Analyzing the Impact of GNSS Spoofing on the Formation of Unmanned Vehicles Swarms

Aanjhan Ranganathan, *Northeastern University, Boston, MA*
Adam Belfki, *Northeastern University, Boston, MA*
Pau Closas, *Northeastern University, Boston, MA*

BIOGRAPHY

Aanjhan Ranganathan is Assistant Professor at Northeastern University, Boston, MA. His research revolves around the security and privacy of wireless networks with a strong focus on autonomous cyber-physical systems and smart ecosystems. Aanjhan is a recipient of several awards, including the prestigious NSF CAREER award, the outstanding dissertation award from ETH Zurich, the regional winner of European Space Agency's Satellite Navigation competition, and the Cyber Award from armasuisse (Switzerland's Department of Defense).

Adam Belfki is a fifth year undergraduate student at Northeastern University Khoury College of Computer Sciences. He is majoring in Computer Science with a minor in Math and a concentration in Artificial Intelligence. His research focuses on distributed and decentralized systems with applications in distributed coverage of swarms and blockchain consensus protocols.

Pau Closas is Associate Professor at Northeastern University, Boston, MA. He received the MS and PhD degrees in Electrical Engineering from UPC in 2003 and 2009. He also holds a MS in Advanced Mathematics from UPC, 2014. His primary areas of interest include statistical signal processing, robust stochastic filtering, and machine learning, with applications to positioning systems and wireless communications. He is the recipient of multiple awards including the 2014 EURASIP Best PhD Thesis Award, the 9th Duran Farell Award, the 2016 ION Early Achievements Award, 2019 NSF CAREER Award, and 2022 IEEE AESS Harry R. Mimno Award.

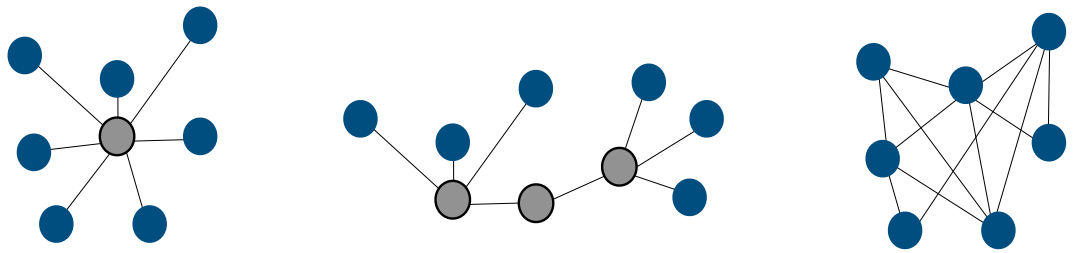
ABSTRACT

In this paper, we explore the vulnerabilities and resilience of drone swarms to potential attacks, particularly focusing on the significance of accurate position information in the successful completion of swarm missions. We delve into the role of location spoofing attacks and assess the robustness of centralized versus distributed swarm communication architectures against such threats. Emphasizing the increasing adoption of distributed and decentralized algorithms, due to their adaptability and elimination of central control, our research centers on a scenario wherein a swarm aims for uniform distribution across a region, ensuring each drone covers equivalent areas. Such coverage tasks are vital for various applications, including surveillance and navigation. Using Voronoi tessellations and Lloyd relaxation, we identify possible attack vectors these missions might encounter. To assess the swarm behavior under location manipulations, we employ both a comprehensive framework integrating Gazebo, Ardupilot, and QGroundControl and a Python-based simulator. Our findings illuminate the challenges inherent to ensuring robust swarm operations and underscore avenues for future research aimed at bolstering swarm technology's defenses.

I. INTRODUCTION

Global Navigation Satellite Systems (GNSS) is a widely-spread Positioning, Navigation and Timing (PVT) technology, which uses the signals transmitted from one or more satellite constellations, including Global Positioning System (GPS), Galileo, GLONASS and BeiDou (Morton et al., 2021). Standalone or fused with complementary technologies, GNSS is the de facto technology for positioning in many applications requiring reliable navigation, when it is available. However, it has been widely demonstrated (Ioannides et al., 2016) that GNSS-based localization technologies are vulnerable to signal jamming and spoofing/replay attacks, whereby a receiver can be either denied positioning (Borio et al., 2016) or deceived to compute a forged PVT solution (Psiaki and Humphreys, 2016). It is today possible to spoof a GPS receiver to any arbitrary location and time in the world with many incidents being reported in the wild (MIT Tech Review, 2019). This vulnerability potentially jeopardizes the use of GNSS in critical applications (Amin et al., 2016), a situation that has attracted the research community which has gained a substantial understanding of those threats and their potential solutions (Dovis, 2015).

A recent work (Sathaye et al., 2022b) delved into the utilization of location information within intricate autonomous cyber-physical systems, specifically focusing on unmanned aerial vehicles (UAVs). These systems rely on multiple sensor modalities to make critical navigation decisions. Specifically, experimental analysis assessing the feasibility and requirements of exerting full control over a UAV's movements solely by spoofing GPS signals were presented. The research described the challenges associated with achieving a comprehensive takeover of a UAV through GPS spoofing while ensuring controlled flight without



(a) Centralized **(b) Hierarchical (Hybrid)** **(c) Decentralized (Emergent Consensus)**

Figure 1: Swarm Architectures: (a) centralized, where a single central node (presented in light gray in the graph) is in charge of the swarm decision-making; (b) hierarchical, where the swarm contains clusters, each locally governed by a central node; and (c) decentralized, where all nodes have the same hierarchical tasks and no central node is required.

collisions and emphasized that while commercial off-the-shelf (COTS) UAVs remain susceptible to GPS spoofing attacks, achieving complete control over a single UAV necessitates the intricate manipulation of the spoofing signals in real-time. In parallel, it was shown to be possible to take control of a UAV by spoofing signals that would change the course of the UAV to that which the spoofer designs, for which a sensing/actuation loop is required (Kling et al., 2022).

The above work looked at the problem of GNSS interference in the context of a single UAV and there is little work on investigating the effect of such spoofing and jamming attacks on GNSS technology when it comes to swarms of unmanned systems. That is, a taxonomy of those threats, attack vectors and associated consequences is missing when groups of collaborative agents are spoofed or jammed. There is an increasing interest in the use of autonomous swarms of robots in safety- and security-critical applications such as search and rescue missions, emergency support, construction efforts, or delivery of goods (Abdelkader et al., 2021). Swarms also have significant applications in the military both from a tactical as well as logistic perspective. For example, swarms of unmanned systems can be used to saturate and overwhelm targets. The aforementioned swarm-based applications require precise positioning, navigation, and communication systems. In parallel, successful and coordinated attacks are deemed more complex to implement than the case of attacking standalone devices, but still feasible. However, it is unclear how such spoofing and jamming attacks on localization systems affect swarms and what effect it has on the overall swarm ecosystem. For example, a spoofing attack on a swarm of unmanned aerial vehicles can potentially lead to swarm collision and collapse or, simply, displacing its course. Swarms also form mesh networks to communicate and coordinate their movements and an adversarial attack on these communication links needs to be studied. Furthermore, given the increasing availability of low-cost unmanned "agents", adversaries are also bound to use them to increase the effectiveness of the attack and it is essential to understand the fundamental limits and requirements of executing such attacks.

In this paper, we take the first step to investigate the vulnerabilities and resilience of swarms to those attacks. First, we describe the various swarm architectures and their corresponding use cases. We will then delve into the importance of position information in achieving a successful swarm mission, and how it can affect the overall outcome of the mission. Specifically, we investigate the impact of location spoofing attacks, and compare and contrast the effectiveness of centralized and distributed swarm communication architectures in countering these attacks using a sample case study. Given the drive towards distributed and decentralized algorithms due to their effectiveness against rapidly changing environments and removal of a central control authority, our example scenario focuses on a specific distributed and decentralized swarm architecture whose goal is to uniformly distribute a swarm of drones across a geographic region to ensure each drone covers an equal area. The so-called coverage task is required in a wide variety of swarm missions including tracking, surveillance, or simply navigating an area. We will analyze popularly deployed Voronoi tessellations (Du et al., 1999) with Lloyd relaxation (Lloyd, 1982) and gain understanding on the attack vectors that a coverage mission can suffer. We will leverage a combination of custom-built framework using Gazebo, Ardupilot, and QGroundControl as well as a Python based simulation framework to evaluate the behavior of swarms to location manipulation threats. Finally, we will summarize the key takeaways and offer suggestions for future research and the challenges that must be addressed in order to improve swarm technology.

II. POSITION INFORMATION IN SWARM ARCHITECTURES AND ALGORITHMS

Drone swarms have certain advantages that make them appealing over traditional single-agent systems. Importantly, swarms provide enhanced coverage (i.e. being able to expand wider geographical areas), accuracy, safety, and flexibility. Some swarm systems exhibit, additionally, low-cost deployments when compared to single-drone setups, which enables using large numbers of expendable agents thus increasing the aforementioned benefits.

1. Swarm Architectures

There are several swarm architectures based on the the specific strategy used in directing the swarm behaviour and specific applications. Broadly, they can be classified into three classes (see Figure 1): i) centralized, ii) decentralized, and iii) hybrid architectures. Centralized architectures rely on a central controller or a central decision-making unit that coordinates the behavior of the swarm drones. In centralized algorithms, the controller collects information from the individual drones and processes it to determine the optimal behavior for the swarm as a whole. In a centralized scheme, the central node (which could be a drone in the swarm itself) is in charge of $i)$ gathering relevant information from the drone entities (e.g. their computed location) and $ii)$ computing and communicating the next position for each agent. Centralized schemes are typically less robust to attacks since that central node performing the main tasks in the swarm becomes the single point of failure and thus can be easily targeted by an attack.

In a decentralized architecture, there is no single leader or structured hierarchy. Instead, individual members follow rules and react to localized information, leading to a global behaviour "emerging" from these local interactions. Sometimes, these architectures are also referred to as "emergent consensus" algorithm based architectures. The main advantage of decentralized architectures is there is no single point of failure and thereby robust and adaptable to changing environment and mission status.

Finally, a hybrid setup combines elements of multiple architectures. For instance, a system might be partly centralized for certain tasks but also allow emergent behavior for other tasks. Entities are organized in layers or levels of responsibility. Decisions or directives might flow from higher to lower levels, but not everything is dictated by a single central entity. Instead, there can be multiple leaders or decision points in the system. Such a setup can be referred to as "hierarchical" architecture. An example scenario would be a robot rescue team where some robots are designated as scouts, some as carriers, and others as medics. The scouts might decide which areas to explore, the carriers transport goods or injured beings based on scouts' feedback, and medics treat injuries. While there's a hierarchical division of roles, decisions within those roles can be made autonomously based on local conditions.

Distributed algorithms rely on each drone making its own decisions based on local information and communication with other drones in its vicinity. In distributed algorithms, each drone has a set of rules or behaviors that it follows, and the collective behavior of the swarm is determined by those rules and emerges from the interactions between individual drones. Typically, there are sequential steps that occur for a distributed swarm to decide on next steps: $i)$ each agent reads sensor data and learns about its local state (e.g. position estimation); $ii)$ the local states are communicated to neighboring agents through some (potentially secure) communication link; $iii)$ the control algorithm running locally, uses its own data and the received information from other agents to determine where the next position should be. As opposed to the centralized counterpart, a distributed scheme is typically more resilient to attacks since the intelligence of the swarm behavior is shared across the agents and thus no single point of failure exists.

Besides the architecture, various sensing, communication, processing, and control modalities play a critical role in the specific algorithms selected for deployment. It is important to note that in many real-world swarm applications, the line between distributed and decentralized can blur. A swarm might use distributed algorithms to process data across its members but use centralized or decentralized algorithms for decision-making and actions based on that data.

2. Swarm Behaviours and Algorithms

There are several functions for a swarm depending on the specific application and mission they are assigned to accomplish. We classify broadly some of the algorithms with a brief description.

a) Spatial Organization.

These behaviors and algorithms (Brambilla et al., 2013) pertain to how agents arrange themselves in space, either relative to each other or within an environment. Some examples problems include formation (drones flying in a specific pattern), flocking (cohesive movement), and coverage (environment monitoring ensuring every part of space is attended to).

b) Resource Foraging.

Some algorithms (Bayındır, 2016) specifically address the challenge of searching for or distributing resources e.g., target search and tracking or foraging for food in the case of natural swarms like ants. Some class of algorithms aim to distribute resources to ensure a balanced workload and can be classified under this category of behaviours.

c) Co-ordination and Decision Making.

In these set of algorithms (Bayındır, 2016), agents co-ordinate their actions or make collective decisions. For example, the agents reach an agreement on certain variables or states, co-ordinate their actions in time, or take over different roles at different time and spatial instances.

d) Adaptive Learning Behaviours.

As the name suggests, these algorithms are designed to allow the swarm to adapt to changing environments or to improve performance over time. They can learn from past experiences or based on feedback from the environment.

3. Spoofing and Jamming GNSS and their effect on Swarms

The focus of this work is on the dependence of swarm architectures and algorithms to information regarding its agents' location. To have access to such information, drones rely on a variety of sensors. Among those, GNSS-based positioning stands out in outdoor environments due to its availability, reliability, and accuracy. However, as argued earlier, GNSS known vulnerabilities include the ability of attackers to either jam or spoof the used ranging signals. In the context of a swarm of agents, different threat models can be considered depending on the swarm centralized/distributed architectures.

Jamming and spoofing attacks provide an attacker with different ways of intervening in the positioning and swarming process. While a jamming attack (to one agent, a subset, or the totality of the swarm) would deny positioning for the affected agent(s), a spoofing attack would deceive the agent(s) into believing forged position information. It has been proven that an attacker can perform both attacks, while when it comes to spoofing single entities on a swarm there are many practical challenges associated to the directivity of the transmission that potentially makes this attack unfeasible with current technology.

Additionally, when it comes to spoofing, one typically assumes that an attacker has *accurate* knowledge of the position of the target agent(s) (Sathaye et al., 2022a). Again, there are practical challenges associated to this capability such as $i)$ being able to compute the victim's position estimation with the necessary accuracy, and $ii)$ that the processing latency should be small enough such that the forged GNSS signal reaches the target on-time to fake the position at the desired instant in time (Kerns et al., 2014; Kling et al., 2022).

Finally, the attacker is assumed to not be capable of injecting forged positions directly over the communication links, as they generally have some sort of security layer (such as cryptography coding). Therefore, for the attacker to manipulate positions, the only viable backdoor is to forge GNSS signals and perform a spoofing attack.

For centralized swarms, the central agent that is in charge of the main computations constitutes a single point of failure, which an attacker might aim to target by attacking the position estimates from the central agent, the other agents in the swarm, or a combination of both attacks. The attacks could be: $i)$ to jam the signal from one or several agents in the swarm, in which case their location would not be available to the central agent; $ii)$ to jam the location of the central agent, which is likely to disrupt the entire operation on a centralize scheme; $iii)$ to spoof the position of swarm agent(s), which would cause the centralize algorithm to incorrectly predict next location, thus potentially leading to collisions unless additional measures are taken; or $iv)$ spoof the central agent location, which could cause general misbehaviour of the swarm algorithm.

In decentralized swarm schemes, the control algorithm is distributed across the agents such that no single point of failure exists. However, the effects of both jamming and spoofing to a subset (or the entirety) of the swarm are unknown. Similarly, $i)$ jamming would cause the position of the specific agent to be unavailable to itself and its neighbors, which could cause erratic behavior of that agent(s); and $ii)$ spoofing would cause a misleading position to be spread over the communication network, thus making the control algorithm to have mismatched information.

While the above classification of swarm architectures and GNSS attacks provide a broad overview, it's essential to understand that in real-world applications, many swarm behaviors can overlap or fall into multiple categories. In many practical scenarios, spatial organization becomes almost a precursor to several task-specific algorithms. For example, a swarm of robots have to cover an agricultural field (a spatial organizaton problem) to identify areas that need irrigation or pesticide treatment (a resource foraging problem). Therefore, in this work, we focus on the impact of GNSS spoofing within the context of spatial organization (e.g., formation and flocking)

III. ANALYSIS OF THE IMPACT OF GNSS SPOOFING ON SWARM FORMATION

At its core, swarm robotics seeks to leverage the collective power of numerous simple agents to perform complex tasks. Key to this is ensuring that agents can effectively distribute themselves and coordinate in an environment i.e., the spatial organization or formation problem becomes imperative. Although there are several algorithms as mentioned in this extensive survey (Brambilla et al., 2013) addressing the spatial organization problem, in the paper, we focus on Voronoi diagrams and Lloyd's relaxation algorithm. The combination of Voronoi diagrams, swarm principles, and Lloyd's relaxation plays a pivotal role in advancing swarm robotics. In the following, we give a brief overview of the Voronoi diagrams and Lloyds relaxation algorithm and how position manipulation can impact the outcome.

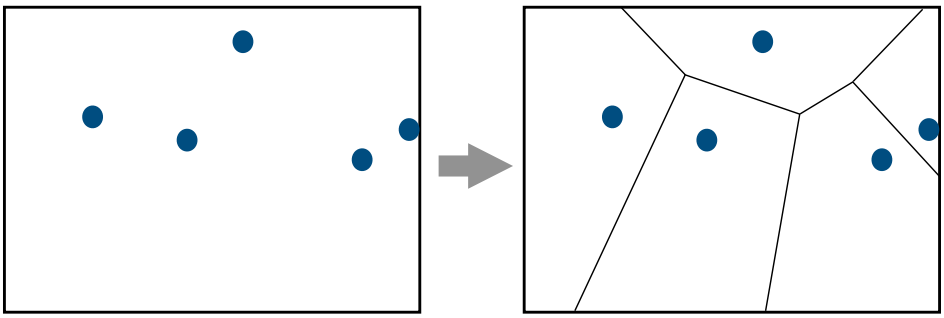


Figure 2: Example of the Voronoi diagram with $n = 5$ seeds. Cells contain points in space that are closer to the corresponding seed (in blue).

1. Voronoi tessellation and Lloyds algorithm

Voronoi tessellation, also known as Voronoi diagram, is a well-known mathematical concept in computational geometry and is used in various fields. It consists of the partitioning of a plane with n generated seeds into convex polygons called Voronoi cells with each polygon containing exactly one seed, such that for any given point in a given cell it is the closest to its seed than to any other. Notice that this is precisely, the purpose of coverage algorithms in swarms of unmanned vehicles.

For a general definition of Voronoi diagrams, consider $\mathcal{X}$ to be a metric space with distance function $d : \mathcal{X} \mapsto \mathbb{R}$ (for instance, the Euclidean distance is commonly used in 2-D applications of Voronoi, $\mathcal{X} = \mathbb{R}^2$, and will serve as the distance function for our experiment without loss of generality). Given a set $\mathcal{S}$ of n seeds in the space, we wish to associate with each seed $\mathbf{s}_i \in \mathcal{X}$, $i \in \{1, \ldots, n\}$, a cell region consisting of all the points in the space closer to $\mathbf{s}_i$ than to any other $\mathbf{s}_j$, $j \in \{1, \ldots, n\} \backslash \{i\}$. We can formalize this definition of a Voronoi region for the $\mathbf{s}_i$ seed (or central node) as:

$$\mathsf{Vor}(\mathbf{s}_i) = \left\{ \mathbf{p} \in \mathcal{X} \mid d(\mathbf{s}_i, \mathbf{p}) \leq d(\mathbf{s}_j, \mathbf{p}) \, , \forall \mathbf{s}_j \in \mathcal{S} \backslash \{\mathbf{s}_i\} \right\} \tag{1}$$

with $\mathsf{Vor}(\mathbf{s}_i)$ being the Voronoi cell for seed s_i, and $\mathbf{p} \in \mathcal{X}$ denoting an arbitrary point in the space to be tessellated. This point $\mathbf{p}$ belongs to the cell associate to site $\mathbf{s}_i$ when its distance to any other site point $\mathbf{s}_j$ is larger. Again, in the context of UAVs and maximizing coverage, the site points $\mathbf{s}_i$ would correspond to UAVs locations.

Constructing the Voronoi diagram involves defining vertices and edges delimiting each region. Vertices are points with three or more equally distant regions. The fundamental property of Voronoi cells is preserved by drawing edges perpendicularly to the midpoint between every pair of neighboring seeds (see Figure 2 for an example tessellation on $n = 5$ arbitrarily placed site points $\mathbf{s}_1, \ldots, \mathbf{s}_5$). Several efficient algorithms have been proposed for constructing Voronoi diagrams leveraging the Delaunay triangulation.

Lloyd's algorithm, or Voronoi relaxation, is an iterative computational technique for redistributing points evenly in a region and partitioning subsets of a Euclidean space into uniformly sized convex cells. After the Voronoi tessellation has been computed for the initial seed locations, Lloyd's algorithm iteration propagates each seed to the centroid of its corresponding Voronoi cell and this operation is repeated until the position of the seed is equal to the centroid and a guaranteed convergence is achieved; a degree of tolerance can be introduced to reach the state earlier (see Figure 3 for an example of how the iterative procedure modifies the Voronoi cells). The centroid of each Voronoi cell is calculated at every iteration and this is accomplished using the vertices of the closed polygonal regions formed by the Voronoi diagram around each seed. Let $\mathcal{V}_i$ be the set of m_i vertices for the corresponding cell of the seed $\mathbf{s}_i \in \mathcal{S}$. In the case of interest here $\mathcal{X} = \mathbb{R}^2$, we denote the position of the k-th vertex for the i-th seed by $\mathbf{p}_{k,i} = (x_{k,i}, y_{k,i})^\top$, in which case $\mathcal{V}_i$ is a set that contains those locations $\mathcal{V}_i = \{\mathbf{p}_k \mid k \in \mathbb{N},\ 0 < k \leq m\}$ and the position of the centroid $\mathbf{c}_i = (c_{x,i}, c_{y,i})^\top$ can be computed in closed-form as (Cortes et al., 2004):

$$c_{x,i} = \frac{1}{6A_i} \sum_{k=1}^{m_i} (x_{k,i} + x_{k+1,i})(x_{k,i}y_{k+1,i} - x_{k+1,i}y_{k,i}) \tag{2}$$

and

$$c_{y,i} = \frac{1}{6A_i} \sum_{k=1}^{m_i} (y_{k,i} + y_{k+1,i})(x_{k,i}y_{k+1,i} - x_{k+1,i}y_{k,i}) \, , \tag{3}$$

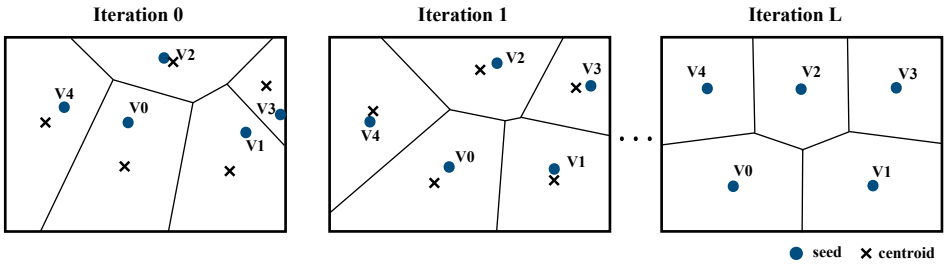


Figure 3: Results of Lloyd's Algorithm on a plane with $n = 5$ nodes and L iterations. The initial location of the nodes (in blue) is the seed to the next iteration of the algorithm, which produces a new target position for the nodes ($\times$) as the centroid of the newly computed Voronoi cell.

where A_i is the signed area of the cell i:

$$A_i = \frac{1}{2}\sum_{k=1}^{m_i}(x_{k,i}y_{k+1,i} - x_{k+1,i}y_{k,i}) \ . \tag{4}$$

2. System Assumptions and Threat Model

For our analysis, we assume the following system setup. A group of drones is tasked with surveilling a specific geographic region, with each drone designated to cover an equal area. To achieve this uniform distribution, the system employs Voronoi tessellation in conjunction with Lloyd's relaxation. Each drone determines its location using GNSS and broadcasts this position to its peers. In real-time, every drone receives the locations of its neighboring drones and computes its Voronoi tessellation based on this localized information. After determining its Voronoi cell, each drone navigates to the centroid of its designated cell and subsequently broadcasts its updated location. Upon receiving these new positions, the drones recalibrate, recompute their tessellations, and adjust again. This iterative process continues until the variance in the area covered by each drone falls below a pre-defined threshold. Once this condition is met, indicating a balanced coverage, the system is considered to have reached a stable state, and no further iterations are pursued.

As described previously, in this work, we focus on GNSS spoofing attack and its impact on swarm formation. In our research, we examine an adversary with the ability to create and send GPS signals. While the attacker's specific objective could vary widely ranging from forcing a drone to collide with other drones to creating a surveillance blackout area, we assume the primary outcome of the spoofing attack is to force the swarm to behave differently than originally intended. It's presumed that the UAV(s) falls within the radio range of the attacker and can, therefore, pick up these counterfeit signals. We further hypothesize that the adversary has successfully compromised the UAVs' GPS receiver, either through a seamless takeover technique as detailed in (Tippenhauer et al., 2011; Ranganathan et al., 2016) or via a non-coherent overshadow attack. The susceptibility of standalone GPS receivers to spoofing has been thoroughly explored in previous studies (Tippenhauer et al., 2011; Nighswander et al., 2012; Psiaki and Humphreys, 2016; Jiang et al., 2013).

3. Impact of Position Manipulation

To gauge the repercussions of GNSS spoofing on drones, we selected specific key impact metrics for evaluation. Chief among these are the change in the number of iterations required for convergence, which directly relates to the time taken for convergence, and the alteration in the distance covered by the members of the swarm, indicative of their resource or power consumption.

GNSS spoofing can distort the location perceived by drones. It's reasonable to assume that such spoofing would impact only a subset of a drone swarm. This is largely due to the challenges of spoofing over a broad geographical region that a full swarm might occupy. Drones that are targeted will then estimate and broadcast incorrect locations, leading to flawed positional data being incorporated into the Voronoi tessellation calculations of neighboring drones. The consequences of this are multifaceted. For one, the algorithm may not converge, potentially causing drones to remain in motion for extended durations and depleting their batteries. Furthermore, this can create uneven surveillance patterns, with some drones covering areas too vast for them, leading to surveillance blind spots. If anti-collision measures are absent, drones might even collide. The algorithm's dynamics become particularly intriguing when multiple drones, affected by spoofing, report identical locations. In such cases, the algorithm has to interpret this as two drones occupying the same spot, introducing additional complexities and potential system discrepancies.

In our analysis, we delve into three distinct spoofing strategies to understand their impact on drone behaviors: i) "Fixed Spoofing," where the false location remains constant throughout the spoofing duration, ii) "Relative Spoofing," in which the deceitful location is set as a subtle deviation from the drone's genuine location, and iii) "Random Spoofing," where the

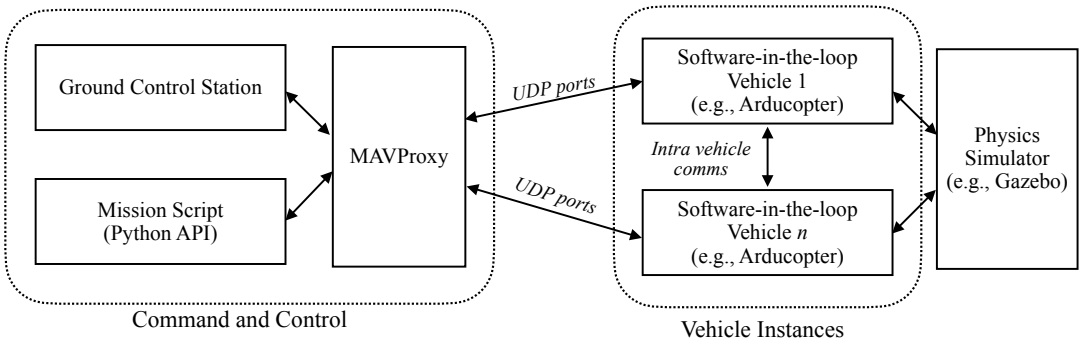


Figure 4: Real-world swarm simulation framework.

misleading location is chosen arbitrarily. By examining these strategies, we aim to capture a broad spectrum of potential threats, providing valuable insights into the swarm formation algorithm's resilience. Naturally, the depth of this study goes beyond these strategies, as several other parameters merit attention for their potential influence on the outcome, such as the choice of the drone targeted, the specific location used for the spoof, and the swarm's physical configuration at the time of the attack. Another pivotal variable is the spoofing duration. We would like to highlight the distinction between continuous spoofing, where false signals are sent during the entire algorithm's operation, and intermittent spoofing. Our initial hypothesis suggests that unrelenting spoofing might prevent the algorithm from converging, tilting the threat more towards a denial-of-service attack. Moreover, constant spoofing signal transmission could betray the attacker's location, potentially making it an imprudent choice for a savvy adversary. Given these considerations, our research places significant emphasis on the effects of partial spoofing, which will be the focal point of our ensuing discussions in the paper.

IV. EXPERIMENTAL EVALUATION

1. Simulation Setup

For our experimental analysis, we employ two distinct setups: a near real-world swarm simulation framework (Figure 4) and a python-based algorithm simulator. The former provides a comprehensive representation of real-world conditions. It is composed of three primary components: a command and control center, vehicle instances, and a physics simulator. The ground control post is replicated in the command and control center, using QGroundControl and MAVProxy — two applications facilitating drone control through the MAVLink protocol. Each vehicle instance merges an Arducopter with a software-in-the-loop model, allowing seamless communication with the control center over UDP ports. Our framework can simulate intra swarm communication through standard networking ports wherever necessary. To replicate the environment and its physics, Gazebo is integrated, enabling high-fidelity sensor streams specific to diverse physical settings.

On the other hand, the python-based algorithm simulator offers an interactive platform, focusing predominantly on the algorithm's intricacies. This simulator allows us to methodically iterate the algorithm and analyze the swarm's behavior, eschewing detailed environmental or drone modeling. At every iteration, users can choose which vehicle to spoof and decide on the nature of the location spoofing, be it absolute or relative. Additionally, the simulator can operate in a non-interactive mode, taking in parameters at the start, such as spoofing duration, vehicle ID, and spoofing location. A practical feature of the simulator is its ability to provide visual feedback, aiding in understanding how location manipulation affects the swarm's behavior.

2. Evaluation Metrics and Results

In our study, we primarily focused on two evaluation metrics: i) the percentage change in the number of iterations required for the algorithm to converge, and ii) the percentage change in the total distance traveled by the drones. For our experiments, we varied the starting positions of the swarm's drones, referred to as initial seed locations, and applied each of the three spoofing techniques: fixed, relative, and random location manipulation. Furthermore, we probed the consequences of manipulating the location for 1, 2, and 3 iterations of the algorithm's execution time, considering each initial seed location and spoofing method. It's noteworthy that each simulation was executed 75 times for every initial seed location. During each run, the drone being spoofed and the exact spoofed location were varied in accordance with the chosen spoofing technique.

a) Impact on Algorithm Convergence Time

Our analysis results regarding the effects of location manipulation within the Voronoi / Lloyd's relaxation algorithm for swarm formation can be viewed in Figure 5. From the data, it's evident that manipulating drone locations does influence the algorithm's convergence time for all the initial seed locations analyzed. An intriguing observation from these results is that for certain

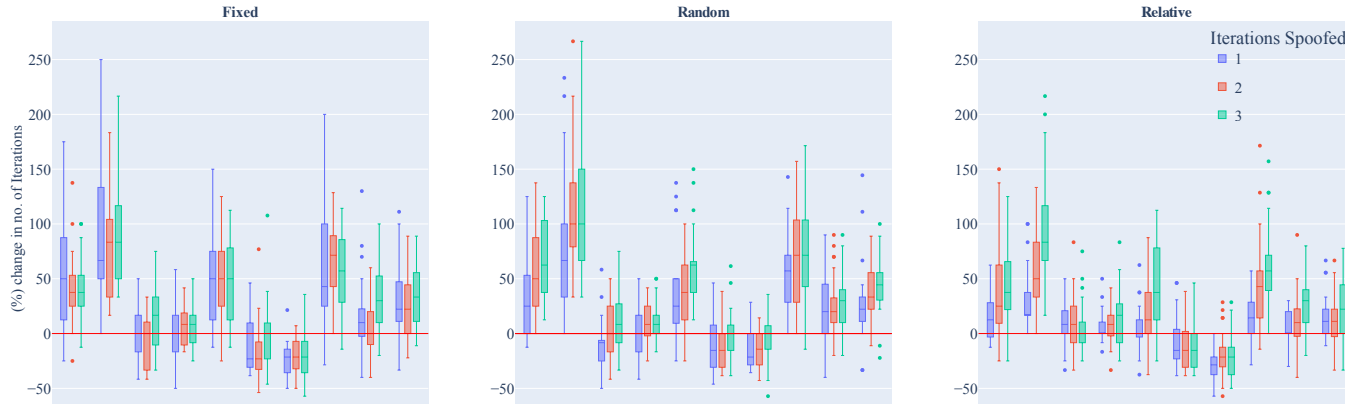


Figure 5: Impact of spoofing on algorithm convergence time for 'fixed', 'random', and 'relative' spoofing attacks. The plots show results for 10 different, randomly-drawn, initial locations of the drones.

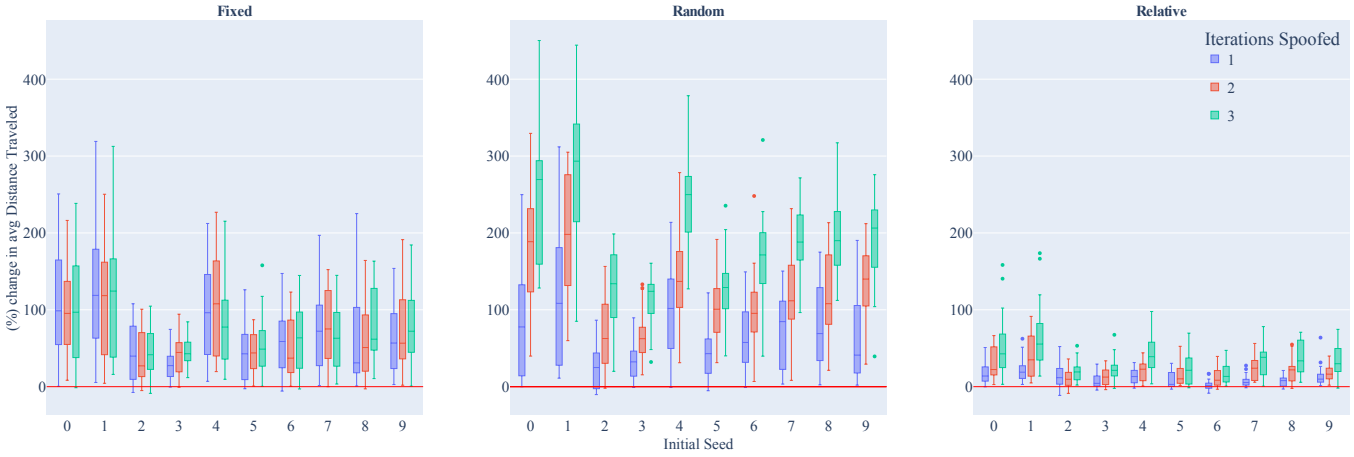


Figure 6: Impact of spoofing on distance traveled by the swarm entities before convergence for 'fixed', 'random', and 'relative' spoofing attacks. The plots show results for 10 different, randomly-drawn, initial locations of the drones.

initial location seeds, spoofing can unexpectedly expedite the algorithm's convergence. This phenomenon can be attributed to the fact that some inherent swarm configurations naturally lead to a prolonged convergence, and location manipulation, in these instances, might inadvertently streamline the process. However, notice that this faster convergence might not yield to a desirable formation, and instead one which the attacker designed. This presents a potential strategy for adversaries: recognizing and consistently spoofing locations of specific drones in such configurations. It is worth noting that for certain seed locations, the algorithm's convergence time increased by more than 200%, even with location spoofing limited to a single iteration. This highlights the imperative for a deeper exploration in this domain.

b) Impact on Travel Distance

In a subsequent analysis, we shifted our focus to evaluate the impact of spoofing location on the distance traveled by the swarm entities. The results, illustrated as a box plot, represent the percentage change in the cumulative distance covered by the entire swarm. It's observable from the data that spoofing typically results in an extended travel distance compared to the baseline distance in the absence of spoofing. Notably, certain swarm configurations, when subjected to spoofing, led to a near 400% increase in the distance traveled by the swarm entities, amounting to roughly five times the typical distance. When considering relative spoofing, the impact is more restrained, with the distance traveled not exceeding twice the usual amount. This outcome aligns with expectations, given that relative spoofing introduces only a marginal location offset, as opposed to a more drastic location change.

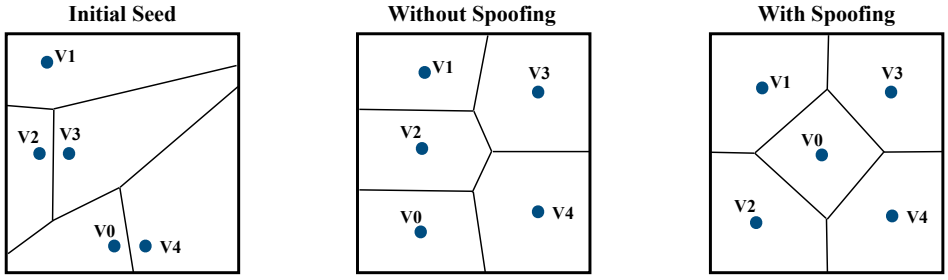


Figure 7: Example case illustrating the impact of spoofing on the ultimate spatial formation with and without spoofing for the same initial seed location for the swarm.

c) Impact on Final Spatial Organization

In our experimental analysis, we observed that the final spatial configuration upon convergence could differ based on location manipulations for specific durations during the algorithm's execution. As illustrated in Figure 7, the ultimate regions designated to each drone vary, depending on the presence or absence of spoofing. This observation suggests the potential for an attacker to craft and time spoofing signals, aiming to influence the swarm's spatial orientation post-convergence. For instance, it's conceivable that a compromised drone might be manipulated to monitor a particular area, thereby inducing a surveillance blind spot. Similarly, an adversary could aim to allocate a vast region to a drone, placing its centroid at a distance from the drone's surveillance scope, which would result in a coverage gap. Such findings indicate an avenue for further research, exploring the prospect of achieving specific outcomes by merely spoofing the GNSS signals within a swarm.

V. CONCLUSION

In our study, we were guided by previous research highlighting the challenges of GNSS spoofing and the nuances of controlling individual UAVs. However, the broader impact of GNSS spoofing on drone swarms remained less explored. Our work sought to understand this area, particularly focusing on the swarm formation algorithms, as formation is an essential precursor to many swarm tasks. Using our custom-designed simulation frameworks, we examined the effects of GNSS spoofing. Some significant findings include: spoofing led to nearly a $3\times$ increase in convergence time, caused more than a $5\times$ surge in the cumulative distance traveled by the drones, and notably influenced the final spatial formation of the swarm. Moving forward, we see a need to further explore the practical considerations for adversaries aiming to exploit GNSS spoofing. Additionally, a thorough evaluation of various swarm algorithms against potential adversarial tactics is warranted to develop more robust solutions.

ACKNOWLEDGEMENTS

This work has been partially supported by the NSF under Awards CISE-2144914, ECCS-1845833, and CCF-2326559.

REFERENCES

Abdelkader, M., Güler, S., Jaleel, H., and Shamma, J. S. (2021). Aerial swarms: Recent applications and challenges. *Current robotics reports*, 2:309–320.

Amin, M. G., Closas, P., Broumandan, A., and Volakis, J. L. (2016). Vulnerabilities, threats, and authentication in satellite-based navigation systems [scanning the issue]. *Proceedings of the IEEE*, 104(6):1169–1173.

Bayındır, L. (2016). A review of swarm robotics tasks. *Neurocomputing*, 172:292–321.

Borio, D., Dovis, F., Kuusniemi, H., and Presti, L. L. (2016). Impact and Detection of GNSS Jammers on Consumer Grade Satellite Navigation Receivers. *Proceedings of the IEEE*, 104(6):1233–1245.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41.

Cortes, J., Martinez, S., Karatas, T., and Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255.

Dovis, F. (2015). *GNSS Interference Threats and Countermeasures*. Artech House.

Du, Q., Faber, V., and Gunzburger, M. (1999). Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*,

41(4):637–676.

Ioannides, R. T., Pany, T., and Gibbons, G. (2016). Known vulnerabilities of global navigation satellite systems, status, and potential mitigation techniques. *Proceedings of the IEEE*, 104(6):1174–1194.

Jiang, X., Zhang, J., Harding, B. J., Makela, J. J., Domı, A. D., et al. (2013). Spoofing GPS receiver clock offset of phasor measurement units. *IEEE Transactions on Power Systems*.

Kerns, A. J., Shepard, D. P., Bhatti, J. A., and Humphreys, T. E. (2014). Unmanned aircraft capture and control via GPS spoofing. *Journal of field robotics*, 31(4):617–636.

Kling, M. T., Lau, D., Witham, K. L., Closas, P., and LaMountain, G. M. (2022). System for Closed-Loop GNSS Simulation. US Patent App. 17/662,822.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

MIT Tech Review (2019). Ghost Ships, Crop Circles, and Soft Gold: A GPS Mystery in Shanghai.

Morton, Y. J., van Diggelen, F., Spilker Jr, J. J., Parkinson, B. W., Lo, S., and Gao, G. (2021). *Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications*. John Wiley & Sons.

Nighswander, T., Ledvina, B., Diamond, J., Brumley, R., and Brumley, D. (2012). GPS software attacks. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*.

Psiaki, M. and Humphreys, T. (2016). GNSS Spoofing and Detection. *Proceedings of the IEEE*, 104(6):1258–1270.

Ranganathan, A., Ólafsdóttir, H., and Capkun, S. (2016). SPREE: A spoofing resistant GPS receiver. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*.

Sathaye, H., LaMountain, G., Closas, P., and Ranganathan, A. (2022a). Semperfi: Anti-spoofing GPS receiver for UAVs. In *Network and Distributed Systems Security (NDSS) Symposium 2022*.

Sathaye, H., Strohmeier, M., Lenders, V., and Ranganathan, A. (2022b). An experimental study of {GPS} spoofing and takeover attacks on {UAVs}. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3503–3520.

Tippenhauer, N. O., Pöpper, C., Rasmussen, K. B., and Capkun, S. (2011). On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*.