an attack. While the vast majority of the literature in robotics and cyber-physical systems security deal primarily with attacks on the sensing and communication infrastructure of a system [101], in this section we consider attacks that interfere with the control logic to hijack a system. For this class of attacks, controller parameter gains can be altered to trigger an undesirable behavior under certain states or tracking errors. For example, in Fig. 5.1 shows a motivational case in which a robot needs to turn to the right but ends up turning to the left, away from the desired trajectory and into an obstacle when the tracking error is within a compromised region.

One of the key principles that we leverage is that such robotic systems, in nominal conditions, i.e., when uncompromised, have well-designed dynamical models that enable accurate predictions of output measurements from their control input signals. A cyber attack on the on-board controller can cause inconsistencies from the expectation of this input-output model, leading to observable deviations from its nominal behavior. To this end, we consider a residual-based monitoring approach that leverages the chi-squared detection scheme to reduce the residual vector into a scalar test measure to detect controller integrity inconsistencies. Regions of the state space or the tracking error space that are deemed compromised are monitored for future operations to avoid them. A compensator to alter information provided to the controller (i.e., reference signal and state vector) is built to avoid any compromised regions within the state or tracking error spaces. Moreover,

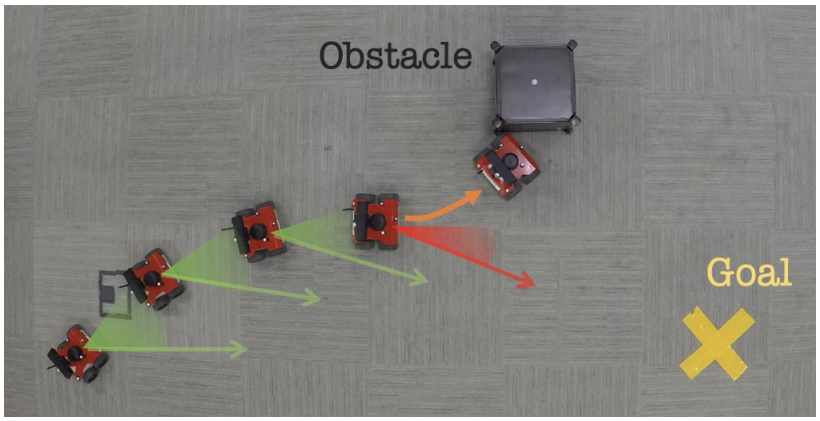


Figure 5.1: Pictorial representation of the problem investigated in this paper in which a robot is tasked to navigate toward a goal under a controller attack that gets activated only when the tracking error crosses a certain threshold (red region in the figure).

to deal with this problem of maintaining desirable control performance during operations, the altered information is designed to minimize the difference in the *compensated* control input signal in comparison to the originally intended (but compromised) control input.

The contribution within this chapter is twofold: 1) a detection framework to discover compromised regions of the state or tracking error space within a controller that cause anomalous system behavior and 2) a compensator that alters the reference signal and state information provided to the controller in order to bypass compromised regions to achieve desired control performance to resiliently continue operations.

5.2 Preliminaries

This chapter considers robotic systems modeled as discrete-time linear time-invariant (LTI) systems of the form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{\nu}_k, \tag{5.1}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{\eta}_k, \tag{5.2}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ denotes the state vector, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input, and $\boldsymbol{y}_k \in \mathbb{R}^s$ represents the output vector at every time instance $k \in \mathbb{N}$. The state, input, and output matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are of appropriate dimensions, while $\boldsymbol{\nu}_k \sim \mathcal{N}(0, \boldsymbol{Q}) \in \mathbb{R}^n$ and $\boldsymbol{\eta}_k \sim \mathcal{N}(0, \boldsymbol{R}) \in \mathbb{R}^{N_s}$ are i.i.d. Gaussian process and measurement uncertainties.

During operations, a Kalman Filter (KF) is implemented to provide a state estimate $\hat{\boldsymbol{x}}_k \in \mathbb{R}^n$ in the form:

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{A}\hat{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{L}(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k) \tag{5.3}$$

where $\boldsymbol{L} = \boldsymbol{P}\boldsymbol{C}^{\mathsf{T}}(\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{R})^{-1}$ is defined as the Kalman gain matrix which is solved by the algebraic Riccati equation.

5.2.1 Threat Model

We assume a general feedback controller with a nominal control input signal that is described as

$$\boldsymbol{u}_k = \boldsymbol{K}\left(\boldsymbol{x}_k^{\text{ref}} - \hat{\boldsymbol{x}}_k\right) = \boldsymbol{K}\boldsymbol{x}_k^e \tag{5.4}$$

where $\boldsymbol{x}_k^e = \boldsymbol{x}_k^{\text{ref}} - \hat{\boldsymbol{x}}_k$ is the tracking error between a ref-erence signal (i.e., desired state) and the state estimate, while $\boldsymbol{K}$ is a feedback gain to provide desired control performance of the system. Additionally, we assume the true control inputs to the system

$$-\boldsymbol{u}_{max} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max} \tag{5.5}$$

are constrained to due to actuation limitations.

We consider control inputs (5.4) that can be altered due to undesired (and unknown) changes in controller parameters and/or additive inputs, as depicted in Fig. 5.2. These changes occur as signals fed to the controller satisfy specific compromised ranges of tracking error $\widetilde{\mathcal{E}}$ and state $\widetilde{\mathcal{X}}$ within a finite tracking error space $\widetilde{\mathcal{E}} \subset \mathcal{E}$ and/or finite state space $\widetilde{\mathcal{X}} \subset \mathcal{X}$. Within these compromised regions, we consider scenarios such as: i) cyber attacks that are able to maliciously modify control parameters and/or introduce control signal biases at runtime or ii) faulty code that is defined before operations begin; resulting in undesirable control inputs provided to the system.

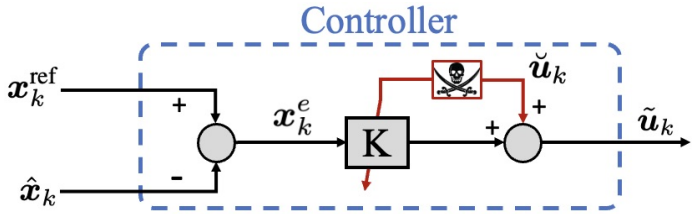


Figure 5.2: A diagram describing altered control parameters and additive inputs to result in undesired control behavior.

The altered control inputs $\widetilde{\boldsymbol{u}}_k \neq \boldsymbol{u}_k$ are presented as

$$\widetilde{\boldsymbol{u}}_k = \widetilde{\boldsymbol{K}}\boldsymbol{x}_k^e + \breve{\boldsymbol{u}}_k \in \mathbb{R}^m \tag{5.6}$$

with the feedback gain $\widetilde{\boldsymbol{K}} \neq \boldsymbol{K}$ and additive input signal $\breve{\boldsymbol{u}}_k \in \mathbb{R}^m$ when the following condition is satisfied:

$$\boldsymbol{x}_k^e \in \widetilde{\mathcal{E}}, \quad \hat{\boldsymbol{x}}_k \in \widetilde{\mathcal{X}}. \tag{5.7}$$

5.3 Problem Formulation

An attack or fault to an on-board controller will consequently result in anomalous behavior due to unreliable control signals $\widetilde{\boldsymbol{u}}_k$ being applied to the system. We focus on discovering specific operating conditions (i.e., regions and ranges) from the information (i.e., reference signal $\boldsymbol{x}_k^{\text{ref}}$ and state estimate $\hat{\boldsymbol{x}}_k$) provided to the controller that cause undesired behaviors.

Problem 5.1 (Anomalous Behavior Detection) *Given the nominal and altered control inputs represented in* (5.4) *and* (5.6)*, we want to detect compromised regions within the state $\mathcal{X}$ and error $\mathcal{E}$ spaces. Formally, the objective is to find conditions of the reference signal and state information*

$$\boldsymbol{u}_k = \begin{cases} \widetilde{\boldsymbol{K}}\boldsymbol{x}_k^e + \breve{\boldsymbol{u}}_k, & \text{if } \{\boldsymbol{x}_k^{\text{ref}}, \hat{\boldsymbol{x}}_k\} \in \widetilde{\mathcal{X}}, \widetilde{\mathcal{E}} \\ \boldsymbol{K}\boldsymbol{x}_k^e, & \text{otherwise} \end{cases} \tag{5.8}$$

that trigger undesired control inputs which are sent to the robot, hence resulting in undesired system behavior.

Upon detection of compromised regions of state $\widetilde{\mathcal{X}} \subset \mathcal{X}$ and tracking error $\widetilde{\mathcal{E}} \subset \mathcal{E}$ spaces, the robot aims to avoid triggering these undesired behaviors such that resilient operation can continue. Formally:

Problem 5.2 (System Recovery) *Design a policy such that the robot computes a compensated reference signal $\bar{\boldsymbol{x}}_k^{\text{ref}}$ and state $\bar{\boldsymbol{x}}_k$ information for the controller, where $\bar{\boldsymbol{x}}_k^{\text{ref}} \neq \boldsymbol{x}_k^{\text{ref}}$ and $\bar{\boldsymbol{x}}_k \neq \hat{\boldsymbol{x}}_k$, in order to avoid malicious regions within the state and error spaces to maintain desirable control performance. Furthermore, the compensated input $\bar{\boldsymbol{u}}_k$ which is computed using the compensated reference signal and state information, seeks to minimize the following:*

$$\bar{\boldsymbol{u}}_k = \left\{ \arg\min_{\bar{\boldsymbol{u}}_k} (\bar{\boldsymbol{u}}_k - \boldsymbol{u}_k^*) : \{\bar{\boldsymbol{x}}_k^{\text{ref}}, \bar{\boldsymbol{x}}_k\} \notin \widetilde{\mathcal{X}}, \widetilde{\mathcal{E}} \right\} \tag{5.9}$$

where $\boldsymbol{u}_k^*$ is the desired control input before compensation and $\arg\min(\bar{\boldsymbol{u}}_k - \boldsymbol{u}_k^*)$ *represents the objective to find a compensated input with minimum difference from the desired.*

5.4 Framework

In this section we describe the monitoring and recovery framework to detect anomalous controller behavior during specific ranges of information, then implement a recovery mechanism for an autonomous robot to provide uncompromised control inputs for motion. The overall control architecture is highlighted in Fig. 5.3 where a detector monitors the residual vector to determine

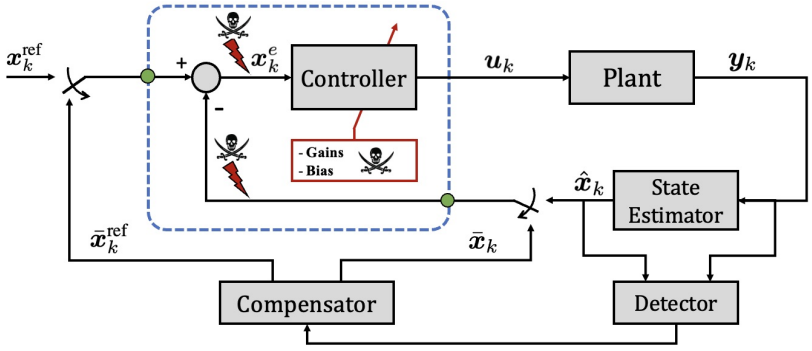


Figure 5.3: Overall detection and compensation architecture of our framework. The compensator manipulates the reference and state estimate vectors to compute a compensated input $\bar{\boldsymbol{u}}_k$ when $\hat{\boldsymbol{x}}_k \in \widetilde{\mathcal{X}}$ or $\hat{\boldsymbol{x}}_k^e \in \widetilde{\mathcal{E}}$ are satisfied.

if anomalous system behavior is occurring. This allows for compensation of information into the controller (i.e., reference signal and state estimate vector) to ensure uncompromised control inputs are sent to the robot. Our focus is on attacks or faults taking place on the on-board controller as specific ranges of input information fed to the controller are provided. Within these ranges, unknown controller parameter changes and/or additive control signals are included to the control input, causing undesired behavior of the robot.

5.4.1 Space Partitioning

In this chapter, we want to discover specific regions within the error and state spaces that may be compromised due to cyber attacks (or possibly faulty code) that alters the control inputs computed by the on-board controller. To monitor for compromised regions within tracking error space $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ or the state space $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ for an ith state, $i = 1, \ldots, n$, we first partition the spaces into a finite number of bins.

For generalization, lets define any given space by the set $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. An ith state in space $\mathcal{S}_i$ to be monitored is partitioned into N_b bins to check for inconsistent behavior within each bin (i.e., partitioned region). The set of $j = 1, \ldots, N_b$ partitioned bins in an ith state are represented as $\mathcal{B}_i = \{b_{i,1}, \ldots, b_{i,j}, \ldots, b_{i,N_b}\}$ that span the entire space. Each partitioned bin of arbitrary size is a subset of the set of bins (i.e., $b_{i,j} \subset \mathcal{B}_i$) within a space $\mathcal{S}_i$ and described by:

$$b_{i,j} = \begin{cases} \mathcal{S}_{i,\min} \leq b_{i,j} \leq \mathcal{S}_{i,(b_{i,j,\max})} & \text{if } j = 1 \\ \mathcal{S}_{i,(b_{i,j-1,\max})} < b_{i,j} \leq \mathcal{S}_{i,(b_{i,j,\max})} & \text{if } j = 2, \ldots, N_b - 1 \\ \mathcal{S}_{i,(b_{i,j-1,\max})} < b_{i,j} \leq \mathcal{S}_{i,\max} & \text{if } j = N_b \end{cases} \tag{5.10}$$

such that

$$\bigcup_{j=1}^{N_b} b_{i,j} = \mathcal{S}_i \quad \text{and} \quad \bigcap_{j=1}^{N_b} b_{i,j} = \emptyset \tag{5.11}$$

are satisfied. As an example, in Fig. 5.4 we show an ith state that is partitioned into the N_b bins (i.e., subspaces) of equal size that cover all possible values of its state space $\mathcal{X}_i$.

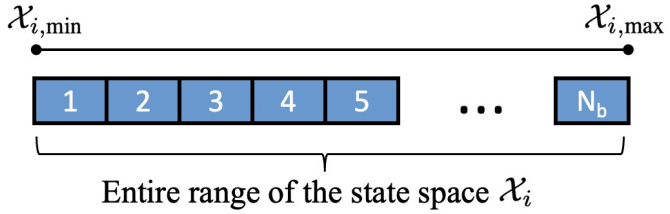


Figure 5.4: Depiction of the N_b partitioned bins spanning a state space $\mathcal{X}_i$.

The objective is to discover if specific regions within a space lead to faulty/anomalous behavior due to a compromised on-board controller providing unknown, malicious control signals to the system. For ease in the remainder of this paper, we describe the bin subspaces of a state i in general terms that may be utilized for either the tracking error or state space. At every time iteration, the system determines which bin j the information that is monitored belongs to (i.e., tracking error $x^e_{k,i} \in b_{i,j}$ or state $\hat{x}_{k,i} \in b_{i,j}$). In the next subsection, we describe how to monitor for anomalous behavior within each bin $b_{i,j}$ in an ith state or error space.

5.4.2 Residual-based State and Error Consistency Monitoring

During an operation, the robot checks for anomalous system behavior due to attacks or faults within the on-board controller. In particular, each error/state subspace (i.e., denoted by the partitioned bins described in Section 5.4.1) is monitored as the conditions of the reference and state estimate information sent to the controller are met. If specific subspaces during operations present anomalous behavior, they are flagged as compromised regions. We leverage a residual-based fault detection scheme to check for anomalous system behavior at runtime. The idea behind this scheme is to utilize the state predictions $\hat{\boldsymbol{x}}_{k+1}$ in (2.6) to determine if the system is responding to the computed control inputs $\boldsymbol{u}_k$ accordingly. To monitor for inconsistent system behavior, we compute the measurement residual vector

$$\boldsymbol{r}_k = \boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k \in \mathbb{R}^{N_s} \tag{5.12}$$

which is defined as the difference between sensor measurements and the state prediction that was computed at the previous time $k-1$. The measurement residual is modeled by a zero-mean Gaussian distribution $\boldsymbol{r}_k = \mathcal{N}(0, \boldsymbol{\Sigma})$ with an expected covariance matrix $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{r}_k \boldsymbol{r}_k^\mathsf{T}] = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^\mathsf{T} + \boldsymbol{R} \in \mathbb{R}^{N_s \times N_s}$, where $\boldsymbol{P}$ is the estimation error covariance matrix. A system that is behaving in a consistent manner

will display residuals that follow this expected distribution, while misbehaving systems violate this expectation. We utilize the well-known chi-square detection scheme by reducing the residual vector into a scalar test measure:

$$z_k = \boldsymbol{r}_k^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_k \tag{5.13}$$

which is chi-squared distributed that follows $z_k \sim \chi^2(N_s)$.

To determine if undesired control inputs $\widetilde{\boldsymbol{u}}_k$ are being computed by the compromised controller in a specific bin $b_{i,j} \subset \mathcal{B}_i$, we monitor for expected behavior of the test measure. Similar to the monitoring method in Chapter 4, we monitor for unexpected sign switching rates to detect inconsistent behavior. The sign of each incoming test measure (5.13) with respect to a user-defined reference value $z^{\text{ref}} \in \mathbb{R}_{>0}$ is computed at every time k. Moreover, the signed test measure value $\text{sgn}(z_k - z^{\text{ref}})$ is compared with the sign of the previous signed test measure value when the system belonged to the same jth bin in state i, $b_{i,j} \subset \mathcal{B}_i$, at time a $k - T_{b_{i,j}}$. If the test measure comparison is of opposite signs, then an alarm is triggered, otherwise an alarm is not triggered. Formally, the procedure to trigger an alarm follows:

$$\zeta_k = \begin{cases} 1, & \text{if } \text{sgn}(z_k - z^{\text{ref}}) = -\text{sgn}(z_{k-T_{b_{i,j}}} - z^{\text{ref}}) \\ 0, & \text{otherwise} \end{cases} \tag{5.14}$$

where $T_{b_{i,j}} \in \mathbb{N}$ denotes the number of time steps since the jth bin in the space $\mathcal{B}_i$ was entered. The alarm $\zeta_k \in \{0, 1\}$ signifies that the test measure at time k is of the opposite sign (i.e., a sign switch) with respect to the previous test measure within the same bin at time $k - T_{b_{i,j}}$ to trigger an alarm $\zeta_k = 1$, otherwise $\zeta_k = 0$. The alarm ζ_k is placed into a runtime alarm rate estimator:

$$\hat{A}_{k|b_{i,j'}} = \begin{cases} \hat{A}_{k-1|b_{i,j'}} + \frac{\zeta_k - \hat{A}_{k-1|b_{i,j'}}}{\ell} & \text{if } j' = j \\ \hat{A}_{k-1|b_{i,j'}} & \text{if } j' \neq j \end{cases} \tag{5.15}$$

to compute an updated estimate for the jth bin within the set $\mathcal{B}_i$, where ℓ is a "pseudo-window" length. All other alarm rate estimates corresponding to a bin $j' \neq j$ are carried over from the previous time step, since they are unaffected as the system did not belong to the space pertaining to the j'th bin. All alarm rate estimates $\forall j \in b_{i,j}$ are initialized to $\hat{A}_{0|b_{i,j}} = \mathbb{E}[A]$ at time $k = 0$, and alarm rate estimates should follow

$$\hat{A}_k = \mathbb{E}[A] = \Pr[\zeta_k = 1] \tag{5.16}$$

where $\mathbb{E}[A]$ is the expected rate which alarms are triggered. Bounds on the estimated alarm rates, denoted by $[\Omega_-, \Omega_+] \in (0, 1)$ that satisfies $\Omega_- < \mathbb{E}[A] < \Omega_+$, can be computed to signify anomalous

system behavior. To summarize, when the alarm (i.e., sign switching) rate for the detection of anomalous controller behavior satisfies,

$$\hat{A}_{k|b_{i,j}} \notin [\Omega_-, \Omega_+] \longrightarrow \textit{Anomalous Behavior} \tag{5.17}$$

the robot detects anomalous behavior within bin j on an ith state. The robot then places the bin j presenting the anomalous behavior into a compromised bin set $b_{i,j} \rightarrow \widetilde{\mathcal{B}}_i$, where $\widetilde{\mathcal{B}}_i \subset \mathcal{B}_i$, to allow for compensation to avoid this region that results in undesired control performance.

5.4.3 State and Reference Compensation

The goal for the compensator is to provide compensated values of the reference signal $\bar{\boldsymbol{x}}_k^{\text{ref}}$ and state $\bar{\boldsymbol{x}}_k$ to the controller in order to avoid any compromised regions within the error $\widetilde{\mathcal{E}}$ and/or state $\widetilde{\mathcal{X}}$ space. We define the control input

$$\bar{\boldsymbol{u}}_k = \boldsymbol{K}\big(\bar{\boldsymbol{x}}_k^{\text{ref}} - \bar{\boldsymbol{x}}_k\big) = \boldsymbol{K}\bar{\boldsymbol{x}}_k^e \tag{5.18}$$

as the computed input that utilizes the compensated reference and state vectors. Additionally, when providing compensated information $\{\bar{x}_{k,i}^{\text{ref}}, \bar{x}_{k,i}\}$ on an ith state to the controller, our objective is to update the information in a manner to minimize the difference in compensated control input signal from the nominal (i.e., desired) input

$$\bar{\boldsymbol{u}}_k = \arg\min(\bar{\boldsymbol{u}}_k - \boldsymbol{u}_k^*) = \arg\min\big(\boldsymbol{K}(\bar{\boldsymbol{x}}_k^e - \boldsymbol{x}_k^e)\big) \tag{5.19}$$

where $\bar{\boldsymbol{x}}_k^e = \bar{\boldsymbol{x}}_k^{\text{ref}} - \bar{\boldsymbol{x}}_k$ and $\boldsymbol{x}_k^e = \boldsymbol{x}_k^{\text{ref}} - \hat{\boldsymbol{x}}_k$.

In Fig. 5.5 we show a high-level view of the compensation approach to satisfy (5.19). The compensator determines whether the incoming information (i.e., the reference $x_{k,i}^{\text{ref}}$ and state estimate $\hat{x}_{k,i}$) belong to any compromised bins $\widetilde{\mathcal{B}}_i \subset \mathcal{B}_i$ (highlighted by the red bins in Fig. 5.5) within an ith state/error space. If the information belongs to a compromised bin, then the compensator chooses the nearest uncompromised bin (colored in orange) from the current state/error, which is leveraged to compute a compensated reference signal $\bar{x}_{k,i}^{\text{ref}}$ and state estimate $\bar{x}_{k,i}$, respectively. Next, we provide the compensation procedure for the cases when an attack to the controller affects regions within the state $\widetilde{\mathcal{X}}_i \subset \mathcal{X}_i$ and tracking error $\widetilde{\mathcal{E}}_i \subset \mathcal{E}_i$ spaces.

Compromised State Space

In this subsection, we describe compensation that occurs as one or more bins within the state space $\mathcal{X}_i$ of any ith state are deemed compromised. The compensation occurs only when the ith state

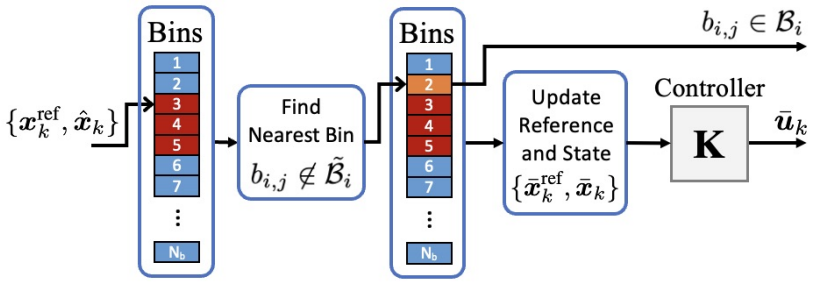


Figure 5.5: The compensation process to provide altered references and states to the controller to avoid compromised regions (red) within an ith state.

estimate element is within a compromised jth bin $\hat{x}_{k,i} \in b_{i,j}$ where $b_{i,j} \subset \widetilde{\mathcal{B}}_i$. The objective is to find a compensated state

$$\bar{x}_{k,i} = \{\bar{x}_{k,i} \in b_{i,j} : \min(\hat{x}_{k,i} - \bar{x}_{k,i}), b_{i,j} \in \mathcal{B}_i \setminus \widetilde{\mathcal{B}}_i\} \tag{5.20}$$

that is provided to the controller which belongs to an uncompromised bin $b_{i,j} \subset \mathcal{B}_i \setminus \widetilde{\mathcal{B}}_i$. To maintain desired reference tracking performance, we also compensate the reference signal $\bar{x}_{k,i}^{\text{ref}}$ by the same amount as the state compensation

$$\bar{x}_{k,i}^{\text{ref}} = x_{k,i}^{\text{ref}} + \Delta x_{k,i} \tag{5.21}$$

such that the tracking error remains unchanged, where $\Delta x_{k,i} = (\bar{x}_{k,i} - \hat{x}_{k,i})$ is the change in state.

Lemma 5.1 *Given the compensated state estimation and reference signal that is provided to the controller to avoid compromised regions in an ith state space $\bar{x}_{k,i} \notin \widetilde{\mathcal{X}}_i$, the computed control input $\bar{\boldsymbol{u}}_k$ is equal to the desired input $\boldsymbol{u}_k^*$.*

Proof: We observe that the difference in tracking error

$$(\bar{x}_{k,i}^{\text{ref}} - \bar{x}_{k,i}) - (x_{k,i}^{\text{ref}} - \hat{x}_{k,i}) = \bar{x}_{k,i}^e - x_{k,i}^e = 0 \tag{5.22}$$

between the compensated and uncompensated information of an ith state that is provided to the controller is zero. The resulting tracking error vector $\bar{\boldsymbol{x}}_k^e = \boldsymbol{x}_k^e$ remains unchanged, thus the control input signal to the system is the same (i.e., $\boldsymbol{K}\bar{\boldsymbol{x}}_k^e = \boldsymbol{K}\boldsymbol{x}_k^e \rightarrow \bar{\boldsymbol{u}}_k = \boldsymbol{u}_k^*$). ■

Compromised Error Space

Next, we describe the scenario where compensation occurs to avoid regions that are within the compromised tracking error space which result in undesired control performance of the system. Similar to the previous subsection for anomalies in the state space, we characterize the compensation effort that occurs as anomalous behaviors are detected within any jth bin of the error space

$x^e_{k,i} = x^{\text{ref}}_{k,i} - \hat{x}_{k,i} \in b_{i,j}$, where $b_{i,j} \subseteq \widetilde{\mathcal{B}}_i$. However, attackers may reduce the usable tracking errors to a much smaller subset of the original error space $\mathcal{E}_i$.

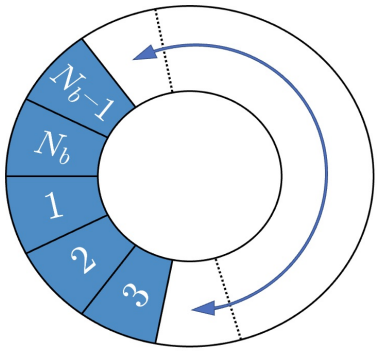


Figure 5.6: Viewing bins within an ith error space as a loop.

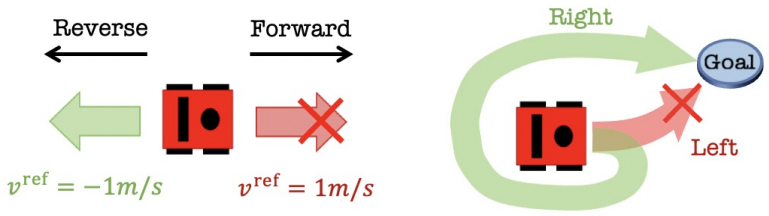
Depending on feasible control inputs (5.5) for a given system (2.1), we can describe the tracking error space for these suitable states as a *loop*, as depicted in Fig. 5.6. In particular, reference signals for these states can be switched to the opposite sign (i.e., reverse direction) to enable the robot to reach specific positions within the environment. Furthermore, a user-defined *buffer* region between bins 1 and N_b may be included to any suitable ith error element whose state can leverage the *loop* method to control when a switching (i.e., reversal) of reference signals is applied. In Fig. 5.7, we provide two examples of this method for both the velocity and heading angle of a robot. The examples depict: 1) a robot navigating with a negative velocity (i.e., velocity reference signal is of the opposite sign) and 2) a robot turning in the opposite direction by creating a loop (i.e., heading angle reference is shifted by $\pm 2\pi$). These scenarios can be exploited such that a robot can reach desired positions within the environment when certain control input conditions cannot be attained due to compromised regions within the error space. If the current tracking error satisfies $x^e_{k,i} \in b_{i,j} \subset \widetilde{\mathcal{B}}_i$ and the nearest tracking error $\breve{x}^e_{k,i} \neq x^e_{k,i}$ within an uncompromised bin

$$\breve{x}^e_{k,i} = \arg\min(x^e_{k,i} - \breve{x}^e_{k,i}), \quad \breve{x}^e_{k,i} \in b_{i,j} \subset \mathcal{B}_i \setminus \widetilde{\mathcal{B}}_i \tag{5.23}$$

crosses over the buffer region (i.e., $1 \to N_b$ or $N_b \to 1$), we update the ith reference signal to the opposite direction with the function $f : \mathbb{R} \mapsto \mathbb{R}$ defined by

$$\bar{x}^{\text{ref}}_{k,i} = f(x^{\text{ref}}_{k,i}) \implies x^e_{k,i} = \bar{x}^{\text{ref}}_{k,i} - \hat{x}_{k,i} \tag{5.24}$$

and then the tracking error $x^e_{k,i}$ is updated accordingly.



(a) (b)

Figure 5.7: Examples of system states such as (a) velocity and (b) heading angle that are capable of providing an "opposite" reference signal.

From the given tracking error $x^e_{k,i}$, the objective is to find the nearest tracking error to provide to the controller

$$\bar{x}^e_{k,i} = \{\bar{x}^e_{k,i} \in b_{i,j} : \arg\min_{\bar{x}^e_{k,i}}(x^e_{k,i} - \bar{x}^e_{k,i}),\, b_{i,j} \subset \mathcal{B}_i \setminus \widetilde{\mathcal{B}}_i\} \tag{5.25}$$

which belongs to an uncompromised bin $b_{i,j} \subset \mathcal{B}_i \setminus \widetilde{\mathcal{B}}_i$ such that $\bar{x}^e_{k,i} \notin \widetilde{\mathcal{E}}_i$. We compensate the reference signal $\bar{x}^{\text{ref}}_{k,i}$ by

$$\bar{x}^{\text{ref}}_{k,i} = \hat{x}_{k,i} + \bar{x}^e_{k,i} \tag{5.26}$$

to achieve the compensated tracking error in (5.25).

After compensation for an ith reference signal has occurred, the compensated control input is no longer equal to the desired control input $\bar{\boldsymbol{u}}_k \neq \boldsymbol{u}^*_k$ as $\boldsymbol{K}\bar{\boldsymbol{x}}^e_k \neq \boldsymbol{K}\boldsymbol{x}^e_k$. The goal is to find a feasible solution to update reference signals for any state $i' \neq i$, $i' = \{1, \ldots, n\} \setminus i$ to minimize the difference between the compensated control input $\bar{\boldsymbol{u}}_k$ with compensated reference signals and the desired control input $\boldsymbol{u}^*_k$. To minimize the difference in control input to the system, the following objective function is solved

$$J(\bar{\boldsymbol{x}}^e_k) = \arg\min_{\check{\boldsymbol{x}}^e_k}\big(\|\check{\boldsymbol{K}}\check{\boldsymbol{x}}^e_k - \boldsymbol{u}^*_k + \boldsymbol{K}_i\bar{x}^e_{k,i}\|\big) \tag{5.27}$$

where $\check{\boldsymbol{K}}$ is the feedback control matrix with the ith column removed, $\check{\boldsymbol{x}}^e_k$ is the tracking error vector with the ith element removed, and $\boldsymbol{K}_i$ is the ith column in the feedback matrix $\boldsymbol{K}$. To satisfy the compensated tracking error vector in (5.27) that minimizes the change in the control input, we compensate the reference signals elements i' for any altered tracking error

$$\bar{x}^{\text{ref}}_{k,i'} = x^{\text{ref}}_{k,i'} + (\bar{x}^e_{k,i'} - x^e_{k,i'}), \quad \forall i' \neq i. \tag{5.28}$$

The following Lemma provides details to show that the true system state converges to the desired reference signal, even as the reference signal is being compensated.

Lemma 5.2 (System Stability) *Given the compensated reference signals in* (5.26) *and* (5.28) *to minimize* (5.27) *in order to avoid any compromised regions within the tracking error space* $\bar{\boldsymbol{x}}_k^e \notin \widetilde{\mathcal{E}}$ *with control input* $\bar{\boldsymbol{u}}_k = \boldsymbol{K}\bar{\boldsymbol{x}}_k^e$*, the reference tracking closed-loop system is asymptotically stable.*

Proof: We omit the full proof due to page limitations. However, Lemma 5.2 can be proved via Lyapunov stability to show that reference tracking is globally asymptotically stable such that the system state converges toward any desired bounded reference signal $\boldsymbol{x}_k^{\text{ref}}$ during attacks/faults to the tracking error space (i.e., $\widetilde{\mathcal{E}} \neq \emptyset$). In other words, the expectation of the true tracking error $\boldsymbol{x}_k^t = \boldsymbol{x}_k^{\text{ref}} - \boldsymbol{x}_k$ is always converging (i.e., $\mathbb{E}[\boldsymbol{x}_{k+1}^t - \boldsymbol{x}_k^t] \to 0$ as $k \to \infty$) for any compensated reference signal $\bar{\boldsymbol{x}}_k^{\text{ref}}$ and compensated state $\bar{\boldsymbol{x}}_k$ computed in (5.24)–(5.28). ■

It is noted that when the compensator is providing compensated signals to the controller $\{\bar{\boldsymbol{x}}_k^{\text{ref}}, \bar{\boldsymbol{x}}_k\} \to \bar{\boldsymbol{u}}_k$, the resulting computed compensated control input $\bar{\boldsymbol{u}}_k$ is utilized in the state estimation process in (5.3).

5.5 Simulation Results

For the simulation case studies, we consider a differential drive UGV with the following general dynamical model (2.49). The system is linearized and it is modeled with a sensor sampling rate $t_s = 0.05$s to satisfy the system model in (5.1) and (5.2).

For all simulations, the robot is tasked to visit a series of waypoints (i.e., goals) within an obstacle-filled environment while maintaining a velocity $v^{\text{ref}} = 0.15$m/s. We present two case studies, one each in the state and error spaces where attacks occur when velocity information provided to the controller is within a compromised region $\hat{v}_k \in \widetilde{\mathcal{X}}$ or $\hat{v}_k^e \in \widetilde{\mathcal{E}}$. Three simulations are provided for each case study that highlight the *Nominal* (i.e., no attack), *Uncorrected*, and *Corrected* scenarios where attacks start at $k = 2700$ and the robot begins from the same initial state $\boldsymbol{x}_0 = [0, 1.5, 0, 0, 0]^\intercal$.

State Space Attack

The case study for attacks within the state space is presented in Fig. 5.8. As the robot velocity estimate provided to the controller satisfies $\hat{v} = [0.12, 0.17] \in \widetilde{\mathcal{X}}_i$, then control feedback gains for velocity are reduced by 80% and the gains for turning are multiplied by -1. In the *Uncorrected* scenario, the system deviates from its intended trajectory as it tries to maintain its desired velocity within a compromised region in the state space. Moreover, the robot collides with an obstacle, although obstacle avoidance is implemented. Instead, in the *Corrected* scenario with our framework implemented, the robot discovers the bins $b_{i,j} \subset \widetilde{\mathcal{B}}_i$ where anomalous behavior is occurring (red region in Fig. 5.8(c)) within the velocity state space, which allows the robot to recover and resiliently maintain its operation shown in Figs. 5.8(a) and 5.8(d).

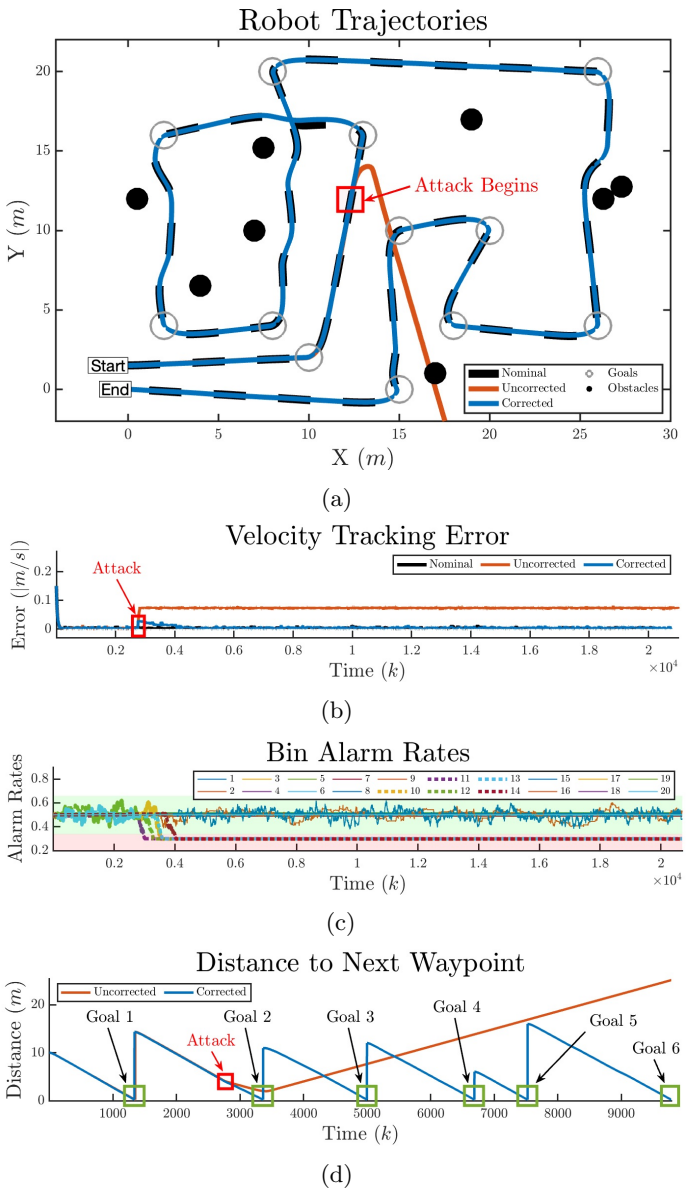


Figure 5.8: A waypoint follower during the compromised state space case study showing: (a) the resulting robot trajectories, (b) velocity tracking error, (c) alarm rates for the $N_b = 20$ bins, and (d) distance to the next goal/waypoint.

Error Space Attack

In Fig. 5.9 we highlight a case study where attacks occur within the error space $\mathcal{E}$. Attacks occur as the velocity tracking error is any positive value, causing feedback gains for velocity to be reduced by 90% and an input bias of $\breve{\boldsymbol{u}}_k = -0.08$. When not monitoring for malicious behavior in the *Uncorrected* scenario, the robot continues to attempt a forward (positive) velocity and is driven away from its next intended goal point. In the *Corrected* scenario, the robot discovers anomalous

behavior in bins $b_{i,j} \subset \widetilde{\mathcal{B}}_i$ corresponding to positive velocity tracking errors (Fig. 5.9(c)). This allows the robot to recover by updating its reference velocity to the opposite direction to navigate the environment in reverse to resiliently maintain the operation.

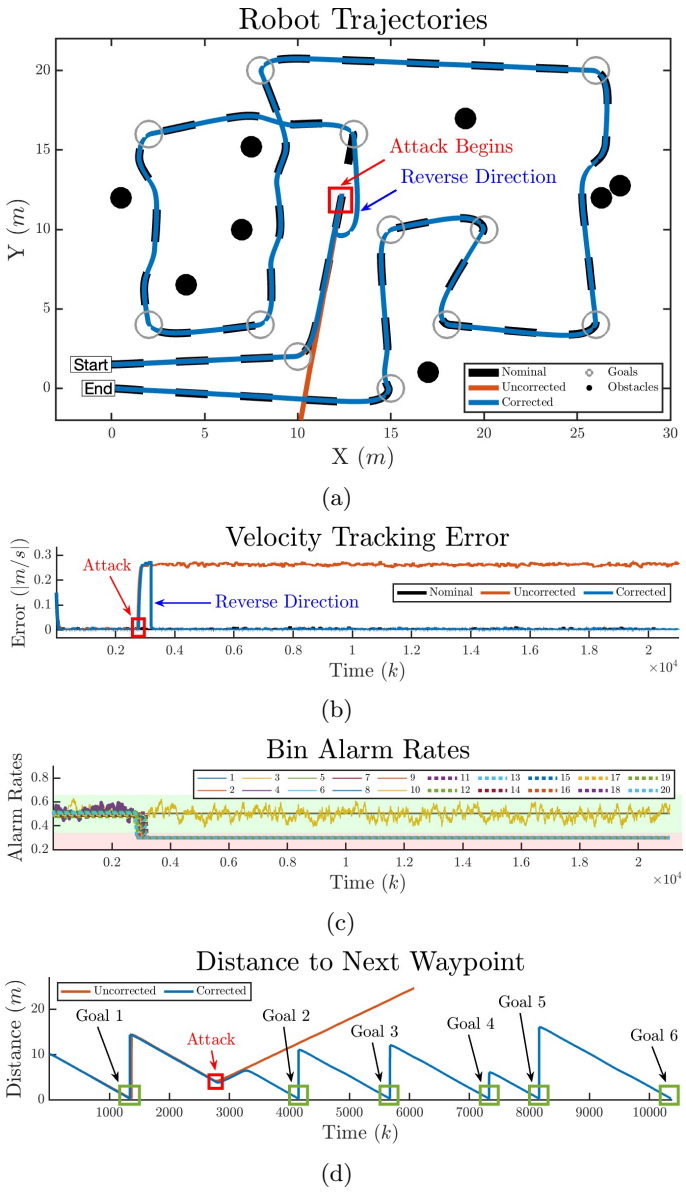
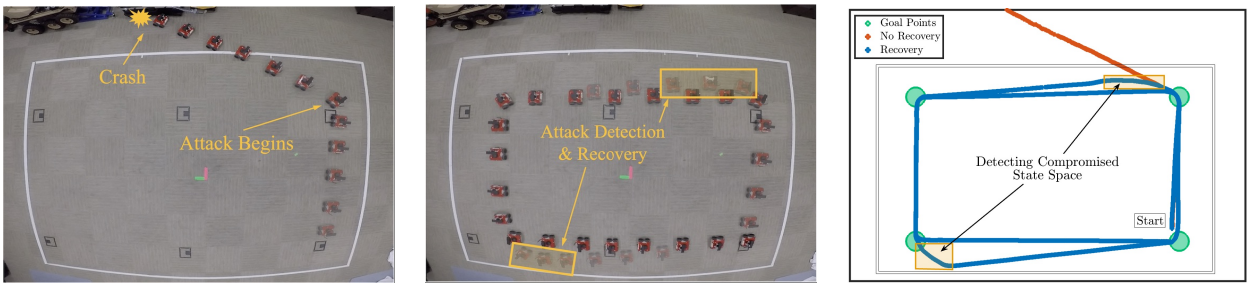


Figure 5.9: The compromised tracking error space case study displaying: (a) the resulting robot trajectories, (b) velocity tracking error, (c) alarm rates for the $N_b = 20$ bins, and (d) distance to the next goal/waypoint.

5.6 Experiment Results

Experimental validations are implemented on Husarion Rosbot 2.0 robots performing a go-to-goal operation within a lab environment. We show two case studies where attacks are triggered based on information from the heading angle this time; the first with an attack scenario within the state space and the second within the error space. For each experiment case study, we provide results where the robot is: a) unprotected from attacks/faults, b) protected from attacks/faults while leveraging our compensation framework for recovery, and c) a MATLAB representation of the robot positions during both the unprotected and protected experiments.

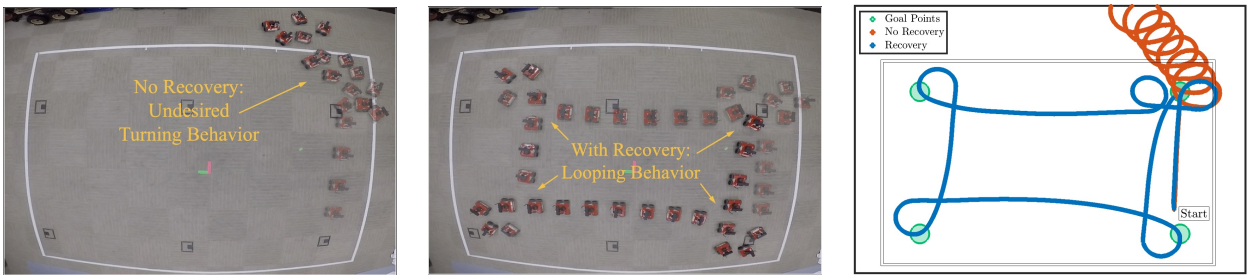
Snapshots of the first case study experiment are presented in Fig. 5.10, which captures the robot navigating to a series of goals. The malicious threat to the controller occurs when the robot heading angle (in degrees) satisfies $\theta \in [140, 245]$ or $\theta \in [-75, 30]$. Without our detection and recovery being performed, shown in Fig. 5.10(a), the control signal to the system is compromised and eventually the robot crashes into a wall. We see in Fig. 5.10(b) where our framework is leveraged, the robot compensates the information to the controller upon discovering anomalous regions within the state space to allow for continued navigation to each of the goal points.



(a) (b) (c)

Figure 5.10: An experiment showing a robot detecting anomalous behavior due to an attack within the state space of the heading angle estimate $\hat{\theta}$, then compensating information provided to the controller to avoid any compromised states $\hat{\theta} \in \mathcal{X}_i \setminus \widetilde{\mathcal{X}}_i$.

Our second case study demonstrates the robot that is subject to attacks in the heading angle error space, as shown in Fig. 5.11. The attack occurs at any instance the robot desires to turn left (i.e., negative tracking error), but instead the attack causes a turning action to the right. When our framework is not implemented (Fig. 5.11(a)), the robot fails to complete the operation due to continued circling motion. As shown in Fig. 5.11(b), since the robot is not able to turn left, the compensator alters the reference signal such that the robot performs a "looping" action by always turning right (i.e., positive tracking error) as this is the only possible action that the robot can do to avoid the compromised tracking error space to continue the operation.



(a) (b) (c)

Figure 5.11: A robot resiliently navigates to a series of waypoints while control parameters corresponding to angular velocity are altered during an attack within the error space. The attacks occur as the robot's tracking error for heading angle is negative (i.e., desired turn to the left).

5.7 Discussion

In this chapter, we have introduced a detection and recovery framework for autonomous mobile robots to maintain uncompromised control actions to resiliently perform a desired operation. The robot is able to identify cyber attacks or faults to its on-board controller within specific regions of the state and tracking error spaces by leveraging a residual-based attack detection scheme within the partitioned spaces. Furthermore, the robot uses a compensator to alter reference signal and state estimation vector information which is fed into the controller to maintain desired performance while avoiding any compromised regions in the state/tracking-error spaces. Future extensions to this chapter include extending the current framework to trigger replanning operations (e.g., changes in mission goals and trajectories) in the event that control recovery is not possible. Further implementation on different vehicles such as aerial robots and multi-robot systems are also potential areas that could exploit desirable recovery behaviors. In the next chapter, we investigate recovery frameworks within multi-robot systems when agents experience cyber attacks or faults to on-board positioning sensors.

Chapter 6

Multi-robot System Cooperative Recovery from Loss of Localization in Unknown Environments

6.1 Introduction

One of the essential capabilities for a mobile autonomous system is to localize itself within an environment. The ability to perform an accurate and robust localization allows unmanned systems to achieve truly autonomous operations. These autonomous operations can be accomplished in various ways, by relying on positioning sensors like global positioning systems (GPS), odometry, and IMU or through range sensors such as LiDAR, infrared (IR), and camera systems. The sensing information can then be leveraged via implementation of localization methods such as Particle filters and Simultaneous Localization and Mapping (SLAM) techniques.

When considering multi-agent system (MAS) applications, for example robotic swarms, consensus algorithms are typically considered where agents share their states to attain coordinated behaviors in a decentralized fashion to accomplish a desired goal [86]. When information being exchanged is incorrect, the entire MAS can be hijacked and lead to unsafe conditions [93]. A variety of issues can cause undesirable information to be exchanged between vehicles, such as cyber attacks or faults to on-board sensors or malicious man-in-the-middle attacks to communication broadcasts. If known landmarks/obstacles are present in the operating space, range sensors can be utilized for localization or to determine if the system is performing as expected. However, if agents within a MAS are navigating in open spaces (e.g., in the middle of the ocean), landmarks may not be available, thus leaving compromised agents unable to reliably localize themselves.

Localization sensing is one of the most critical information required to achieve intelligent and autonomous capabilities in unmanned systems like autonomous mobile robots. Global Positioning Systems (GPS) are the most common sources for outdoor positioning, but there exists other sensing methods to localize, such as: proximity sensors, radio frequency (RF), external cameras, and LiDAR [21]. However, security-related threats and faults to positioning sensors can compromise system

operations, with examples demonstrating catastrophic consequences that include GPS spoofing to divert vessels off course [8] and GPS interference on unmanned aerial vehicles that cause undesirable control behavior leading to crashes [36].

The issue of undesirable on-board positioning sensing is exacerbated within multi-robot systems (MRSs). When left unchecked, compromised robots can negatively impact the entire system's control performance by hijacking the MRS motion to unsafe regions. Resilient measures have been incorporated to MRSs to ensure continued safe operations in the presence of uncooperative robots within a swarm [93, 89]. To accomplish this, uncompromised robots typically "ignore" any misbehaving agents to remove undesirable effects that could incur to the MRS. In the case of proximity-based formations of robots, misbehaving actors are isolated from the remaining robots (as implemented in Chapters 3 and 4); thus robots experiencing cyber attacks or faults are essentially discarded without a recovery method implemented. In turn, discarded agents can potentially enter undesirable/restricted regions within the environment.

In this chapter, we introduce two frameworks for multi-robot systems to cooperatively recover (i.e., re-localize) compromised robots that experience cyber attacks or faults to on-board positioning sensors. Our first framework leverages stochastic system information to hide randomness-based information to alert neighboring robots of the impending attack/fault, thus protecting information from interception by potentially malicious attackers. When a robot detects that its on-board positioning sensor has been compromised, it broadcasts the signal which is overlaid within state information to create a hidden signature that alerts neighboring robots of its unreliable sensor. Furthermore, after detection of its unreliable on-board positioning sensor, compromised robots perform checkpointing for state reconstruction and compute reachable sets to ensure safety while continuing to navigate in the environment. The detection of the hidden signature triggers the nearby uncompromised robots to perform a cooperative motion behavior specifically to come within sensing/visual range, i.e., leveraging robots as mobile landmarks to aid in re-localizing the compromised robot. Our decentralized framework is designed to be robust within various environments that may or may not include known landmarks that could be used for localization while also considering that robots operate beyond distance/visual sensing range from each other. The first framework in this chapter builds upon frameworks in previous chapters where we leveraged randomness-based detection techniques to discover anomalous behaving agents subject to stealthy cyber attacks in MRSs (in Chapter 3) and to infer safety-critical information via hidden signatures within broadcast data exchanges (in Chapter 4).

In our second framework, which differs from our first framework, agents are tasked to maintain desired formations while remaining resilient to faulty on-board positioning sensors. Similarly though, the MAS operates within open or unknown environments that do not offer identifiable landmarks

and also operate beyond range sensing of nearby agents for use in localization when nominal on-board positioning sensors are unreliable. To deal with this, the proposed framework enables compromised agents to leverage Received Signal Strength Indication (RSSI) and received position information from neighboring agents for localization (i.e., mobile landmarks). Multilateration is performed using the noisy RSSI measurements and received neighboring agent's positions to provide position measurements in replacement of the unreliable on-board position sensors. To minimize the RSSI-based position measurement error, a weighted least squares method is used that leverages the commonly utilized log-normal shadowing path loss model [33]. Moreover, the RSSI-based position measurements have unknown covariances which differ from the nominal on-board position sensor. To improve state estimation performance, compromised agents leverage an adaptive Kalman Filter to estimate the position measurement covariance matrix at runtime. The proposed framework is introduced in a generalized manner that may be used on any formation control technique for swarms of homogeneous LTI modeled agents. As a specific case study in this paper, a virtual spring-damper physics model [5] for proximity-based formation control is considered on MASs in a 2-dimensional coordinate frame. However, our MAS framework can be expanded to heterogeneous systems [112], non-linear modeled agents (i.e., by using Extended KF) [59], and higher dimensional coordinate frames [109].

6.2 Preliminaries

This section introduces the multi-robot system dynamical model, communication and threat models, and assumptions used throughout this chapter.

6.2.1 Multi-robot System Model

Let us consider a multi-robot system of N homogeneous robots modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We denote $\mathcal{V} = \{1, \ldots, N\}$ as the robot set and the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, where an edge $(i, j) \in \mathcal{E}$ indicates a connection for control from robot $i \in \mathcal{V}$ to robot $j \in \mathcal{V}$. We consider each robot i can be represented in an LTI state space form:

$$\begin{aligned} \boldsymbol{x}_i^{(k+1)} &= \boldsymbol{A}\boldsymbol{x}_i^{(k)} + \boldsymbol{B}\boldsymbol{u}_i^{(k)} + \boldsymbol{\nu}_i^{(k)}, \\ \boldsymbol{y}_i^{(k)} &= \boldsymbol{C}\boldsymbol{x}_i^{(k)} + \boldsymbol{\eta}_i^{(k)}, \end{aligned} \tag{6.1}$$

with state $\boldsymbol{A}$ and input $\boldsymbol{B}$ matrices, state vector $\boldsymbol{x}_i^{(k)} \in \mathbb{R}^n$, and $\boldsymbol{\nu}_i^{(k)} \in \mathbb{R}^n$ denoting zero-mean Gaussian process noise. All N robots rely on noisy sensors represented by the output vector $\boldsymbol{y}_i^{(k)} \in \mathbb{R}^{N_s}$ with $\boldsymbol{C}$ denoting the output matrix and $\boldsymbol{\eta}_i^{(k)} \in \mathbb{R}^{N_s}$ representing zero-mean Gaussian measurement noise at discrete time iterations $k \in \mathbb{N}$. Each robot i utilizes a Kalman Filter to

provide a state estimate $\hat{\boldsymbol{x}}_i^{(k)} \in \mathbb{R}^n$ and a range sensor providing 360° field of view with a limited range $\delta_r > 0$ for collision avoidance.

In an effort to cooperatively maintain a desired proximity-based formation, the robots within the MRS exchange state information with each other. Each robot $i \in \mathcal{V}$ obeys a control consensus to achieve the desired proximities by:

$$\boldsymbol{u}_i^{(k)} = \mathcal{U}\big(\hat{\boldsymbol{x}}_i^{(k)}, \hat{\boldsymbol{x}}_j^{(k)}\big), \quad i \neq j \tag{6.2}$$

such that $j \in \mathcal{S}_i$, where $\mathcal{S}_i \subset \mathcal{V}$ is a neighbor set utilized for control. Given the neighbor set $\mathcal{S}_i$ for each ith robot, we represent the edge set by $\mathcal{E} = \big\{(i,j) \;\big|\; j \in \mathcal{S}_i, \forall i \in \mathcal{V}\big\}$ that is defined within our graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the robot set $\mathcal{V}$.

6.2.2 Threat Model

We assume the multi-robot system is navigating within an adversarial environment, such that individual robots may be subject to malicious cyber attacks or faults to position sensors. On an unprotected proximity-based formation of robots, a single compromised robot can affect the entire system of N robots as the effects of the attack are propagated throughout the system. During a sensor attack, we assume that an attacker can continuously intercept and modify position measurements (i.e., GPS data) with false, yet plausible information in an attempt to intentionally hijack the multi-robot system. Furthermore, position sensors may experience various faults, such as: drift, scaling, and hard faults [3].

For simplicity, we characterize both attacks and faults to received sensor measurements on-board a robot i as

$$\tilde{\boldsymbol{y}}_i^{(k)} = \boldsymbol{y}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{6.3}$$

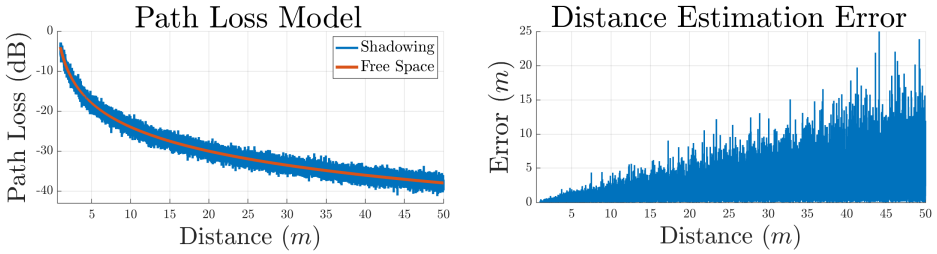
where $\boldsymbol{\xi}_i^{(k)}$ denotes the additive vector to localization measurements and $\tilde{\boldsymbol{y}}_i^{(k)}$ is the compromised measurement vector.

6.2.3 Communication Model

To overcome malicious cyber attacks or faults to position sensors, in this work, agents measure RSSI from the received communications of nearby agents. A commonly-used path loss model is the log-normal shadowing model [33], that is defined by:

$$P_{ij,[rx]}^{(k)} = P_{[tx]} - PL(d_0) - 10\beta \log \frac{d_{ij}^{(k)}}{d_0} + \Lambda \tag{6.4}$$

where $P^{(k)}_{ij,[rx]}$ is the measured received power by an agent i of an agent j, $PL(d_0)$ is the power loss (in dB) from a reference distance d_0, and $d^{(k)}_{ij} = \|\boldsymbol{p}^{(k)}_i - \boldsymbol{p}^{(k)}_j\|$ denotes the true distance between agents i and j. The channel shadowing $\Lambda \sim \mathcal{N}(0, \sigma^2_\Lambda)$ is modeled as a zero-mean Gaussian noise and β is the path loss exponent. It is assumed that all agents have the same transmitting power $P_{[tx]}$ which is known by the agents beforehand. In Fig. 6.1, we provide an example of received signal strength that follows the assumed path loss model with shadowing and the impact to the corresponding distance estimation as distance between agents increases.



(a) (b)

Figure 6.1: An example of the path loss model and the incurred distance estimation error magnitude from RSSI measurements as distance increases.

6.3 Problem Formulation

We assume the nominal behavior of a MRS is to navigate within a desired proximity-based formation where robots are beyond sensing range from each other while accomplishing tasks during operation. However, robots that experience cyber attacks or faults to positioning sensors can impede on the ability of all robots within the MRS to successfully execute the mission. A problem that we intend to solve is to have robots identify compromised agents that have lost localization capabilities and then behave in a cooperative manner to aid in recovery (i.e., re-localization) of any compromised agents.

The first problem we wish to solve relies on a control policy to trigger a specific motion for cooperative recovery of compromised agents. Formally:

Problem 6.1 *(Control-based Cooperative Recovery) Consider an MRS tasked to navigate an unknown environment towards a goal at position $\boldsymbol{p}_g$. Create a decentralized policy for any robot $i \in \mathcal{V}$ to detect that a neighboring robot $j \in \mathcal{V}$ has lost localization capability (i.e., $\boldsymbol{\xi}_j \neq 0$) and switch to a cooperative recovery control mode $\breve{\boldsymbol{u}}^{(k)}_i$ to recover (i.e., re-localize) the compromised robot j such that:*

$$\mathbb{E}[\boldsymbol{p}_j - \hat{\boldsymbol{p}}_j] = 0. \tag{6.5}$$

Agents that have detected their on-board positioning sensors have been compromised are tasked to generate an identifiable signal to notify nearby agents of the undesirable conditions without explicitly broadcasting of this information. The neighboring robots are able to infer the alerting signal to enable the switch in control behavior to perform cooperative recovery.

An additional framework for cooperative recovery is introduced later in this section which relies on leveraging the known communication model characterized in Section 6.2.3. Differing from the control-based problem described in Problem 6.1, we wish to preserve the desired proximity-based formation while still re-localizing any compromised agents. For this second framework, we are interested in solving the following two problems:

Problem 6.2 *(Detection and Sensor Reconfiguration) Create a policy such that any agent $i \in \mathcal{V}$ that detects anomalous position sensor measurement behavior can reconfigure its sensor model in the D-dimensional space to satisfy:*

$$\widetilde{\boldsymbol{y}}_{i,[1:D]}^{(k)} \longrightarrow \bar{\boldsymbol{y}}_{i,[1:D]}^{(k)} \tag{6.6}$$

by leveraging the known communication model to provide reliable position measurements $\bar{\boldsymbol{y}}_{i,[1:D]}^{(k)}$ to re-localize itself.

Upon detection of sensor attacks/faults and sensor reconfiguration, we want to improve state estimation performance to accommodate the updated sensor measurement model.

Problem 6.3 *(Estimation Error Minimization) Create a policy $\mathcal{P}$ where an agent $i \in \mathcal{V}$ adaptively updates an estimate for the unknown position measurement covariance within the Kalman Filter given the updated sensor model by:*

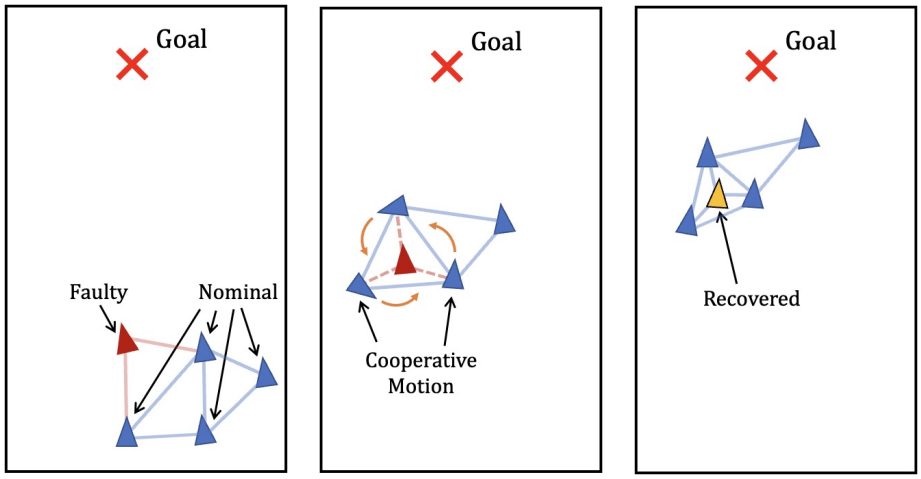
$$\mathcal{P}\big(\hat{\boldsymbol{R}}_i^{(k)}\big) \longrightarrow \min\Big(\big(\boldsymbol{e}_i^{(k)}\big)^{\mathsf{T}}\boldsymbol{e}_i^{(k)}\Big) \tag{6.7}$$

to minimize its estimation error $\boldsymbol{e}_i^{(k)} = \boldsymbol{x}_i^{(k)} - \hat{\boldsymbol{x}}_i^{(k)}$ within the state estimation process to optimized control performance within the multi-agent formation.

6.4 Motion-based Cooperative Recovery

In this section, we introduce a MRS framework where robots cooperatively recover (i.e., re-localize) compromised agents that experience cyber attacks or faults to positioning sensors, as depicted in Fig. 6.2. Our framework leverages randomness to contain hidden data within stochastic system information to alert neighboring robots of the impending attacks/faults. Similar to the framework Chapter 4, the transfer of hidden signals to notify neighboring agents of safety-critical conditions enables the protection of information from interception by potentially malicious eavesdroppers.

When a robot detects that its on-board position sensors have been compromised, it broadcasts a detectable hidden signal to alert neighboring robots of its unreliable on-board sensor(s). This signal triggers the nearby uncompromised robots to perform a cooperative recovery behavior to come within sensing/visual range (i.e., mobile landmarking) to aid in re-localizing the compromised robot. We target autonomous MRSs that may perform operations within unknown adversarial environments that are absent of landmarks that could be used for localization and also operate beyond distance/visual sensing range from each other.



(a) (b) (c)

Figure 6.2: A pictorial representation of the cooperative recovery problem we wish to solve. Nominal (uncompromised) robots (in blue) perform (a) detection of the faulty robot, (b) cooperative behavior mode to act a mobile landmarks, and (c) aid in re-localization (recovery) of compromised robots (red) that experience cyber attacks or faults to on-board position sensors.

The diagram in Fig. 6.3 highlights the overall architecture of our approach to allow for resilient MRS operations. Each robot i monitors on-board sensors for consistent behavior and upon detection of anomalous position sensor behavior, the robot performs state reconstruction to maintain safety during the absence of localization capabilities. After state construction, compromised robots generate an alerting signal that is hidden within state information broadcasts, which the nearby robots can infer the hidden signature to identify the alert. The detection of the hidden alert signal triggers a cooperative control mode in neighboring agents to aid in recovery by coming within sensing range to provide mobile landmarks for the compromised robot, thus allowing the compromised robot to leverage a particle filtering method to re-localize itself within the environment.

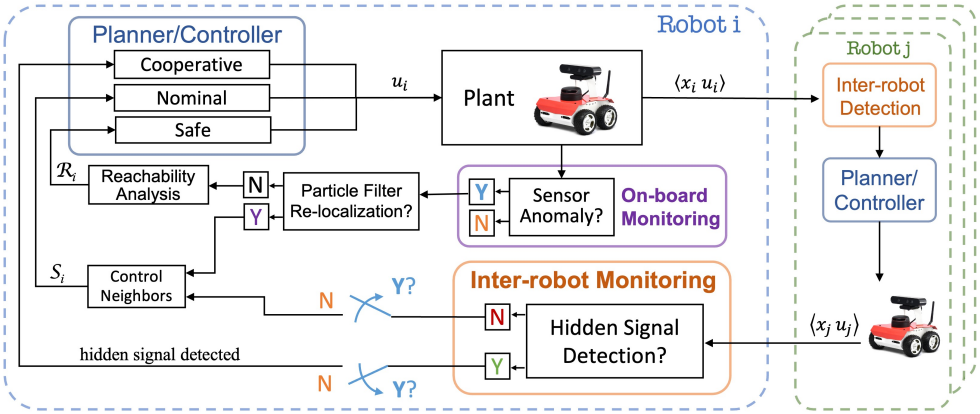


Figure 6.3: The overall architecture of our framework followed by each robot.

6.4.1 On-board Sensor Anomaly Detection

During operations, each robot computes the measurement residual to monitor for consistent behavior from its on-board position sensor measurements. We define the measurement residual $\boldsymbol{r}_i^{(k)}$ on a robot i by

$$\boldsymbol{r}_i^{(k)} = \tilde{\boldsymbol{y}}_i^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_i^{(k)} \in \mathbb{R}^{N_s}, \tag{6.8}$$

that is a normally distributed vector $\boldsymbol{r}_i^{(k)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_i)$ where $\boldsymbol{\Sigma}_i$ is the expected measurement residual covariance during nominal conditions by a robot i. To monitor whether the incoming measurements are behaving in an expected manner, we employ the chi-square detection scheme where the measurement residual vector is reduced to a chi-squared scalar *test measure* defined by:

$$z_i^{(k)} = \left(\boldsymbol{r}_i^{(k)}\right)^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1}\, \boldsymbol{r}_i^{(k)} \tag{6.9}$$

The chi-square detection scheme on-board a robot i triggers an alarm $\zeta_i = 1$ when $z_i^{(k)} > \tau$ is satisfied (otherwise $\zeta_i^{(k)} = 0$) where $\tau \in \mathbb{R}_+$ is a user-defined threshold, then alarms are sent to the runtime estimator in (2.78). When the estimated alarm rate $\hat{A}_i^{(k)}$ no longer follows the expectation:

$$\hat{A}_i^{(k)} \neq \mathbb{E}[A] \quad \longrightarrow \quad \text{Anomaly Detection} \tag{6.10}$$

a robot i concludes that a cyber attack or fault has occurred to its on-board positioning sensor, where $\mathbb{E}[A]$ is the expected alarm rate.

6.4.2 Checkpointing and Reachability for Safe Motion

With the positioning sensor deemed compromised, a robot can no longer rely on its position estimate while navigating. A compromised robot performs state reconstruction via checkpointing

and reachability analysis along with a virtual physics-based scheme to maintain the system safety to avoid reaching undesired areas $\mathcal{A} \subset \mathcal{M}$ in the environment $\mathcal{M}$.

Checkpointing for State Reconstruction

Once a robot i detects a cyber attack or fault to its localization sensor, the robot leverages saved historical information over a window of length $L \in \mathbb{N}$ of the robot's states and control inputs. To recover the system state, the robot "goes back" to a time when the system state was considered safe to reconstruct an estimate of the failed state elements, by using a checkpointing and recovery procedure [46]. The robot then *rolls forward* the checkpointed state by utilizing the stored control inputs from the checkpointing time $k - L$ up until the present time k. From this roll-forward procedure [46], a reconstructed state $\hat{\hat{\boldsymbol{x}}}_i^{(k)} \in \mathbb{R}^n$ is computed by:

$$\hat{\hat{\boldsymbol{x}}}_i^{(k)} = \boldsymbol{A}_d^L \hat{\boldsymbol{x}}_i^{(k-L)} + \sum_{l=1}^{L} \boldsymbol{A}_d^{l-1} \boldsymbol{B}_d \boldsymbol{u}_i^{(k-l)} \tag{6.11}$$

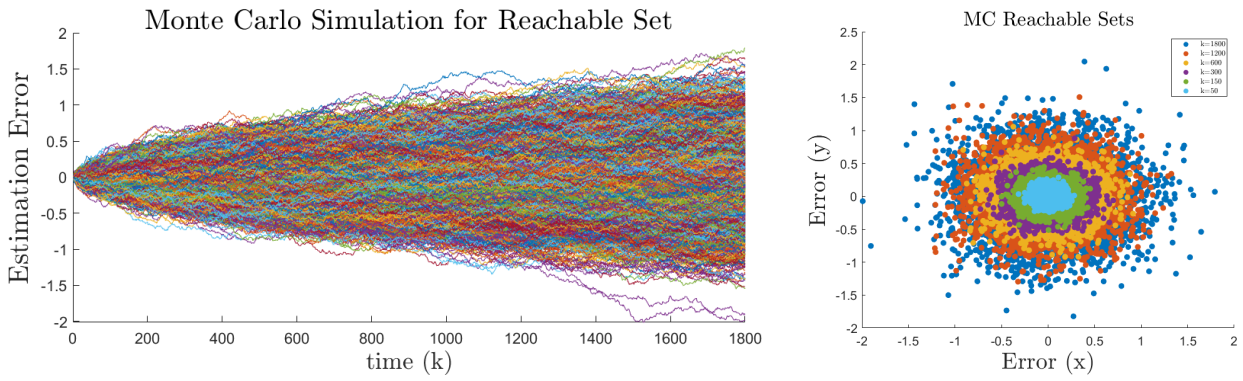
that is then used for control purposes. We assume the window length L of historical data is chosen to contain enough information that extends longer than the time necessary to detect position sensor anomalies [46].

Reachable Sets

With the loss of localization capabilities, we want to ensure that compromised robots do not enter an undesired area $\mathcal{A}$ within the environment $\mathcal{M} \subseteq \mathbb{R}^2$. Compromised robots are tasked to compute a reachable set $\mathcal{R}_i \subset \mathcal{M}$ that is based on the dynamical and noise models to provide all possible positions that the robot's true state may be located given the localization uncertainties to help avoid navigating into any undesired areas. While other techniques exist [52, 48], we utilize Monte Carlo simulations to generate reachable sets [23] where ellipsoid bounds $\varepsilon_i^{(k^*)}$ of the reachable set $\mathcal{R}_i$ are computed, as illustrated with the example in Fig. 6.4. This method encompasses all possible locations of an ith compromised robot, where k^* is the time since beginning the reachable set computation.

Safe Motion

For a compromised robot i to ensure safety while in motion, the goal is to maintain a safe distance away from any known undesired restricted area $\mathcal{A} \subset \mathcal{M}$. To do this, the robot includes a virtual spring from the minimum distance between the reachable set bounds $\varepsilon_i^{(k^*)}$ and the undesired region



(a) (b)

Figure 6.4: Monte Carlo Reachable Set Example.

to provide a repulsive force modeled as:

$$\boldsymbol{u}_{i,\mathcal{A}}^{(k)} = \kappa_{\mathcal{A}}(l_{i\mathcal{A}} - l_o^0)\vec{\boldsymbol{d}}_{i\mathcal{A}}, \quad \text{if } l_{i\mathcal{A}} < l_o^0, \tag{6.12}$$

where $l_{i\mathcal{A}} = \min \left\| \varepsilon_i^{(k^*)} - \boldsymbol{p}_{\mathcal{A}} \right\|$, $l_o^0 > 0$ is a safety distance, and $\boldsymbol{p}_{\mathcal{A}} \in \mathbb{R}^2$ is a position within the undesired region $\mathcal{A}$. The *safe* control input of compromised robot i follows:

$$\bar{\boldsymbol{u}}_i^{(k)} = \boldsymbol{u}_{i,\mathcal{A}}^{(k)} + \kappa_{ig} l_{ig} \vec{\boldsymbol{d}}_{ig} - \gamma_i \hat{\bar{\boldsymbol{v}}}_i^{(k)} \tag{6.13}$$

where $l_{ig} = \|\boldsymbol{p}_g - \hat{\bar{\boldsymbol{p}}}_i^{(k)}\|$ is the distance between the goal point and the reconstructed position estimate and $\hat{\bar{\boldsymbol{v}}}_i^{(k)}$ is the reconstructed velocity estimate. This allows the compromised robot to safely continue navigating toward the goal point while a reliable localization estimate is unavailable.

6.4.3 Inter-robot Residual Characteristics

In this subsection, we formalize the generation of a hidden signature that compromised robots emit to notify neighboring robots of its loss of localization capabilities. We describe the necessary conditions that neighboring robots monitor for to detect the hidden signature to trigger cooperative recovery.

We begin by modeling an inter-robot residual that robots leverage to monitor for consistent behavior of neighboring agents. Each robot exchanges its state estimate and control input at every time instant k to enable neighboring robots to predict its motion. A robot i is then able to make state predictions of any nearby robot $j \in \mathcal{V}$ by:

$$\hat{\boldsymbol{x}}_{ij}^{(k+1)} = \boldsymbol{A}\hat{\boldsymbol{x}}_j^{(k)} + \boldsymbol{B}\boldsymbol{u}_j^{(k)} \tag{6.14}$$

where $\hat{\boldsymbol{x}}_{ij}^{(k+1)} \in \mathbb{R}^n$ is the prediction made by a robot i of a robot j. A robot i then computes an *inter-robot residual*

$$\boldsymbol{r}_{ij}^{(k)} = \hat{\boldsymbol{x}}_{j}^{(k)} - \hat{\boldsymbol{x}}_{ij}^{(k)} \tag{6.15}$$

which enables robot i to monitor for nominal state information behavior received from a robot j. The expectation of the inter-robot residual vector follows $\boldsymbol{r}_{ij}^{(k)} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{ij}) \in \mathbb{R}^n$, where each element $q = \{1, \ldots, n\}$ is characterized by:

$$\mathbb{E}[r_{ij,q}] = 0, \quad \text{Var}[r_{ij,q}] = \sigma_{ij,q}^2 = \sum_{s=1}^{N_s} \Big(K_{(q,s)} \sigma_{j,s} \Big)^2 \tag{6.16}$$

such that $K_{(q,s)}$ represents the element of the qth row and sth column of the Kalman gain $\boldsymbol{K}$. Next, we formalize a procedure to generate a hidden signal to alert nearby robots.

6.4.4 Identifiable Hidden Signal

Once any compromised robot $i \in \mathcal{V}_i^C$ detects anomalous on-board sensor behavior (6.10), it desires to notify neighboring robots of its unreliable sensor without explicitly broadcasting the information. The objective is to produce a detectable randomness-based signature, all while the inter-robot residual (6.16) continues to follow the expected distribution. The compromised robot leverages the reconstructed state $\hat{\tilde{\boldsymbol{x}}}_i^{(k)}$ and safe control input (6.13) to compute the next reconstructed state estimate to time $k+1$ by integrating forward system dynamics, as follows:

$$\hat{\tilde{\boldsymbol{x}}}_i^{(k+1)} = \boldsymbol{A} \hat{\tilde{\boldsymbol{x}}}_i^{(k)} + \boldsymbol{B} \bar{\boldsymbol{u}}_i^{(k)}. \tag{6.17}$$

Additionally, a compromised robot i overlays a hidden signal $\boldsymbol{w}_i^{(k)} \in \mathbb{R}^n$ to the reconstructed state by

$$\hat{\tilde{\boldsymbol{x}}}_{i,\mathcal{H}}^{(k)} = \hat{\tilde{\boldsymbol{x}}}_i^{(k)} + \boldsymbol{w}_i^{(k)} \tag{6.18}$$

which is an additive Gaussian distributed vector that emulates the stochastic state estimate behavior from the viewpoint of neighboring robots. The updated state (6.18) containing the overlaid signal is then broadcast to the nearby robots. In the following Lemma, we show how a compromised robot $j \in \mathcal{V}_j^C$ constructs the hidden signal $\boldsymbol{w}_j^{(k)}$ such that the inter-robot residual (6.15) maintains the expected distribution as in nominal conditions (6.16) from the perspective of a robot i.

Lemma 6.1 *A compromised robot $j \in \mathcal{V}_j^C$ covertly disguises the hidden signal $\boldsymbol{w}_j^{(k)}$ such that the inter-robot residual distribution* (6.16) *from the perspective of neighboring robots emulates nominal behavior if each element $q = \{1, \ldots, n\}$ of the hidden signal vector follows $\mathbb{E}[w_{j,q}] = 0$ and* $\text{Var}[w_{j,q}] = \frac{\sum_{s=1}^{N_s} \left(K_{(q,s)} \sigma_{j,s} \right)^2}{2}$.

Proof: We first consider the forward dynamics in (6.17) and inter-robot prediction in (6.14) where both leverage the same state space dynamics and control input but compute one step ahead projections with differing states. The compromised robot j projects forward with the deterministic dynamics the state $\hat{\hat{\boldsymbol{x}}}_i^{(k)}$ while the neighboring robot i uses the state $\hat{\hat{\boldsymbol{x}}}_{i,\mathcal{H}}^{(k)}$ from (6.18) which contains the hidden signal. A neighboring robot i monitoring inter-robot residuals of robot j, is in effect monitoring the difference between two consecutive Gaussian hidden signals overlaid in the state information at times k and $k-1$, denoted by the difference vector $\boldsymbol{d}_{ij}^{(k)} = \boldsymbol{w}_j^{(k)} - \boldsymbol{w}_j^{(k-1)}$. Given this, the expected distribution of each qth difference vector element follows:

$$\begin{aligned} \mathbb{E}[d_{ij,q}^{(k)}] &= \mathbb{E}[w_{j,q}^{(k)}] - \mathbb{E}[w_{j,q}^{(k-1)}] = 0, \\ \mathrm{Var}[d_{ij,q}^{(k)}] &= \mathrm{Var}[w_{j,q}^{(k)}] + \mathrm{Var}[w_{j,q}^{(k-1)}] = 2\mathrm{Var}[w_{j,q}]. \end{aligned} \tag{6.19}$$

Now, setting the difference vector $\boldsymbol{d}_{ij}^{(k)} \sim \mathcal{N}(\boldsymbol{0}, 2\mathrm{Var}[\boldsymbol{w}_j])$ equal to the expectation of the inter-robot residual in (6.16), we establish that the hidden signal must contain properties:

$$\mathbb{E}[\boldsymbol{w}_j] = \mathbb{E}[\boldsymbol{r}_{ij}], \quad \mathrm{Var}[\boldsymbol{w}_j] = \frac{\boldsymbol{\Sigma}_{ij}}{2}, \tag{6.20}$$

thus each element of the hidden signal $w_{j,q}$ follows:

$$\mathbb{E}[w_{j,q}] = 0, \quad \mathrm{Var}[w_{j,q}] = \frac{\sum_{s=1}^{N_s} \left(K_{(q,s)}\sigma_{j,s}\right)^2}{2}, \tag{6.21}$$

thereby, concluding the proof. ■

By employing the deterministic forward dynamics (6.17) with the hidden signal (6.18), the observation of the inter-robot residual from the perspective of a neighboring robot remains unchanged (i.e., preserved Gaussian distributed properties). However, the inter-robot residual contains a differing sign randomness signature from the nominal operating conditions. Next, we describe the conditions for nearby robots to implement in order to observe a change in inter-robot residual sign switching rate (i.e., randomness properties).

6.4.5 Hidden Signal Detection

To detect a randomness-based hidden signal within (6.18), a robot i monitors an inter-robot residual sign switching rate $\boldsymbol{\psi}_{ij} = \left[\psi_{ij,1}, \ldots, \psi_{ij,n}\right]^\mathsf{T}$ of nearby robots $j \in \mathcal{V}$. The following procedure triggers an alarm (i.e., $\zeta_{ij,q}^{(k)} = 1$) when a sign switch occurs on any qth inter-robot residual element at a time

k by:

$$\zeta_{ij,q}^{(k)} = \begin{cases} 1, & \text{if } \operatorname{sgn}\big(r_{ij,q}^{(k)}\big) = -\operatorname{sgn}\big(r_{ij,q}^{(k-1)}\big) \\ 0, & \text{otherwise.} \end{cases} \tag{6.22}$$

The sign switching alarm $\zeta_{ij,q}^{(k)} \in \{0,1\}$ is then sent to a runtime alarm rate estimation algorithm to provide an updated estimate of the sign switching rate $\hat{\psi}_{ij,q}^{(k)} \in [0,1]$.

Lemma 6.2 *Given a inter-robot residual* (6.15) *during attack-free conditions, the expected value and variance of the sign switching rate to signify nominal random behavior follows* $\mathbb{E}[\psi] = \frac{1}{2}$ *and* $\operatorname{Var}[\psi] = \frac{1}{4(2\ell-1)}$*, respectively.*

Proof: See Lemmas 4.1 and 4.2 in Chapter 4 for similarly designed proofs. ■

The overlaid hidden signal $\boldsymbol{w}_j^{(k)}$ by a robot j transforms the stochastic inter-robot residual variable to contain serial randomness characteristics while observed by a robot i. The following Lemma provides the required observed sign switching behavior by a robot i to determine that a neighboring robot j is emitting the hidden signal.

Lemma 6.3 *A robot* $i \in \mathcal{V}$ *detects the hidden signal* $\boldsymbol{w}_j^{(k)}$ *from a compromised robot* j *when the observed inter-robot residual* (6.15) *sign switching rate for all* $q = 1, \ldots, n$ *elements satisfy* $\hat{\psi}_{ij,q}^{(k)} \in [\Psi'_-, \Psi'_+]$.

Proof: Let us begin by leveraging the expectation of the sign switching alarm rate distribution from known properties a sequence of serial data. The expected value $\mathbb{E}[\psi'] = \frac{2}{3}$ and variance $\operatorname{Var}[\psi'] = \frac{16}{90(2\ell_\psi - 1)}$ of the inter-robot residual sign switching rate is emitted with the hidden signal during safe control (6.13). Considering this, we say that the detection limits are $\Psi' = \{\Psi'_-, \Psi'_+\} \in [0,1]$ described as:

$$\Psi' = \mathbb{E}[\psi'] \pm \left|\Phi^{-1}\Big(\frac{\alpha}{2}\Big)\right| \sqrt{\operatorname{Var}[\psi']} \tag{6.23}$$

where $\alpha \in (0,1)$ denotes the level of significance and $\Phi^{-1}\big(\frac{\alpha}{2}\big)$ describes how many standard deviations from the expected value (i.e., the z-score). To summarize (6.23), when the estimated inter-robot residual sign switching alarm rate for all elements $q = \{1, \ldots, n\}$ of a robot j satisfy:

$$\hat{\psi}_{ij,q}^{(k)} \in [\Psi'_-, \Psi'_+] \;\longrightarrow \textit{Hidden Signal Detection}, \tag{6.24}$$

then robot i detects a hidden signal behavior in robot j. ■

After a robot i identifies that a robot j is compromised, robot j is included into the compromised robot set $j \to \mathcal{V}_i^C$. In Fig. 6.5, we provide examples of the expected distributions of the residual sign switching rate during nominal and hidden signal conditions for various window lengths ℓ_ψ.

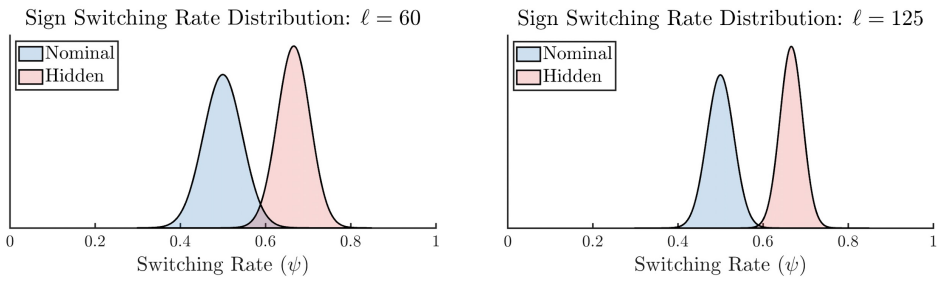


Figure 6.5: Expected distributions for nominal (blue) vs hidden signal (red) inter-robot residual sign switching rates for window lengths $\ell = \{60, 125\}$.

Corollary 6.1 *Given a user-defined window length $\ell_\psi > 0$ for a robot i to estimate inter-robot sign switching rate elements $\hat{\psi}_{ij,q}$, $q = 1, \ldots, n$ of a robot j, the probability of false detection (FD) is described as* $\Pr(FD) = \beta$.

Proof: Under the assumption that each qth inter-robot residual element is independent, the probability β that a qth sign switching alarm rate $\hat{\psi}_{ij,q}$ travels above the lower bound for hidden signal detection (i.e., $\hat{\psi}_{ij,q} > \Psi'_{-}$) follows:

$$\begin{aligned} \beta &\approx 1 - \int_{-\infty}^{\Psi'_{-}} f\big(X \mid \mathbb{E}[\psi], \mathrm{Var}[\psi]\big), \\ &\approx 1 - \int_{-\infty}^{\Psi'_{-}} \frac{1}{\sqrt{2\pi \mathrm{Var}[\psi]}} \exp\left\{ -\frac{1}{2} \left(\frac{X - \mathbb{E}[\psi]}{\sqrt{\mathrm{Var}[\psi]}} \right)^2 \right\} \end{aligned} \tag{6.25}$$

where $f\big(X \mid \mathbb{E}[\psi], \mathrm{Var}[\psi]\big)$ denotes the probability density function of the sign switching alarm rate under nominal conditions. Then, the probability for at least $n_s \in \{1, \ldots, n\}$ sign switching rate elements to travel above the lower detection bound (during nominal conditions) is found by:

$$\Pr(FD) = \sum_{n_s=1}^{n} \binom{n}{n_s} \beta^{n_s} (1 - \beta)^{n - n_s} \tag{6.26}$$

such that when $n_s = n$ (i.e., a false detection) results in $\Pr(FD) = \beta^n$, thus concluding the proof. ■

6.4.6 Cooperative Recovery

In this subsection, we describe the decentralized cooperative behavior of a robot i that detects a hidden signature (6.24) from a compromised robot $j \in \mathcal{V}_i^C$ that also belongs to its control neighbor

set $j \in \mathcal{S}_i$. The cooperative robot i switches to a cooperative control mode to move within sensing range of the compromised agent to provide mobile landmarks for recovery (i.e., re-localization).

Neighboring robots employ cooperative motion via a commonly-used spiral pattern to search for and recover compromised robots [62] (i.e., encircling, rotating, and converging motion). First, each cooperative robot i estimates who the other cooperative neighbors h of the compromised robot $j \in \mathcal{V}_i^C$ are for cooperative recovery by leveraging Gabriel Graph (GG) rule [29]. We utilize GG rule to construct a set of nearest neighbor robots to the compromised robot j that form a connected graph without crossing control edges. A robot i estimates the cooperative set $\hat{\mathcal{C}}_{j,i}$ of robot j by:

$$\hat{\mathcal{C}}_{j,i} = \left\{ h' \in \mathcal{V} \;\middle|\; \widehat{jhh'} \leq \pi/2,\ h, h' \in \mathcal{V} \setminus \mathcal{V}_i^C \right\} \tag{6.27}$$

where $\widehat{jhh'}$, $i \neq h \neq h'$ is the interior angle of the three robot position configuration and positions of robots $h, h' \in \mathcal{V} \setminus \mathcal{V}_i^C$ are received from information broadcasts. Each cooperative robot i utilizes the estimated set $\hat{\mathcal{C}}_{j,i}$ to maneuver into locations to surround the compromised robot j in equal angular intervals. The desired angle between the other cooperative robots follows $\hat{\theta}_i^{des} = \frac{2\pi}{|\hat{\mathcal{C}}_{j,i}|}$. To maintain the desired intervals between its cooperative neighbors while encircling around the compromised robot j, each robot i includes a control input to generate the tangential force:

$$\boldsymbol{u}_{i,E}^{(k)} = \Big[\kappa_E(\hat{\theta}_i^{des} - \widehat{ijh_i^L}) - \kappa_E(\hat{\theta}_i^{des} - \widehat{ijh_i^R}) \Big] \vec{\boldsymbol{d}}_{ij}(\perp) \tag{6.28}$$

where κ_E is a user-defined control gain, while h_i^L and h_i^R are the nearest cooperative neighbors to the left and right from the perspective of robot i, respectively. Additionally, $\widehat{ijh_i^L}$ and $\widehat{ijh_i^R}$ denote inner angles between the nearest left and right cooperative neighbors and $\vec{\boldsymbol{d}}_{ij}(\perp)$ denotes the direction of the tangential force from the vector of robots i to j.

Simultaneously, each cooperative robot i rotates around and converges toward the received position coordinate of the compromised robot j that are computed by:

$$\begin{aligned} \boldsymbol{u}_{i,R}^{(k)} &= \big[\kappa_R(l_{ij} - l_{v'}^0) \big] \vec{\boldsymbol{d}}_{ij}(\perp), \\ \boldsymbol{u}_{i,C}^{(k)} &= \big[\kappa_C(l_{ij} - l_{v'}^0) \big] \vec{\boldsymbol{d}}_{ij}, \end{aligned} \tag{6.29}$$

to generate the rotational $\boldsymbol{u}_{i,R}^{(k)}$ and converging $\boldsymbol{u}_{i,C}^{(k)}$ motion. Parameters that determine control behavior are κ_R and κ_C which denote rotate and converge. The desired distance $l_{v'}^0$ is reduced to $0 < l_{v'}^0 < \delta_r$ to enable cooperative robots to come within sensing range δ_r of the compromised robot j. The position of robot j used for rotation and converging (i.e., utilized in $l_{ij} = \|\hat{\boldsymbol{p}}_i^{(k)} - \check{\boldsymbol{p}}_j^{(k)}\|$) is

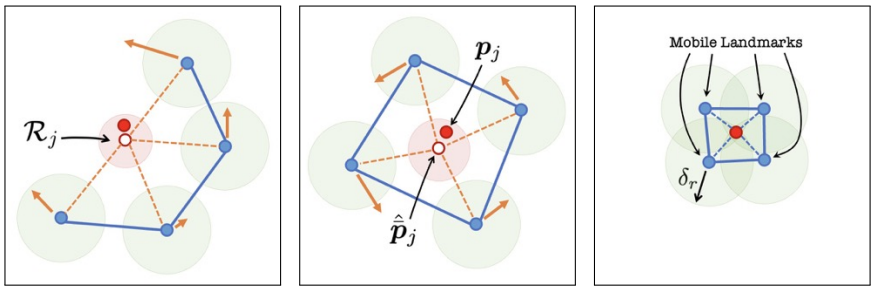
determined by:

$$\breve{\boldsymbol{p}}_j^{(k)} = \begin{cases} \hat{\boldsymbol{p}}_{j,i}^{(k)}, & \text{if } \|\boldsymbol{p}_i^{(k)} - \boldsymbol{p}_j^{(k)}\| \leq \delta_r, \\ \hat{\hat{\boldsymbol{p}}}_{j,\mathcal{H}}^{(k)}, & \text{otherwise,} \end{cases} \tag{6.30}$$

where $\hat{\hat{\boldsymbol{p}}}_{j,\mathcal{H}}^{(k)}$ is the received position (containing the hidden signal) from compromised robot $j \in \mathcal{V}_i^C$ and $\hat{\boldsymbol{p}}_{j,i}^{(k)}$ is the observed estimated position of robot j when within sensing range of robot i. The control input for each robot i during cooperative recovery follows

$$\breve{\boldsymbol{u}}_i^{(k)} = \boldsymbol{u}_{i,E}^{(k)} + \boldsymbol{u}_{i,R}^{(k)} + \boldsymbol{u}_{i,C}^{(k)} + \kappa_{ig} l_{ig} \vec{\boldsymbol{d}}_{ig} - \gamma_i \boldsymbol{v}_i^{(k)} \tag{6.31}$$

to allow for cooperative behavior to find the compromised robot j while still navigating towards the goal. Once a robot i and all cooperative neighbors have come within sensing range of the compromised robot, the robots return to the nominal formation controller (6.2) while maintaining reduced distances $l_{v'}^0$ between the cooperative and compromised robots. Pictured in Fig. 6.6 is an example of cooperative recovery motion from nearby robots (blue disks) to aid in re-localization of a compromised robot (red disk). Cooperative robots rotate and converge to the location of the distressed agent to come within sensing range δ_r (depicted by green disks) to provide mobile landmarks.



(a) (b) (c)

Figure 6.6: Cooperative behavior of robots (blue) that: (a) encircle the estimated location of a compromised robot j (red), (b) rotate around and converge toward the estimated location, and (c) remain within sensing range (green disk) of a compromised robot to provide mobile landmarks for localization.

Remark 6.1 (Lost Robot) *In a scenario where the compromised robot j is not found, each cooperative robot i places robot j into its set of lost robots $j \in \mathcal{V}_i^L$. A robot i determines the compromised robot j is unrecoverable (and no longer used for control) when the distance between the robot j and all cooperative neighbors, including itself, are within sensing range and the robot j is not observed. The cooperative robots return to nominal control mode when this scenario occurs.*

6.4.7 Mobile Landmarks for Re-localization

Once one or more cooperative neighboring robots come within sensing range, a compromised robot $i \in \mathcal{V}_i^C$ begins to re-localize itself within the environment. In this work, compromised robots use a particle filtering-based method for re-localization by leveraging the mobile landmarks (i.e., in-sensing range robots), which are obtained by measurements from on-board range sensors that are assumed to still be available and also utilizing the received position coordinates from nearby robots. Given the known reachable set $\mathcal{R}_i \subset \mathcal{M}$ and the sensing range δ_r, a robot i assumes that any robot j with position coordinates that satisfies:

$$\mathcal{P}_i = \begin{cases} j \in \mathcal{P}_i, & \text{if } \hat{\boldsymbol{p}}_j^{(k)} \in \mathcal{R}_i \oplus \delta_r, \\ j \notin \mathcal{P}_i, & \text{otherwise,} \end{cases} \tag{6.32}$$

can potentially be the observed robot(s) within range sensing, denoted by the robot set $\mathcal{P}_i \subset \mathcal{V}$ where we represent $\mathcal{R}_i \oplus \delta_r$ as the summation of areas from the computed reachable set $\mathcal{R}_i$ and the disk with sensing radius $\delta_r > 0$. The output of our particle filter which utilizes the mobile landmarks provided by nearby robots $j \in \mathcal{P}_i$ (i.e., resulting position coordinates) acts as a replacement for position measurements, in place of the compromised on-board position sensor. This measurement is then used for state estimation and control.

6.4.8 Multiple Compromised Robots

Up to this point, we have characterized the cooperative recovery framework for robots when only a single control neighbor robot j is emitting a hidden signal. However, a robot that encounters a scenario when two or more neighbors are emitting a hidden alert signal must make a decision of which compromised neighbor to choose to aid in re-localization. We present a flow chart in Fig. 6.7 of the decision process made by any robot when this condition is present.

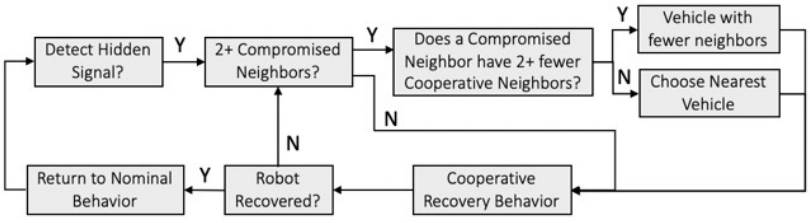


Figure 6.7: Cooperative behavior logic of robots when more than one neighboring robot is emitting a hidden signal.

To summarize, a robot first infers the number of cooperative neighbors that each compromised robot has by using (6.27). If the number of cooperative neighbors for one compromised robot is at least 2 fewer than any other compromised robot, then the robot chooses to aid the robot

with the minimum number of encircling neighbors. If the difference in cooperative neighbors does not satisfy the previous conditions, then the robot will choose the nearest robot (i.e., in terms of euclidean distance). At every time instant k, the robot infers the number of cooperative neighbors per compromised robot, allowing the robot to switch to another compromised robot if network topology conditions change.

6.4.9 Simulation Results

Our approach is validated with MATLAB simulations and experiments using a swarm of ROSbot 2.0 robots. In all case studies, the agents resiliently maintain a desired proximity-based formation when leveraging a virtual spring-damper physics model defined by:

$$\boldsymbol{u}_i^{(k)} = \left[\sum_{j \in \mathcal{S}_i} \kappa_i (l_{ij} - l_v^0) \vec{\boldsymbol{d}}_{ij} + \kappa_g l_{ig} \vec{\boldsymbol{d}}_{ig} \right] - \gamma_i \boldsymbol{v}_i^{(k)}. \tag{6.33}$$

The MRSs are tasked to perform go-to-goal operations in unknown environments where agents are susceptible to cyber attacks and faults to vulnerable on-board position sensors.

We consider $N = 10$ agents treated as double integrator point masses that are navigating in formation to a goal point within an environment $\mathcal{M}$ where desired distances between agents is 10m (i.e., virtual spring rest lengths) and on-board sensing range is limited to $\delta_r = 3$m. A sequence of snapshots presented in Fig. 4.4 highlight our cooperative recovery framework to aid in re-localization of two agents $\{1, 7\}$ that experience malicious cyber attacks that falsify their position sensor measurements. Once the compromised agents detect the cyber attacks, they emit the hidden signature in $\hat{\tilde{\boldsymbol{x}}}_{i,\mathcal{H}}$ to notify neighboring robots (blue disks), allowing them to perform cooperative recovery behavior as shown in Fig. 6.8(b). In Fig. 6.8(c), the cooperative agents come within sensing range of the compromised agents to aid in recovery by acting as a mobile landmark (recovered agents are depicted by yellow disks). Fig. 6.8(d) shows that all compromised agents have been recovered and the multi-robot system continues to navigate toward the desired goal point.

From the perspective of robot 10, we show in Fig. 4.5 the observed inter-robot residual sign switching rates of all neighboring agents. The compromised agents $\{1, 7\}$ have an observed increase in sign switching rates (shown only for position in x-direction), allowing agents to switch to the cooperative recovery control mode to aid in robot re-localization.

Shown in Table 6.1 is the success rate of our cooperative recovery framework during varying MRS sizes and number of compromised robots. For each of the six scenarios, 100 simulations were ran to result in an observed success rate, which is defined as the percent of robots that were successfully re-localized. We observe as the number of compromised robots increases, the overall rate of success decreases since there are fewer uncompromised robots available to aid in recovery.

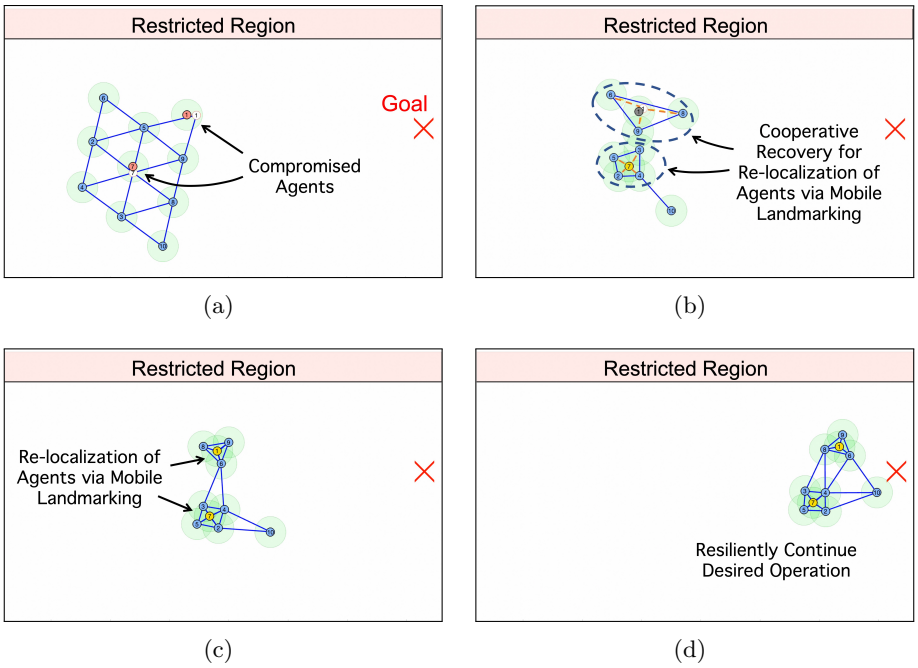


Figure 6.8: A formation of $N = 10$ agents resiliently navigating to a desired goal point (red 'X'). Robots perform cooperative recovery to aid in re-localization of the two compromised robots.

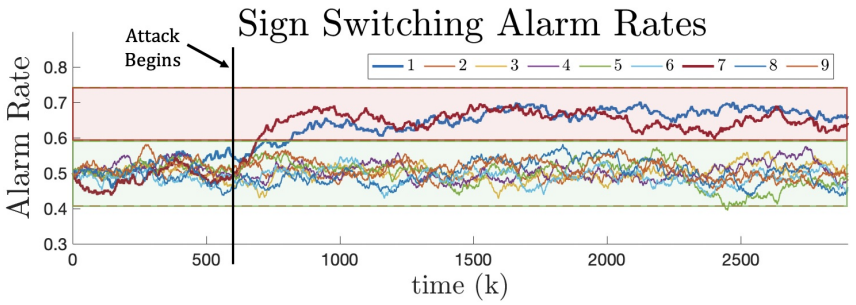


Figure 6.9: Observed inter-robot residual sign switching rate (position in the x-direction) from the perspective of robot 10 throughout the simulation.

Table 6.1: Cooperative Recovery Success Rate

Total Agents	10			25		
Compromised	1	3	5	3	5	10
Success (%)	100	91	56	100	96	64

6.4.10 Experiment Results

In the experiment case study, we consider $N = 6$ Husarion ROSbot 2.0 robots that are navigating in a lab environment while resiliently performing a go-to-goal operation, as depicted in Fig. 4.6. Robot 5 is subject to attacks to its position sensor (Fig. 4.6(b)) in an attempt to hijack the robot

to an undesirable (i.e., restricted) region. Upon detection of the cyber attack, robot 5 generates a hidden signal in $\hat{\tilde{\boldsymbol{x}}}_{i,\mathcal{H}}$ to alert its neighbors. The snapshots in Fig. 4.6(c)-(d) show the cooperative motion of neighboring robots $\{1, 2, 3, 4\}$ to aid in recovery of robot 5 by acting as mobile landmarks. Once robot 5 is recovered (Fig. 4.6(e)), the entire swarm of robots is able to safely navigate to the desired goal region, as shown in Fig. 4.6(f)-(g).

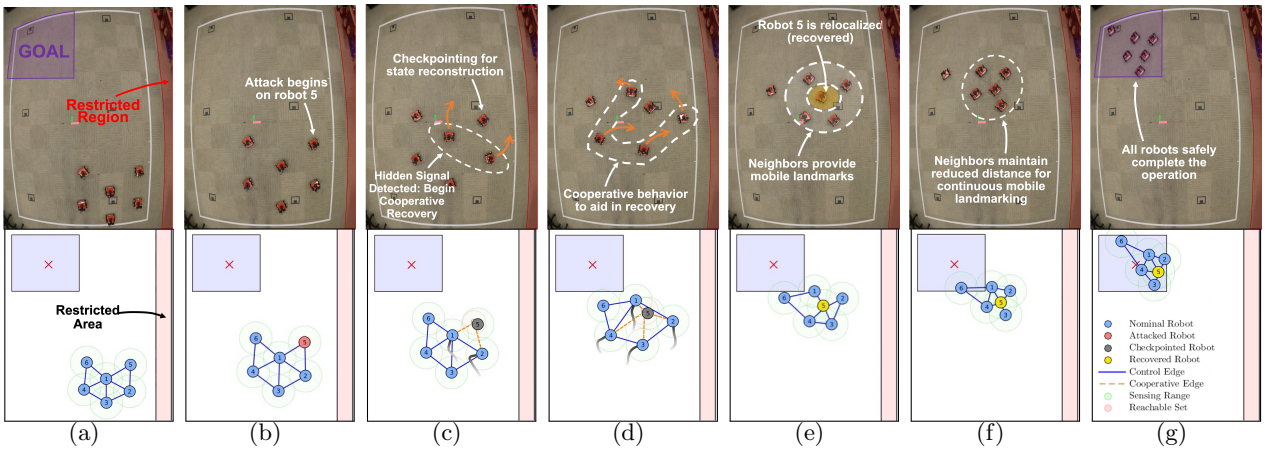


Figure 6.10: An experiment showing $N = 6$ robots navigating to a goal (purple region) with robot $i = 5$ experiencing a cyber attack to its position sensor. The neighboring robots detect the hidden signal from robot 5, then perform a cooperatively aid in re-localization via mobile landmarking.

Furthermore, in Fig. 6.11 we highlight the case where our recovery framework is not implemented. Upon detection of the misbehaving robot, the remaining robots in the swarm isolate the compromised robot without a recovery method in place; thus allowing it to navigate into an undesired region.

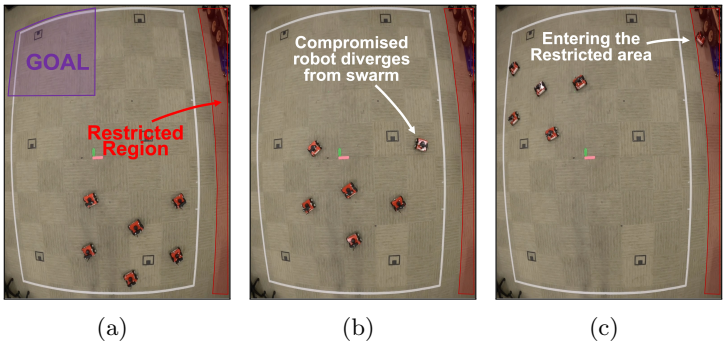


Figure 6.11: A multi-robot system depicting: (a) nominal control performance, (b) a compromised robot diverging from the swarm, and (c) the compromised robot crashing within an undesirable region without our framework.

6.5 Cooperative Recovery via Received Signal Strength Indication

The focus on this section resides on the problem of resilient coordination in multi-agent systems that leverage control consensus schemes when one or more agents lose localization capabilities. More specifically, agents are tasked to maintain desired formations within open or unknown environments that do not offer identifiable landmarks and also operate beyond range sensing of nearby agents for use in localization when nominal on-board positioning sensors are unreliable. To deal with this, the proposed framework enables compromised agents to leverage Received Signal Strength Indication (RSSI) and received position information from neighboring agents for localization (i.e., mobile landmarks). Multilateration is performed using the noisy RSSI measurements and received neighboring agent's positions to provide position measurements in replacement of the unreliable on-board position sensors. To minimize the RSSI-based position measurement error, a weighted least squares method is used that leverages the commonly utilized log-normal shadowing path loss model [33]. Moreover, the RSSI-based position measurements have unknown covariances which differ from the nominal on-board position sensor. To improve state estimation performance, compromised agents leverage an adaptive Kalman Filter (KF) to estimate the position measurement covariance matrix at runtime. The proposed framework is introduced in a generalized manner that may be used on any formation control technique for swarms of homogeneous LTI modeled agents. As a specific case study in this paper, a virtual spring-damper physics model [5] for proximity-based formation control is considered on MASs in a 2-dimensional coordinate frame. However, our MAS framework can be expanded to heterogeneous systems [112], non-linear modeled agents (i.e., by using Extended KF) [59], and higher dimensional coordinate frames [109].

In the remaining of this section, we describe our decentralized framework for detection of cyber attacks and faults to on-board localization sensors and the recovery method by utilizing the RSSI measurements from nearby agents (i.e., mobile landmarks) in the swarm to replace the compromised on-board sensor. The block diagram in Fig. 6.12 summarizes the proposed framework followed by each agent in the swarm to recover from localization sensor/fault to maintain control performance within the formation. As an agent i discovers anomalous behavior to its localization sensors, it switches to a recovery mode which relies on RSSI measurements from nearby uncompromised agents to replace the unreliable on-board sensor measurements. Then, the agent adaptively updates its KF to accommodate the unknown RSSI-based position measurement covariance for improved control performance.

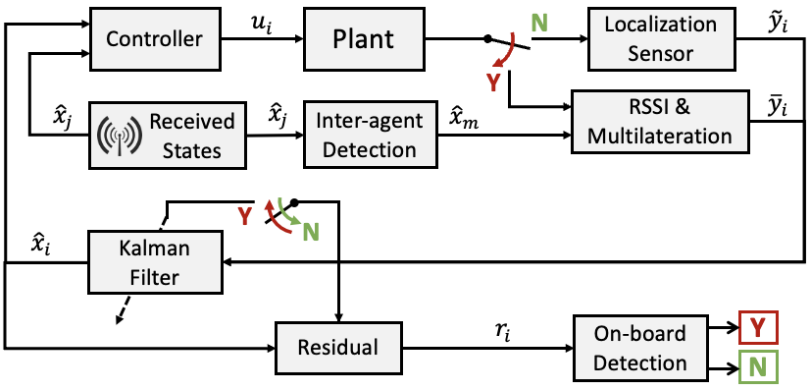


Figure 6.12: Overall control architecture followed by each agent $i \in \mathcal{V}$ to remain resilient from localization (i.e., position) sensor attacks and/or faults.

6.5.1 Anomaly Detection

Each agent $i \in \mathcal{V}$ in our proposed framework monitors for inconsistent behavior of both its on-board position sensor and position information from nearby agents. Let us define the *measurement residual* on an agent i:

$$\boldsymbol{r}_i^{(k)} = \tilde{\boldsymbol{y}}_i^{(k)} - \boldsymbol{C}\hat{\boldsymbol{x}}_i^{(k|k-1)}, \tag{6.34}$$

as the difference between the measurements and the prediction. We can model the measurement residual covariance matrix (assuming the KF has converged to steady state) during attack-free conditions that is described as $\boldsymbol{\Sigma}_i = \boldsymbol{C}\boldsymbol{P}_i^{(\infty)}\boldsymbol{C}^\mathsf{T} + \boldsymbol{R} \in \mathbb{R}^{N_s \times N_s}$ where the steady state estimation covariance $\boldsymbol{P}_i^{(\infty)}$ is found from the discrete Riccati equation. Each agent i monitors its D-dimensional position sensor measurements for anomalies by way of the commonly-used chi-square scheme [67], which produces a scalar quadratic *on-board test measure* computed by

$$z_i^{(k)} = \left(\boldsymbol{r}_{i,[1:D]}^{(k)}\right)^\mathsf{T} \bar{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{r}_{i,[1:D]}^{(k)} \tag{6.35}$$

which has an expected chi-squared distribution $z_i^{(k)} \sim \chi(D)$ with D degrees of freedom. The matrix $\bar{\boldsymbol{\Sigma}}_i \in \mathbb{R}^{D \times D}$ represents the position sensor measurement covariance block within

$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_i & * \\ *^\mathsf{T} & \breve{\boldsymbol{\Sigma}}_i \end{bmatrix} \tag{6.36}$$

where $\breve{\boldsymbol{\Sigma}}_i$ represents the non-position covariance block corresponding to the remaining sensors.

Similarly, each agent monitors for expected behavior of nearby agents according to the control consensus model that is followed by all agents. Each agent i receives both state $\hat{\boldsymbol{x}}_j^{(k)}$ and control input $\boldsymbol{u}_j^{(k)}$ information from any agent $j \in \mathcal{V} \setminus \{i\}$. State evolution predictions $\hat{\boldsymbol{x}}_{ij}^{(k+1)} \in \mathbb{R}^n$ for an

agent j are given as:

$$\hat{\boldsymbol{x}}_{ij}^{(k+1)} = \boldsymbol{A}\hat{\boldsymbol{x}}_{j}^{(k)} + \boldsymbol{B}\boldsymbol{u}_{j}^{(k)} \in \mathbb{R}^{n}, \tag{6.37}$$

which is computed by an agent i. At every time iteration $k \in \mathbb{N}$, an agent i computes the *inter-agent residual* $\boldsymbol{r}_{ij}^{(k)}$ —defined as the difference between the received state estimate $\hat{\boldsymbol{x}}_{j}^{(k)}$ and predicted state $\hat{\boldsymbol{x}}_{ij}^{(k)}$ of an agent j— by

$$\boldsymbol{r}_{ij}^{(k)} = \hat{\boldsymbol{x}}_{j}^{(k)} - \hat{\boldsymbol{x}}_{ij}^{(k)} \in \mathbb{R}^{n}. \tag{6.38}$$

If the agent j is behaving in a nominal fashion, each inter-agent residual element $q = 1, \ldots, n$ follows the distribution $r_{ij,q}^{(k)} \sim \mathcal{N}(0, \sigma_{ij,q}^{2})$ given $\sigma_{ij,q}^{2} = \sum_{s=1}^{N_s} \left(K_{(q,s)}\sigma_{j,s}\right)^{2}$ where $K_{(q,s)}$ represents the element of the qth row and sth column of the Kalman gain $\boldsymbol{K}$, and $\sigma_{j,s}$ is the sth diagonal element in the known measurement covariance matrix $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}_i$. For ease, we construct the inter-agent covariance matrix $\boldsymbol{\Sigma}_{ij} \in \mathbb{R}^{n\times n}$ with diagonal elements equal to $\sigma_{ij,q}$, i.e. $\boldsymbol{\Sigma}_{ij} = \mathrm{diag}(\sigma_{ij,1}, \ldots, \sigma_{ij,n})$. In a similar fashion to the on-board test measure in (6.35), the *inter-agent test measure* is computed as

$$z_{ij}^{(k)} = \left(\boldsymbol{r}_{ij,[1:D]}^{(k)}\right)^{\mathsf{T}} \bar{\boldsymbol{\Sigma}}_{ij}^{-1} \boldsymbol{r}_{ij,[1:D]}^{(k)} \tag{6.39}$$

by an agent i to monitor for consistency of an agent j where $\bar{\boldsymbol{\Sigma}}_{ij} \in \mathbb{R}^{D\times D}$ represents the inter-agent residual covariance block for position within $\boldsymbol{\Sigma}_{ij} = \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_{ij} & * \\ *^{\mathsf{T}} & \breve{\boldsymbol{\Sigma}}_{ij} \end{bmatrix}$.

To monitor for expected behavior of either test measure in (6.35) and (6.39), simply denoted as $z^{(k)}$, an agent i creates an alarm-based mechanism when the test measure exceeds a user-defined threshold τ which follows:

$$\zeta^{(k)} = \begin{cases} 1, & \text{if } z^{(k)} > \tau, \\ 0, & \text{if } z^{(k)} \leq \tau. \end{cases} \tag{6.40}$$

The threshold parameter $\tau \in \mathbb{R}_{>0}$ is tuned to satisfy a user-defined desired false alarm rate $A^{\mathrm{des}} \in (0, 1)$.

Lemma 6.4 (Threshold) *Let us assume that the sensors on agent i are both free of cyber attacks and faults while considering the alarm procedure in* (6.40) *for the test measure $z^{(k)} \sim \chi(D)$ with a threshold $\tau \in \mathbb{R}_{>0}$. To tune for a desired false alarm rate $A^{des} \in (0, 1)$, the threshold τ is found by*

$$\tau = 2\Gamma^{-1}\Big(1 - A^{des}, \frac{D}{2}\Big) \tag{6.41}$$

to achieve a desired alarm rate, where $\Gamma^{-1}(\cdot, \cdot)$ is the inverse regularized lower incomplete gamma function [91].

Formally, the probability $\Pr(\cdot)$ that the test measure exceeds the user-defined threshold is described as $\Pr(z^{(k)} > \tau) \approx A^{\text{des}}$. Each agent i updates its alarm rate estimate $\hat{A}^{(k)}$ with the runtime update $\hat{A}^{(k)} = \hat{A}^{(k-1)} + \frac{\zeta^{(k)} - \hat{A}^{(k-1)}}{\ell}$ where $\hat{A}^{(0)} = A^{\text{des}}$, $\ell > 0$, and the alarm rate estimate can be approximated to a Normal distribution with a variance that shares properties of the exponential moving average [30].

Corollary 6.2 (Detection Bounds) *Given the tuned threshold* (2.69) *for a desired false alarm rate* $A^{des} \in (0, 1)$*, the position sensor measurements are behaving as expected with a level of significance* $\alpha \in (0, 1)$ *if the estimated alarm rate* $\hat{A}^{(k)} \in [0, 1]$ *satisfies the detection bounds* $\hat{A}^{(k)} \in [\mathrm{T}_-, \mathrm{T}_+]$.

Proof: We construct confidence intervals with a level of significance $\alpha \in (0, 1)$ for a Normally distributed random variable, that provide the detection bounds

$$\mathrm{T}_\pm = A^{\text{des}} \pm \left|\Phi^{-1}\left(\frac{\alpha}{2}\right)\right| \sqrt{\frac{A^{\text{des}}(1 - A^{\text{des}})}{2\ell - 1}} \tag{6.42}$$

where alarm rate estimates that go beyond these bounds are exhibiting anomalous behavior, thus concluding the proof. ■

To summarize Corollary 6.2, when the estimated alarm rate $\hat{A}_i^{(k)}$ for detection of inconsistencies to the on-board test measure $z_i^{(k)}$ no longer satisfies the detection bounds $\hat{A}_i^{(k)} \notin [\mathrm{T}_-, \mathrm{T}_+]$, agent i detects that a cyber attack or fault to its positioning sensors is present. In the case of inter-agent monitoring where $\hat{A}_{ij}^{(k)} \notin [\mathrm{T}_-, \mathrm{T}_+]$, an agent i deems an agent j compromised and places the agent into a compromised agent set $\mathcal{V}_i^C \subset \mathcal{V}$, i.e., $j \in \mathcal{V}_i^C$.

6.5.2 RSSI-based Position Measurements

In this subsection, we discuss an RSSI-based localization method that leverages the modeled communication channel described in Section 6.2.3 to replace the compromised/faulty sensor providing position measurements. Our proposed method is performed by any compromised agent i that utilizes the measured RSSI and received position information from uncompromised neighboring agent's communication broadcasts to compute RSSI-based position measurements. We present RSSI-based multilateration in MASs within 2-dimensional environments (i.e., $D = 2$); however, our approach is also valid in 3-dimensional spaces [109].

Once an agent i detects anomalous position sensor behavior, the agent no longer relies on the on-board position sensor and begins to measure RSSI from any trustworthy nearby agents. An agent i utilizes estimated distances from RSSI measurements to each uncompromised mobile agent m in the set $\mathcal{M}_i \subset \mathcal{V} \setminus \{\mathcal{V}_i^C \cup i\}$ where the set $\mathcal{M}_i = \{1, 2, \ldots, M\}$ represents the M uncompromised

mobile agents (i.e., mobile landmarks). From the known communication model, the observed path loss $PL_{im}^{(k)}$ (in dB) by an agent i from an agent m from RSSI $P_{im,[rx]}^{(k)}$ (in dBm) is:

$$PL_{im}^{(k)} = P_{[tx]} - P_{im,[rx]}^{(k)}. \tag{6.43}$$

The estimated distance $\hat{d}_{im}^{(k)}$ based on received (i.e., measured) signal strength of an agent m that is computed by a compromised agent i follows:

$$\hat{d}_{im}^{(k)} = 10^{\frac{PL_{im}^{(k)} - PL(d_0)}{10\beta}}. \tag{6.44}$$

The RSSI-based distance estimate to an agent $m \in \mathcal{M}_i$ are log-normal random variables [103], provided the assumption that the communication channel follows a log-normal shadowing path loss model in (6.4), (6.43), and (6.44). Consequently, the distance estimate in (6.44) is a biased estimate. We leverage the assumed log-normal distribution to compensate for the biased estimate. The log-normal random variables are described by the parameters μ_d and σ_d [103]:

$$\mu_d = \ln d_{ij}^{(k)}, \qquad \sigma_d = \frac{\sigma_\Lambda \ln 10}{10\beta} \tag{6.45}$$

with an RSSI distance estimate expectation that follows

$$\mathbb{E}\big[\hat{d}_{im}^{(k)}\big] = \exp\left\{\mu_d + \frac{\sigma_d^2}{2}\right\} \tag{6.46}$$

such that the expected estimation bias

$$e_{im,d}^{(k)} = \mathbb{E}\big[\hat{d}_{im}^{(k)}\big] - d_{im}^{(k)} \in \mathbb{R}_{>0} \tag{6.47}$$

is defined as the difference between the expectation of the RSSI-based distance measurement $\mathbb{E}\big[\hat{d}_{im}^{(k)}\big]$ and the true distance $d_{im}^{(k)}$. Given that the true distance $d_{im}^{(k)}$ is unknown in (6.45), an agent i leverages the distance $\hat{d}_{im}^{(k)} = \|\hat{\boldsymbol{p}}_i^{(k)} - \hat{\boldsymbol{p}}_m^{(k)}\|$ which is the estimated distance between agents i and m from the position estimate of agent i and the received position estimate from an agent m. Using (6.47), we can rewrite equation (6.44) to compensate for the estimation bias by

$$\hat{d}_{im}^{(k)} = 10^{\frac{PL_{im}^{(k)} - PL(d_0)}{10\beta}} - e_{im,d}^{(k)}. \tag{6.48}$$

Each mth agent's estimated distances are leveraged with their corresponding received positions $\hat{\boldsymbol{p}}_m^{(k)} = [\hat{p}_{m,x}^{(k)}\ \hat{p}_{m,y}^{(k)}]^\mathsf{T}$ to find an optimal position $\bar{\boldsymbol{p}}_i^{(k)}$ by minimizing the distance error residuals

$\epsilon_m \in \mathbb{R}$, $m = 1, \ldots, M$ using the following set of equations:

$$\begin{cases} \hat{d}_{i1}^{(k)} = \sqrt{\left(\hat{p}_{1,x}^{(k)} - \bar{p}_{i,x}^{(k)}\right)^2 + \left(\hat{p}_{1,y}^{(k)} - \bar{p}_{i,y}^{(k)}\right)^2 + \epsilon_1} \\ \hat{d}_{i2}^{(k)} = \sqrt{\left(\hat{p}_{2,x}^{(k)} - \bar{p}_{i,x}^{(k)}\right)^2 + \left(\hat{p}_{2,y}^{(k)} - \bar{p}_{i,y}^{(k)}\right)^2 + \epsilon_2} \\ \vdots \\ \hat{d}_{iM}^{(k)} = \sqrt{\left(\hat{p}_{M,x}^{(k)} - \bar{p}_{i,x}^{(k)}\right)^2 + \left(\hat{p}_{M,y}^{(k)} - \bar{p}_{i,y}^{(k)}\right)^2 + \epsilon_M} \end{cases} \tag{6.49}$$

Next, we subtract the first equation from the remaining equations to obtain a system of $M-1$ linear equations

$$\boldsymbol{\Omega}_i \bar{\boldsymbol{p}}_i^{(k)} = \boldsymbol{\phi}_i + \boldsymbol{\varepsilon}_i \tag{6.50}$$

with $\boldsymbol{\Omega}_i$, $\boldsymbol{\phi}_i$, and $\boldsymbol{\varepsilon}_i$ computed by:

$$\boldsymbol{\Omega}_i = \begin{bmatrix} 2\left(\hat{p}_{2,x}^{(k)} - \hat{p}_{1,x}^{(k)}\right) & 2\left(\hat{p}_{2,y}^{(k)} - \hat{p}_{1,y}^{(k)}\right) \\ 2\left(\hat{p}_{3,x}^{(k)} - \hat{p}_{1,x}^{(k)}\right) & 2\left(\hat{p}_{3,y}^{(k)} - \hat{p}_{1,y}^{(k)}\right) \\ \vdots & \vdots \\ 2\left(\hat{p}_{M,x}^{(k)} - \hat{p}_{1,x}^{(k)}\right) & 2\left(\hat{p}_{M,y}^{(k)} - \hat{p}_{1,y}^{(k)}\right) \end{bmatrix} \tag{6.51}$$

$$\boldsymbol{\phi}_i = \begin{bmatrix} \left(\hat{d}_{i2}^{(k)}\right)^2 - \left(\hat{d}_{i1}^{(k)}\right)^2 + b_2^{(k)} - b_1^{(k)} \\ \left(\hat{d}_{i3}^{(k)}\right)^2 - \left(\hat{d}_{i1}^{(k)}\right)^2 + b_3^{(k)} - b_1^{(k)} \\ \vdots \\ \left(\hat{d}_{iM}^{(k)}\right)^2 - \left(\hat{d}_{i1}^{(k)}\right)^2 + b_M^{(k)} - b_1^{(k)} \end{bmatrix} \tag{6.52}$$

$$\boldsymbol{\varepsilon}_i = \begin{bmatrix} \epsilon_2 - \epsilon_1 & \epsilon_3 - \epsilon_1 & \ldots & \epsilon_M - \epsilon_1 \end{bmatrix}^\mathsf{T} \tag{6.53}$$

where $b_m^{(k)} = \left(\hat{p}_{m,x}^{(k)}\right)^2 + \left(\hat{p}_{m,y}^{(k)}\right)^2$, $\forall m \in \mathcal{M}_i$. To optimize the position $\bar{\boldsymbol{p}}_i^{(k)}$, we use the following objective function:

$$J\left(\bar{\boldsymbol{p}}_i^{(k)}\right) = \arg\min \left[\left\| \boldsymbol{W}_i^{-\frac{1}{2}} \left(\boldsymbol{\Omega}_i \bar{\boldsymbol{p}}_i^{(k)} - \boldsymbol{\phi}_i \right) \right\|^2 \right] \tag{6.54}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{(M-1)\times(M-1)}$ is a weighting matrix which minimizes the sum of squares of the distance error residual vector $\boldsymbol{\varepsilon}_i$. A compromised agent i's RSSI-based position measurement is found by solving the objective function in (6.54) with a weighted least squares (WLS) estimator:

$$\bar{\boldsymbol{y}}_{i,[1:2]}^{(k)} = \bar{\boldsymbol{p}}_i^{(k)} = (\boldsymbol{\Omega}_i^\mathsf{T} \boldsymbol{W}_i^{-1} \boldsymbol{\Omega}_i)^{-1} \boldsymbol{\Omega}_i^\mathsf{T} \boldsymbol{W}_i^{-1} \boldsymbol{\phi}_i. \tag{6.55}$$

Remark 6.2 *For an agent i to compute a single feasible solution for an RSSI-based position measurement (from* (6.50) *and* (6.55)*), the number of nearby uncompromised mobile agents M must satisfy $M \geq D + 1$. This is because under-determined systems have an infinite number of solutions.*

Next, the weighting matrix $\boldsymbol{W}_i$ is characterized to compute an optimal RSSI-based measurement error from (6.55).

6.5.3 Hyperbolic Weighting Matrix

To compute the weighting matrix, we first assume that the RSSI-based estimated distances $\hat{d}_{im}^{(k)}$ from each uncompromised agent $m \in \mathcal{M}_i$ are independent from each other. Thus, the weighting matrix $\boldsymbol{W}_i$ is the covariance of the vector $\boldsymbol{\phi}_i$ that can be calculated by a hyperbolic weighting matrix with each (p_w, q_w)th element (i.e., $w_{p_w q_w}$) given as [103]:

$$w_{p_w q_w} = \begin{cases} \mathrm{Var}\Big[\big(\hat{d}_{i1}^{(k)}\big)^2\Big] + \mathrm{Var}\Big[\big(\hat{d}_{i(p_w+1)}^{(k)}\big)^2\Big] & \text{if } p_w = q_w \\ \mathrm{Var}\Big[\big(\hat{d}_{i1}^{(k)}\big)^2\Big] & \text{if } p_w \neq q_w \end{cases} \tag{6.56}$$

where the elements of the weighting matrix are inter-agent RSSI-based estimation variances $\mathrm{Var}[\cdot]$. Given the log-normal shadowing path loss communication model in (6.4), (6.43), and (6.44), the RSSI-based estimated distances $\hat{d}_{im}^{(k)}$ in (6.48) are log-normal random variables where we can leverage the modeled variances. The estimation variances are computed by [103]:

$$\mathrm{Var}\big[\big(\hat{d}_{im}^{(k)}\big)^2\big] = \mathbb{E}\big[\big(\hat{d}_{im}^{(k)}\big)^4\big] - \big(\mathbb{E}\big[\big(\hat{d}_{im}^{(k)}\big)^2\big]\big)^2 \tag{6.57}$$

which are found from the cth moment of a given log-normal distributed variable that is represented as $\mathbb{E}[(\hat{d}_{im})^c] = \mathrm{e}^{c\mu_d + \frac{c^2\sigma_d^2}{2}}$ with log-normal parameters. Therefore, the second and fourth moments from (6.57) are:

$$\mathbb{E}\big[\big(\hat{d}_{im}^{(k)}\big)^4\big] = \mathrm{e}^{4\mu_d + 8\sigma_d^2}, \tag{6.58}$$

$$\mathbb{E}\big[\big(\hat{d}_{im}^{(k)}\big)^2\big] = \mathrm{e}^{2\mu_d + 2\sigma_d^2}, \tag{6.59}$$

and substituting (6.58) and (6.59) into (6.57), we result in

$$\mathrm{Var}\big[\big(\hat{d}_{im}^{(k)}\big)^2\big] = \mathrm{e}^{4\mu_d}\big(\mathrm{e}^{8\sigma_d^2} - \mathrm{e}^{4\sigma_d^2}\big) \tag{6.60}$$

which are used in the construction of the weighting matrix (6.56) to compute the optimized RSSI-based position measurement $\bar{\boldsymbol{y}}_{i,[1:2]}^{(k)}$ by using the WLS algorithm (6.55).

6.5.4 Adaptive Position Measurement Covariance Estimation

With the position sensors reconfigured to leverage RSSI-based measurements, the measurement covariance matrix $\boldsymbol{R}$ used during nominal conditions is no longer suitable for optimal state estimation.

To deal with this issue, the compromised agent leverages the recursive KF process to allow for adaptive updates (i.e., time varying) of the measurement covariance matrix $\boldsymbol{R}_i^{(k)}$. The adaptive KF can be described in three phases, as follows:

Kalman Filter Phase: Prediction

As in the same manner of a typical recursive KF, the following equations denote

$$\hat{\boldsymbol{x}}_i^{(k|k-1)} = \boldsymbol{A}\hat{\boldsymbol{x}}_i^{(k-1|k-1)} + \boldsymbol{B}\boldsymbol{u}_i^{(k-1)}, \tag{6.61}$$

$$\boldsymbol{P}_i^{(k|k-1)} = \boldsymbol{A}\boldsymbol{P}_i^{(k-1|k-1)}\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{Q}, \tag{6.62}$$

the predicted state and estimation error covariance.

Kalman Filter Phase: Correction

We utilize a similar principal to previous literature ([1], [118]) that have characterized adaptive adjustments to estimate unknown process and measurement covariance matrices in dynamic systems. In these works, a residual-based method (i.e., the measurement residual in (6.34)) is used to adaptively update estimates of the unknown noise covariances. However, the authors assumed that sensor measurements are always zero-mean Gaussian distributed (i.e., attack-free conditions), thus the state estimates are zero-mean Gaussian distributed random variables. In the presence of unreliable sensor measurements and state estimates, the covariance matrix estimation can also be compromised, hence leading to unreliable covariance estimates.

Here, we also leverage a residual-based method to estimate the measurement covariance matrix. However, due to the unreliable position estimates from attacks/faults that occur to position sensors, the residual for estimation must omit the use of the estimates for position $\hat{\boldsymbol{x}}_{i,[1:D]}^{(k)}$ within the state estimate vector. To deal with this problem, we leverage the RSSI-based position measurements (6.55), multi-agent system model, and the remaining states in the state estimation vector $\hat{\boldsymbol{x}}_{i,[(D+1):n]}^{(k)}$ to reconstruct the RSSI-based position measurement covariance. Furthermore, we utilize known properties from serial randomness over the sequence of data for RSSI measurement covariance estimation.

First, to reconfigure to RSSI-based position measurements, an agent i updates the first D rows of the output matrix $\boldsymbol{C}$ to form the updated output matrix $\bar{\boldsymbol{C}}_i$ by

$$\bar{\boldsymbol{C}}_{i,[1:D]} = \begin{bmatrix} \boldsymbol{I}_D & \boldsymbol{0}_{D\times(N_s-D)} \end{bmatrix} \tag{6.63}$$

where $\boldsymbol{I}_D \in \mathbb{R}^{D\times D}$ denotes a D-dimension identity matrix, as the RSSI-based position measurements have a 1:1 mapping with the position states (i.e., $\bar{\boldsymbol{y}}_{i,[1:D]}^{(k)} = \bar{\boldsymbol{p}}_i^{(k)}$ where $\bar{\boldsymbol{p}}_i^{(k)}$ is the direct mapping of

the state from the RSSI-based position measurement in (6.55)). The updated output matrix used for the adaptive KF is denoted by $\bar{\boldsymbol{C}}_i \in \mathbb{R}^{N_s \times n}$ such that the first D rows are described in (6.63) and the remaining $N_s - D$ rows (if applicable) are the same as the output matrix $\boldsymbol{C}$.

Assumption 6.1 *The RSSI-based position measurements can be approximated as a Gaussian distributed vector with covariance $\bar{\boldsymbol{R}}_i \in \mathbb{R}^{D\times D}$ centered over the true position $\boldsymbol{p}_i^{(k)}$ of the compromised agent i (i.e., $\bar{\boldsymbol{y}}_{i,[1:D]}^{(k)} \approx \mathcal{N}(\boldsymbol{p}_i^{(k)}, \bar{\boldsymbol{R}}_i)$).*

Next, a compromised agent i computes the RSSI position measurement residual described as

$$\bar{\boldsymbol{r}}_i^{(k)} = \bar{\boldsymbol{y}}_{i,[1:D]}^{(k)} - \bar{\boldsymbol{C}}_{i,[1:D]}\hat{\bar{\boldsymbol{x}}}_i^{(k)} \tag{6.64}$$

which is a comparison between the RSSI-based position measurement (6.55) and a position prediction $\bar{\boldsymbol{C}}_{i,[1:D]}\hat{\bar{\boldsymbol{x}}}_i^{(k)}$. The prediction vector $\hat{\bar{\boldsymbol{x}}}_i^{(k)}$ in (6.64) is found by

$$\hat{\bar{\boldsymbol{x}}}_i^{(k)} = \boldsymbol{A}\bar{\boldsymbol{x}}_i^{(k-1)} + \boldsymbol{B}\boldsymbol{u}_i^{(k-1)} \tag{6.65}$$

where $\bar{\boldsymbol{x}}_i^{(k-1)}$ is a change to the previous state estimate vector at time $k-1$ where the position estimate $\hat{\boldsymbol{x}}_{i,[1:D]}^{(k-1|k-1)}$ is replaced with the position mapping from the RSSI-based position measurement $\bar{\boldsymbol{y}}_{i,[1:D]}^{(k-1)} = \bar{\boldsymbol{p}}_i^{(k-1)}$, i.e.,

$$\bar{\boldsymbol{x}}_i^{(k-1)} = \left[\left(\bar{\boldsymbol{p}}_i^{(k-1)}\right)^{\mathsf{T}} \ \left(\hat{\boldsymbol{x}}_{i,[(D+1):n]}^{(k-1|k-1)}\right)^{\mathsf{T}}\right]^{\mathsf{T}}. \tag{6.66}$$

At each time $k \in \mathbb{N}$ the difference (i.e., the residual in (6.64)) $\bar{\boldsymbol{r}}_i^{(k)} = \bar{\boldsymbol{p}}_i^{(k)} - \hat{\bar{\boldsymbol{p}}}_i^{(k)}$ where $\hat{\bar{\boldsymbol{p}}}_i^{(k)} = \hat{\bar{\boldsymbol{x}}}_{i,[1:D]}^{(k)}$ from (6.65) is monitored between two consecutive positions with respect to the true state $\boldsymbol{x}_i$ at times k and $k-1$. The noise characteristics for the measured position $\bar{\boldsymbol{p}}_i^{(k)}$ at a time k are subject to both measurement noise $\bar{\boldsymbol{R}}_i^{(k)}$ and process noise $\bar{\boldsymbol{Q}}$ of the position, where $\bar{\boldsymbol{Q}}$ is the noise covariance block within $\boldsymbol{Q}$ that corresponds to the position states of the compromised sensors. Given these noise properties, the expectation of the RSSI measured residual are described as:

$$\begin{aligned}\mathbb{E}[\bar{\boldsymbol{r}}_i^{(k)}] = \mathbb{E}[\boldsymbol{\eta}_{i,[1:D]}^{(k)}] - \mathbb{E}[\boldsymbol{\eta}_{i,[1:D]}^{(k-1)}] \\ + \mathbb{E}[\boldsymbol{\nu}_{i,[1:D]}^{(k-1)}] - \mathbb{E}[\boldsymbol{\nu}_{i,[1:D]}^{(k-2)}] = \boldsymbol{0},\end{aligned} \tag{6.67}$$

$$\begin{aligned}\mathrm{Var}[\bar{\boldsymbol{r}}_{i,[1:D]}^{(k)}] &= \mathrm{Var}[\boldsymbol{\eta}_{i,[1:D]}^{(k)}] + \mathrm{Var}[\boldsymbol{\eta}_{i,[1:D]}^{(k-1)}] \\ &\quad + \mathrm{Var}[\boldsymbol{\nu}_{i,[1:D]}^{(k-1)}] + \mathrm{Var}[\boldsymbol{\nu}_{i,[1:D]}^{(k-2)}] \\ &= 2\bar{\boldsymbol{R}}_i^{(k)} + 2\bar{\boldsymbol{Q}} \approx \bar{\boldsymbol{\Sigma}}_i^{(k)}.\end{aligned} \tag{6.68}$$

The objective is to estimate the positive definite covariance matrix of the RSSI measurement residual

$$\bar{\boldsymbol{\Sigma}}_i^{(k)} = \mathbb{E}\left[\bar{\boldsymbol{r}}_i^{(k)}\left(\bar{\boldsymbol{r}}_i^{(k)}\right)^{\mathsf{T}}\right] \in \mathbb{R}^{D\times D} \tag{6.69}$$

by using a rolling runtime estimate similar to [1]

$$\bar{\boldsymbol{\Sigma}}_i^{(k)} = (1-\delta)\bar{\boldsymbol{\Sigma}}_i^{(k-1)} + \delta\Big(\bar{\boldsymbol{r}}_i^{(k)}\big(\bar{\boldsymbol{r}}_i^{(k)}\big)^{\mathsf{T}}\Big) \tag{6.70}$$

where $\delta \in (0,1)$ is a *forgetting* parameter in adaptively updating the covariance[1]. From the approximated estimated covariance in (6.68), we can compute the estimated RSSI-based position measurement covariance block

$$\bar{\boldsymbol{R}}_i^{(k)} = \frac{\bar{\boldsymbol{\Sigma}}_i^{(k)} - 2\bar{\boldsymbol{Q}}}{2} \tag{6.71}$$

within the updated measurement covariance matrix

$$\boldsymbol{R}_i^{(k)} = \begin{bmatrix} \bar{\boldsymbol{R}}_i^{(k)} & \mathbf{0}_{D\times(N_s-D)} \\ \mathbf{0}_{(N_s-D)\times D} & \breve{\boldsymbol{R}} \end{bmatrix} \tag{6.72}$$

for all N_s sensors, where $\breve{\boldsymbol{R}} \in \mathbb{R}^{(N_s-D)\times(N_s-D)}$ is the measurement covariance of the remaining non-position sensors.

Kalman Filter Phase: Update

We incorporate the correction phase that adaptively updated the position measurement covariance to optimize the state estimate, we then have the updates to:

$$\boldsymbol{K}_i^{(k)} = \boldsymbol{P}_i^{(k|k-1)}\bar{\boldsymbol{C}}_i^{\mathsf{T}}\big(\bar{\boldsymbol{C}}_i\boldsymbol{P}_i^{(k|k-1)}\bar{\boldsymbol{C}}_i^{\mathsf{T}} + \boldsymbol{R}_i^{(k)}\big)^{-1}, \tag{6.73}$$

$$\hat{\boldsymbol{x}}_i^{(k|k)} = \hat{\boldsymbol{x}}_i^{(k|k-1)} + \boldsymbol{K}_i^{(k)}\big(\bar{\boldsymbol{y}}_i^{(k)} - \bar{\boldsymbol{C}}_i\hat{\boldsymbol{x}}_i^{(k|k-1)}\big), \tag{6.74}$$

$$\boldsymbol{P}_i^{(k|k)} = \big(\boldsymbol{I}_n - \boldsymbol{K}_i^{(k)}\bar{\boldsymbol{C}}_i\big)\boldsymbol{P}_i^{(k|k-1)}, \tag{6.75}$$

the Kalman gain, state estimate, and estimation covariance.

6.5.5 Simulation Results

Our approach is validated with MATLAB simulations on swarms of $N = 12$ mobile agents modeled as double integrator dynamics. The MAS performs a go-to-goal operation within the x-y plane while experiencing cyber attacks and faults to on-board positioning sensors. As a case study, we employ a virtual spring-damper mesh [5] for decentralized proximity-based formation control to validate the RSSI-based localization framework for resilient formation control. For all simulations, the agents begin with randomized initial positions and are tasked to perform a go-to-goal operation while maintaining $l^{\text{des}} = 8$m distance from any neighboring agent. Additionally, at time $k = 350$, five

[1]We note that a smaller δ puts a greater weight on the previous RSSI measurement residual covariance estimate at time $k-1$, thus resulting in less variation in the updated estimate.

agents are randomly chosen to suffer from a cyber attack (false data injection) and two experience sensor faults. The communication model has a path loss exponent $\beta = 2$ and shadowing noise of $\Lambda = \mathcal{N}(0, 2)$, while the forgetting parameter used is $\delta = 0.01$ for measurement covariance estimation.

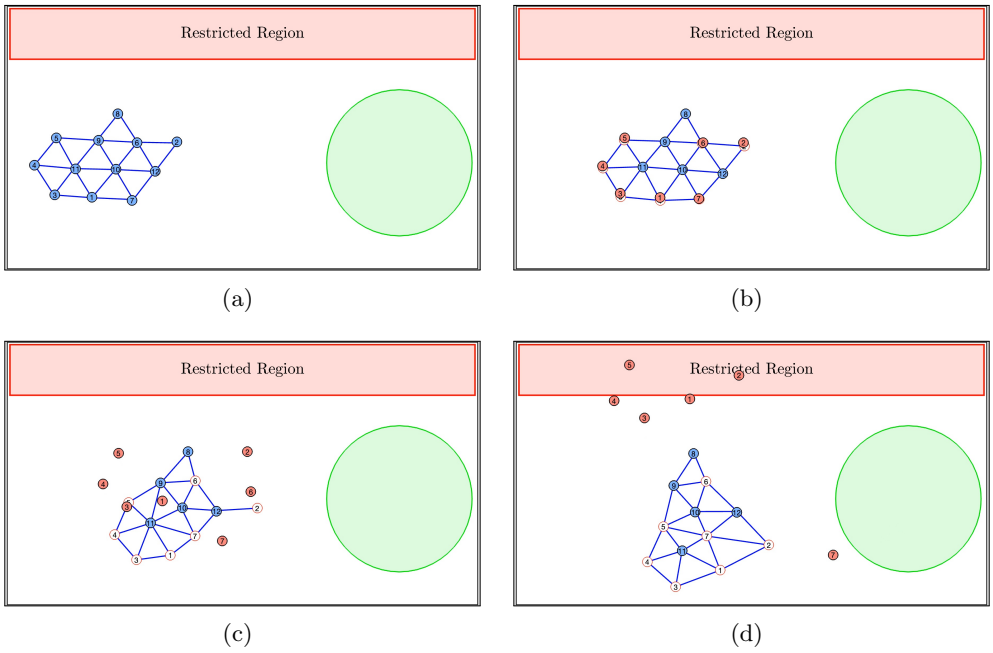


Figure 6.13: An unprotected system of $N = 12$ agents compromised by cyber attacks and faults to on-board position sensors on seven agents (red disks) resulting in them being diverted away from the goal (green region).

In the first simulation presented in Fig. 4.4, we show a sequence of snapshots for an unprotected multi-agent system navigating toward a goal region (green disk). The implemented cyber attacks occur simultaneously on the five compromised agents with the intent of diverting their true positions toward the undesired region (red region). Beginning in Fig. 6.13(a), all agents are uncompromised (blue disks) and perform in a nominal manner; however in Fig. 6.13(b), the compromised agents are subject to malicious cyber attacks and faults to their positioning sensors. In Figs. 6.13(c)-(d), the true states of the compromised agents (red disks) are driven to undesired regions in the environment, while the remaining uncompromised agents (blue disks) along with the corresponding unreliable position estimates of the compromised agents (empty disks) continue navigating toward the goal. In Fig. 5.9 we perform the same simulation as in Fig. 4.4, but this time the MAS is utilizing our framework for resiliency. As shown in the series of snapshots, all seven agents are able to: 1) detect the anomalous on-board positioning sensor behavior and 2) perform sensor reconfiguration to RSSI-based position measurements to resiliently maintain desired control performance within the MAS such that all agents safely reach the desired goal.

We provide a comparison between various MAS scenarios by showing the true proximity

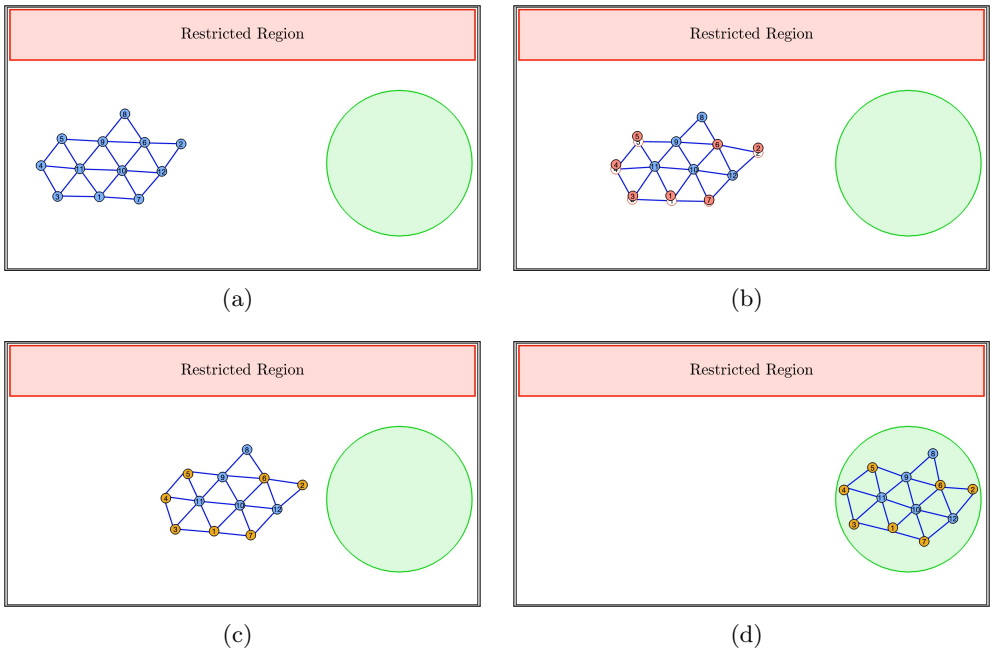


Figure 6.14: A system of $N = 12$ agents leveraging our framework resiliently navigates to the desired goal point while experiencing cyber attacks and faults to on-board position sensors (recovered agents in yellow).

error between neighboring agents $(i, j) \in \mathcal{E}$. We highlight the scenarios which include: 1) no detection and recovery, 2) a non-robust noise covariance update method of $\boldsymbol{R}^{(k)}$ in [1], 3) detection and recovery without updating $\boldsymbol{R}^{(k)}$, and 4) our proposed robust method for updating $\boldsymbol{R}^{(k)}$. Fig. 6.15 shows the average formation proximity error over 400 simulations with randomized initial positions for each scenario. At each time step k, the formation proximity error is computed as $E^{(k)} = \frac{1}{|\mathcal{E}|} \sum_{\forall (i,j) \in \mathcal{E}} \left| \|\boldsymbol{p}_i^{(k)} - \boldsymbol{p}_j^{(k)}\| - l^{\text{des}} \right|$.

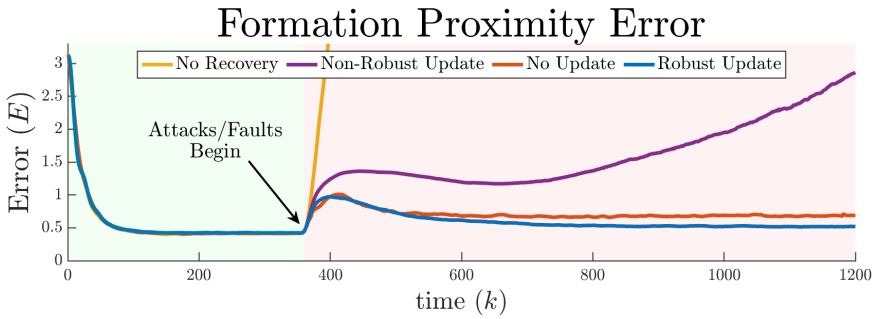


Figure 6.15: A comparison of inter-agent proximity error within the formation.

In Table 6.2, the results of position estimation error $\boldsymbol{e}_{i,x/y}$ in the 2-dimensional x-y coordinate frame are presented for any compromised agent i. Our proposed method to adaptively update the unknown RSSI-based position measurement covariance improves estimation error for more desirable

control performance within the swarm.

Table 6.2: Position Estimation Error.

	No Update	Robust Update
Variance $[e_{i,x/y}]$	0.249/0.253	0.151/0.144

6.6 Discussion

In this chapter, we have presented two decentralized frameworks for a multi-robot system to safely and resiliently perform operations by utilizing cooperative recovery techniques to aid in re-localizing compromised robots that experience positioning sensor cyber attacks or faults. The first cooperative recovery framework leverages motion to recover any compromised agents subject to cyber attacks or faults to positioning sensors. Compromised robots generate a randomness-based hidden signature to alert neighboring robots of the impending attack, allowing them to switch to a cooperative recovery control mode to aid in robot re-localization via mobile landmarking. The second method provides a decentralized framework is an alternative method for recovery within multi-robot systems without altering the nominal network topology. In a similar manner, multi-agent systems are able to resiliently navigate in the presence of cyber attacks and/or faults to on-board positioning sensors within open or unknown environments that lack identifiable landmarks (i.e., anchors) and also operate beyond distance/range sensing of other nearby agents. Upon detection of anomalous sensor behavior, an agent performs sensor reconfiguration to leverage RSSI-based measurements from the nearby agents (i.e., mobile landmarks) as a replacement for the original position sensor. An adaptive Kalman Filtering method accommodates the updated position sensor by estimating its unknown RSSI-based measurement covariance to reduce estimation error for improved control performance within the swarm.

Future ideas to extend these cooperative recovery frameworks could include exploring within heterogeneous multi-robot systems where we can exploit differing system dynamics to more efficiently aid with recovery. For the RSSI-based recovery framework, future work includes improving the robustness when the assumed path loss model does not hold due to instability of RSSI signals, for example, due to the presence of cluttered environments that create multi-path behavior in the communication broadcasts.

Chapter 7

Cooperative Robotic Teams for Defending Against Malicious Intruders

In this chapter, we present a framework for defensive robot teams to defend protected regions from malicious intruding robots. We provide preliminary results to show our progress on this framework, as it is a continuation of the cooperative multi-robot system frameworks introduced in Chapter 6. This paper is in preparation to be submitted to a journal in robotics.

7.1 Introduction

Recent advances in multi-robot systems have enabled the development of cooperative control strategies in applications such as: escorting [37], exploration [58], capturing/entrapment [80], and military platooning [61]. Another interesting, yet similar, concept is shepherding, where the objective is to guide one or more actors from an initial to a final location by using a team of cooperative robots. In defensive applications leveraging robotic teams, the actor(s) are maneuvered to a desirable region in the environment to fulfill a task.

Recently, the research community has began working on shepherding solutions where the actors are assumed non-cooperative with the robotic team, thus the cooperative robots require complex control algorithms to influence the motion of the actors. Typically, the assumption of the actor(s) that need to be shepherded is that their behavior model is passive, meaning that their motion is only influenced by nearby positioning of the robotic herders (e.g., utilizing a potential function model). The problem of shepherding is increasingly challenging when the actors no longer behave passively, and instead perform in a malicious manner. For example, a malicious actor (or in this case, called an *intruder* or *attacker*) may have an objective to reach a specific location in the environment to cause damage or harm to an object, person, or region. With this in mind, a team of cooperative robots may be tasked to prevent this harmful event from occurring by working together in a defensive manner to block the attempted intrusion. Within this chapter, we combine the

concept of shepherding with cooperative defensive behavior by leveraging multi-robot systems to safeguard *protected* regions within an environment. Moreover, we assume that the intruder continues to maintain its malicious behavior, despite having been intercepted by the defensive robots. In turn, the intruder may create avoidance behaviors to navigate around the defensive robots that are behaving as "obstacles" to impede its motion. We introduce a cooperative control policy utilized by a team of defensive robots which: 1) intercept an incoming intruder agent, 2) impedes the motion of the intruder by constructing a barrier formed by the defender robots, and 3) shepherd the malicious intruder to a safe region within the environment.

7.2 Preliminaries

We consider a team of N_d defender robots denoted as $i \in \mathcal{D} = \{1, 2, \ldots, N_d\}$ tasked to defend against a single robotic intruder within a 2-dimensional environment $\mathcal{M} \subset \mathbb{R}^2$. Within the environment is a *protected region* $\mathcal{P} \subset \mathcal{M}$ defined by $\mathcal{P} = \{m \in \mathbb{R}^2 \mid \|m - c_\mathrm{p}\| < r_\mathrm{p}\}$ and a *safe region* $\mathcal{S} \subset \mathcal{M}$ described as $\mathcal{S} = \{m \in \mathbb{R}^2 \mid \|m - c_\mathrm{s}\| < r_\mathrm{s}\}$. The variables $(c_\mathrm{p}, r_\mathrm{p})$ and $(c_\mathrm{s}, r_\mathrm{s})$ characterize the centers and radii of the protected and safe regions, respectively. We assume the defenders and intruder robot are subject to double integrator dynamics

$$\dot{\boldsymbol{p}}_{\mathrm{d},i} = \boldsymbol{v}_{\mathrm{d},i}, \quad \dot{\boldsymbol{v}}_{\mathrm{d},i} = \boldsymbol{u}_{\mathrm{d},i} \tag{7.1}$$

$$\dot{\boldsymbol{p}}_\mathrm{a} = \boldsymbol{v}_\mathrm{a}, \quad \dot{\boldsymbol{v}}_\mathrm{a} = \boldsymbol{u}_\mathrm{a} \tag{7.2}$$

$$\|\boldsymbol{u}_{\mathrm{d},i}\| \leq u_{\mathrm{d,max}}, \quad \|\boldsymbol{u}_\mathrm{a}\| \leq u_{\mathrm{a,max}} \tag{7.3}$$

where $\boldsymbol{p}_{\mathrm{d},i} \in \mathbb{R}^2$ is the position of an ith defender and $\boldsymbol{p}_\mathrm{a} \in \mathbb{R}^2$ is the position of the intruder robot in an (x-y)–coordinate frame, respectively. The vectors $\boldsymbol{v}_{\mathrm{d},i}$ and $\boldsymbol{v}_\mathrm{a}$ denote velocities for defenders and the attacker, while $\boldsymbol{u}_{\mathrm{d},i}$ and $\boldsymbol{u}_\mathrm{a}$ represent the accelerations, which function as control inputs for the defenders and attacker robots. Accelerations are limited by maximum values $u_{\mathrm{d,max}}$ and $u_{\mathrm{a,max}}$, where $u_{\mathrm{a,max}} \leq u_{\mathrm{d,max}}$. In Fig. 7.1 is an example of the environment $\mathcal{M}$ considered, consisting of the protected and safe regions, along with the defender and attacker robots.

7.2.1 Attack Model

In this work, we assume the attacker's goal is to navigate into the known protected region $\mathcal{P}$. To do this, the attacker consists of an attacking force

$$\boldsymbol{u}_\mathrm{attack} = u_\mathrm{attack,mag}\vec{\boldsymbol{d}}_{ap} \tag{7.4}$$

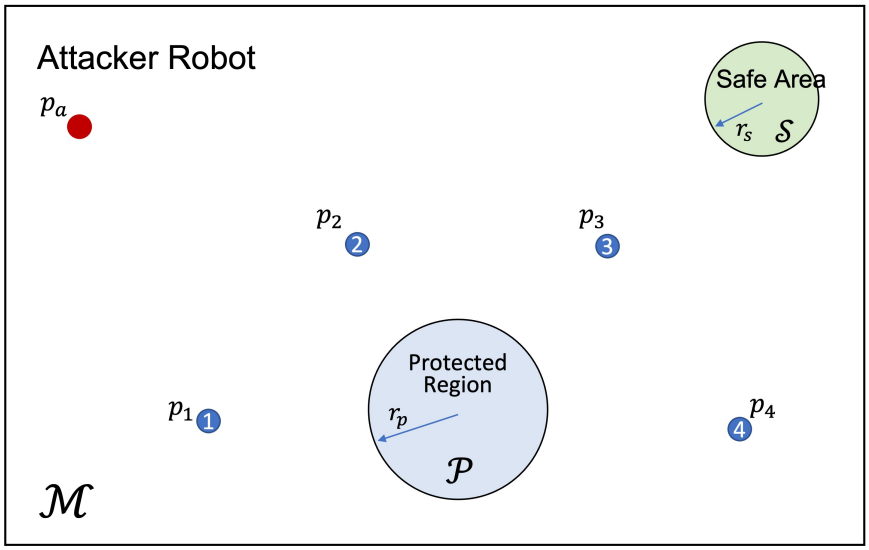


Figure 7.1: An example of the environment that is considered in this work.

where $u_{\text{attack,mag}}$ is the attacking force magnitude and $\vec{\boldsymbol{d}}_{ap}$ is the unit vector for force directionality, which attracts the attacker toward the center of the protected region. When the attacker encounters defender robots or obstacles during its approach of the protected region, the attacker leverages virtual springs for avoidance to navigate around the impediments.

7.3 Problem Formulation

The objective for defender robots is to ensure no intruding agents enter the protected region $\mathcal{P}$, which has a safety-critical importance that must remain un-encroached. The defender robots aim to block the advance of the intruder robot before it reaches the protected region. Formally, we write the problem we wish to solve as:

Problem 7.1 *Give the N_d defender robots in the set $\mathcal{D}$, design a decentralized control policy such that the defenders construct a blockade formation between the position of the attacker $\boldsymbol{p}_a$ and the region $\mathcal{P} \subset \mathcal{M}$ to prevent the attacker from entering the protected region, i.e.:*

$$\boldsymbol{p}_a \notin \mathcal{P}. \tag{7.5}$$

A second problem that we wish to solve is upon impeding the advance of the attacker, the defensive robots influence the attacker's motion toward the safe region $\mathcal{S}$ in the environment. Formally:

Problem 7.2 *Design a decentralized control policy such that defenders cooperatively utilize a control input $\overline{\boldsymbol{u}}_{d,i}$ to maneuver the blockade formation in a manner to steer the intruder away from the*

protected region $\mathcal{P}$ and towards the safe region $\mathcal{S} \subset \mathcal{M}$ where $\mathcal{S} \cap \mathcal{P} = \emptyset$ by:

$$\overline{\boldsymbol{u}}_{d,i} \rightarrow \boldsymbol{p}_a \in \mathcal{S} \tag{7.6}$$

where the defenders accomplish their task when the attacker's position lies within the safety region.

7.4 Cooperative Defensive Framework

In this section, we present our cooperative framework that aims to solve the problems defined in Section 7.3. We first introduce the full motion behavior of the intruder agent which aims to navigate into the known protected region while avoiding any obstruction (i.e., the defensive robots), then we present the control models of the defender robots to impede motion via barrier/wall construction and cooperative shepherding of the malicious intruder agent.

7.4.1 Attacker Motion

To begin, we first introduce the avoidance algorithm on-board the malicious intruder to avoid any obstruction in its path while navigate to its goal destination (i.e., the protected region). The avoidance control input of an attacker when navigating follows:

$$\boldsymbol{u}_{\text{avoid}} = \sum_{i \in \mathcal{N}_a} \kappa_r (l_{ai} - l_r^0) \vec{\boldsymbol{d}}_{ai} - \sum_{o \in \mathcal{O}_a} \kappa_o (l_{ao} - l_o^0) \vec{\boldsymbol{d}}_{ao} \tag{7.7}$$

where $l_{ai} = \|\boldsymbol{p}_a - \boldsymbol{p}_{\text{d},i}\|$ and $l_{ao} = \|\boldsymbol{p}_a - \boldsymbol{p}_o\|$ are distance between the attacker and defender robots and obstacles, respectively, with $\boldsymbol{p}_o \in \mathbb{R}^2$ representing the position of an oth obstacle. The variables κ_r and κ_o are virtual spring constants, while $\vec{\boldsymbol{d}}_{ai}$ and $\vec{\boldsymbol{d}}_{ao}$ denote the direction of repulsive forces on the attacker robot. The sets $\mathcal{N}_a \subseteq \mathcal{D}$ and $\mathcal{O}_a$ represent nearby defenders and obstacles that cause repulsive forces onto the attacker. The nearby defender robot set represents any defender $i \in \mathcal{D}$ that is within a reactive distance $l_r^0 \in \mathbb{R}_+$ and is computed by:

$$\mathcal{N}_a = \begin{cases} i \in \mathcal{N}_a, & \text{if } l_{ai} \leq l_r^0, \\ i \notin \mathcal{N}_a, & \text{otherwise}, \end{cases} \tag{7.8}$$

and similarly, the set of nearby obstacles is found by:

$$\mathcal{O}_a = \begin{cases} o \in \mathcal{O}_a, & \text{if } l_{ao} \leq l_o^0, \\ o \notin \mathcal{O}_a, & \text{otherwise}, \end{cases} \tag{7.9}$$

for any obstacle $o \in \mathcal{O}$ within a distance $l_o^0 \in \mathbb{R}_+$.

The overall control input for the attacker robot is the summation of the attack vector and any avoidance maneuvering by

$$\boldsymbol{u}_a = \boldsymbol{u}_{\text{attack}} + \boldsymbol{u}_{\text{avoid}}. \tag{7.10}$$

Next, we introduce the motion models that the defender robots follow in order to prevent the intruder robot from entering the protected region $\mathcal{P}$.

7.4.2 Defensive Barrier Construction

Upon detection of a malicious intruder within the environment, the defensive robots in the set $\mathcal{D}$ begin a cooperative control behavior to intercept the incoming attacker before it reaches the protected region. The objective is to form a defensive line (i.e., barrier) to stop the motion of the attack towards the protected region. This defensive line formation is perpendicular to the attacker to protected region vector $\vec{\boldsymbol{d}}_{ap}$, denoted by the line $\vec{\boldsymbol{d}}_{ap}[\perp]$, with separation $\delta_d > 0$ between defenders as illustrated in Fig. 7.2.

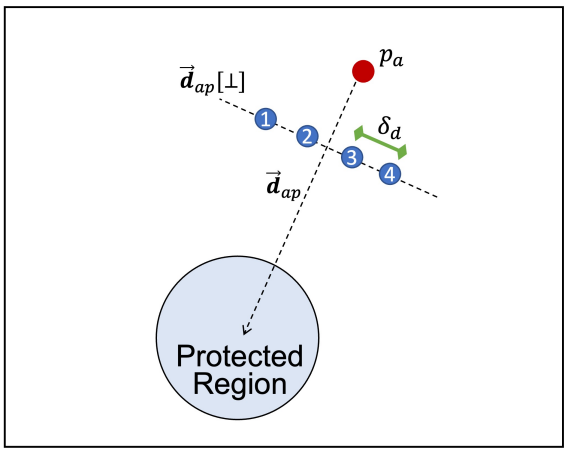


Figure 7.2: An example of a desirable defensive line that is located between the attacker position and protected region.

The point of intersection of the vectors $\vec{\boldsymbol{d}}_{ap}[\perp]$ and $\vec{\boldsymbol{d}}_{ap}$ is represented as a central defensive position $\boldsymbol{p}^{\text{def}}$. In this subsection, we characterize how defensive robots align themselves with respect to this defensive position to block an incoming attacker; however, we discuss how the defensive position $\boldsymbol{p}_{\text{def}}$ is computed in the following subsection (Section 7.5.3).

In this work, we assume the detection of an incoming attacker is discovered by the defender robots scattered throughout the environment. Given a limited sensing range $\delta_r > 0$ on-board the defender robots, the attacker is detected once it comes within sensing range of any robot i when the following is satisfied

$$\|\boldsymbol{p}_{\text{a}} - \boldsymbol{p}_{\text{d},i}\| \leq \delta_r, \quad \forall i \in \mathcal{D}. \tag{7.11}$$

Once the attacker has been detected, the observing defender robot i communicates the location of the attacker to the remaining robots. The initial robot that detects the attacker (i.e., the defender with the shortest distance to the attacker) converges to the position $\boldsymbol{p}^{\text{def}}$ to intercept the attacker. The remaining robots converge toward positions on the line with a separation distance δ_d from any other defensive robots. The location chosen along the defensive line is dependent on their current position with respect to the locations of the attacker and protected region (i.e., to join along the left or right side of the defensive line). An example of this choice of defensive positioning is depicted in Fig. 7.3 where the robot selects the correct positioning along the defensive line when joining.

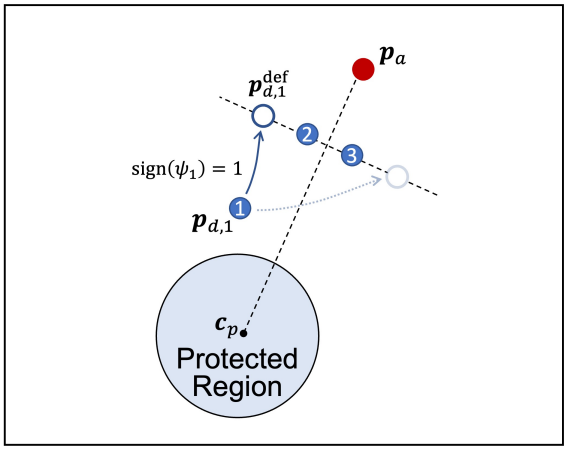


Figure 7.3: An example of a desirable defensive line that is located between the attacker position and protected region.

A defensive robot i determines whether to move to the left or right side of the line by first computing the decision variable ψ_i by:

$$\psi_i = (p_{\text{ax}} - c_{\text{px}})(p_{\text{dy},i} - c_{\text{py}}) - (p_{\text{ay}} - c_{\text{py}})(p_{\text{dx},i} - c_{\text{px}}) \tag{7.12}$$

where $(p_{\text{ax}}, p_{\text{ay}})$, $(c_{\text{px}}, c_{\text{py}})$, and $(p_{\text{dx},i}, p_{\text{dy},i})$ are the (x, y) position coordinates of the attacker, central position of the protected region, and the position of the ith defensive robot. Then, the positioning along the line is determined by:

$$\boldsymbol{p}_{\text{d},i}^{\text{def}} = \begin{cases} \boldsymbol{p}_c^{\text{def}} - \delta_d(\theta), & \text{if } \text{sign}(\psi_i) = 1, \\ \boldsymbol{p}_c^{\text{def}} + \delta_d(\theta), & \text{otherwise}, \end{cases} \tag{7.13}$$

to choose whether to move to the left side of the line (i.e., $\text{sign}(\psi_i) = 1$) or to the right (i.e., $\text{sign}(\psi_i) = -1$). If two or more defensive robots are tasked to navigate to the either side of the defensive line, then the agents perform a linear assignment problem to decide which agent moves to available positions along the line.

7.4.3 Attacker Tracking

Here we discuss the control behavior performed by each defensive robot contained on the defensive line $\mathcal{D}_L$ to maintain desired positions between the attacker and protected region. The objective is to sustain defensive positioning such that the line/barrier is in an ideal location (i.e., directly between the attacker and protected region). To make this determination, the "center of mass" of the defensive robots or also the average positions denoted by the central defensive position $\boldsymbol{p}_{\mathrm{c}}^{\mathrm{def}}$ is directly between the attacker and the center of the protected region. More specifically, if l_{ac_p} denotes the line from the attacker ($\boldsymbol{p}_{\mathrm{a}}$) and the center of the protected region ($\boldsymbol{c}_p$), then the desired central defensive position lies on l_{ac_p}, i.e., $\boldsymbol{p}_{\mathrm{c}}^{\mathrm{def}} \in l_{\mathrm{ac}_p}$. From the central defensive position, the respective defensive positions $\boldsymbol{p}_{\mathrm{d},i}^{\mathrm{def}}$ where $i \in \mathcal{D}_L$ are located accordingly given separation distance δ_d with respect to the central position. The central defensive position is continuously updated as the attacker navigates toward the protected region.

The defensive robots leverage the central defensive position and the motion of the attacker to remain in ideal defensive line positioning. In effect, each robot $i \in \mathcal{D}_L$ follows a proportional-derivative (PD) control algorithm to enable for continuous tracking. The tracking control algorithm is expressed as:

$$\boldsymbol{u}_i = k_P(\boldsymbol{p}_{\mathrm{d},i}^{\mathrm{def}} - \boldsymbol{p}_{\mathrm{d},i}) + k_D(\dot{\boldsymbol{p}}_{\mathrm{a}} - \dot{\boldsymbol{p}}_{\mathrm{d},i}\Big[\frac{r_{\mathrm{a}}}{r_{\mathrm{d},i}}\Big])\vec{\boldsymbol{d}}_{ap}(\perp) \tag{7.14}$$

where $k_P > 0$ and $k_D > 0$ are proportional and derivative control parameter gains while $\dot{\boldsymbol{p}}_{\mathrm{d},i}$ and $\dot{\boldsymbol{p}}_{\mathrm{a}}$ are velocities of the attacker and ith defensive robot. The unit vector $\vec{\boldsymbol{d}}_{ap}(\perp)$ expresses that all velocities considered are tangential to the vector $\vec{\boldsymbol{d}}_{ap}$ (i.e., from attacker to protected region positions). Furthermore, we denote $\dot{\boldsymbol{p}}_{\mathrm{d},i}[\frac{r_{\mathrm{a}}}{r_{\mathrm{d},i}}]$ as the desired tangential velocity of the defensive robot which is a function of the position of the attacker and itself with respect to the protected region. As highlighted in Fig. X, the defensive robot requires a tracking velocity which is less than the attacker, as it is at a shorter radial distance to the protected region than the attacker. The desired tangential tracking velocity of the ith defender is:

$$\dot{\boldsymbol{p}}_{\mathrm{d},i} = \frac{\boldsymbol{r}_{\mathrm{d},i}}{\boldsymbol{r}_{\mathrm{a}}}\dot{\boldsymbol{p}}_{\mathrm{a}} \tag{7.15}$$

to maintain a radial velocity that enables the defender to track an ideal defensive position to block the attacker. In the next section, we will begin to model the overall *shepherding system* consisting of the cooperative defensive robots attempting to steer the intruder agent away from the protected region and toward the safe region in the environment.

7.5 Shepherding System

In this section, we first characterize the high-level shepherding system model followed by the low-level control model performed by each of the cooperative defensive robots to shepherd the intruder to the safe region.

7.5.1 Shepherding Model

We first represent the shepherding system dynamical model in the following state space form:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}_{\text{shep}} + \boldsymbol{D}u_{\text{attack,mag}} \tag{7.16}$$

where the shepherding state vector $\boldsymbol{x} \in \mathbb{R}^6$ is

$$\boldsymbol{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & \theta & \dot{\theta} \end{bmatrix}^{\mathsf{T}} \tag{7.17}$$

within the x-y coordinate frame and system matrices are described as:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ -\sin(\phi) & 0 \\ 0 & 0 \\ -\cos(\phi) & 0 \\ 0 & 0 \\ 0 & \beta \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} 0 \\ \cos(\theta) \\ 0 \\ \sin(\theta) \\ 0 \\ 0 \end{bmatrix} \tag{7.18}$$

such that ϕ is the angle of the defensive line, θ defines the angle (direction) of the attack force, and β is the moment of the defensive line. The control input vector to the system (7.16) is written as:

$$\boldsymbol{u}_{\text{shep}} = \begin{bmatrix} u_{\text{lin}} & u_{\text{ang}} \end{bmatrix}^{\mathsf{T}} \tag{7.19}$$

where u_{lin} and u_{ang} are linear and angular force control inputs, respectively. Next, we formalize the high-level closed-loop shepherding system control model.

7.5.2 High-level Shepherding Control Model

To control the shepherding system, we consider a reference tracking feedback controller

$$\boldsymbol{u}_{\text{shep}} = -\boldsymbol{K}_{\text{x}}\boldsymbol{x}_{\text{e}} - \boldsymbol{u}_{\text{attack}} + \boldsymbol{K}_{\text{r}}\boldsymbol{x}_{\text{ref}} \tag{7.20}$$

where $\boldsymbol{x}_{\text{e}} = \boldsymbol{x}_{\text{ref}} - \boldsymbol{x}_{\text{a}}$ denotes the state tracking error of the attacker with respect to a reference state $\boldsymbol{x}_{\text{ref}}$ to follow. The matrices $\boldsymbol{K}_{\text{x}}$ and $\boldsymbol{K}_{\text{r}}$ represent the state feedback control gain and reference

gain. Next, by combining (7.20) into (7.16), we can represent the closed-loop shepherding control system as:

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_{\mathrm{x}})\boldsymbol{x}_{\mathrm{e}} + \boldsymbol{B}\boldsymbol{K}_{\mathrm{r}}\boldsymbol{x}_{\mathrm{ref}} \tag{7.21}$$

to shepherd the attacker agent along the desired reference state. We first make the assumption that the defensive robots are able to generate a feasible trajectory $\boldsymbol{x}_{\mathrm{ref}}$ for the shepherded attacker can follow. This is due to constraints in the shepherding control input based on limitations of the defensive robot capabilities and forces generated by the attacker's on-board obstacle avoidance algorithm. Given these limitations, we assume the shepherding control inputs is constrained such that $\boldsymbol{u}_{\mathrm{shep}} \leq \boldsymbol{u}_{\max}$. In the next subsection, we will discuss how we leverage the defensive robot positions to generate the linear and angular control components of the shepherding control input.

7.5.3 Defensive Robot Positioning for Intruder Steering

The objective of the defensive robots, after impeding the motion of the attacker, is to position themselves in a way to influence the attacker's motion away from the protected region and into the safe region. To enable this behavior, the defensive robot positioning is placed in a desirable location with respect to the attacker as well as a steering angle ϕ_{steer} of the defensive line to guide the attacker in a desired direction to generate desirable shepherding control inputs.

This desired motion to steer the attacker away is constructed from the desired shepherding force $\boldsymbol{u}_{\mathrm{shep}}$ which counteracts both the attack force $\boldsymbol{u}_{\mathrm{attack}}$ and utilizes the force computed from the on-board obstacle avoidance algorithm $\boldsymbol{u}_{\mathrm{avoid}}$. The defensive line is positioned in a specific manner to counteract the attack input $\boldsymbol{u}_{\mathrm{attack}}$ and by leveraging the potential functions from the on-board obstacle avoidance algorithm of the attacker. More specifically, the defenders use the repulsive force from the attacker's avoidance algorithm to their advantage to coerce its motion in a desired manner (i.e., toward the safe region). As previously described in (7.19), the defenders position themselves within the defensive line to satisfy the linear u_{lin} and angular u_{ang} shepherding control inputs, respectively.

We begin with the defensive line steering for the angular input component to the shepherding system. The goal is to align the defensive line such that it is perpendicular to the direction of the desired shepherding force direction (to simultaneously create a defensive barrier and to influence the attacker motion via its avoidance algorithm). To this end, we compute a steering control input for the defensive robots to correctly align themselves with respect to the attacker to satisfy u_{ang}. The steering control input corrects any error between the desired versus the current angle of the defensive line. Formally, the angular control input is:

$$u_{\mathrm{steer}} = k_{\mathrm{steer}}\Big(\|\boldsymbol{u}_{\mathrm{shep}}\|\vec{\boldsymbol{d}}_{\mathrm{shep}} - \big(\psi + \frac{\pi}{2}\big)\Big) \tag{7.22}$$

where $\|\boldsymbol{u}_{\text{shep}}\|\vec{\boldsymbol{d}}_{\text{shep}}$ denotes the direction of the desired angle to influence the attacker's motion, k_{steer} is a control gain, and $(\boldsymbol{\psi} + \frac{\pi}{2})$ is the current angle of the defensive line (offset by $\frac{\pi}{2}$ to account for the desired defensive line positioning being perpendicular).

Next, we compute the central location of the defensive line by reverse engineering the obstacle avoidance algorithm to satisfy u_{lin}. In terms of positions of the defenders and attacker, with the known potential field for obstacle avoidance, the resultant *linear* shepherding control input is computed in general terms by:

$$u_{\text{lin}} = \sum_{i=1}^{|\mathcal{D}_L|} k_{\text{lin}} \|\boldsymbol{p}_{\text{d},i} - \boldsymbol{p}_{\text{a}}\| \vec{\boldsymbol{d}}_{ia} \tag{7.23}$$

where $k_{\text{lin}} > 0$ is the parameter gain for avoidance and $\vec{\boldsymbol{d}}_{ia}$ is the unit vector denoting direction of forces from the ith defender to the attacker. We can approximate that the defensive line is in proper symmetrical position in obstruction of the attacker's motion, we can then rewrite (7.23) in terms of the central defensive position as:

$$u_{\text{lin}} = \sum_{i=1}^{|\mathcal{D}_L|} k_{\text{lin}} \|\boldsymbol{p}_{\text{c}}^{\text{def}} - \boldsymbol{p}_{\text{a}}\| \cos(\psi_{ia}) \tag{7.24}$$

where $\cos(\psi_{ia})$ denotes the difference between the direction of the resultant input vector and the vector between the ith defender and the attacker. To choose a proper defensive positioning of all defensive robots, we solve for a desired central defensive position by:

$$\|\boldsymbol{p}_{\text{c}}^{\text{def}} - \boldsymbol{p}_{\text{a}}\| = \frac{u_{\text{lin}}}{k_{\text{lin}} \sum_{i=1}^{|\mathcal{D}_L|} \cos(\psi_{ia})} \tag{7.25}$$

where $\|\boldsymbol{p}_{\text{c}}^{\text{def}} - \boldsymbol{p}_{\text{a}}\| \in \mathbb{R}_+$ is a scalar separation distance between the attacker and central defensive position. The location of the central defensive position is placed at:

$$\boldsymbol{p}_{\text{c}}^{\text{def}} = \boldsymbol{p}_{\text{a}} + \|\boldsymbol{p}_{\text{c}}^{\text{def}} - \boldsymbol{p}_{\text{a}}\| \vec{\boldsymbol{d}}_{ap} \tag{7.26}$$

With the central defensive position computed, the robots are able to position themselves with respect to this location and utilize it as the center of the defensive line. This allows the defensive robots to align themselves between the intruder and protected region, while also influencing the motion of the attacker in a desirable manner.

7.6 Simulation Results

Our preliminary results for our approach have been implemented in MATLAB on cooperative swarms of $N_d = 3$ and $N_d = 4$ defensive robots in a 2-dimensional environment $\mathcal{M}$. For each of

the simulation case studies, a malicious intruder aims to navigate into a protected region $\mathcal{P}$ from an unknown location within the environment from the perspective of the defensive robots. The objective of the defensive robots in the set $\mathcal{D}$ is to intercept the intruder and impede its motion before it enters the protected region by constructing a defensive barrier. Then, the defensive robots cooperatively work together to shepherd the intruder to a safe region $\mathcal{S}$ located away from the protected region. The multi-robot system operation is considered successful if the protected region remains unimpeded and the intruder is guided into the safe region.

Our first case study is presented in Fig. 7.4, where we have considered $N_d = 3$ defensive robots to complete the operation. Initial conditions of the simulation case study are shown in Fig. 7.4(a), and the intruder interception phase shown in Fig. 7.4(b) where defensive robot $i = 1$ first performs the engagement. In Fig. 7.4(c), the defensive robots have constructed a defensive barrier to impede motion and shepherd the intruder along a desired trajectory. Finally, the intruder has been successfully guided to the safe region in Fig. 7.4(d) to accomplish the mission.

Our second case study is represented in Fig. 7.5, where this time we consider $N_d = 4$ defensive robots. In Fig. 7.5(b), we see two of the defender robots have intercepted the intruder and have begun constructing the defensive barrier to impede intruder motion. During this time, two more robots have not yet joined the defensive barrier, but are navigating toward defensive positions located along the line. As shown in Figs. 7.5(c) and 7.5(d), the four defensive robots successfully shepherd the intruder along a trajectory to the safe region.

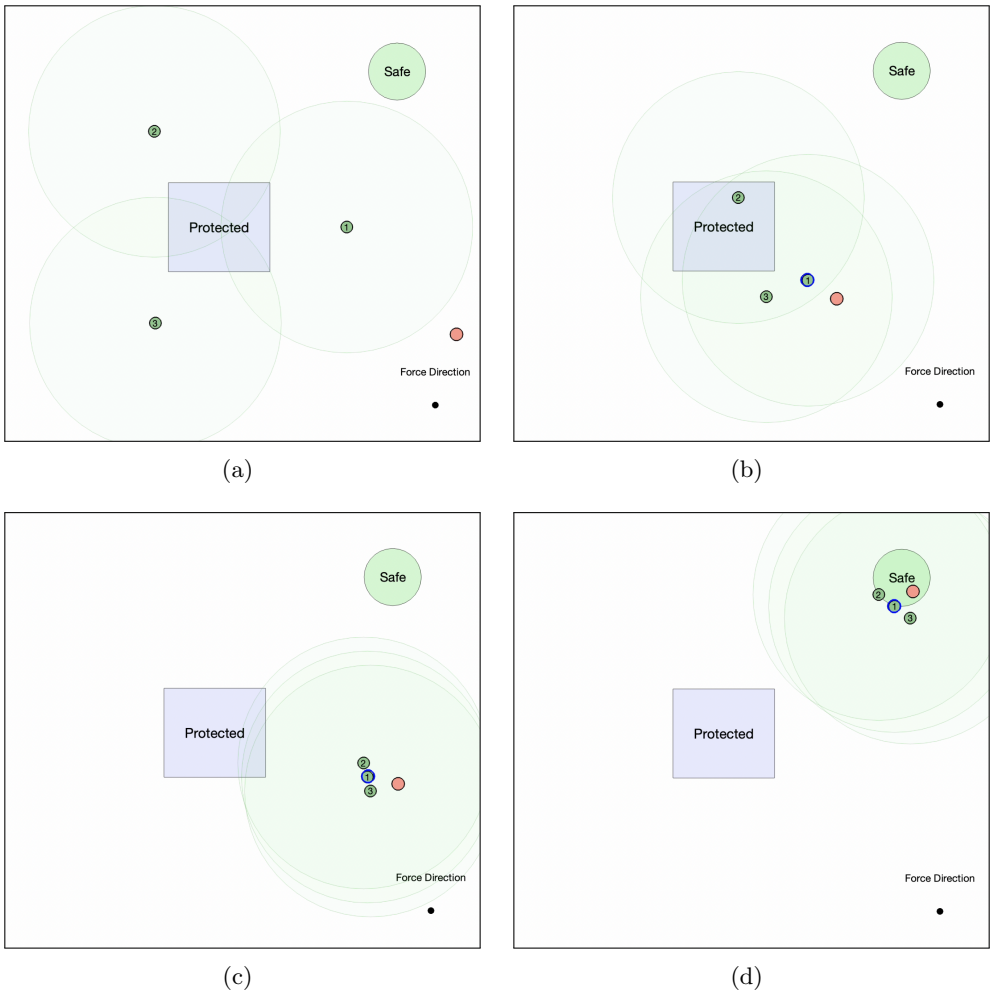


Figure 7.4: A multi-robot system of $N_d = 3$ agents (small green disks) leveraging our defensive framework to block the motion of a malicious intruder (red disk) then shepherd it to a safe region in the environment.

7.7 Discussion

In this chapter, we have introduced our current defensive robotic team framework to block malicious intruders from entering protected regions within the environment. We have devised a cooperative control strategy for the defensive robot team to: 1) impede the motion of a intruder to prevent it from entering a known protected region, and 2) cooperative strategy to shepherding the maliciously behaving intruder to a safe region within the environment. Our cooperative framework enables the defensive robot team to construct a defensive barrier/wall to both impede motion and shepherd the intruder in a desirable manner. Unique to the related literature in this field, we assume that the intruder agents continue to behave in a malicious manner even after being engaged by the cooperative shepherding robots. To highlight our work on this cooperative framework thus far, we

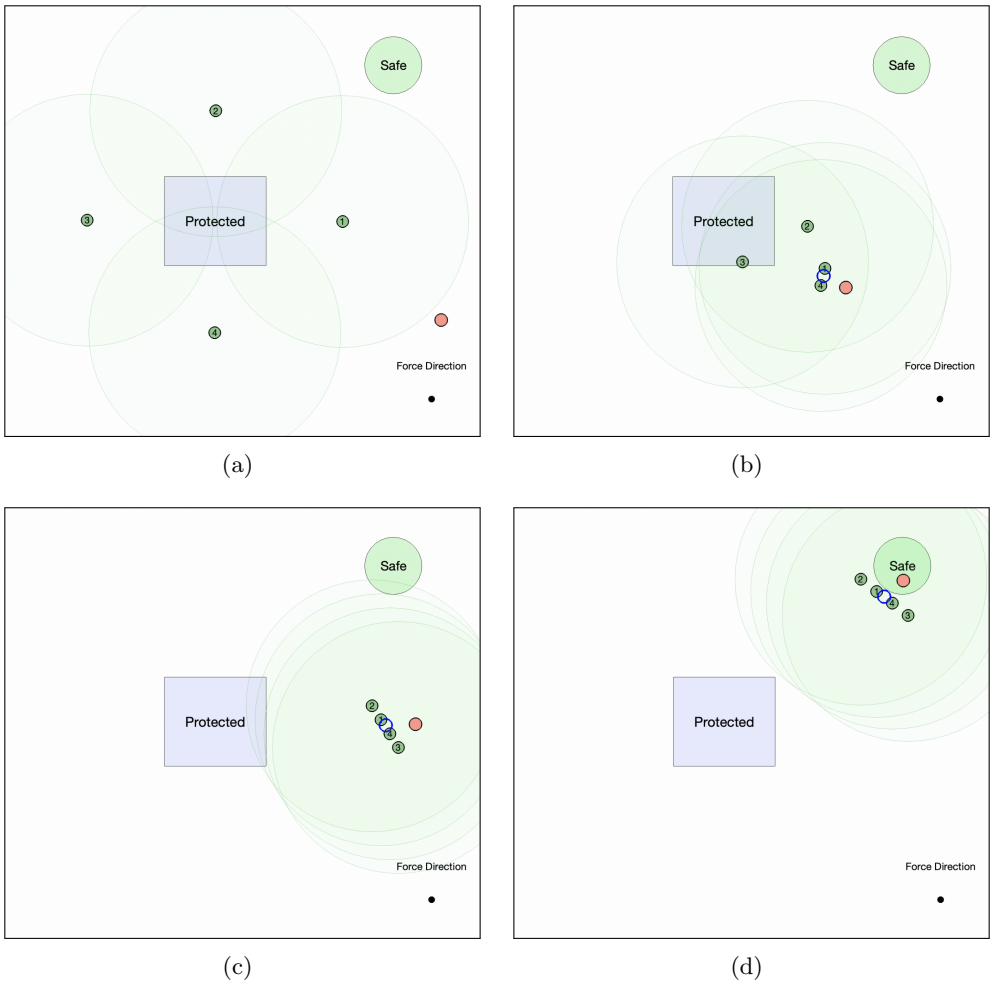


Figure 7.5: A multi-robot system of $N_d = 4$ agents leveraging our defensive framework to block the motion of a malicious intruder (red disk) then shepherd it to a safe region in the environment.

present multiple simulations to show the effectiveness of our strategy to block and shepherd the intruding agent.

7.7.1 Future Directions

The direction of the work presented in this chapter can go in a multitude of ways. To begin, the first task we wish to find a solution for is proving controllability/stability of the shepherding portion of our framework. The goal is to show that based on the positioning of the defensive robots, we are able control the intruder agent along a feasible desired trajectory to shepherd the intruder to a final destination (i.e., the safe region). An idea to which came to mind is treating the shepherding agents (i.e., both the cooperative defensive robots and malicious intruder) as a single combined system, where the positioning of the defensive robots with respect to the intruder acts as a "control

input" to the shepherding system. Furthermore, the attack vector and obstacle avoidance algorithm on-board the intruder are assumed to still be active while engaged by the defenders, are treated as disturbances to the system. To prove stability of the overall system, we want to use a similar theory in quadrotor control (such as in [63]) to control the shepherding system in a 2-dimensional environment.

Another aspect we wish to investigate in the future is how the number of engaged defender robots affect the shepherding performance of the malicious attacker. For example, questions that we could pose are, can as few as two defender robots be able to block and shepherd the attacker (assuming similar dynamical capabilities) to the safe region? Can we do this with one defender? Or, when does the effectiveness of an increasing number of defender robots begin to level off, such that having more defenders does not improve barrier/shepherding performance? This leads us to another future consideration of the effectiveness of our framework when more than one attacker is attempting to navigate into the protected region. If we have theoretical guarantees on the number of defenders it takes to shepherd a single intruder, then we can provide guarantees on the number of attacker agents our defender robot set $\mathcal{D}$ can handle when using our cooperative framework.

As of now, we have preliminary simulation results in MATLAB to highlight our progress thus far. However, our future plans include implementing this framework on a team of defensive unmanned ground robots in a lab environment. To make for interesting case studies, we want to include experiment results where the attacker/intruder agent behaves autonomously according to its combined attack and avoidance force models (7.10), and also a case where the attacker is manually controlled by a human with the same objective of gaining access into the protected region.

Part IV

Epilogue

Chapter 8

Conclusions and Future Work

In this chapter, we will conclude the dissertation with an overview of what we have accomplished and learned, followed by a discussion of real-world applications for this work and also any possible directions we could take for future work to build on what we have achieved thus far.

8.1 Conclusions

In this dissertation, we have presented frameworks for detection and recovery to maintain resilience to cyber attacks and faults that may occur on single- and multi-robot systems. The foundation of this work resides upon randomness-based monitoring for attack detection by leveraging residual-based detection schemes. The primary contribution of our randomness-based techniques is to monitor for randomness-based inconsistencies of measurement residuals when performing sensor attack detection. Our techniques have improved upon state-of-the-art residual-based methods, as they are designed to discover stealthy sensor attacks that intentionally hide within noise profiles on autonomous robots. Previous methods typically monitor whether the residual magnitudes follow an expected distribution, while our techniques monitor for pattern inconsistencies within the residual sequence that may be caused by false data injections to sensor measurements. We then carried over the randomness-based techniques to provide greater resiliency in multi-robot systems. More specifically, with our detection frameworks implemented within multi-robot systems, individual robots can determine if information being received from other robots is behaving consistently based on an expected behavior. With this implemented, we also provide detection capabilities to stealthy man-in-the-middle attacks on communication broadcasts exchanged between robots, where attackers can potentially intercept broadcast information and replace it with plausible data that can deceive traditional detection schemes. Upon detection of cyber attacks, the impacted robots are isolated and removed from the multi-robot system through network reconfiguration to allow for operations to continue performing in a resilient manner. Furthermore, we have developed a framework such that robots are able to safely exchange safety-critical information with other robots via hidden signatures based on hidden

motion models. This is beneficial because safety-critical information (e.g., a task) is no longer explicitly broadcast over communication channels that are vulnerable to interception by malicious eavesdroppers. With these aforementioned techniques, we have provided resilience to stealthy cyber attacks within multi-robot systems to allow for continued safety-critical task-based operations.

The final phase in this dissertation focused on recovery when autonomous systems experience on-board cyber attacks and faults. In order to recover agents within multi-robot systems, we introduced cooperative recovery frameworks to aid in re-localizing compromised robots that have unreliable on-board localization/positioning sensors. This improves upon our other multi-robot frameworks and other similar works in prior literature that utilize isolation and reconfiguration to maintain system resilience, as our method is designed to fully recover any compromised agent such that all robots complete missions in a safe manner. Differing from prior works, we assume that these formations of robots operate in unknown or landmark-free environments, such that localization by use of objects (i.e., landmarks) in the environment is not feasible. To cooperatively recover any compromised robots, we have developed a framework that relies on a control-based method such that the remaining robots coordinate their motion to provide mobile landmarks for re-localization. A second cooperative framework relies on leveraging received signal strength indication (RSSI) from information exchanges over radio frequency broadcasts for re-localization. By utilizing both RSSI-based measurements and the received position information, the neighboring robots are treated as mobile landmarks for localization. To recover a single autonomous robot system, we developed a detection and recovery framework to provide resiliency to cyber attacks and faults within on-board controllers that cause anomalous control behavior. Different from previous literature, we make the assumption that the anomalous behavior is triggered when information sent to the controller is in specific operating regions. We then developed a novel compensation scheme that is deigned to alter information sent to the controller to avoid any regions causing malicious behavior, thus allowing an autonomous system to maintain a desirable control signal to continue performing operations in a resilient fashion.

8.2 Discussion and Possible Future Work

Throughout this dissertation, we have presented extensive simulations with realistic system models and lab experiments using ground robots. The results to these relevant scenarios highlight the applicability of our techniques and frameworks to solve real-world problems within mobile autonomous systems, such as use in autonomous vehicles or military applications involving swarming robotics. Other interesting real-world applications that the frameworks introduced in this dissertation could be used in beyond mobile robotics include: attack detection in power grids (utilizing the material in Chapter 2), resilient control/motion in industrial robotics (using the framework in Chapter 5), and

ground target tracking (i.e., geolocation) using multiple video surveillance sensors/cameras (utilizing frameworks in Chapter 6). Moreover, we have demonstrated that randomness-based monitoring is a vital method for discovering cyber attacks that are designed to remain hidden within system noise profiles. While this area of research has been investigated for a number of years, attackers will always find security vulnerabilities or ways to evade intrusion detection schemes. While much of our results are based on theory and simulations, it would be interesting to investigate the effectiveness of these detection techniques work on more real-world systems with complex dynamics that operate outside of a lab environment. As an example, how well can these randomness-based detection mechanisms work when a system is experiencing external disturbances such as wind or ground surfaces that are not level. In our results shown throughout the dissertation, we assumed that we had a perfect model of our system, which includes the system dynamics and noise models. However, if these models are not completely accurate, the effectiveness of our detection mechanisms can be compromised. Future work can be implemented to test and/or characterize the effects on detection performance under various unknown uncertainties that may occur to a system during operations. Another strong assumption that we made in our attack detection frameworks is the attacker has full knowledge of the system model (e.g., dynamical, noise, estimator, and controller). An appealing idea would be to put effort into relaxing such a strong assumption to examine what an attacker's worst-case capabilities on a system are when less knowledge of the system is known.

We have also exhibited methods for autonomous multi-robot systems to recover from stealthy cyber attacks and faults by way of network configuration and cooperative recovery techniques within multi-robot systems. While the assumptions in this dissertation mainly covered spoofs/faults to on-board sensors and alterations of information within communication broadcasts, there are numerous other methods for attackers and system faults to compromise a multi-robot system. For example, in our cooperative recovery frameworks we assumed only on-board positioning sensors are compromised (due to faults/attacks) to impact localization capabilities; a scenario where a vehicle sustains damage which impacts both positioning sensing and system dynamics/actuation would potentially not be recoverable or could negatively influence system operations. This case would affect both localization and control performance due to an unknown change to the system model. More thought would be needed in the recovery process, such that cooperative vehicles aiding in recovery must provide re-localization support while also realizing that the compromised agents control performance has been diminished. Possible solutions to this problem could include altering the control behavior of the entire swarm (e.g., navigating at lower velocities) or re-plan operations by making changes to trajectories or mission tasks. Other issues that could arise include a complete loss of on-board positioning sensing (e.g., GPS) on all vehicles, losses in communication capability between agents, and unforeseen disturbances or increasingly difficult to navigate environments, all of

which could burden performance to the extent that the entire swarm operation can be compromised. Although recent literature has addressed many of these aforementioned issues provided specific assumptions and constraints to each of their frameworks, many holes and deficiencies still remain to prevent autonomous systems from fully recovering due to undesirable circumstances that may occur at runtime. However, within the realm of multi-robot systems, there are many possibilities for them to significantly improve our lives in the near future. Applications ranging from autonomous cars and transportation to smart cities, warehouse logistics, delivery services, defense weapon systems, and mobile sensor networks for natural disaster prediction all have much to gain in the future with the increased utilization of multiple robots/vehicles that can resiliently perform tasks.

In summary, the techniques presented in this dissertation have made it increasingly difficult for a malicious actor to negatively influence autonomous single- and multi-robot systems and have made these systems more resilient to unforeseen circumstances. However, we are only in the beginning phase of a world dominated by autonomous systems that are able to perform complex tasks to improve our daily lives. The detection and recovery techniques highlighted in this dissertation bring these autonomous systems one step closer to being completely safe and reliable.

Bibliography

[1] Shahrokh Akhlaghi, Ning Zhou, and Zhenyu Huang. "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation". In: *2017 IEEE Power Energy Society General Meeting*. 2017, pp. 1–5. DOI: 10.1109/PESGM.2017.8273755.

[2] C. Bai and V. Gupta. "On Kalman filtering in the presence of a compromised sensor: Fundamental performance bounds". In: *2014 American Control Conference*. June 2014, pp. 3029–3034. DOI: 10.1109/ACC.2014.6859155.

[3] Edward Balaban, Abhinav Saxena, Prasun Bansal, Kai F. Goebel, and Simon Curran. "Modeling, Detection, and Disambiguation of Sensor Faults for Aerospace Applications". In: *IEEE Sensors Journal* 9.12 (2009), pp. 1907–1917. DOI: 10.1109/JSEN.2009.2030284.

[4] Etienne Bertin, Jean-Michel Billiot, and Rémy Drouilhet. "Continuum percolation in the Gabriel graph". In: *Advances in Applied Probability* 34.4 (2002). DOI: 10.1239/aap/1037990948.

[5] N. Bezzo, B. Griffin, P. Cruz, J. Donahue, R. Fierro, and J. Wood. "A Cooperative Heterogeneous Mobile Wireless Mechatronic System". In: *IEEE/ASME Transactions on Mechatronics* 19.1 (2014), pp. 20–31.

[6] Nicola Bezzo, Patricio J. Cruz, Francesco Sorrentino, and Rafael Fierro. "Decentralized identification and control of networks of coupled mobile platforms through adaptive synchronization of chaos". In: *Physica D: Nonlinear Phenomena* 267 (2014). Evolving Dynamical Networks, pp. 94–103. ISSN: 0167-2789.

[7] Nicola Bezzo, Yuan Yan, Rafael Fierro, and Yasamin Mostofi. "A Decentralized Connectivity Strategy for Mobile Router Swarms". In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 4501–4506. ISSN: 1474-6670.

[8] Jahshan Bhatti and Todd E. Humphreys. "Hostile Control of Ships via False GPS Signals: Demonstration and Detection". In: *Navigation* 64.1 (2017), pp. 51–66. DOI: 10.1002/navi.183.

[9] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. New York: Elsevier, 1976.

[10] D. Brook and D. A. Evans. "An Approach to the Probability Distribution of Cusum Run Length". In: *Biometrika* 59.3 (1972), pp. 539–549. ISSN: 00063444.

[11] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Princeton University Press, 2009. ISBN: 978-0-691-14195-4.

[12] Camillo Cammarota. "The difference-sign runs length distribution in testing for serial independence". In: *Journal of Applied Statistics* 38.5 (2011), pp. 1033–1043. DOI: 10.1080/02664761003758984.

[13] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. "Attacks against Process Control Systems: Risk Assessment, Detection, and Response". In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ASIACCS '11. Hong Kong, China: Association for Computing Machinery, 2011, pp. 355–366. ISBN: 9781450305648. DOI: 10.1145/1966913.1966959.

[14] Luca Carlino, Di Jin, Michael Muma, and Abdelhak M Zoubir. "Robust distributed cooperative RSS-based localization for directed graphs in mixed LoS/NLoS environments". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–20.

[15] Luis C. Carrillo-Arce, Esha D. Nerurkar, José L. Gordillo, and Stergios I. Roumeliotis. "Decentralized multi-robot cooperative localization using covariance intersection". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 1412–1417. DOI: 10.1109/IROS.2013.6696534.

[16] J. Chen and Q. Zhu. "Resilient and decentralized control of multi-level cooperative mobile networks to maintain connectivity under adversarial environment". In: *2016 IEEE 55th Conference on Decision and Control*. 2016.

[17] Qifeng Chen, Yunhe Meng, and Jianjun Xing. "Shape control of spacecraft formation using a virtual spring-damper mesh". In: *Chinese Journal of Aeronautics* 29.6 (2016), pp. 1730–1739. ISSN: 1000-9361.

[18] Y. Chen, S. Kar, and J. M. F. Moura. "Resilient Distributed Estimation Through Adversary Detection". In: *IEEE Transactions on Signal Processing* 66.9 (2018), pp. 2455–2469.

[19] Daniel Claes, Frans Oliehoek, Hendrik Baier, Karl Tuyls, et al. "Decentralised online planning for multi-robot warehouse commissioning". In: *AAMAS'17: Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*. 2017, pp. 492–500.

[20] M. Conti, N. Dragoni, and V. Lesyk. "A Survey of Man In The Middle Attacks". In: *IEEE Communications Surveys Tutorials* 18.3 (2016), pp. 2027–2051.

[21] Georgios A Demetriou. "A Survey of Sensors for Localization of Unmanned Ground Vehicles (UGVs)." In: *IC-AI*. Citeseer. 2006, pp. 659–668.

[22] K. Derr and M. Manic. "Extended Virtual Spring Mesh (EVSM): The Distributed Self-Organizing Mobile Ad Hoc Network for Area Exploration". In: *IEEE Transactions on Industrial Electronics* 58.12 (2011), pp. 5424–5437. DOI: 10.1109/TIE.2011.2130492.

[23] Alex Devonport and Murat Arcak. "Data-Driven Reachable Set Computation using Adaptive Gaussian Process Classification and Monte Carlo Methods". In: *2020 American Control Conference (ACC)*. 2020, pp. 2629–2634. DOI: 10.23919/ACC45564.2020.9147918.

[24] Dany Dionne, Hannah Michalska, Yaakov Oshman, and Josef Shinar. "Novel Adaptive Generalized Likelihood Ratio Detector with Application to Maneuvering Target Tracking". In: *Journal of Guidance, Control, and Dynamics* 29.2 (2006), pp. 465–474. DOI: 10.2514/1.13447.

[25] Dany Dionne, Hannah Michalska, Yaakov Oshman, and Josef Shinar. "Novel Adaptive Generalized Likelihood Ratio Detector with Application to Maneuvering Target Tracking". In: *Journal of Guidance, Control, and Dynamics* 29.2 (2006), pp. 465–474. DOI: 10.2514/1.13447.

[26] M. Fachri, S. Juniastuti, S. M. S. Nugroho, and M. Hariadi. "Crowd evacuation using multi-agent system with leader-following behaviour". In: *2017 4th International Conference on New Media Studies*. 2017.

[27] Alberto Ferrari. *A note on sum and difference of correlated chi-squared variables*. 2019. arXiv: 1906.09982 [stat.ME]. URL: https://arxiv.org/abs/1906.09982.

[28] P.M. Frank. "Robust Model-Based Fault Detection in Dynamic Systems". In: *IFAC Proceedings Volumes* 25.4 (1992). IFAC Symposium on On-line Fault Detection and Supervision in the Chemical Process Industries, Newark, Delaware, 22-24 April, pp. 1–13. ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)50209-8.

[29] K. Ruben Gabriel and Robert R. Sokal. "A New Statistical Approach to Geographic Variation Analysis". In: *Systematic Biology* 18.3 (Sept. 1969), pp. 259–278. ISSN: 1063-5157. DOI: 10.2307/2412323.

[30] S I Gass and C M Harris. "Encyclopedia of Operations Research and Management Science". In: *Journal of the Operational Research Society* 48.7 (1997), pp. 759–760. DOI: 10.1057/palgrave.jors.2600798.

[31] Stephanie Gil, Swarun Kumar, Dina Katabi, and Daniela Rus. "Adaptive communication in multi-robot systems using directionality of signal strength". In: *The International Journal of Robotics Research* 34.7 (2015), pp. 946–968. DOI: 10.1177/0278364914567793.

[32] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. "A Survey of Physics-Based Attack Detection in Cyber-Physical Systems". In: *ACM Comput. Surv.* 51.4 (July 2018). ISSN: 0360-0300. DOI: 10.1145/3203245.

[33] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511841224.

[34] L. Guerrero-Bonilla, A. Prorok, and V. Kumar. "Formations for Resilient Robot Teams". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 841–848.

[35] Fredrik Gustafsson. *Adaptive filtering and change detection*. Vol. 1. John Wiley & Sons, Ltd, 2000. ISBN: 9780470841617. DOI: https://doi.org/10.1002/0470841613.ch1.

[36] David Hambling. *Drone Crash Due To GPS Interference In U.K. Raises Safety Questions*. 2020. URL: https://www.forbes.com (visited on 08/10/2020).

[37] Yazied A. Hasan, Arpit Garg, Satomi Sugaya, and Lydia Tapia. "Defensive Escort Teams for Navigation in Crowds via Multi-Agent Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5645–5652. DOI: 10.1109/LRA.2020.3010203.

[38] N. Hashemi, C. Murguia, and J. Ruths. "A Comparison of Stealthy Sensor Attacks on Control Systems". In: *2018 Annual American Control Conference (ACC)*. June 2018, pp. 973–979. DOI: 10.23919/ACC.2018.8431300.

[39] Kiyohiko Hattori, Naoki Tatebe, Toshinori Kagawa, Yasunori Owada, Lin Shan, Katsuhiro Temma, Kiyoshi Hamaguchi, and Keiki Takadama. "Deployment of Wireless Mesh Network Using RSSI-Based Swarm Robots". In: *Artif. Life Robot.* 21.4 (Dec. 2016), pp. 434–442. ISSN: 1433-5298. DOI: 10.1007/s10015-016-0300-y.

[40] Cheng-Chung Hsu, Syh-Shiuh Yeh, and Pau-Lo Hsu. "Particle filter design for mobile robot localization based on received signal strength indicator". In: *Transactions of the Institute of Measurement and Control* 38.11 (2016), pp. 1311–1319.

[41] G. Jing and L. Wang. "Multiagent Flocking With Angle-Based Formation Shape Control". In: *IEEE Transactions on Automatic Control* 65.2 (2020), pp. 817–823.

[42] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[43] Chitra R. Karanam, Belal Korany, and Yasamin Mostofi. "Magnitude-Based Angle-of-Arrival Estimation, Localization, and Target Tracking". In: *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2018, pp. 254–265. DOI: 10.1109/IPSN.2018.00053.

[44] A. Khazraei, H. Kebriaei, and F. R. Salmasi. "Replay attack detection in a multi agent system using stability analysis and loss effective watermarking". In: *2017 American Control Conference (ACC)*. 2017.

[45] Bernhard Klar. "A note on gamma difference distributions". In: *Journal of Statistical Computation and Simulation* 85.18 (2015), pp. 3708–3715. DOI: 10.1080/00949655.2014.996566.

[46] Fanxin Kong, Meng Xu, James Weimer, Oleg Sokolsky, and Insup Lee. "Cyber-Physical System Checkpointing and Recovery". In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. 2018, pp. 22–31. DOI: 10.1109/ICCPS.2018.00011.

[47] R. Kurazume, S. Nagata, and S. Hirose. "Cooperative positioning with multiple robots". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. 1994, 1250–1257 vol.2. DOI: 10.1109/ROBOT.1994.351315.

[48] Alexander B. Kurzhanski and Pravin Varaiya. "Ellipsoidal Techniques for Reachability Analysis". In: *Hybrid Systems: Computation and Control*. Ed. by Nancy Lynch and Bruce H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 202–214. ISBN: 978-3-540-46430-3.

[49] C. Kwon, W. Liu, and I. Hwang. "Security analysis for Cyber-Physical Systems against stealthy deception attacks". In: *2013 American Control Conference*. June 2013, pp. 3344–3349.

[50] Cheolhyeon Kwon, Scott Yantek, and Inseok Hwang. "Real-Time Safety Assessment of Unmanned Aircraft Systems Against Stealthy Cyber Attacks". In: *Journal of Aerospace Information Systems* 13.1 (2016), pp. 27–45. DOI: 10.2514/1.I010388.

[51] Cheolhyeon Kwon, Scott Yantek, and Inseok Hwang. "Real-Time Safety Assessment of Unmanned Aircraft Systems Against Stealthy Cyber Attacks". In: *Journal of Aerospace Information Systems* 13.1 (2016), pp. 27–45. DOI: 10.2514/1.I010388.

[52] Andreas Lawitzky, Anselm Nicklas, Dirk Wollherr, and Martin Buss. "Determining states of inevitable collision using reachability analysis". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 4142–4147. DOI: 10.1109/IROS.2014.6943146.

[53] S. Lee and B. Min. "Distributed Direction of Arrival Estimation-Aided Cyberattack Detection in Networked Multi-Robot Systems". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594465.

[54] Gaoqi Liang, Steven R. Weller, Junhua Zhao, Fengji Luo, and Zhao Yang Dong. "The 2015 Ukraine Blackout: Implications for False Data Injection Attacks". In: *IEEE Transactions on Power Systems* 32.4 (2017), pp. 3317–3318. DOI: 10.1109/TPWRS.2016.2631891.

[55] Fu Lin, Makan Fardad, and Mihailo R. Jovanovic. "Optimal Control of Vehicular Formations With Nearest Neighbor Interactions". In: *IEEE Transactions on Automatic Control* 57.9 (2012), pp. 2203–2218. DOI: 10.1109/TAC.2011.2181790.

[56] T. X. Lin, E. Yel, and N. Bezzo. "Energy-aware Persistent Control of Heterogeneous Robotic Systems". In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 2782–2787. DOI: 10.23919/ACC.2018.8431238.

[57] Fen Liu, Jing Liu, Yuqing Yin, Wenhan Wang, Donghai Hu, Pengpeng Chen, and Qiang Niu. "Survey on WiFi-based indoor positioning techniques". In: *IET Communications* 14.9 (2020), pp. 1372–1383. DOI: https://doi.org/10.1049/iet-com.2019.1059.

[58] R. Maeda, T. Endo, and F. Matsuno. "Decentralized Navigation for Heterogeneous Swarm Robots With Limited Field of View". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 904–911.

[59] Renan Maidana, Alexandre Amory, and Aurélio Salton. "Outdoor Localization System with Augmented State Extended Kalman Filter and Radio-Frequency Received Signal Strength". In: *2019 19th International Conference on Advanced Robotics (ICAR)*. 2019, pp. 604–609. DOI: 10.1109/ICAR46387.2019.8981637.

[60] Joaquin Mass-Sanchez, Erica Ruiz-Ibarra, J Cortez-González, Adolfo Espinoza-Ruiz, and Luis A Castro. "Weighted hyperbolic DV-hop positioning node localization algorithm in WSNs". In: *Wireless Personal Communications* 96.4 (2017), pp. 5011–5033.

[61] C. J. R. McCook and J. M. Esposito. "Flocking for Heterogeneous Robot Swarms: A Military Convoy Scenario". In: *2007 Thirty-Ninth Southeastern Symposium on System Theory*. 2007, pp. 26–31.

[62] Malika Meghjani and Gregory Dudek. "Search for a rendezvous with lost target at sea". In: *ICRA Workshop on Persistent Autonomy for Aquatic Robotics*. 2015.

[63] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[64] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas. "Coding sensor outputs for injection attacks detection". In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 5776–5781. DOI: 10.1109/CDC.2014.7040293.

[65] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas. "Coding sensor outputs for injection attacks detection". In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 5776–5781. DOI: 10.1109/CDC.2014.7040293.

[66] Charlie Miller and Chris Valasek. "Remote exploitation of an unaltered passenger vehicle". In: *Black Hat USA* 2015.S 91 (2015), p. 93.

[67] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. "False data injection attacks against state estimation in wireless sensor networks". In: *49th IEEE Conference on Decision and Control*. 2010, pp. 5967–5972. DOI: 10.1109/CDC.2010.5718158.

[68] Yilin Mo and B. Sinopoli. "False Data Injection Attacks in Control Systems". In: *First Workshop on Secure Control Systems*. 2010.

[69] Yilin Mo and Bruno Sinopoli. "On the Performance Degradation of Cyber-Physical Systems Under Stealthy Integrity Attacks". In: *IEEE Transactions on Automatic Control* 61.9 (2016), pp. 2618–2624. DOI: 10.1109/TAC.2015.2498708.

[70] C. Murguia and J. Ruths. "Characterization of a CUSUM model-based sensor attack detector". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. Dec. 2016, pp. 1303–1309. DOI: 10.1109/CDC.2016.7798446.

[71] C. Murguia and J. Ruths. "On model-based detectors for linear time-invariant stochastic systems under sensor attacks". In: *IET Control Theory Applications* 13.8 (2019), pp. 1051–1061. ISSN: 1751-8644. DOI: 10.1049/iet-cta.2018.5970.

[72] A. Nourian and S. Madnick. "A Systems Theoretic Approach to the Security Threats in Cyber Physical Systems Applied to Stuxnet". In: *IEEE Transactions on Dependable and Secure Computing* (2018).

[73] James J Nutaro. *Building software for simulation: theory and algorithms, with applications in C++*. John Wiley & Sons, 2011.

[74] R. Olfati-Saber, J. A. Fax, and R. M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1 (2007), pp. 215–233.

[75] Luis Oliveira, Hongbin Li, Luis Almeida, and Traian E Abrudan. "RSSI-based relative localisation for mobile robots". In: *Ad Hoc Networks* 13 (2014), pp. 321–335.

[76] Omur Ozel, Sean Weerakkody, and Bruno Sinopoli. "Physical watermarking for securing cyber physical systems via packet drop injections". In: *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE. 2017, pp. 271–276.

[77] E. S. Page. "Continuous Inspection Schemes". In: *Biometrika* 41.1/2 (1954), pp. 100–115. ISSN: 00063444.

[78] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas. "Robustness of attack-resilient state estimators". In: *2014 ACM/IEEE International Conference on Cyber-Physical Systems*. 2014.

[79] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. "Attack Detection and Identification in Cyber-Physical Systems". In: *IEEE Transactions on Automatic Control* 58.11 (2013), pp. 2715–2729. DOI: 10.1109/TAC.2013.2266831.

[80] Alyssa Pierson and Mac Schwager. "Controlling noncooperative herds with robotic herders". In: *IEEE Transactions on Robotics* 34.2 (2017), pp. 517–525.

[81] Anderson G. Pires, Douglas G. Macharet, and Luiz Chaimowicz. "Towards Cooperative Localization in Robotic Swarms". In: *Distributed Autonomous Robotic Systems*. Ed. by Nak-Young Chong and Young-Jo Cho. Tokyo: Springer Japan, 2016, pp. 105–118. ISBN: 978-4-431-55879-8.

[82] Anderson Grandi Pires, Douglas G. Macharet, and Luiz Chaimowicz. "Exploring heterogeneity for cooperative localization in Swarm Robotics". In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015, pp. 407–414. DOI: 10.1109/ICAR.2015.7251488.

[83] Junqi Qu, Xinguang Li, and Gongwu Sun. "Optimal Formation Configuration Analysis for Cooperative Localization System of Multi-AUV". In: *IEEE Access* 9 (2021), pp. 90702–90714. DOI: 10.1109/ACCESS.2021.3090514.

[84] T. R, C. Murguia, and J. Ruths. "Tuning Windowed Chi-Squared Detectors for Sensor Attacks". In: *2018 Annual American Control Conference (ACC)*. June 2018, pp. 1752–1757. DOI: 10.23919/ACC.2018.8431073.

[85] T. R., C. Murguia, and J. Ruths. "Tuning Windowed Chi-Squared Detectors for Sensor Attacks". In: *2018 Annual American Control Conference*. 2018. DOI: 10.23919/ACC.2018.8431073.

[86] W. Ren, R. W. Beard, and E. M. Atkins. "Information consensus in multivehicle cooperative control". In: *IEEE Control Systems Magazine* 27.2 (2007), pp. 71–82.

[87] Wei Ren. "Formation Keeping and Attitude Alignment for Multiple Spacecraft Through Local Interactions". In: *Journal of Guidance, Control, and Dynamics* 30.2 (2007), pp. 633–638. DOI: 10.2514/1.25629.

[88] V. Renganathan and T. Summers. "Spoof resilient coordination for distributed multi-robot systems". In: *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. 2017, pp. 135–141.

[89] Venkatraman Renganathan, Kaveh Fathian, Sleiman Safaoui, and Tyler Summers. "Spoof Resilient Coordination in Distributed and Robust Robotic Networks". In: *IEEE Transactions on Control Systems Technology* 30.2 (2022), pp. 803–810. DOI: 10.1109/TCST.2021.3063924.

[90] Sheldon M. Ross. *Introduction to Probability Models, Ninth Edition*. Orlando, FL, USA: Academic Press, Inc., 2006. ISBN: 0125980620.

[91] Sheldon M. Ross. *Introduction to Probability Models, Ninth Edition*. Orlando, FL, USA: Academic Press, Inc., 2006. ISBN: 0125980620.

[92] S.I. Roumeliotis and G.A. Bekey. "Collective localization: a distributed Kalman filter approach to localization of groups of mobile robots". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 3. 2000, 2958–2965 vol.3. DOI: 10.1109/ROBOT.2000.846477.

[93] K. Saulnier, D. Saldaña, A. Prorok, G. J. Pappas, and V. Kumar. "Resilient Flocking for Mobile Robot Teams". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1039–1046.

[94] Eugene Seneta. "Fitting the variance-gamma model to financial data". In: *Journal of Applied Probability* 41.A (2004), pp. 177–187. DOI: 10.1239/jap/1082552198.

[95] Yilun Shang. "Scaled consensus of switched multi-agent systems". In: *IMA Journal of Mathematical Control and Information* 36.2 (2018).

[96] S. Sharmin, S. I. Salim, and K. R. I. Sanim. "A Low-Cost Urban Search and Rescue Robot for Developing Countries". In: *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*. 2019, pp. 60–64.

[97] B. Shucker, T. D. Murphey, and J. K. Bennett. "Convergence-Preserving Switching for Topology-Dependent Decentralized Systems". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1405–1415.

[98] Brian Shucker. "Control of Distributed Robotic Macrosensors". PhD thesis. University of Colorado at Boulder, 2006.

[99] Sidney Siegel. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill New York, 1956, p. 312.

[100] Dan Simon. *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons, Jan. 2006. ISBN: 978-0-47-170858-2.

[101] Florian Sommer, Jürgen Dürrwang, and Reiner Kriesten. "Survey and Classification of Automotive Security Attacks". In: *Information* 10.4 (2019). ISSN: 2078-2489. DOI: 10.3390/info10040148.

[102] Paul Sundvall and Patric Jensfelt. "Fault detection for mobile robots using redundant positioning systems". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 3781–3786.

[103] Paula Tarrío, Ana M Bernardos, and José R Casar. "Weighted least squares techniques for improved received signal strength based localization". In: *Sensors* 11.9 (2011), pp. 8569–8592.

[104] D. van der Walle, B. Fidan, A. Sutton, Changbin Yu, and B. D. O. Anderson. "Non-hierarchical UAV formation control for surveillance tasks". In: *2008 American Control Conference*. 2008, pp. 777–782.

[105] A. Wald and J. Wolfowitz. "On a Test Whether Two Samples are from the Same Population". In: *Ann. Math. Statist.* 11.2 (1940). DOI: 10.1214/aoms/1177731909.

[106] Rong Wang, Junnan Du, Zhi Xiong, Xin Chen, and Jianye Liu. "Hierarchical Collaborative Navigation Method for UAV Swarm". In: *Journal of Aerospace Engineering* 34.1 (2021), pp. 1–14. DOI: 10.1061/(ASCE)AS.1943-5525.0001216.

[107] Y. Wang and H. Ishii. "Resilient Consensus Through Event-Based Communication". In: *IEEE Transactions on Control of Network Systems* 7.1 (2020), pp. 471–482.

[108] B. P. Welford. "Note on a Method for Calculating Corrected Sums of Squares and Products". In: *Technometrics* 4.3 (1962), pp. 419–420.

[109] Jihoon Yang, Haeyoung Lee, and Klaus Moessner. "Multilateration localization based on Singular Value Decomposition for 3D indoor positioning". In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2016, pp. 1–8. DOI: 10.1109/IPIN.2016.7743627.

[110] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. "A Survey of Indoor Localization Systems and Technologies". In: *IEEE Communications Surveys Tutorials* 21.3 (2019), pp. 2568–2599. DOI: 10.1109/COMST.2019.2911558.

[111] Amirreza Zaman, Behrouz Safarinejadian, and Wolfgang Birk. "Security analysis and fault detection against stealthy replay attacks". In: *International Journal of Control* 0.0 (2020), pp. 1–14. DOI: 10.1080/00207179.2020.1862917.

[112] M. Zamani and A. Pedro Aguiar. "Distributed localization of heterogeneous agents with uncertain relative measurements and communications". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 4702–4707. DOI: 10.1109/CDC.2015.7402952.

[113] W. Zeng and M. Chow. "Resilient Distributed Control in the Presence of Misbehaving Agents in Networked Control Systems". In: *IEEE Transactions on Cybernetics* 44.11 (2014), pp. 2038–2049.

[114] Dan Zhang, Gang Feng, Yang Shi, and Dipti Srinivasan. "Physical Safety and Cyber Security Analysis of Multi-Agent Systems: A Survey of Recent Advances". In: *IEEE/CAA Journal of Automatica Sinica* 8.2 (2021), pp. 319–333. DOI: 10.1109/JAS.2021.1003820.

[115] Dan Zhang, Gang Feng, Yang Shi, and Dipti Srinivasan. "Physical Safety and Cyber Security Analysis of Multi-Agent Systems: A Survey of Recent Advances". In: *IEEE/CAA Journal of Automatica Sinica* 8.2 (2021), pp. 319–333. DOI: 10.1109/JAS.2021.1003820.

[116] Dan Zhang, Gang Feng, Yang Shi, and Dipti Srinivasan. "Physical Safety and Cyber Security Analysis of Multi-Agent Systems: A Survey of Recent Advances". In: *IEEE/CAA Journal of Automatica Sinica* 8.2 (2021), pp. 319–333. DOI: 10.1109/JAS.2021.1003820.

[117] C. Zhao, J. He, and J. Chen. "Resilient Consensus with Mobile Detectors Against Malicious Attacks". In: *IEEE Transactions on Signal and Information Processing over Networks* 4.1 (2018), pp. 60–69.

[118] Binqi Zheng, Pengcheng Fu, Baoqing Li, and Xiaobing Yuan. "A robust adaptive unscented Kalman filter for nonlinear estimation with uncertain noise covariance". In: *Sensors* 18.3 (2018), p. 808.

[119] Y. Zheng and L. Wang. "Consensus of Switched Multiagent Systems". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 63.3 (2016), pp. 314–318.

[120] Yunru Zhu, Yuanshi Zheng, and Long Wang. "Containment control of switched multi-agent systems". In: *International Journal of Control* 88.12 (2015). DOI: 10.1080/00207179.2015.1050698.