Towards Trustworthy Swarming of Autonomous Vehicles

A Thesis

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Masters of Science (Computer Science)

by

Maggie Gates

August 2020

Approval Sheet

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

Masters of Science (Computer Science)

Maggie Gates

This thesis has been read and approved by the Examining Committee:

Nicola Bezzo, Advisor

Jack Davidson, Committee Chair

Yonghwi Kwon

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

August 2020

Abstract

Though research in Cyber Physical Systems (CPS) continues to push forward the capabilities of autonomous robotic systems in many military and civilian applications, there exist tasks that require the use of more than a single robot. Such tasks can be addressed with swarms of autonomous robots. There have been several studies on the topic of robot swarms in the last few years, however these studies are mostly centered around motion planning and coordination design. Swarm technology is improving rapidly and will soon find its way into society. With this possibility in the near future it becomes critical to consider security of these systems to guarantee safety. The work presented in this thesis develops around these premises and proposes a framework to detect and overcome cyber-attacks in robotic swarms. The proposed framework utilizes virtual physics and graph theory rules to create a decentralized mesh of robotic vehicles. By leveraging inter-swarm interactions between neighbors in the swarms we then propose a trust-based detection algorithm to identify spoofing and hijacking attacks. Finally, we propose a checkpointing protocol to recover the state of attacked victim nodes.

Acknowledgements

Thank you to my advisor, Dr. Nicola Bezzo. Thank you for your patience when I overextended myself and the research suffered, for always cheering on my successes whether they were related to our work or not, and for ensuring that I produced a work I am proud of. I couldn't have asked for a better advisor.

Thank you to Dr. Damian Lyons for welcoming me into your lab when I had no experience, for providing me with responsibility and your trust, and for introducing me to the world of robotics.

Thank you to Squires for all the help and advice over the years. When I think of what it means to be a hacker I think of you patiently explaining how registries work to me.
Thank you to Moose for pushing me to get good by simply being confident that I would be.

Thank you to my brother, Thomas for exposing me to adversity from a young age ;)
To my parents, thank you for everything. I grew up always knowing you were proud of me but I hope this makes you a little prouder.

And finally, I'd like to thank Vatsal. Thank you for reading every draft, looking at every minute image adjustment, and always asking me "Shouldn't you be working right now?". You support me in more ways than I can list. Also, degree++ you're behind by one, handsome.

Contents

List of Figures

Chapter 1

Introduction

Modern life is filled with Cyber Physical Systems (CPS). From the systems that manufacture the tools we use to the vehicles that drive us from place to place. As these systems become more advanced we move towards a reality of autonomous systems fulfilling these tasks. Furthermore, while there already exists numerous civilian and military applications for such autonomous systems, CPS research is now looking towards networks of systems because this interconnection, exchange of information, and cooperative behavior can have even further societal impact. Applications for these networks include smart grids to power and connect cities [29] as well as swarms of mobile autonomous vehicles to transport goods across countries and seas [48] or to perform data collection in un-mapped, potentially hazardous, territory [2].

With such wide and invasive applications it is critical that we focus on security measures for these CPS networks now, before they become deployed on a wide scale. In this thesis, we will first provide an overview of the landscape of CPS and the state of the art in CPS attacks and security measures. We will then narrow the focus to swarms of mobile autonomous vehicles. The landscape of attack vectors for such swarms is significant and there has already been quite a bit of work done on resilient motion planning and control, but not as much work has been done on detecting and mitigating attacks.

The technology behind swarms of mobile autonomous robots relies heavily on sensing and communication. In order for each node in the swarm to interact with the world around it as well the other nodes in the swarm, it has to be able to rely on both sensing and communicated data. Should these lines of communication be infiltrated (e.g. man in the middle attacks), the consequences could include the swarm being misdirected, damaged, or used to compromise the world around it. This is the main concern of this research: *How do we identify and mitigate communication and sensor-based attacks on robotic swarms?*

1.1 Contribution

The contribution of this thesis is four-fold:

Assessment and Review of State of the Art Research in CPS Security – In the first part of the thesis we will provide an overview of the state of the art on defense mechanisms of CPS security by examining the existing knowledge of attack vectors as well as the current work in security mechanisms and identify the gaps in research and the potential benefits and challenges of moving from single- to multi-vehicle systems.

Resilient Coordination of Multi-Vehicle Systems Under Cyber-Attack – Next, we will provide a process for resilient planning in autonomous multi-vehicle systems under attack by developing on a consensus-based algorithm for the swarm. The proposed controller utilizes the inter-swarm interactions and leverages virtual physics between neighboring nodes. These relationships are then used to build confidence/trust amongst the nodes in the swarm. This planning framework is built in such a way as to support discovery of sensor and measurement inconsistencies that might indicate a spoofing attack.

Trust Measurement and Attack Detection The main contribution of this work is the trust and detection algorithm. The attack detection algorithm proposed in this thesis is built to accommodate and account for noise inherent to the system. CPS are particularly noisy as they typically have sensor noise as well as process noise. However this noise, which is random and bounded, has an expected and known behavior. The general principle is that if we expect random noise within certain error bounds we can identify when there is abnormal behavior. Using these measurements of error we propose a method for building trust over time. Although the concept of trust over time is not a new one, we believe this application to be a novel and beneficial approach to swarm security. In the proposed method, each node builds trust for its neighbor in a decentralized fashion. This process involves each node comparing the communicated position of its neighbor node to a predicted position. Then, if those two sets are sufficiently similar that node builds positive trust for its neighbor, however if the two sets are sufficiently different, negative trust is built. Looking at the trend over time allows us to identify when a node is behaving unexpectedly as well as checkpoint the point in time in which the system started to misbehave.

Checkpointing and Recovery Finally this work contributes a checkpointing and verification method to rollback the state of a detected malicious vehicle and predict its state at the current time. This procedure is demonstrated to be effective when a vehicle has undergone a communication spoofing attack. The checkpointed

information can then be used to either verify the attack, or attempt to determine if the node has instead been victim to large amounts of natural disturbance or another attack.

1.2 Outline

In the first part of this thesis we will provide a survey on the current state of the art of CPS security research for mobile autonomous robots and multi-vehicle systems. We will then discuss cyber issues and their implications as society moves toward swarm technology and interconnected and autonomous systems. Finally, this thesis will attempt to solve some of these issues through the following framework: First, we will introduce a motion planning scheme for multi-vehicle systems that leverages virtual physics and graph theory to build a consensus and disperse the agents in the environment. Next, we will show that each node in the swarm can build trust over time for each neighboring node by predicting the future position of each neighbor at each time step and comparing those to the communicated positions. We will then show that we can use this trust as well as the information gained performing the comparison to detect when an attack has occurred. Finally, if enough memory exists, by storing previous swarm positions and rolling back to a time in which all nodes were trusted we can use this information to retrieve the state of the compromised vehicle.

The designed algorithm will be demonstrated through MATLAB simulations with in depth analysis on different case studies in which a victim node undergoes both communication-based and sensor-based attacks.

Chapter 2

Survey of Related Work and the State of the Art in CPS Security

Before discussing the research contributions of this thesis, we present an in depth review of Cyber Physical Systems (CPS) security in Section 2.1 as well as the current state of the art of research for mobile autonomous robots and multi-vehicle system security in Section 2.2. Specifically, we will review the architecture of single and multi-vehicle systems in order to discuss how they can be attacked. We will also review current research in autonomous mobile vehicles and swarms in order to highlight the foundational work relied on in this paper and identify the deficiencies in swarm security research that led to this thesis.

A note on terminology: Throughout this thesis the terms node, agent, robot, and vehicle are used interchangeably. Although they can be distinct concepts, for the use cases of this work they represent the same thing.

2.1 Survey of CPS Security

2.1.1 CPS Architecture

A cyber-physical system (CPS) is a class of computing system that refers to the integration of computation and communication with sensing and actuation in a way that allows the system to interact directly with the physical world. Some examples of these systems include robotics, autonomous vehicles, smart-homes, and industrial systems. For the purpose of this research the focus is on autonomous multi-vehicle systems.

Generally speaking, the architecture of a swarm of mobile autonomous vehicles from a CPS perspective can be represented by Fig. 2.1:

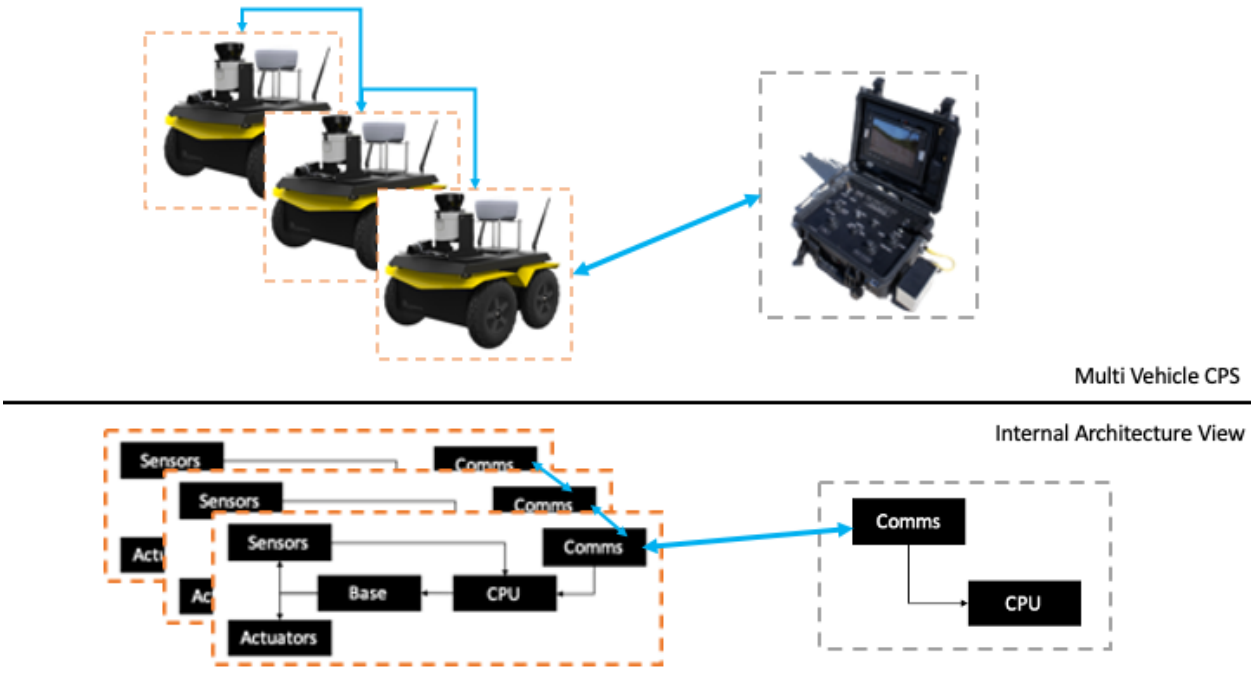


Figure 2.1: Multi Vehicle CPS Architecture

A swarm can contain N nodes and it can be either centralized or decentralized. Decentralized means each node performs its own computations and interacts with its neighbors in the swarm. In Fig.2.1 there is also a command unit, however unlike in a centralized deployment it does not perform all computations or issue directions to each node.

The boxes below the images represent the inner architecture of each system. The typical architecture of a mobile autonomous vehicle has a communication vector which receives and sends data to the other nodes in the swarm, the command unit, and potentially other remote mediums. The communications vector shares information with the CPU such as the states (e.g.,positions) of the neighboring nodes. This data gets factored into the calculations performed for path planning and decision making. In addition, the CPU receives data from any onboard sensors. The communicated data can include position information, obstacle and environment information, as well as various other system measurements. The base houses the sensors as well as the actuators which are responsible for interacting with the robots' environment.

The key takeaway from this architecture diagram is that the CPU interacts with both sensor data as well as communicated data. In this work we leverage the redundancy of those two vectors to compare position information shared through communication and position information calculated by the CPU using sensor data.

2.1.2 Types of Attacks

There exist 5 main attack surfaces on these systems:

1. Sensor Attacks: The attacker arbitrarily changes sensor data. A pertinent example of this would be GPS spoofing attacks [35]. In this example the attacker convinces the GPS sensing system on board the vehicle that it is located at point B when it is actually located at point A.

2. Actuator Attacks: The attacker can arbitrarily change actuator values [18]. An example of this attack would be if an attacker was able to cause a vehicle to go faster than intended by sending the wheels increased speed commands.

3. Communication Attacks: The attacker can change messages. These messages could be between the sensors and the controllers, between the controllers and actuators, or between different nodes in a swarm [26]. This last example will be the main focus of this thesis and we will return to it shortly.

4. Controller Attacks: The attacker can change the controller's parameters or code [7]. For example an attacker could change the controller code that tells the system to stop when it sees a stop sign or manipulate gains in the controller thus changing its behavior.

5. Environment attacks: The attacker can manipulate the environment. For example an attacker could put a fake stop sign in the middle of the road to get the vehicle to stop.[33]

Although there has been some interesting research on attacks and attack vectors in each of the five categories, we will focus primarily on communication attacks. We focus on this category in particular because it is the most common and often the most detrimental to the integrity of a swarm. One approach that has been studied for protecting communication between the nodes in a swarm is encryption. While there has been a great deal of research in encryption [42][50][17] this thesis focuses on the case where an attacker is able to bypass any security measures by, for example, spoofing sensor data prior to it being encrypted and sent. Although it is certainly possible to attack the actuators and controllers directly, these vectors often require physical access. Similarly, environment attacks require proximity to and knowledge of where the swarm is and/or will be in order to alter the environment in a way beneficial to the attacker. These constraints make them less common vectors of attack.

In addition to the above types of attacks, there are 5 main classes of CPS attacks which we define below.

1. Spoofing: An attacker creates data that is similar to what the system is expecting but modifies it. For example, an attacker sending fake GPS data [1].

2. Denial of Service: An attacker overwhelms the system with so many requests or so much data that the other components of the system are unable to communicate with it [51] [22]. For example sending a GPS position request with such frequency that another node in the swarm is unable to get a response to its own position request.

3. Replay Attack: An attacker records the contents of a message or interaction and replays it at a later date to replicate the behavior [30]. For example an attacker could record the authentication sequence involved with unlocking a car and then replay it at a later date to gain access to the car.

4. Man-in-the-Middle Attack: An attacker intercepts a message and changes the contents before forwarding it to the original recipient [12]. For example, an attacker intercepted a the GPS coordinates of one node being sent to another and changes the data to say that node is somewhere else.

5. Relay Attack: An attacker records the contents of a message or interaction between entity A and B and plays it for entity C hoping to recreate the behavior [11]. For example if the message were communicating a stop directive from A to B the attacker could relay this message to C in order to get it to stop as well.

Of the classes listed above this work focuses on communication and sensor spoofing attacks. We chose spoofing attacks because we believe them to be the most dangerous of the aforementioned five in the context of a swarm tasked with a goal. Spoofing attacks allow the attacker to control the movements of the victim node remotely and could result in the attacker gaining control of the swarm and its trajectory. Replay and relay attacks lack the level of control that a spoofing attack might have and a denial of service attack, while disruptive to the mission, is unlikely to result in the physical loss of the swarm or its nodes. This work could also be applied to man-in-the-middle attacks but we will consider it a subcategory of the more general spoofing attacks.

It is also important to note, that as CPS becomes more advanced, so too do the attacks. The types and classes of attacks listed above give a high-level description of the attack, however, we must also consider the skill with which the attack is implemented. The implementation of the attacks we are considering are coordinated and stealthy. We are expecting the attackers to be highly skilled and highly aware of the systems they are attacking so as to hide the true intention of their attack until it has already been performed.

The kinds of information a stealthy attacker might care about when performing an attack on a modern autonomous multi-vehicle system include the kinds of sensors on board the vehicles, what the sensor data payload looks like, with whom that data payload is being communicated, the error bounds of the sensors and actuators on board the system, and the environment the system is being deployed in. By knowing this

information the attacker can then implement their attack within the bounds of expectation, allowing them to blend in with the error and environment in order to remain undetected. In Section 3.1.1 we will outline the types of attacks our proposed framework addresses, as well as the information we are expecting the attacker to know about the system.

2.2 Literature Review

This literature review is broken into two main sections. First we will discuss studies on cyber security in CPS in Section 2.2.1. This section focuses on novel attack vectors in CPS and the effects these attacks have. This section is intended to summarize the attack vectors that are known as well as emphasize the importance of researching detection and prevention mechanisms. This leads us to Section 2.2.2. This section will cover detection mechanisms as well as resilient estimation and control algorithms. The intent of this section is to enumerate the current landscape of security in CPS. Finally, we will end this chapter with a brief discussion on what security looks like as autonomous mobile robot research moves from single to multi-vehicle applications in Section 2.3. In this section, we will briefly touch on some of the challenges with this shift as well as some of the benefits.

2.2.1 Studies on Cyber Security in CPS

When considering attack vectors in CPS there is a large diversity of systems and methods to consider. The attack vectors of a manufacturing robot are very different from the attack vectors of a boat or a car. We will touch briefly on all of the main systems, however, for our research we focus on attacks and disturbances from the perspective of swarms of autonomous vehicles.

The autonomous mobile vehicle that is getting the most general attention right now is the car. Although many different CPS systems are widely used, an autonomous car has the most impact for the everyday person. Thus, automotive attack surfaces is where we will begin the literature review. There have been several surveys that enumerate the attack surfaces of cars [5] [19] [28]. These surveys discuss both vulnerabilities when an attacker has physical access as well as attack vectors for remote attacks. The remote attack vectors that are highlighted are bluetooth, FM radio, cellular, and wifi.

Another example of a remote attack in a different class is a recent paper entitled "Phantom of the ADAS"[33]. In this work the authors demonstrated that using a drone and a projector, they could fool a car's main controller into perceiving a person in front of the car. As a result, they could force the car to come to a stop. They also demonstrated that they could control the car's trajectory by projecting a new path on the ground, as well as cause the car to speed up by projecting a fake speed limit sign.

Additional methods for disrupting mobile autonomous vehicles include attacking the vehicle sensors directly. Two such methods include incapacitating drones equipped with gyroscopes by using sound to interfere with its signal. Performing this attack in real-world experiments resulted in a test drone crashing. [44] Another method that demonstrated the power and feasibility of attacking system sensors targeted the inertial sensors of 25 types of devices including a drone. [45] Rather than interfere with the sensors as before, this method involved injecting malicious data and spoofing the sensors resulting in control of the device.

Drones have many more demonstrated attack vectors[21]. In addition to sensor attacks, drones have demonstrated vulnerabilities to their data communication and control communications. The consequences of these attack vectors can be significant. One method [38] demonstrated that by targeting the communications of a drone an attacker could perform drone hijacking, eavesdropping, and even tracking of individuals. These consequences are concerning given the military's interest in the operationalized use of drone.

Another relevant work that demonstrated sensor attacks showed the feasibility of a GPS spoofing attack on a yacht [35]. This work demonstrated the feasibility of spoofing the GPS signal of a boat to make the boat (and its crew) believe that it was 3 degrees off course. A small adjustment like this would be well within the bounds of normal error and would not alarm the crew, instead, as demonstrated in the experiment, the crew would simply make a small adjustment to the boat's trajectory to redirect it on course. The accumulation of constant GPS spoofing over the duration of an hour results in the boat moving well off course. As described in the paper, while this attack is not yet feasibly mainstream it is within the bounds of possibility and is one of the scenarios that inspired this thesis.

Finally, there has been a great deal of interest in adversarial machine learning as control theory moves toward integrating deep neural networks (DNN) for path planning and safety assurance. The principal behind these adversarial machine learning attacks is to introduce noise into the system in such a way that it confuses the decision of the DNN. These attacks have been exploited in particular for sign recognition [13] [43] by introducing noise or small modifications to images that cause the network to classify them incorrectly; one example is a stop sign being classified as a speed limit sign as demonstrated in [13].

2.2.2 Detection, Estimation, and Control

In addition to the work discussed above on novel attack methods there is quite a bit of work on detection measures for attacks and divergent sensor data, as well as resilient estimation and control algorithms. Below we discuss recent and novel approaches for both single and multi-vehicles systems.

We begin by discussing detection mechanisms for sensor attacks.

A key principle in cyber security is defense-in-depth, or redundancy. This is a principle that can be applied in multiple ways when considered from the CPS security perspective. One such way is redundancy of measurements. The idea of measurement redundancy has also been suggested for a resilient cruise controller [3]. One such method is for the cruise controller to utilize three different types of measurement for the same data. For example, if a cruise controller were to measure a vehicle's velocity based on GPS data, IMU data, and encoder data it could then constantly compare the values. Suppose the GPS data was being spoofed. In this case, the redundancy of the IMU and encoder data would aide in detecting the attack. The authors then proposed using a Kalman filter-based approach and a shielding procedure to weight data based on their consistency with a predefined model of a system to detect attacks on a sensor. Another similar approach uses Kalman filters for resiliency in order to filter out false data injections[32].

There is also some relevant work on networks of sensors and actuators. One area of particular interest to the CPS security field is infrastructure. The interest in the field is not spurred solely by the potential improvements and applications to automating daily life but also by the demonstration of what can happen when this networked infrastructure is attacked. The most well known demonstration of these effects was malware now known as Stuxnet. Stuxnet was a worm that targeted systems such as these and caused significant damage to scada systems in 2010. The effects of Stuxnet were significant and triggered a large push for research in to the security of infrastructural networks of sensors and actuators [25]. In systems such as these, often the sensors and actuators are working in conjunction and interacting with each other, similar to swarms of vehicles. A proposed security measure for this type of system is a method known as watermarking [31] [39]. This method involves injecting information into the system in order to authenticate systems data or dynamics.

Moving away from infrastructure, another redundancy based security method is swarm consensus. At a high level, the word consensus means "general agreement." When we apply that idea to swarm security we want the swarm to be in consensus on two topics:

1. The destination and the steps to be taken along the trajectory.

2. Whether or not a node in the swarm is under attack.

We will focus mostly on consensus on the former. It is worth noting that most of the work in this area is theoretical and often assumes knowledge of the attacked nodes in the swarm.

The recent work on consensus for swarm security can be partitioned into several categories, the first of which is consensus and innovation. We start here because it is the broadest topic. Consensus and innovation generally means: at each decision point, each node uses both swarm knowledge (consensus), and knowledge local to the node (innovation). One instance of this method is producing a thresholding value for each

node's innovation depending on expected number of nodes under attack [6]. Similar to the Kalman filter method mentioned above, this approach allows the impact of the node's input to be scaled down. In this case, however, the thresholding value is applied with less granularity, across the swarm rather than for each sensor. Another approach is to compare each node's innovation to the swarm consensus [8]. This approach flags any node's innovation that is significantly divergent from the consensus. In this way each node is being constantly evaluated on the swarm level and is able to flag an attack if and when it attempts to affect the node's relative position to the swarm. There has also been work that takes a combined approach [20]. The combined approach is a set of algorithms that includes a consensus step that weights each node's measurements depending on its neighbors' position information, as well as an innovation step where each node processes its own unweighted measurements.

The second method for resilient consensus based controllers is consensus using the Mean-Subsequence-Reduced (MSR) algorithm. This algorithm builds resiliency into the controller by ignoring values that are too large or too small[40] [9] [14] [36]. The algorithm works on a step by step, node by node basis. At each time step each node takes its measurement and then looks at a sorted list of its neighborhood's measurements. All measurements above or below the high and low cutoffs are ignored. Thus, the swarm reaches consensus by systematically disregarding divergent values. In this way the swarm ignores any updates to state that are beyond the bounds of what is expected and therefore what may be an attack.

MSR has mostly been studied in the case of networks that are completely connected. One work [24] showed conclusively that MSR-type algorithms require the network to form complete graphs in order to achieve consensus and that traditional connectivity is not enough. In its place, a concept called graph robustness was suggested. Graph robustness is an updated definition of the requirements of the connectedness of the graph in order to use MSR-like algorithms for consensus. This notion has been the topic of much research since its introduction and there is quite a bit of work on these MSR-like algorithms for robust swarms[10] [37][15].

A third similar method that relies on input control is proposed for persistent switching topology[46]. In this method inputs are constrained based on swarm size and number of attacked nodes. It is important to note that while the application of this algorithm is similar to the MSR algorithm, the way those thresholds are computed differ.

One drawback of many of the aforementioned resilient controller works is that they are too theoretical to be deployed in the real world. Whether that is due to the large amount of processing power required for the calculations or the reliance on regular inputs at predetermined time intervals. For this, event-triggered-updates for swarm consensus has been proposed[34]. At a high level, the event-triggered-update predetermines the conditions at which time each node updates its measurements and between those events keeps the input steady by providing sampled input. In this way, the processing requirements can be reduced as can the need

for regular timely updates. Two recent papers pair this principal with MSR-like algorithms for more efficient resilient consensus in swarms [49] [47].

The key idea behind all of aforementioned security mechanisms is maintaining the swarm despite noise or an attack, rather than identifying and removing an attack. This works well for point in time bursts of attacks but not as well for stealthy attacks. Furthermore, what is really missing is a usable framework for detecting an attack in the first place and responding to it before it affects the swarm.

2.3 Single vs Multi Vehicle Systems

Loss of precision is one of the main challenges in moving from single to multi vehicle systems. When a command is issued to a single vehicle, there are a handful of known factors that may cause that vehicle not to end up at the location specified. These factors include environmental obstacles, sensor error, and actuator error. However, as we move to a paradigm that involves multiple vehicles those factors increase in number and effect. No longer is the environment the only potential obstacle, but other nodes in the swarm are as well. In order to safely interact with each other, nodes in a swarm must maintain a rest length between each other. While this helps to prevent them from interfering with each other it can also lead to spreading, resulting in a loss of precision and a higher probability of collisions with the environment. The risk of spreading means we must consider more thoroughly how the nodes interact with each other and the environment safely. Furthermore, error becomes a much larger issue. In a single vehicle environment the error is set and we can accurately predict where the vehicle could end up as a result of that error. However, in a multi-vehicle environment not only does each vehicle come with actuator and sensor error, that error gets propagated through the swarm as those noisy measurements get communicated and used to calculate the paths of neighbor nodes.

The aforementioned errors can be challenging enough when considered on their own, however they become even more challenging when considered as a potential attack vector. With the loss of precision comes uncertainty. This large region of error and the possibility for spreading means that an attacker can augment positional data and remain fairly undetected as long as they stay somewhat within the bounds of the error. Over time were an attacker to add small values to the position of even just one node in the swarm, they could remain undetected while causing a significant change in the swarm's trajectory.

The challenges are significant hurdles, however, the move to multi-vehicle's brings with it benefits as well. Application benefits such as the ability to perform tasks that require more than one vehicle aside, there is also the benefit of redundancy. As discussed in the literature review, redundancy, also known as defense-in-depth, is a key principle of security. When we consider security mechanisms for mobile autonomous vehicles it is

easy to see that multiple vehicles provide multiple perspectives, measurements, and processing capabilities. This redundancy is a powerful tool when leveraged for anomaly detection as well as to increase the amount of data these systems have access to. On example of the benefit of redundancy outside from an application perspective is data collection use cases. Whether these MVS be used for mapping unsafe terrain or for aiding in missing persons cases, networked vehicles allow for data streams from multiple angles and perspectives.

Chapter 3

Attack Detection, Checkpointing, and Recovery in Multi-vehicle Systems

In this work we are interested in investigating cyber-security issues and providing resilience during multi-vehicle system operations. As pointed out in the review above, while a lot of work has been done on single vehicle systems, several challenges and opportunities exist when dealing with security in multi-vehicle systems. In this chapter we propose a resilient technique to coordinate swarms of autonomous vehicles while detecting and isolating compromised vehicles that have the intention of hijacking the entire system. Our framework leverages virtual physics interactions, graph theory, and knowledge about the system dynamics to detect and recover a compromised network.

We begin the chapter by discussing the type of attack vectors of interest in multi-vehicle systems in Section 3.1.1 followed by a formal definition of the problem in Section 3.1.2. Finally we present our trust-based resilient scheme in Section 3.2.

3.1 Preliminaries

3.1.1 Type of Attacks

The two categories of attacks we will demonstrate and test our framework against are communication-based attacks and sensor-based attacks. These two are similar in how they affect the swarm but distinct in how they affect the victim node. The effect is similar because the result is the propagation of false and/or malicious data throughout the swarm. In practice this looks like neighbor nodes of the victim node receiving incorrect position data from the victim node to be used in path planning calculations. The difference however, is that

in communication-based attacks the victim node knows where it is located, but is not properly communicating that information. In this attack an attacker resides somewhere between the victim node and the neighbor nodes with which it is communicating. Here, the attacker feeds malicious data to the neighbors while the victim node remains unaware. The result is that the victim node will continue to act as expected maintaining the swarm dynamics and formation despite communicating otherwise. A sensor attack is more devastating as it affects the victim node's awareness of itself. In a sensor attack the attacker resides somewhere within the victim node itself and replaces correct data readings with malicious data. This data is then ingested by the victim node as well as communicated to the rest of the swarm. This puts the victim node into a state of confusion as it attempts to maintain formation but ultimately fails, with the attacker overriding the data at will. This kind of attack can result in the complete loss of the node.

In this thesis we will demonstrate both attacks as well as analyze readings of the swarm side by side to understand the difference. Due to the nature of the simulations we do no assign units to any values.

3.1.2 Problem Formulation and Hypotheses

Consider a multi-vehicle system (MVS) with the objective to navigate toward a goal while maintaining a uniform connected formation. Here, "uniform formation" specifically refers to maintaining a safe distance between each node. The main focus of this research is to find a policy to detect if and when one or more vehicles are compromised and reconfigure the MVS to continue its intended operation. This work relies on the following assumptions:

1. The intention of the attack is to divert the network by hijacking one or more nodes in the swarm while being stealthy.
2. System dynamics including sensor specifications and uncertainties under normal conditions are known.
3. Each vehicle in the swarm has the same known dynamics.
4. Each vehicle receives the position of its neighbor nodes.
5. The vehicles are cooperative and use the exchanged information to coordinate their motion.
6. The expected error between the predicted and communicated state is contained within the noise of the state estimation.
7. There are no memory constraints and each agent in the swarm can store positional data from the other agents for as long as necessary.
8. The communication range is assumed to be larger than the sensing range.

In order to create resilient behaviors in swarms, we propose a consensus-based approach to coordinate the motion of each agent in the swarm based on the received information from the surrounding neighbors. The goal here is to maintain the connectedness of the system at all the times and enable predictable estimation of next state for each vehicle in a decentralized fashion. Additionally we want to use this estimation to detect inconsistent malicious behaviors of any node in the system.

In the framework that we lay out below, we first present a controller that utilizes virtual physics and graph theory rules to create a decentralized mesh of robotic vehicles in Section 3.2.1. We then outline a scheme for trust building and attack detection that leverages inter-swarm interactions between neighbors to identify spoofing and hijacking attacks in Section 3.2.2. We finally propose a checkpointing protocol to recover the state of attacked victim nodes in section 3.2.3.

3.2 Solution

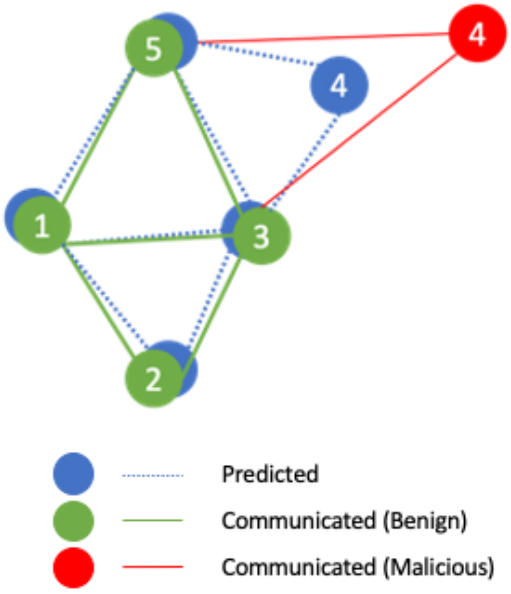


Figure 3.1: Approach overview

The general approach of the solution relies on two things. First, we rely on each node to predict the position of its neighboring nodes at each time step $t + 1$. In Fig. 3.1 this is represented by the blue circle. Secondly, we rely on each node to communicate its position with its neighboring nodes. In Fig. 3.1, correctly communicated nodes are represented by the green circles. Each node in the swarm is then able to compare the predicted positions to the communicated positions and determine the difference between the two for each

of its neighbors. Utilizing this comparison each node is able to keep a running record of how well its neighbor is conforming with respect to the expected behavior. In the image we can see that the red circle represents a malicious node communicating incorrect data. The difference between node 4's predicted position and communicated position is clearly much larger than the predicted position and communicated position of nodes 1,2, 3 and 5.

Along those lines we also keep a running record of trust over time. This means, that after each comparison the nodes can decide how trusted their neighbor is based upon whether or not they are properly conforming. Over time we can then look at the trends of these differences and trust values. Does a particular node have a much lower trust score than the other nodes? Does that node have a pattern of large differences between the predicted and communicated positions? This can be used to detect when a node is not behaving properly.

This is where our trust and detection algorithm comes in. In Fig. 3.2 the top image represents a properly behaving node and the bottom represents a malicious node

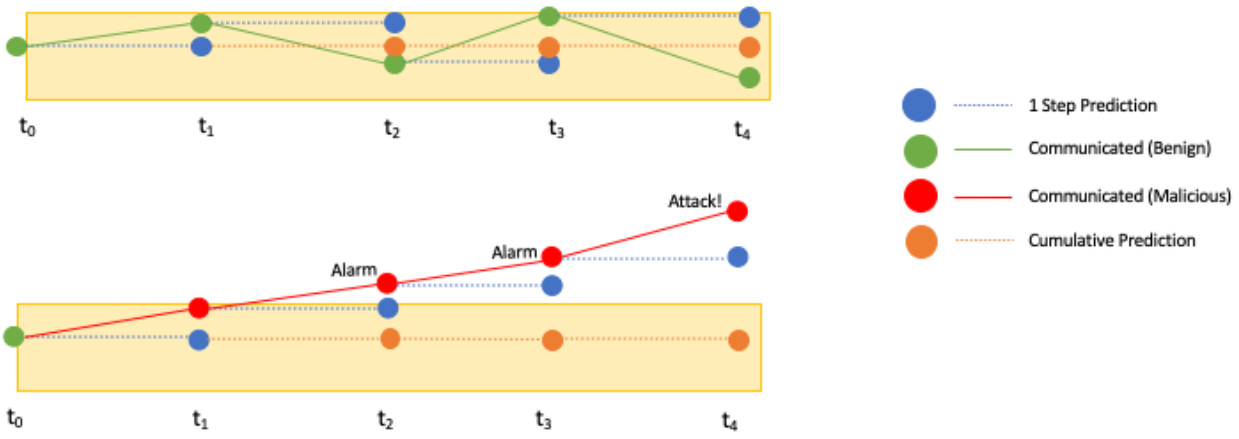


Figure 3.2: A diagram of how an attack might work.

In this diagram we can see how an attack might be performed at each time step. We start at t_0 at which point the communicated position is correct and the node is trusted. Based on that position and other dynamics depending on the swarm and the goal, the node's position at t_1 is predicted, however the communicated position indicates that the node is in fact slightly higher than expected. The difference between the two positions is noted. Even though an attack may hide within the noise of the sensor and system, over time the cumulative error is expected to have a certain distribution indicated by the yellow region in the pictorial representation. In normal conditions, the system's state and its prediction will stay within this bound most of the times. If an attack is hijacking the system pushing it toward a certain direction as displayed in the second image in Fig.3.2 then the cumulative error between the predicted and measured state will increase, sounding an alarm. In our approach, we don't immediately claim an attack has occurred the

first time that the error goes beyond the expected reachable region. We allow for this to account for the random natures of noise in measurements and in order to prevent false negatives. Instead we model the occurrence of such false negative cases and monitor how often the system is sounding an alarm. Once the number of alarms is above a certain threshold, then we consider that the system is under attack because its behavior is inconsistent from normal operations. In our framework, every time that an alarm is sounded a measure of trust is decreased. An attack is then detected if this measure is decreased concurrently for a certain system-dependant period of time. Furthermore, because we keep a record of the trustworthiness over time we can decide at what point that node went from trusted to un-trusted. In the case of Fig.3.2, we know that at t_0 the victim node was trusted. We can then use that data to checkpoint back in time and rerun our prediction algorithms to predict where the un-trusted node might actually be.

This dual trust and alarm procedure allows us to take into consideration benign noise. CPS systems carry much noise due to the combined affect of actuator and sensor noise. We account for this by parmeterizing a threshold of what the expected error is and by comparing the cumulative residual for each to node to that threshold. The cumulative residual is the summation of the differences between the predicted and communicated position over time. While we anticipate noise we expect it to look fairly random and thus this summation over time should have close to a 0 mean. Our proposed trust method allows for this noise to be tracked over time and for malicious patterns to emerge while still allowing the swarm to prioritize the goal. Rather than investigating these disturbances as they arise, the swarm operates under a principle of trust but verify allowing it to move with operational speed, each node adjusting to its neighbors at each step while still remaining constantly aware of the changes in the swarm dynamics that might indicate the infiltration of an attacker. Below we will step through each component of our approach more deeply.

3.2.1 Robot Model and Prediction Algorithm

We build the dynamics of the swarm using spring-mass virtual physics based on previous work in robotic and networked system control [4] [27]. The robots, or nodes in the swarm, are treated as point mass particles and the edges between each neighbor node simulates a spring, as depicted in Fig. 3.3. The spring-like edges are beneficial because they maintain the dissipative force effect of reaching a velocity of zero once the swarm has reached its goal and/or has arrived at a uniform shape.

The dynamics of the i^{th} robot are as follows:

$$\ddot{x}_i = \left[\sum_{j \in S_i} k_{ij}(\ell_{ij} - \ell_{ij}^0)\hat{d}_{ij} \right] + k_{ig}\ell_{ig}\hat{d}_{ig} - \gamma_i \dot{x}_i \tag{3.1}$$

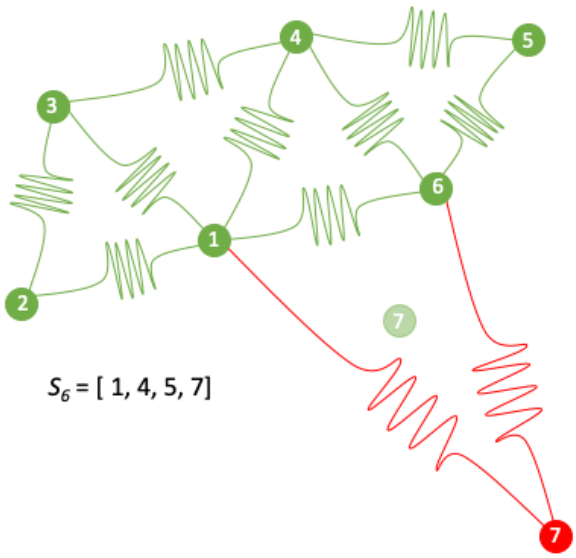


Figure 3.3: A diagram of how the spring mass algorithm works.

Where $i = 1...N$ and $i \neq j$.

We assume a simple path-loss communication model [16] in which communication decays with distance. In order to disperse and cover the environment well while maintaining connectivity, the network of the swarm is formed using the Gabriel Graph theory rule [4]. This rule is a nearest neighbor rule that utilizes a circle of a parameterized diameter. For any two nodes that are near each other and have no other nodes within a circle created using them as vertices, an edge is formed. The parameterized diameter is the maximum length for which a circle can be formed.

Following this rule, we produce S_i, the set of neighbors for node i. If we reference Fig. 3.3 we can see that node 6 has a set $S_6 = [1, 4, 5, 7]$. There exists two other nodes in the swarm that do not share an edge with node 6 and are therefore not in set S_6 and not included in the calculations for $\ddot{x}_6$. This is important to note because we leverage the information passed between neighbors for control purposes.

Here we also assign ℓ_{ij} to be the length of the edge or spring between robot i and j; in practice this is the Euclidean distance between two vehicles or nodes. ℓ^0_{ij} is the rest length of the spring; in practice this is the desired distance to maintain for safety and communication purposes. $\hat{d}_{ij}$ is the unit vector describing the direction of the force of the spring between i and j. This vector is responsible for the directional push and pull of the springs as they attempt to find equilibrium with each node approximately ℓ^0_{ij} apart. k_{ij} is the spring constant between the i and j, and γ_i is a damping coefficient responsible for controlling the magnitude of the repulsive/attractive forces (acceleration) between the robots.

In our simulations we add an additional virtual spring attractive force in the direction of the goal as specified in the second term on the right hand side of (3.1). In this case k_{ig}, ℓ_{ig}, and $\hat{d}_{ig}$ are the spring

constant, Euclidean distance, and unit vector between robot i and the goal g, respectively.

In Fig. 3.4 below we provide an example of the dynamics at work. Notice that the swarm begins in a position in which the nodes are closer together than the desired rest distance, thus they first expand to a safe formation then, maintaining the formation move in the direction of the goal.

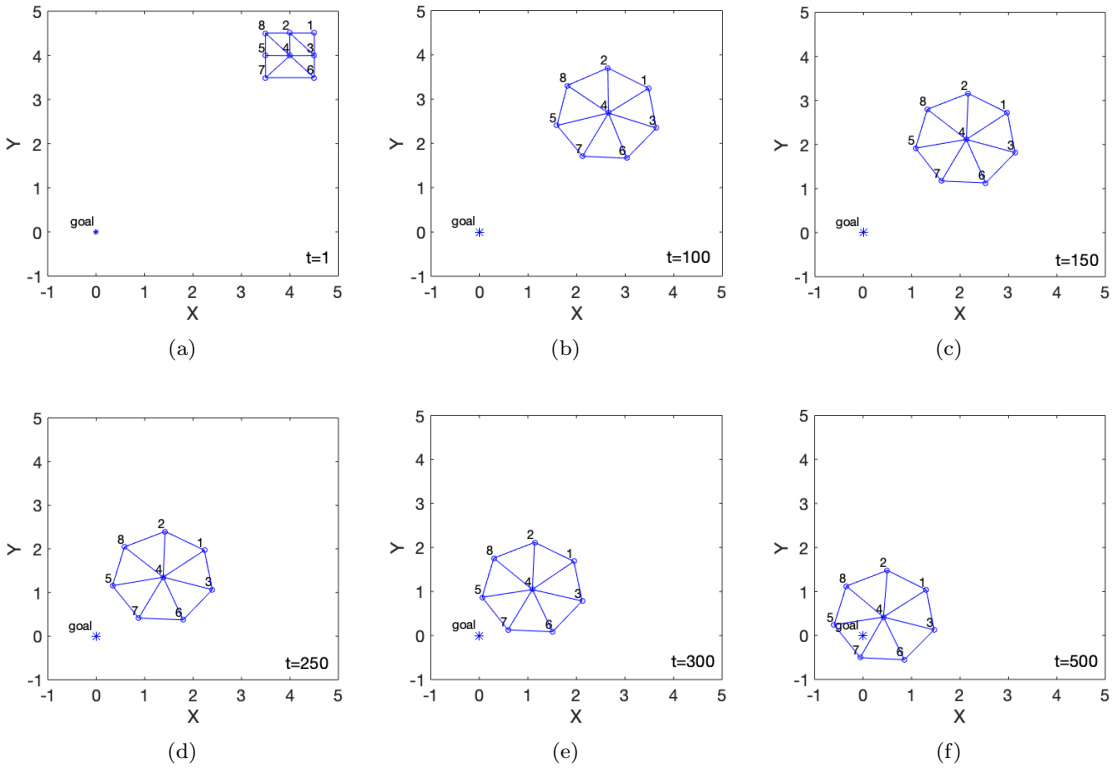


Figure 3.4: The expected behavior of the swarm without an attack.

We chose this spring-mass algorithm because its decentralized nature is more efficient and proven to produce similar results as centralized/global approaches [41]. Additionally, the ability to leverage inter-swarm information makes for more resilience and provides the opportunity for redundancy which we use for attack detection purposes. We also chose the Gabriel graph rule because it creates formations that tend to cover more of the environment by removing edges that cross each other [23]. Thus, the two main components of our controller are ideal for swarm formation and movement, however, while we leverage the inter-swarm communications for detection purposes it is also important to note how an attack can use those same communications to propagate an attack through this controller to the entire swarm.

Now, we explore how an attack affects the dynamics.

An attacker can compromise the operation of a swarm by changing the data propagated from some vehicles to others by spoofing on-board sensors or by manipulating the communication between vehicles. Mathematically

this can be represented as:

$$\tilde{y}_{j,t} = y_{j,t} + \xi \tag{3.2}$$

where $y_{j,t}$ is the true sensor measurement of j at time t, $\xi \neq 0$ is sensor/communication spoof and $\tilde{y}_{j,t}$ is the information received by the neighbor vehicles in the swarm. Since $\ell_{ij} = ||x_i - y_j||$, after an attack happens $\tilde{\ell}_{ij} = ||x_i - \tilde{y}_j|| \neq \ell_{ij}$ and in turn,

$$\tilde{\ddot{x}}_i = \left[\sum_{j \in S_i} k_{ij}(\tilde{\ell}_{ij} - \ell^0_{ij})\hat{d}_{ij}\right] + k_{ig}\ell_{ig}\hat{d}_{ig} - \gamma_i \dot{x}_i \neq \ddot{x}_i \tag{3.3}$$

hijacking the vehicles connected to the malicious node j to undesired locations.

If we indicate with $S_{c,i}$ the subset of S_i with compromised nodes, then (3.3) can be rewritten as

$$\tilde{\ddot{x}}_i = \left[\sum_{j \in S_i \setminus S_{c,i}} k_{ij}(\ell_{ij} - \ell^0_{ij})\hat{d}_{ij}\right] + \left[\sum_{j \in S_{c,i}} k_{ij}(\tilde{\ell}_{ij} - \ell^0_{ij})\hat{d}_{ij}\right] + k_{ig}\ell_{ig}\hat{d}_{ig} - \gamma_i \dot{x}_i \tag{3.4}$$

Although a malicious node directly affects only a subset of the vehicles in the network, the connectedness of the swarm causes this malicious effect to propagate through the entirety of the formation.

We can see one such attack in action in Fig. 3.5. The attack demonstrated is a communication based attack in which the x position is altered with added noise in the positive x direction. Note, the attack in these images begins at $t = 100$ and is a fairly aggressive attack for demonstration purposes; we also test our framework against more stealthy attacks. Furthermore although the nodes begin at the same position as in Fig. 3.4 the slight variation in initial formation is due to noise. Random noise is added in all simulations to replicate more realistic dynamics and the differences can change the formation slightly.

Our goal is to discover at run-time the set $S_{c,i}$ and hence eliminate the second terms on the right hand side in (3.4).

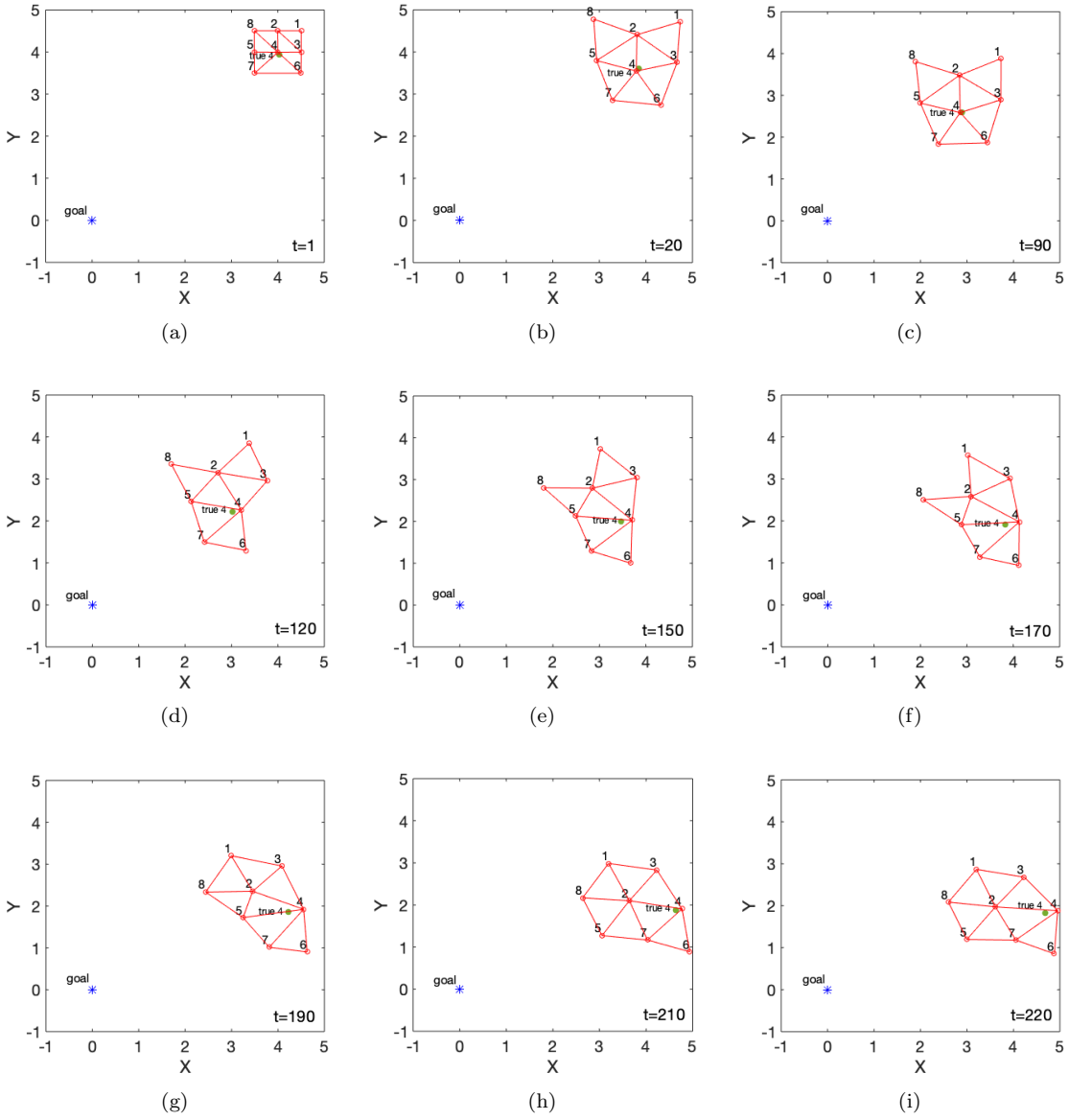


Figure 3.5: The behavior of the swarm during an attack with no recovery mechanisms implemented.

3.2.2 Trust Building and Attack Detection

Swarms of mobile autonomous vehicles are subject to significantly more disturbance than any single vehicle might be. Furthermore, as explained in Section 3.2.1, any disturbance, be it a malicious attack or a benign but significant gust of wind, might affect only one node's position at the moment it occurs, however, each future calculation will propagate that disturbance throughout the swarm to neighboring nodes. For this reason it is important for us to consider the ramifications of this for detecting an attack. We also wanted to challenge ourselves by considering an attacker who was stealthy, that is, an attacker who was aware that disturbances happened and there would always be some amount of error between the predicted and true positions. Taking these factors into account, detecting an attack becomes less straightforward. Thus, the motivation for why we measure trust over time is to be able to identify patterns in the noise that might be otherwise undetected.

The trust over time model we created builds in two different kinds of buffers to account for disturbance and naturally occurring noise errors. First, we set a threshold for the maximum expected size of the error at any given point in time. This accounts for the basic amounts of sensor and processor noise. This value should be set based on the specifications of the robots' sensors and actuators. Next we set a threshold for the number of concurrent time steps each node is allowed to be above the threshold. The reasoning behind this buffer is that while it is not altogether unheard of that a disturbance causes a node to have more noise than normal, the expectation is that each node will bounce back below the threshold. However, should a node carry more noise than expected for a longer period of time than expected, this is suggestive of an attack or otherwise anomalous behavior.

For the purposes of our implementation we assign three variables for use later. $\hat{x}_j^i$ is set to be the predicted position of node j by node i where $j \in S_i$ found in (3.1). These values are calculated by each node for each of its neighbors at every time step. y_j is the true position of node j at each time step and $\bar{y}_j$ is the communicated position of node j. Under normal conditions $y_j = \bar{y}_j$ and $|\hat{x}_j^i - \bar{y}_j| \leq \epsilon_1$ where ϵ_1 is the expected error of the system. Thus, we hypothesize that the swarm has been attacked or is otherwise operating unexpectedly if $\bar{y}_j \neq y_j \neq x_j$. We use this logic to build our trust and detection algorithms.

Our trust building algorithm leverages the information shared by neighboring nodes and each nodes ability to predict the position of its neighbors. Each node compares the communicated information ($\bar{y}_j$) with the predicted information ($\hat{x}_j^i$). The expectation is that the difference or residual between the two is small and within known bounds. In practice, the process is fairly simple. Each i^{th} robot compares the predicted position of the j^{th} robot with j's communicated position. In this way the i^{th} robot is constantly updating its understanding of where j says it is located and where i expects it to be next. We then observe the cumulative

residual over time and if it is within the bounds of error as well as following a normal random pattern we believe that robot j to be communicating correctly. However, if the communicated and predicted positions vary by large amounts over time or demonstrate a consistently positive or negative difference we hypothesize that that robot has been the victim of an attack. Although we specifically designed this trust building and detection in the context of an attack the approach has more general implications, including identifying any pattern of anomalous or divergent sensor data due for example to failure and system dynamics changes.

In order to account for noise and potential other factors that might induce a period of a larger difference between the communicated and predicted position we implement a trust building procedure. For this procedure each node i calculates the residual of each neighbor node j at time step t. We then keep the cumulative residual of these difference from t_0 to t_c, where t_c represents the current time.

$$\tau_j^i(t) = \bar{y}_j(t) - \hat{x}_j^i(t-1) \forall j \in S_i \forall i = 1...N \tag{3.5}$$

where N = total number of robots

$$T_j^i(t_c) = \sum_{t=0}^{t_c} \tau_j^i(t) \tag{3.6}$$

Where $T_j^i(t_c)$ is the cumulative residual for node j by i.

Next we use this $T_j^i(t_c)$ and perform algorithm 1 to calculate the trust value ρ and the attack variable ω.

Algorithm 1 Calculate Trust : ρ

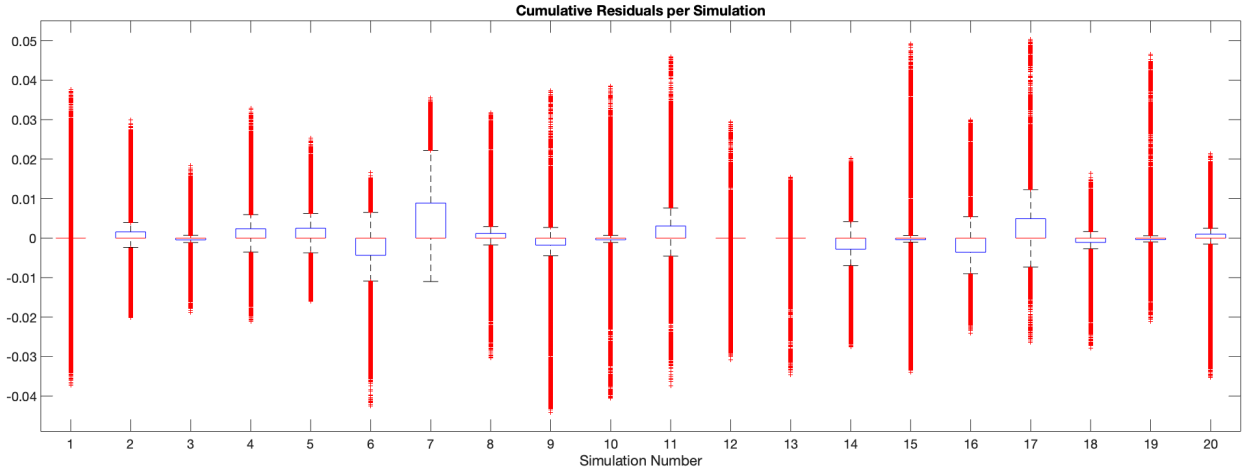
if $\mid T_j^i(t_c) \mid \leq \lambda\epsilon_1$ **then**
  Safe
  $\rho = \rho + k_i$
  $\Omega = \Omega - 1$
else
  Alarm
  $\rho = \rho - k_i$
  $\Omega = \Omega + 1$
end if
if $\Omega \leq \epsilon_2$ **then**
  Safe
else
  Attack
end if

In this algorithm, ϵ_1 is the cumulative residual threshold and ϵ_2 is the number of alarms or concurrent time steps during which trust is negatively impacted before we determine an attack. These parameters should be set on a per system basis as they are determined by the system noise. For the purposes of our simulations and experiments they were set empirically by running simulations of an 8-node swarm, 20 times, for 500 time steps. We then fit a curve to the maximum absolute value at each time step and use linear interpolation

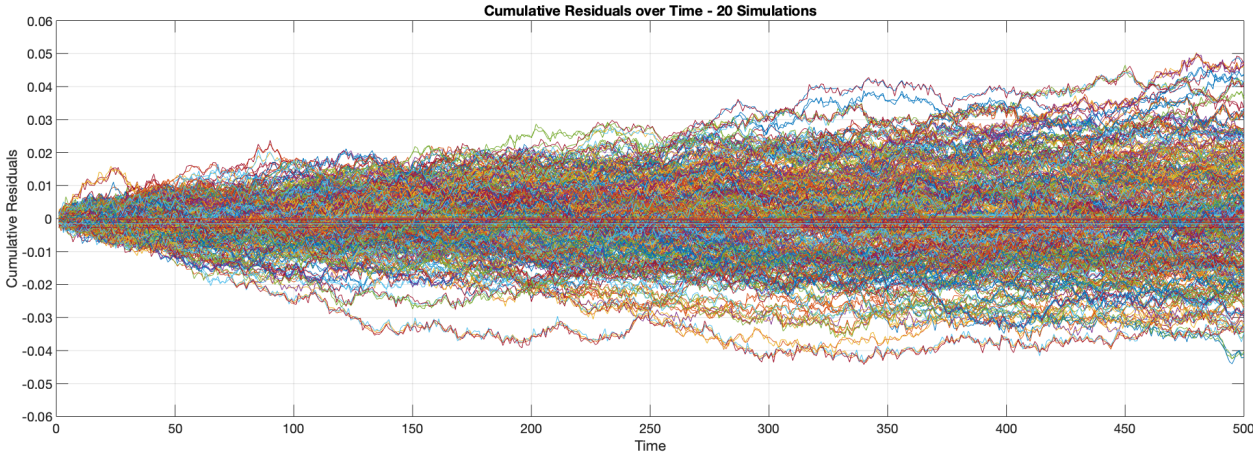
for every time step to construct our thresholding function. The results of these simulations can be seen in Fig. 3.6. Furthermore, we discuss the benefits of a dynamic thresholding function such as this in Section 3.2.4.

Finally, in (1), ρ is used to store the trust over time. If the residual at t, $\tau_j^i(t)$, is within our bounds for the error (ϵ_1) we increment ρ by a constant step k_i. If the difference is outside the bounds of error we decrement ρ by a constant step k_i. We also keep track of an attack variable Ω. If ρ is being negatively impacted we increment Ω and if ρ is positively impacted we decrement Ω. This allows us to account for instances in which there is a short period of negative trust but the system regains control. The Ω value is capped at 0 and ϵ_2 as determined above. 0 means the system is not under attack and at a steady state and a value $\geq \epsilon_2$ means we believe the system to be under attack. At this point, the swarm should perform mitigation and recovery measures.

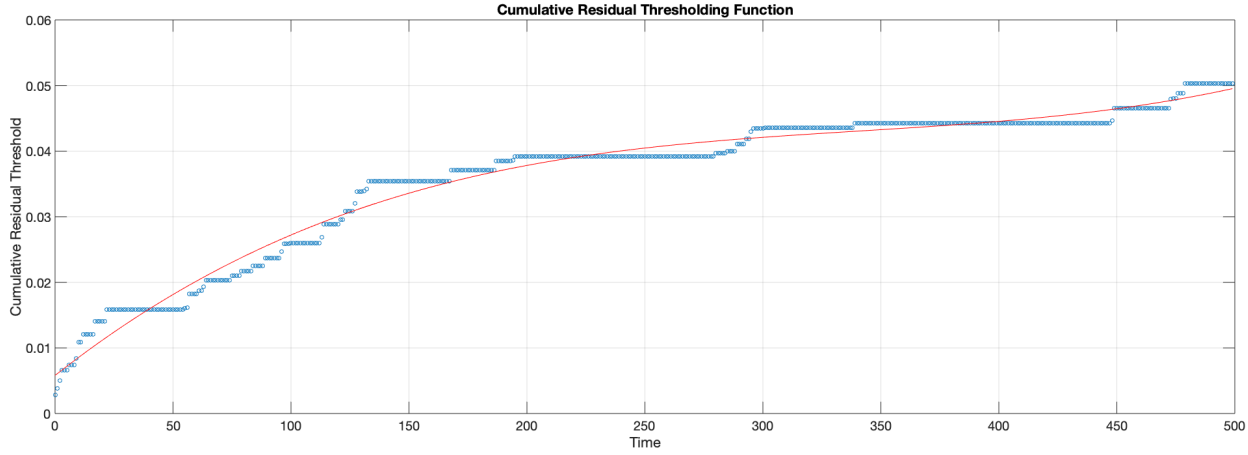
In Fig. 3.7(a) and Fig. 3.7(b) we demonstrate the expected behavior of the cumulative residuals and trust values of a node. Fig. 3.7(a) is a plot showing the cumulative residual of a node under normal conditions. Notice, the pattern is fairly noisy and randomly oscillates about zero, peaking at $10 * 10^{-3}$. Fig. 3.7(b) shows the trust for the node which is steadily increasing as the node's cumulative residual never exceeds the threshold ϵ_1.



(a)

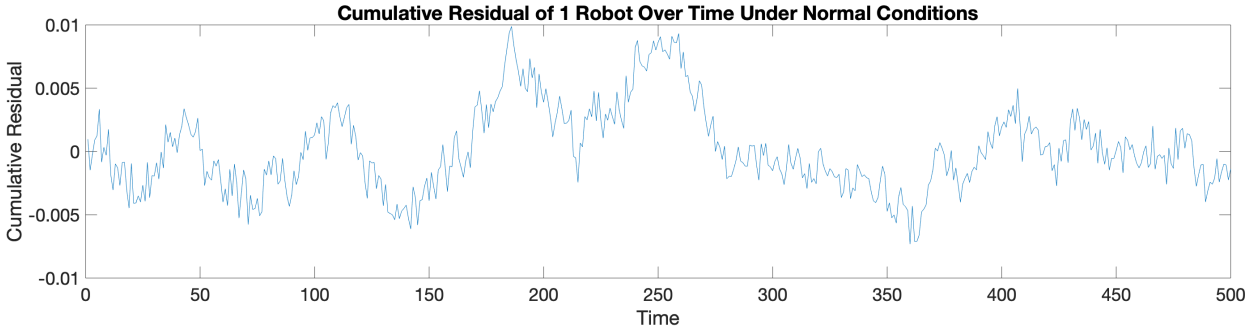


(b)

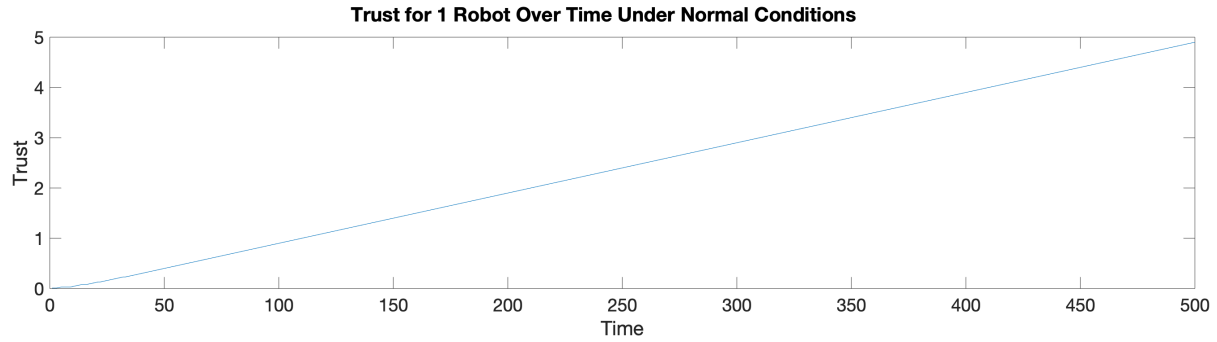


(c)

Figure 3.6: $T_j^i(t_c)$ for swarm under normal conditions



(a)



(b)

Figure 3.7: Cumulative residual and trust plots under normal conditions

3.2.3 Checkpointing

Lastly we discuss our proposed checkpointing procedure for use after detecting a potential attack. This procedure assumes that the victim node has undergone a communication attack and is still maintaining the swarm dynamics.

For each trust calculation between nodes ij we keep a variable $t_{trusted}$. This variable holds the most recent time step at which node i trusts node j. This value is only updated if no alarm flags have been raised and therefore there is no significant discrepancy between the communicated and predicted positions of node j. The update procedure is as follows: for each step in (1), if $\Omega = 0$ and we positively increment ρ, we set $t_{trusted} = t_c$ where t_c represent the current time. Once an alarm flag has been raised this variable stops being updated. Then should the node be determined to be under attack by our detection mechanism the swarm would remove the victim node from the network in order to prevent the potential spread of the attack as well as to stop providing information on the state of the swarm to the attacker. The swarm would then begin the checkpointing procedure.

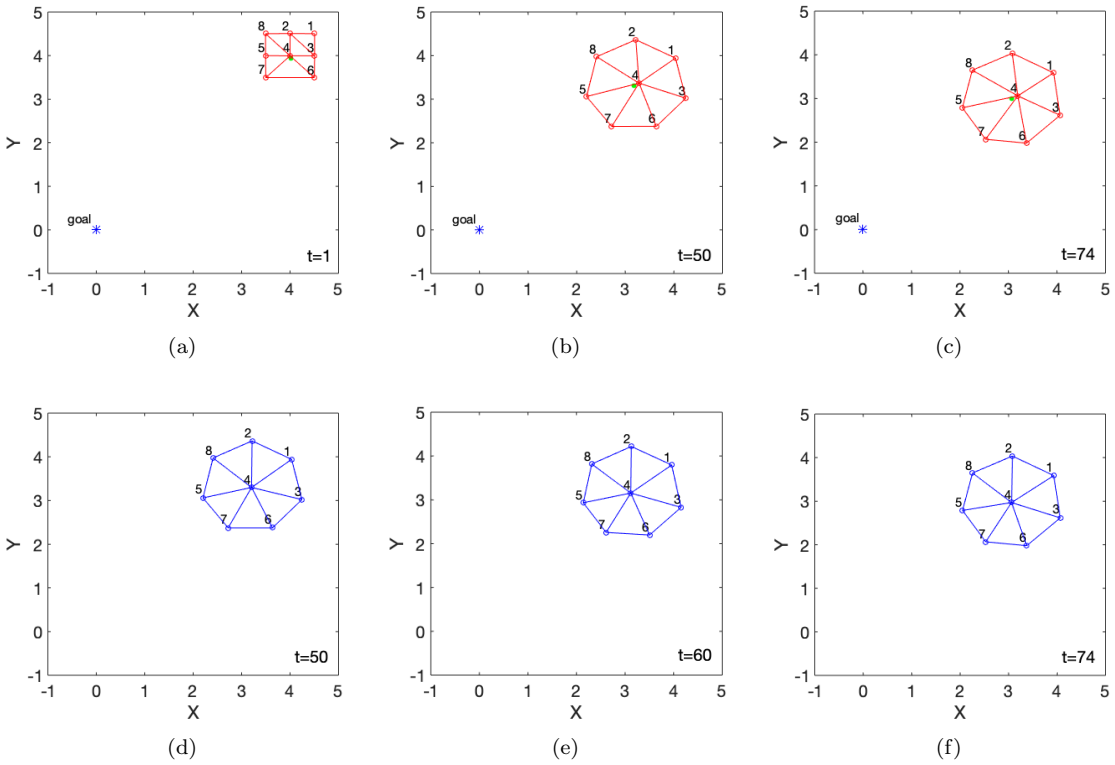
The checkpointing procedure is as follows: using (3.1) we roll back to $t_{trusted}$ and rerun the prediction calculations from $t_{trusted}$ to t_c using only the stored positional data of the trusted nodes and disregarding information sent by the victim node. At each step we predict only what the victim nodes position should be based on the formation and dynamics of the swarm. In this way we attempt to use these predictions to determine where the victim node is actually located. Again, this method of checkpointing presupposes the attack was a communication attack and that the node is still attempting to behave properly. In this scenario we would expect that the node is following the swarm dynamics and attempting to position itself in such a way as to maintain proper formation and rest length between itself and its neighbors. Thus, the checkpointing will accurately locate the compromised node within a small amount of error.

Using this information, the swarm can attempt to confirm the predicted, checkpointed position of the victim node using other sensing data such as obstacle detection sensing capabilities (e.g. short range lidar). If the node is within the error bounds of the predicted location the swarm could attempt to recover it and conclude that it was indeed undergoing a communication attack. If it is not at the predicted position but is at the communicated position the swarm could conclude that the node was a victim of excessive disturbance but can still be trusted and proceed as normal. Finally, if the node is not at either location we should assume that it has been lost and the swarm will remove that node from its calculations and proceed to the goal. At this point, other measures would be needed to attempt to recover the lost victim node.

In Fig. 3.8 we provide an example of an aggressive attack for the purposes of demonstrating the checkpointing procedure more clearly through stills. Sub figures 3.8(a) and 3.8(b) show the swarm behaving as normal

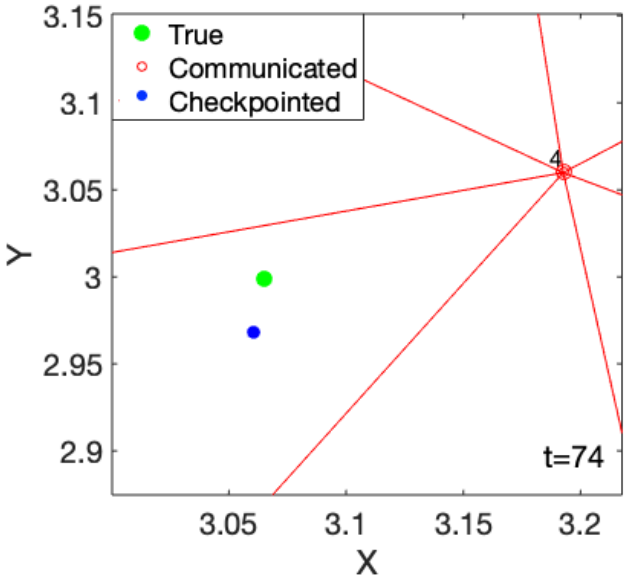
with the attack starting at $t = 50$. Right after, 3.8(b) and 3.8(c) shows the swarm under attack. In this figure we can see the communicated position of node 4, represented by the red circle in the center of the swarm, pulling in the positive x direction with the true position of node 4, represented in green, lagging behind. The next set of images 3.8(d) 3.8(e) 3.8(f) in all blue shows the back in time nature of the checkpointing algorithm as it goes back to a time step more trusted and reruns to the swarms current position predicting the true position of node 4. Finally, in the last image 3.8(g) we see the results of the checkpointing algorithm with the true position in green very close to the predicted position represented by the red dot.

In our simulations we do not use this procedure and instead simply reconfigure the network and continue towards the goal. However, this is a procedure that would work if it were operationally critical to retrieve the node or to predict if the node may be intersecting with any unsafe states.



(a) (b) (c)

(d) (e) (f)



(g)

Figure 3.8: Aggressive Attack Example

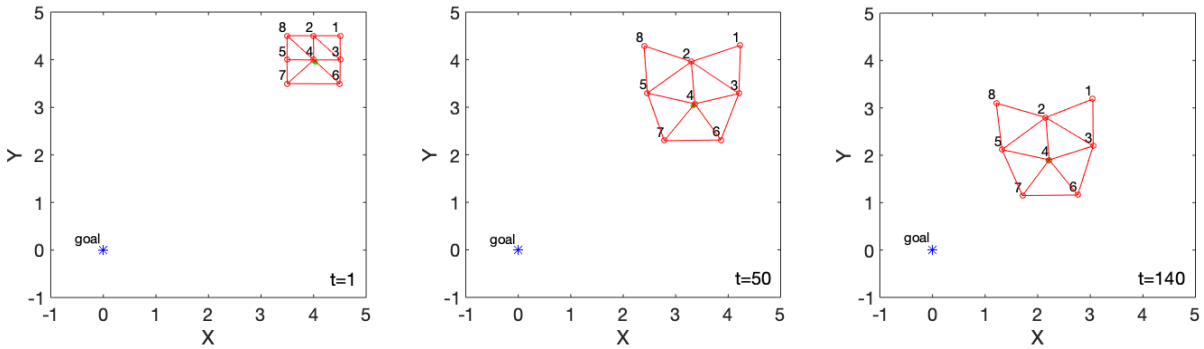
3.2.4 Simulations and Analysis

In this section we provide a series of simulations for our framework under different attacks. The priority for all swarms being simulated in this section is go to goal as such, after detecting an attack the victim node will be removed from the network and the swarm will reconfigure and proceed toward the goal. In Section 3.2.4 we demonstrate our framework against a stealthy communication spoofing attack and in Section 3.2.4 we demonstrate our framework against a stealthy sensor spoofing attack. In Section 3.2.4 we compare the two types of attacks. Finally, in section 3.2.4 we look at the benefits of the dynamic thresholding approach.

Simulation 1: Communication Spoofing

In this first simulation we return to the example of a communication attack. The simulation below is on the attack discussed in Section 3.2.1. For this attack we assume an attacker has managed to infiltrate the communications and is editing the communicated position. Furthermore, in this simulation the swarm has a goal position of (0,0) and assume that the attacker is attempting to stealthily drag the swarm off its course. Because we focus on stealthy attacks it is hard to see from the simulation stills where the attack begins. For this reason we will provide minimal stills from the simulation and instead focus on the trust values and cumulative residual plots.

In Fig. 3.9 we see the familiar starting positions of the swarm in the upper right hand corner, we then see the swarm expand and move towards the goal. For this simulation the attack begins at t=100, and the attacker is adding a positive 0.001 value to each communicated x position. The green dot represents the true position of the victim node (node 4) and the red dot represents the communicated position. To the human eye there is little difference between before the attack in the first two stills and after the attack in the last still.



(a) (b) (c)

Figure 3.9: Communication Attack case study

To understand how an attack was detected, reference Fig. 3.10(a). The first figure demonstrates the expected behavior of the cumulative residual for a benign node from the perspective of a benign node. As one can see, the max cumulative residual is fairly low, falling just below 0.0016. In addition, although there is a slight trend in the positive direction, the plot is very noisy. This indicates that the positive trend is naturally occurring and the node is behaving as expected.

The next figure Fig. 3.10(b) shows the cumulative residual for the attack node from the perspective of a benign node. Notice there is a distinct difference between the two plots both in terms of the scale as well as the pattern. As mentioned above, the first figure peaks at a max value of 0.0016 while the second image peaks at a max value of 0.06, a number almost 37 times larger. More importantly, the noisiness in the second image is significantly reduced starting at time step 100 when the residual begins to increase almost linearly. From this plot it is clear when the attack began.

The third plot Fig. 3.10(c) is for a benign node from the perspective of the attacked node. As mentioned in Section 3.2.3 during a communication attack the victim node is still reading correct data from its sensors. As such, the node will continue to follow the swarm dynamics. When the attack occurs, even though the victim node is the one propagating the malicious data they are unaware that an attack is occurring. Instead, they continue to behave as expected and compute their next position using the information communicated by their neighbors as well as comparing that information to its predicted position for each neighbor. As the attack propagates through the swarm, the neighbor nodes begin to be affected by the malicious data and they too begin to exhibit a pattern in their cumulative residual. This is represented in the third plot Fig. 3.10(c). The behavior is noisy and expected until around time step 125 when the residual begins to display a clear positively increasing pattern that is significantly less noisy than its previous behavior. This indicates that the neighbor is beginning to be pulled in the direction of the malicious communicated values and away from where the victim is expecting it to go.

Sub figure Fig. 3.10(d) plots the trust values for the malicious node from a benign node. Although from the cumulative residual plot we can tell that the attack started at time step 100 the value of the cumulative residual does not pass our threshold until around 140 at which point our trust begins to decrease and at around 145 we determine there to be an attack.

The next phase of the simulation is checkpointing. Reference Fig. 3.11. In this figure we only include the checkpointed predicted position for the attacked node as the simulation stills were too similar to be of interest. In this figure you can see the communicated position of the node represented by the red circle, the true position of the node represented by the green dot and the checkpointed predicted position represented by the red dot. As you can see the checkpointed predicted position falls somewhere between the true and communicated positions and closer to the true.

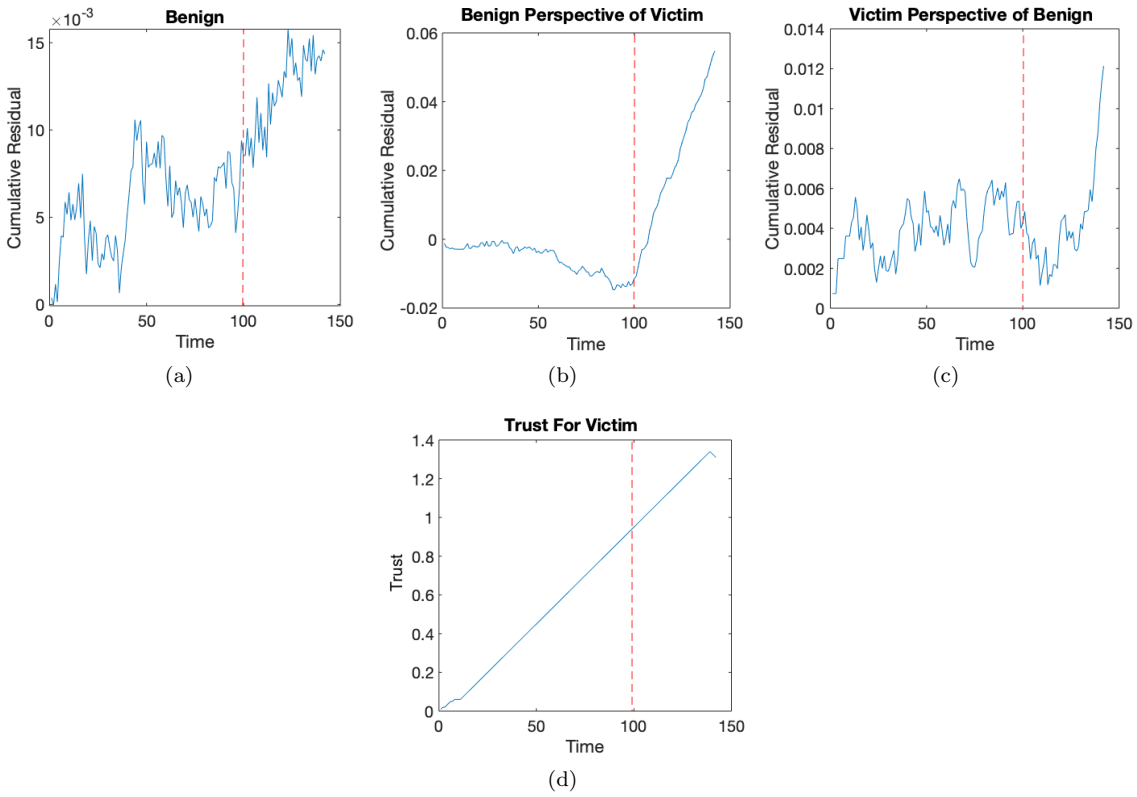
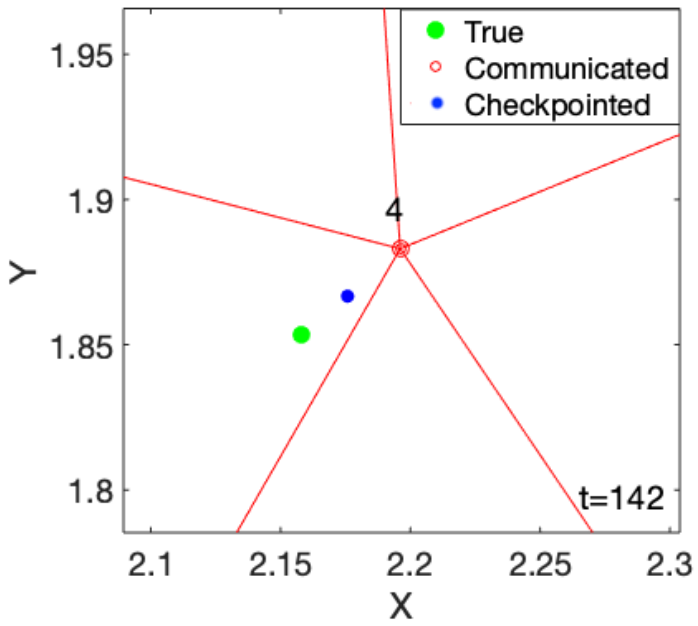


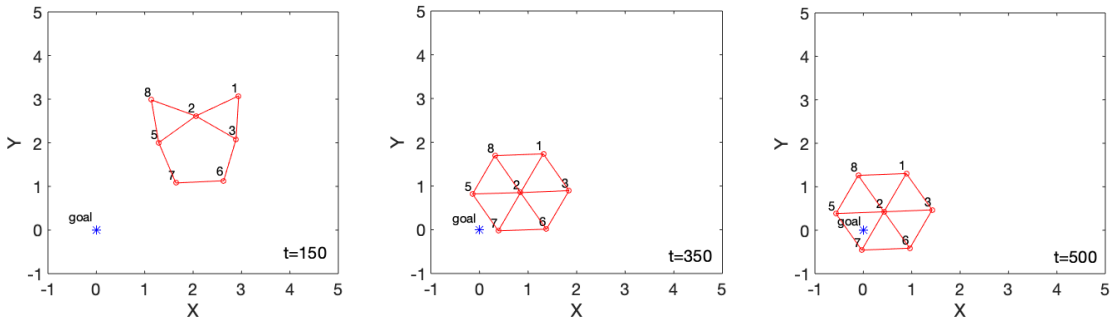
Figure 3.10: Cumulative Residuals and Trust Values for Communication Attack

Finally, in order to successfully reach the goal location, the swarm removes the victim node, reconfigures the network and moves in a safe formation towards the goal as can be seen in Fig. 3.12



(a)

Figure 3.11: Checkpointing for Communication Attack



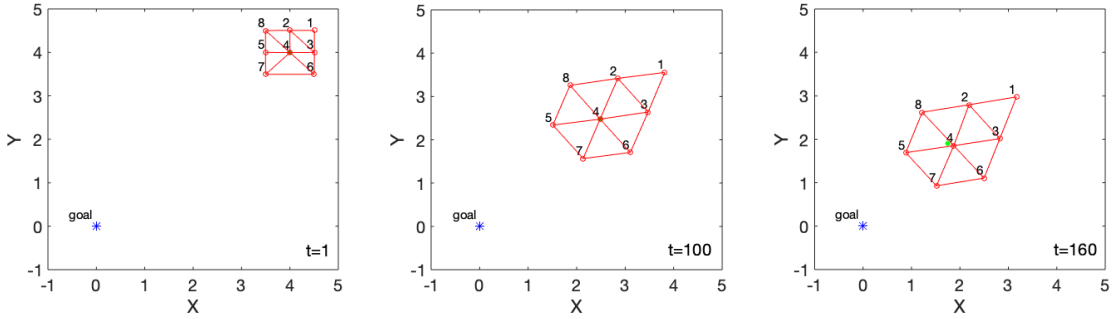
(a) (b) (c)

Figure 3.12: Behavior of the Swarm after Removing Malicious Node for Communication Attack

Simulation 2: Sensor Spoofing

In our second simulation we look at a sensor spoofing attack example. During a sensor spoofing attack the attacker has a lot more control over the node as it is manipulating the position information directly. This means not only is the victim node communicating the malicious data, the node also believe that data to be true and is thus unaware of where it is actually located. This means the node could either be where it is communicating that it is or it could be somewhere else no longer abiding by the swarm dynamics. In this simulation we will represent the second case, all other aspects of the attack will remain the same (i.e. the start position, the goal position, the attack increment, and the attack time start).

The first still looks much the same as the simulation from Section 3.2.4, however the formation of the swarm is slightly different. This is due to the noise that is added into all simulations to represent normal system noise and does not affect the framework or the attack.



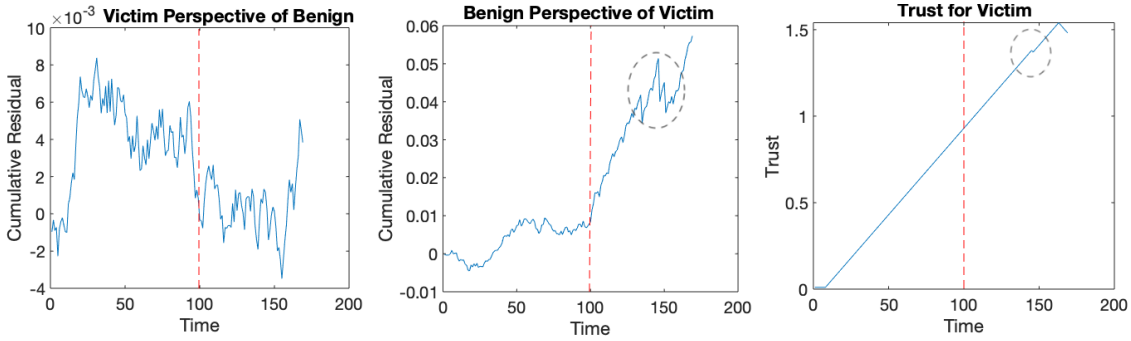
(a) (b) (c)

Figure 3.13: Sensor Attack Scenario

Again we look to the cumulative residual and trust plots to understand what is happening. Fig. 3.14(a) plots the cumulative residual for a benign node from the perspective of a benign node. We can see the noisy and random cumulative residual with a maximum value of 0.008. Next, Fig. 3.14(b) depicts the cumulative residual of the victim node from the perspective of a benign node. Although the plot is a little noisier than the communication attack plot we can still see the steady increase of the residual peaking first at time step 140 at a value of 0.05, if we look at the same time in the trust plot Fig. 3.14(c) we can see a decrease in trust at the moment. However the cumulative residual drops for around 10 time steps before once again peaking at close to 0.06. At this point the trust has decreased enough to determine the node is under attack.

This false peak, circled in Fig. 3.14(b) is interesting because it was not programmed into the simulation. Instead because the attack is a constant positive addition of a value within the bounds of the noise this represents a period during which the added random system noise and the attack noise were diametrically

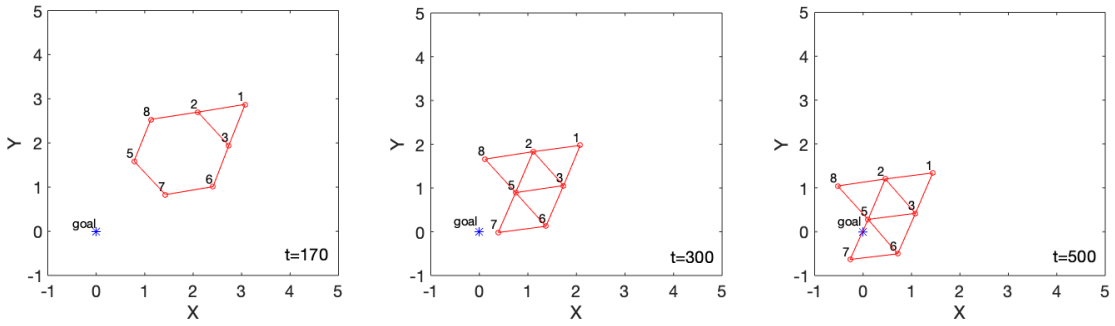
opposed. This shows that despite the stealth with which the attack is being performed we are still able to detect before much damage has been done to the swarm. Note, however that this attack was not detected until roughly time step 170 while the communication attack was detected before time step 150.



(a) (b) (c)

Figure 3.14: Position Difference and Trust Values for Sensor Attack

As mentioned earlier, the checkpointing prediction fails under the sensor attacks. Again, this is due to the more devastating nature of sensor-based attacks. For this reason we do not show the checkpointing step and instead move right to removing the victim node, re-configuring the network, and proceeding towards the goal.

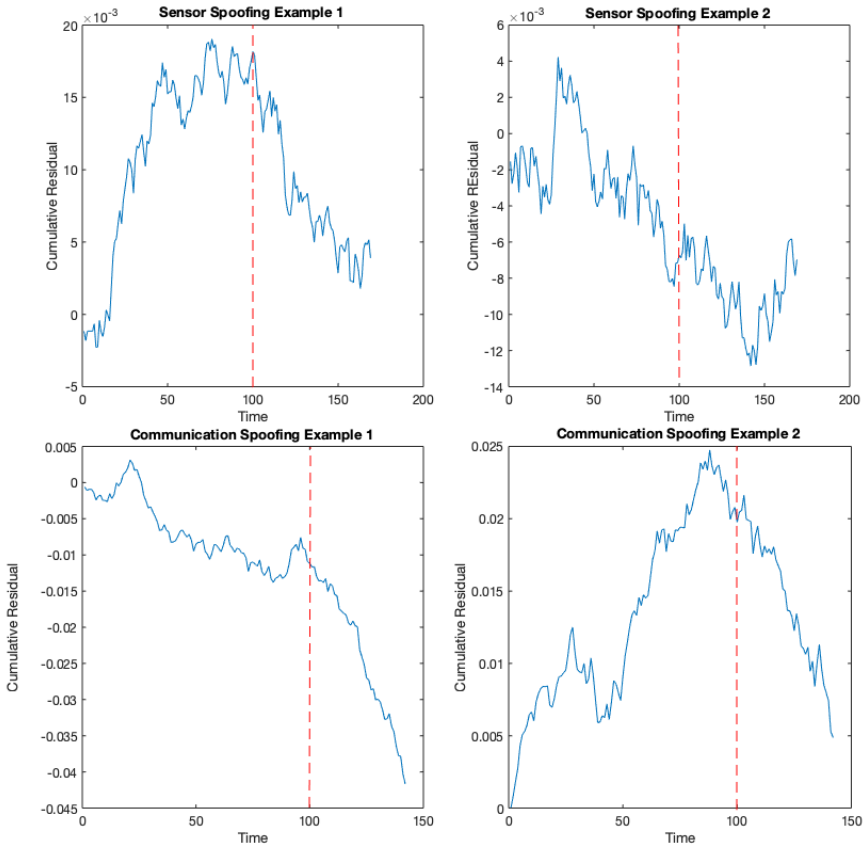


(a) (b) (c)

Figure 3.15: Behavior of the Swarm after Removing Malicious Node for Sensor Attack

Comparing Communication and Sensor Based Attack Cumulative Residuals

As mentioned in Section 3.2.4, during a communication attack the victim node is unaware that it is propagating malicious information and is instead attempting to maintain swarm dynamics. As such, the victim node continues to perform the comparison and detection algorithm for its neighbor nodes. As the attack progresses a pattern in the victim node's neighbor's cumulative residuals will arise. This is result of the neighbors getting pulled in the direction of the spoofed node. This is not the case in a sensor attack because the victim node believes it is where the malicious sensor information is indicating it is. For this reason the shift in its neighbor nodes position will make sense to the victim node. We can see this in the Fig. 3.16. The top row shows two figures plotting the cumulative residual of two different benign nodes from the perspective of the victim node. As expected, the malicious node sees only the normal noisy behavior. In contrast, the bottom row shows two figures plotting the cumulative residual of two different benign nodes from the perspective of the victim node in a communication attack. Here we can see that around the time of the attack the neighbors of the victim begin to move less noisy and distinct direction.



(a)

Figure 3.16: Sensor vs. Communication Attack

For the purposes of this thesis this distinction is more anecdotal, however, is interesting to note as it could lead to fruitful future work.

Results From Dynamic Thresholding

The main benefit of using a cumulative residual rather than point-in-time differences is the ability to observe patterns in node behavior. While point-in-time measurements are great for capturing anomalous behavior the moment it happens, it fails to identify the more stealthy attackers. In this work, we are specifically interested in identifying attackers who are attempting to hide within the noise. In such a use case the attacker is adding very small values to the positional data of one or more nodes. Point in time difference will miss this attacker because at any given time step the attacker is remaining below the threshold of what would be expected in normal system noise. However, over time if the attacker consistently adds small amounts of noise it can lead their victim/s far off course. In order to capture this we needed to observe the pattern and accumulation of noise over time. For this purpose, the cumulative residual works well at identifying expected random noise under normal conditions versus patterned incremental adjustments. While any given node may rise above the threshold at one point in time, at the next time step it is likely to drop back down. With an attack the accumulation of added malicious values will cause the residual to rise above the threshold and remain there, continuing to steadily rise.

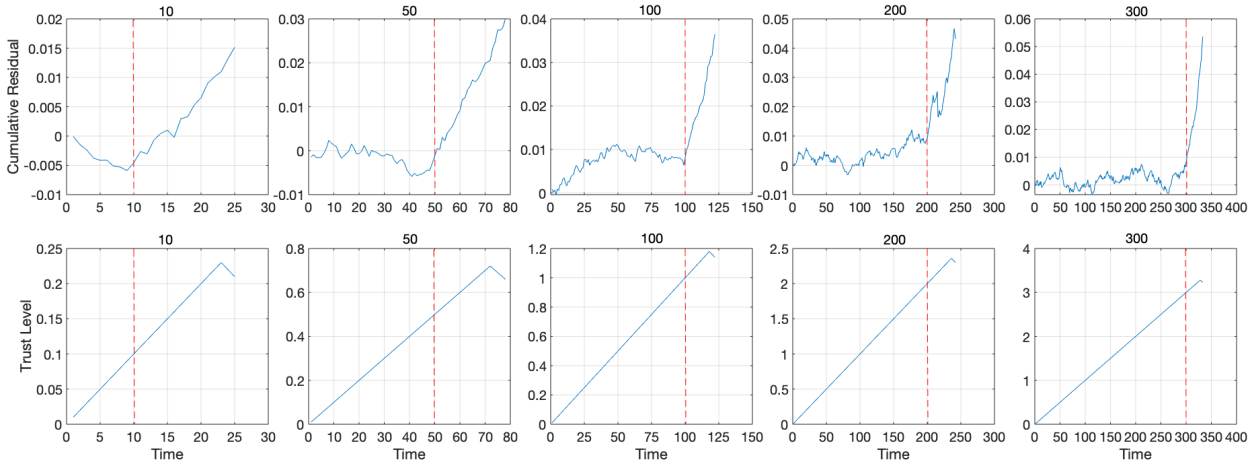
The cumulative residual for each node naturally tends to grow over time. Thus, in order to select the appropriate thresholding value ϵ_1 that minimizes false positives but catches an attack as early as possible we created a dynamic thresholding function. This was done empirically. We collected data on 20 simulations of an 8 node swarm under normal conditions over 500 time steps. We then plotted the maximum cumulative residuals for each time step and performed a simple regression analysis to fit a curve to the data. In order to best set our threshold for ϵ_1 we use this curve and performed interpolation to pick the appropriate threshold at any given time step.

In Fig. 3.17 we have plotted the cumulative residual and trust values for 5 different simulations. In each simulation the attack starts at time step 10, 50, 100, 200, and 300 respectively. Both plots for each simulation are the cumulative residual and trust values for the victim node from a benign node. The trust plots show at what point in the cumulative residual the node passed the set threshold for that time and trust began to decrease. Note, that this threshold is different for each plot and grows over time.

In the two previous simulation examples in Section 3.2.4 and Section 3.2.4 the threshold was not set dynamically and instead was set at a value of 0.05 for clearer demonstrations. We selected 0.05 because it was the largest value of noise observed during normal conditions. When we set a single threshold we want to be sure to account all normally occurring noise. However, because, as we mentioned, this noise grows over

time depending on when in the time period $0:500$ the attack starts, cumulative residual for the victim node has to accumulate over a longer period of time to pass that threshold. This is the case for both simulation examples. This is not ineffective, particularly because the detection framework catches the attack before it is able to make a noticeable change to the swarm. However, it is possible to catch the attack quicker with dynamic thresholding.

Notice in the first example Fig. 3.10 the attack starts at time step 100 but does not pass the threshold and get detected until close to time step 150. In Fig. 3.14 the detection takes even longer and is not detected until slightly after time step 150. In our simulation using dynamic thresholding, for attack start time 100, the attack is detected in half as much time, just before time step 125. This is because the threshold is smaller for time steps in the 100s than in the 400's and while we might expect benign nodes to accumulate more noise naturally by time step 500 it is unlikely they will have accumulated it by 100. Thus, we can safely lower the threshold without the risk of an increased rate of false positives. In this way the dynamic thresholding allows us to narrow the region of expected noise and increase the speed with which the swarm detects and attack.



(a)

Figure 3.17: Dynamic Thresholding Detection at Different Attack Times

Chapter 4

Conclusion and Future Work

In this thesis we provided a survey on current attack vectors and cyber security mechanisms for CPS. Additionally, we outlined a framework for attack detection, checkpointing, and recovery in multi-vehicle systems. Our framework builds on work for resilient swarm control as well as proposes a novel method for attack detection through leveraging inter-swarm interactions and performing regular comparisons between predicted and communicated positional data for neighboring nodes. Furthermore we propose a novel thresholding approach with the collection of cumulative residuals over time and linear interpolation to identify divergent patterns. Finally we propose a checkpointing mechanism that utilizes swarm dynamics and the existing prediction algorithms to determine where a node that has been the victim of a communication attack might be located.

The main benefit of this work is the scalability and ease of application. This framework uses an efficient controller, requires little additional processing power, and no more information than is already required for path planning. Furthermore, the parameterizing of ϵ_1 and ϵ_2 on a per system basis allows this framework to be as sensitive as desired. By creating tight bounds for ϵ_1 there exists the potential for more false positives but a smaller opportunity for an attack to remain undetected. By increasing ϵ_1 the opportunity for an attack to remain undetected increases while the rate of false positives goes down, however, the attacker would have to be more stealthy in order to avoid triggering the detection algorithm at some point and thus would be unable to impact the swarm by any significant amount. The benefit of this attack would mostly be in reconnaissance and gaining privileged knowledge to the whereabouts of the swarm.

4.0.1 Future Work

In this work we have assumed a basic communication model that is restricted to a parameterized distance unhindered within that range. While this is typically true, further work could explore the impact of intermittent or delayed communication. While, our framework is built on collecting uninterrupted data over time, particularly the checkpointing procedure, there exist applications for MVS that make uninterrupted communication challenging. A fertile area of exploration is to examine the effects of this intermittence and leverage more precise communication models to aid the detection of cyber-attacks. The inclusion of extended predictions to determine future intended paths and and analyze how well a neighbor node maintains course on the predicted path could also improve resilience and anticipate unsafe actions.

There also exists room for improvement for the checkpointing algorithm. At present our checkpointing algorithm is accurate only for communication-based attacks. We assume a victim of a communication attack is maintaining swarm dynamics and formation and we are therefore able to predict its location after an attack by rolling back to a trusted state and rerunning the prediction algorithm using only benign swarm data. Our framework however assume complete knowledge and long enough memory to store all past information related to the compromised node and its neighbors. One potential avenue for future research is to relax some of these assumptions and rely on reachability based methods to predict the possible future states of a malicious node after rolling back to a trustworthy state. How to handle the malicious node is also another aspect that will be considered more in future work: specifically in our current framework the vehicle is checkpointed but the swarm is removing its malicious effects by not considering its information. Depending on the prediction a state machine could be implemented to switch the uncompromised swarm into a recovery mode to search and rescue or even to pursue the malicious vehicle and surround it to stop its malicious behavior.

Finally exploring more complicated case studies and applications including heterogeneous robotic systems with mixed sensing and dynamics is also a topic of interest for future work.

Bibliography

[1] I. Akkaya, E. A. Lee, and P. Derler. "Model-based evaluation of GPS spoofing attacks on power grid sensors". In: *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. 2013, pp. 1–6.

[2] Jackson Barnett. *DIU wants to bring swarming drones to the DOD*. 2020. URL: https://www.fedscoop.com/diu-drones-uas-swarm-platforms/.

[3] N. Bezzo et al. "Attack resilient state estimation for autonomous robotic systems". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3692–3698.

[4] Nicola Bezzo et al. "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. DOI: https://doi.org/10.3182/20110828-6-IT-1002.02555. URL: http://www.sciencedirect.com/science/article/pii/S147466701644317X.

[5] Stephen Checkoway et al. "Comprehensive Experimental Analyses of Automotive Attack Surfaces". In: *Proceedings of the 20th USENIX Conference on Security*. SEC'11. San Francisco, CA: USENIX Association, 2011, p. 6.

[6] Y. Chen, S. Kar, and J. M. F. Moura. "Attack Resilient Distributed Estimation: A Consensus+Innovations Approach". In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 1015–1020.

[7] Y. Chen, S. Kar, and J. M. F. Moura. "Cyber-Physical Attacks With Control Objectives". In: *IEEE Transactions on Automatic Control* 63.5 (2018), pp. 1418–1425.

[8] 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.

[9] S. M. Dibaji, H. Ishii, and R. Tempo. "Resilient randomized quantized consensus with delayed information". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 3505–3510.

[10] Seyed Mehran Dibaji and Hideaki Ishii. "Resilient Consensus of Second-Order Agent Networks: Asynchronous Update Rules with Delays". In: *Automatica* 81 (Jan. 2017). DOI: 10.1016/j.automatica.2017.03.008.

[11] Saar Drimer and Steven J. Murdoch. "Keep Your Enemies Close: Distance Bounding against Smartcard Relay Attacks". In: *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. SS'07. Boston, MA: USENIX Association, 2007. ISBN: 1113335555779.

[12] O. Eigner, P. Kreimel, and P. Tavolato. "Detection of Man-in-the-Middle Attacks on Industrial Control Networks". In: *2016 International Conference on Software Security and Assurance (ICSSA)*. 2016, pp. 64–69.

[13] Ivan Evtimov et al. "Robust Physical-World Attacks on Machine Learning Models". In: *CoRR* abs/1707.08945 (2017). arXiv: 1707.08945. URL: http://arxiv.org/abs/1707.08945.

[14] D. Fiore and G. Russo. "Resilient and private consensus in multi-agent systems". In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3478–3483.

[15] W. Fu et al. "Resilient Consensus of Discrete-Time Complex Cyber-Physical Networks Under Deception Attacks". In: *IEEE Transactions on Industrial Informatics* 16.7 (2020), pp. 4868–4877.

[16] Andrea Goldsmith. *Wireless Communications*. USA: Cambridge University Press, 2005. ISBN: 0521837162.

[17] Qian He et al. "Lightweight attribute based encryption scheme for mobile cloud assisted cyber-physical systems". In: *Computer Networks* 140 (2018), pp. 163–173. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.01.038. URL: http://www.sciencedirect.com/science/article/pii/S1389128618300458.

[18] X. Huang and J. Dong. "Reliable Control Policy of Cyber-Physical Systems Against a Class of Frequency-Constrained Sensor and Actuator Attacks". In: *IEEE Transactions on Cybernetics* 48.12 (2018), pp. 3432–3439.

[19] A. Humayed et al. "Cyber-Physical Systems Security—A Survey". In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1802–1831.

[20] S. Kar, J. M. F. Moura, and K. Ramanan. "Distributed Parameter Estimation in Sensor Networks: Nonlinear Observation Models and Imperfect Communication". In: *IEEE Transactions on Information Theory* 58.6 (2012), pp. 3575–3605.

[21] C. G. L. Krishna and R. R. Murphy. "A review on cybersecurity vulnerabilities for unmanned aerial vehicles". In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 194–199.

[22] Marina Krotofil et al. "CPS: Driving Cyber-Physical Systems to Unsafe Operating Conditions by Timing DoS Attacks on Sensor Signals". In: *Proceedings of the 30th Annual Computer Security Applications Conference*. ACSAC '14. New Orleans, Louisiana, USA: Association for Computing Machinery, 2014, pp. 146–155. ISBN: 9781450330053. DOI: 10.1145/2664243.2664290. URL: https://doi.org/10.1145/2664243.2664290.

[23] Steven M. LaValle. *Planning Algorithms*. USA: Cambridge University Press, 2006. ISBN: 0521862051.

[24] H. J. LeBlanc et al. "Resilient Asymptotic Consensus in Robust Networks". In: *IEEE Journal on Selected Areas in Communications* 31.4 (2013), pp. 766–781.

[25] "Lessons from Stuxnet". In: *Computer* 44.04 (Apr. 2011), pp. 91–93. ISSN: 1558-0814. DOI: 10.1109/MC.2011.115.

[26] Y. Li et al. "Jamming Attacks on Remote State Estimation in Cyber-Physical Systems: A Game-Theoretic Approach". In: *IEEE Transactions on Automatic Control* 60.10 (2015), pp. 2831–2836.

[27] 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.

[28] Charlie Miller and Chris Valasek. "A survey of remote automotive attack surfaces". In: *black hat USA* 2014 (2014), p. 94.

[29] David Mitchell. *DEMCO awarded $126 million loan for electrical lines, smart grid technology*. 2020. URL: https://www.theadvocate.com/baton_rouge/news/business/article_78f385d4-b560-11ea-be7f-a341d1b944a9.html.

[30] Y. Mo and B. Sinopoli. "Secure control against replay attacks". In: *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2009, pp. 911–918.

[31] Y. Mo, S. Weerakkody, and B. Sinopoli. "Physical Authentication of Control Systems: Designing Watermarked Control Inputs to Detect Counterfeit Sensor Outputs". In: *IEEE Control Systems Magazine* 35.1 (2015), pp. 93–109.

[32] Yilin Mo et al. "False data injection attacks against state estimation in wireless sensor networks". In: Jan. 2010, pp. 5967–5972.

[33] Ben Nassi et al. *Phantom of the ADAS: Phantom Attacks on Driver-Assistance Systems*. Cryptology ePrint Archive, Report 2020/085. https://eprint.iacr.org/2020/085. 2020.

[34] Cameron Nowzari, Eloy Garcia, and Jorge Cortes. "Event-Triggered Communication and Control of Networked Systems for Multi-Agent Consensus". In: (Dec. 2017). DOI: 10.1016/j.automatica.2019.03.009.

[35] Mark L Psiaki and Todd E Humphreys. *Protecting GPS From Spoofers Is Critical to the Future of Navigation*. July 2016. URL: https://spectrum.ieee.org/telecom/security/protecting-gps-from-spoofers-is-critical-to-the-future-of-navigation.

[36] 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.

[37] D. Saldaña et al. "Resilient consensus for time-varying networks of dynamic agents". In: *2017 American Control Conference (ACC)*. 2017, pp. 252–258.

[38] Fred Samland et al. "AR.Drone: Security threat analysis and exemplary attack to track persons". In: *Proceedings of SPIE - The International Society for Optical Engineering* 8301 (Jan. 2012), pp. 15–. DOI: 10.1117/12.902990.

[39] B. Satchidanandan and P. R. Kumar. "Dynamic Watermarking: Active Defense of Networked Cyber–Physical Systems". In: *Proceedings of the IEEE* 105.2 (2017), pp. 219–240.

[40] Yilun Shang. "Resilient consensus of switched multi-agent systems". In: *Systems & Control Letters* 122 (2018), pp. 12–18. ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2018.10.001. URL: http://www.sciencedirect.com/science/article/pii/S0167691118301750.

[41] Brian Shucker, Todd D. Murphey, and John K. Bennett. "Convergence-preserving switching for topology-dependent decentralized systems". English (US). In: *IEEE Transactions on Robotics* 24.6 (Dec. 2008), pp. 1405–1415. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.2007940.

[42] Shuo Chen and Maode Ma. "A dynamic-encryption authentication scheme for M2M security in cyber-physical systems". In: *2013 IEEE Global Communications Conference (GLOBECOM)*. 2013, pp. 2897–2901.

[43] Chawin Sitawarin et al. "DARTS: Deceiving Autonomous Cars with Toxic Signs". In: (Feb. 2018).

[44] Yunmok Son et al. "Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 881–896. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/son.

[45] Yazhou Tu et al. "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1545–1562. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/tu.

[46] Y. Wang and H. Ishii. "A Distributed Model Predictive Scheme for Resilient Consensus with Input Constraints". In: *2019 IEEE Conference on Control Technology and Applications (CCTA)*. 2019, pp. 349–354.

[47] 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.

[48] Amy Webb. *The Future of Customer Service Is Driving Down the Sidewalk (Slowly)*. 2020. URL: https://www.inc.com/magazine/201908/amy-webb/delivery-drone-robot-autonomous-vehicle-order-fulfillment.html.

[49] Y. Wu et al. "Event-Triggered Resilient Consensus for Multi-Agent Networks Under Deception Attacks". In: *IEEE Access* 8 (2020), pp. 78121–78129.

[50] Shiyong Yin et al. "M2M Security Technology of CPS Based on Blockchains". In: *Symmetry* 9.9 (Sept. 2017), p. 193. ISSN: 2073-8994. DOI: 10.3390/sym9090193. URL: http://dx.doi.org/10.3390/sym9090193.

[51] H. Zhang et al. "Optimal DoS attack policy against remote state estimation". In: *52nd IEEE Conference on Decision and Control*. 2013, pp. 5444–5449.