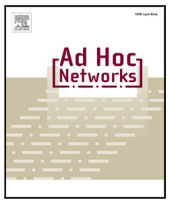
Ad Hoc Networks 184 (2026) 104135

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



DRUID: Coordinating drone movements for compromised node identification

Mauro Farina *, Erica Salvato, Martino Trevisan, Alberto Bartoli

Department of Engineering and Architecture, University of Trieste, Via Alfonso Valerio 6/1, 34100, Trieste, Italy

ARTICLE INFO

Keywords:
UAV network
MANET
Compromised node
Node relocation
Network connectivity
Mobile nodes

ABSTRACT

In recent years, Unmanned Aerial Vehicles (UAVs) (also called drones) networks have become increasingly popular in scenarios where rapid deployment, flexible mobility, and real-time data acquisition are crucial, such as disaster relief, environmental monitoring, military operations, and smart city infrastructure. However, due to their dynamic nature and dependence on wireless communication, they are intrinsically vulnerable to a variety of cyberattacks. In this work, we present DRUID, a decentralized scheme for silently identifying a compromised drone that selectively alters the messages it forwards. The scheme uses a combination of secret sharing and multipath routing to allow a pair of communicating drones, namely A and B, to detect the presence of a compromised drone along any route between them, thereby categorizing each route as either safe or compromised. The scheme operates iteratively and consists of three key modules: (i) an Information Retrieval Procedure that allows A to learn more about the topology, (ii) a binary search-like Identification Procedure, and (iii) if the previous module fails to identify the compromised drone, a Node Repositioning Procedure that relocates nodes closer to the compromised path. We validate DRUID on a large and diverse set of 178 731 graphs representing realistic UAV networks with different communication ranges. Comparing our scheme to previous work, experiments show that DRUID achieves a 97 % identification rate—up from the 54 % of the most recent alternative approach. We analyze the cost associated with the node repositioning procedure in terms of computation time and drone movement, and show that it generally takes a few seconds.

1. Introduction

The past decade has seen a remarkable increase in interest and research in the field of Unmanned Aerial Vehicle (UAV) networks. The ability of UAVs (also known as *drones*) to move in three dimensions freely, together with the flexibility offered by infrastructure-less networks, makes UAV networks ideal for many applications where traditional solutions may not be available, e.g., military [1] and rescue operations [2], disaster relief [3], and wildfire detection [4]. However, UAV networks are inherently vulnerable to a range of cyberattacks due to their dynamic nature and dependency on wireless communications [5–8]. The lack of physical boundaries facilitates intrusion, eavesdropping, and node compromise, making UAV networks subject to several threats. While traditional cryptography and encryption can mitigate the risk of information leakage to unauthorized third parties, a legitimate but compromised drone can silently disrupt the UAV network operations, for example, by sharing false information [9], by increasing the energy consumption of nodes through routing attacks [10], or by altering routed messages [11]. Consequently, promptly detecting and mitigating compromised nodes is crucial for ensuring the secure and effective functioning of UAV networks.

In this work, we present DRUID, a scheme for *silently* detecting and identifying a compromised drone that selectively alters the messages it forwards. The scheme uses a combination of secret sharing and multipath routing to enable a pair of communicating drones, namely A and B, to *detect* the presence of a compromised drone along any route between them, thereby categorizing each route as either safe or compromised. Then, A acts as the coordinator of an iterative procedure for strategic *node repositioning* that enables the *identification* of the compromised drone. The identification procedure relies exclusively on information obtained along safe routes, thereby reducing the number of interactions with the compromised node and minimizing the risk of alerting it. DRUID operates in a decentralized manner, requiring no central authority (such as a ground control station) while also integrating seamlessly with existing routing protocols.

Our proposal builds on previous work in this area. In particular, [12] first introduced SPREAD, a scheme that uses secret sharing and multipath routing to prevent a single adversary from both recovering the original message and disrupting communication. More recently, [11] proposed SpiderMiners, an extension of SPREAD that incorporates an identification scheme capable of identifying the specific

* Corresponding author.
E-mail addresses: mauro.farina@phd.units.it (M. Farina), erica.salvato@dia.units.it (E. Salvato), martino.trevisan@dia.units.it (M. Trevisan), bartoli.alberto@units.it (A. Bartoli).

https://doi.org/10.1016/j.adhoc.2026.104135
Received 24 April 2025; Received in revised form 13 November 2025; Accepted 4 January 2026
Available online 5 January 2026

node altering the routed messages. DRUID builds on SpiderMiners by introducing a modular pipeline consisting of an improved identification scheme and a novel node repositioning strategy, which together drastically increase the adversary identification rate.

We validate DRUID on a large and diverse set of 178 731 graphs representing realistic UAV networks having different communication ranges and route lengths. Using an earlier state-of-the-art proposal (SpiderMiners) as a baseline, our experiments reveal a significant improvement in silent identification rates, with SpiderMiners achieving 54 % and DRUID reaching 97 %. We investigate how graph connectivity and the number of disjoint paths influence the ability of each scheme to identify the attacker, finding a strong correlation with both factors.

The remainder of the paper is organized as follows. Section 2 introduces Secret Sharing and Multipath Routing, two essential concepts behind DRUID. In Section 3 we present DRUID's architecture and the operation of its building blocks, while Section 4 describes the different relocation strategies that we propose. Section 5 details our experimental setup and dataset, which we use to evaluate DRUID in Section 6 and Section 7 in terms of identification rate and execution time, respectively. Finally, we summarize related works in Section 8, while we discuss our findings and draw conclusions in Section 9.

2. Background

A (t, n)-secret sharing scheme is a cryptographic tool that involves three parties: a trusted authority known as the *dealer*, a set of n *participants*, and a *combiner*. In this scheme, the dealer generates and encodes a secret into n parts, commonly referred to as *shares*, and assigns a distinct share to each participant. The combiner decodes the secret when any subset of t or more participants submits their shares.

Shamir [13] and Blakley [14] independently introduced the first schemes in 1979, with the former gaining more popularity due to its simplicity and space efficiency [15]. Both these schemes assume that all the parties involved in the process are trusted, making them vulnerable to different kinds of cheating. For example, a participant could submit a fake share during the secret recovery process, causing the reconstruction of a different secret [15]. In 1995, Wu and Wu [16] proposed an algorithm that can be incorporated in secret sharing schemes, including [13], for detecting cheating by participants and deterministically identifying the cheater(s).

Secret sharing with cheating detection distributes trust and reliability among multiple entities, thereby eliminating single points of failure. As a result, secret sharing has found numerous applications in UAV networks, e.g., to identify compromised drones [11], authenticate nodes [17–19], and improve routing resilience [12].

Routing in UAV networks faces several challenges because the high mobility and speed of drones lead to frequent changes in topology and link disruptions [6,20]. Due to the limited capacity of batteries, nodes may be frequently added or removed from the network to ensure mission accomplishment.

Multipath routing is an effective strategy to mitigate the effects of frequent link disruptions: by establishing multiple routes between the source and the destination, nodes can introduce a redundancy that facilitates continuous communication in both dynamic and adversarial environments, i.e., in the presence of malicious nodes [21]. In this context, [12] introduced SPREAD, a scheme that integrates secret sharing and multipath routing across node-disjoint routes to enhance resilience against eavesdropping, infiltration, and node compromise attacks. In SPREAD, a node A, wishing to send a message to another node B, acts as the dealer, whereas B is responsible for reconstructing the secret (i.e., the combiner). A splits the message into n shares and routes each on a different node-disjoint route; such routes serve the role of participants of the (t, n)-secret sharing scheme. The combination of secret sharing and multipath routing ensures that (i) no node other than B can reconstruct the message, and (ii) B can recover the original message with only t shares out of the n generated and sent by A. Therefore, the communication tolerates up to $n - t$ missing or invalid (altered) shares. However, SPREAD alone cannot identify the adversary.

Recently, [11] proposed SpiderMiners, a scheme that builds upon SPREAD by (i) implementing the aforementioned cheating detection and cheater identification scheme [16], thereby allowing B to identify the specific route where a share has been altered, and (ii) introducing a binary-search like identification procedure that deterministically pinpoints the compromised drone. In the following section, we provide an in-depth description of DRUID, including the changes and improvements applied to SpiderMiners's original procedures.

3. DRUID

In this section, we provide an overview of our DRUID (Detection of Rogue UAV through Iterative Displacement) proposal: an iterative scheme that silently identifies a compromised drone that selectively alters the messages it forwards. It comprises three building blocks, and Fig. 1 illustrates the overall pipeline. DRUID is employed when a drone, which we refer to as B, detects the presence of an adversary along one of the disjoint routes used by another drone, referred to as A, to split and transmit a message. Following this initial detection, B initiates the *Information Retrieval Procedure* (IRP) to enable A to execute the *Identification Procedure* (IDP). Should the IDP fail to pinpoint the compromised drone, A coordinates a strategic relocation of the drones according to the *Node Repositioning Procedure* (NRP), and IRP and IDP are repeated. The process ends when A identifies the compromised drone or when it judges the problem to be unsolvable. In the following sections, we describe each block.

3.1. Network and threat model

Let $G = (N, E)$ be an undirected connected graph representing a 2D UAV network of drones, each with a communication radius R_c. Each node $n_i \in N$ represents a drone, whereas edges represent the ability of the corresponding nodes to communicate between them, i.e., the set of edges is given by $E = \{(n_i, n_j) \mid d(n_i, n_j) \leq R_c,\ n_i, n_j \in N,\ n_i \neq n_j\}$, where $d(\cdot, \cdot)$ denotes the euclidean distance between two nodes.

The network uses Dynamic Source Routing (DSR) [22], a well-known routing protocol for ad hoc networks. DSR is a reactive protocol, meaning that the routes connecting a drone $A \in N$ wishing to communicate with another drone $B \in N$ are discovered on demand. In DSR, A specifies the route in the packet header, allowing each node along the path to see the entire route the packet needs to take.

The adversary is a single compromised node $n_c \in N$ within the UAV network. The following assumptions are made regarding the adversary: it (i) identifies as a legitimate node, (ii) has access to all encryption keys of that node, and (iii) selectively alters routed messages.

We say that a route $r_{X,Y}$ from a node X to a node Y is a sequence of nodes $n_1, n_2, \ldots, n_l$ such that: (i) $l \geq 3$; (ii) $n_1 = X$ and $n_l = Y$; (iii) $\forall i \in [1, l), d(n_i, d_{i+1} \leq R_c)$. We will omit the specification of the route endpoints when ambiguities cannot arise.

Let $A, B \in N$ be two drones such that $d(A, B) > R_c$ and let m be a secret message that A wants to send to B. Transmission of m occurs as follows.

1. Using the SPREAD scheme, A determines a set $\mathcal{R}$ of at least 2 node-disjoint routes from A to B. Let n denote the number of such routes (i.e., $n = |\mathcal{R}| \geq 2$).
2. A selects a value $t \in [2, n]$.
3. Using the (t, n)-threshold secret sharing scheme by Shamir, A generates n shares for m and sends each share to B along one of the routes in $\mathcal{R}$, selecting a different route for each share.
4. To enable B to detect and handle attacks on the routing, A includes the routing information in each packet header using a secret sharing scheme, as described in [11].

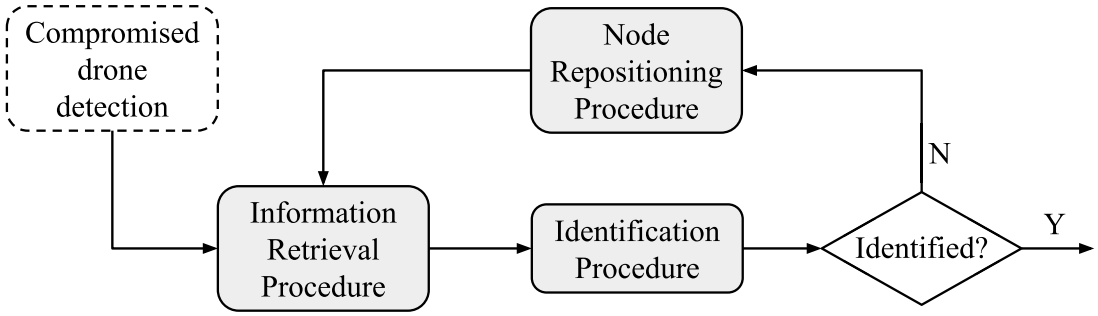


Fig. 1. Pipeline of the modules employed by DRUID.

When B has received a sufficient number of shares, B reconstructs the secret message m. By using Wu and Wu's cheating detection-cheater identification scheme [16], B determines whether the message is intact or one of its shares has been altered. In the latter case, B is also able to determine the *compromised route* $r_c \in \mathcal{R}$, i.e., the route followed by the altered share and that, by definition, is the only route in $\mathcal{R}$ that includes the compromised node n_c. We refer to the remaining routes $\mathcal{R}_s = \mathcal{R} \setminus \{r_c\}$ as *safe routes*, and the nodes composing them as *safe nodes* $\mathcal{N}_S = \bigcup_{r_s \in \mathcal{R}_S} \{n_i \in r_s\}$.

3.2. Information Retrieval Procedure (IRP)

The Information Retrieval Procedure (IRP) module is used upon detection of an adversary on a certain route r_c. When B initiates the IRP, it enables drone A to gather all the information needed to execute the Identification Procedure and the Node Repositioning Procedure. We distinguish two versions of the IRP. The first, $\mathsf{IRP}_{\mathrm{SM}}$, corresponds to the identification procedure used in SpiderMiners [11]. The second, $\mathsf{IRP}_{\mathrm{DR}}$, improves $\mathsf{IRP}_{\mathrm{SM}}$ and allows A to learn more about the topology, thereby accounting for the possibility of relocating the nodes within the deployment area.

IRP_{SM}: SpiderMiners's information retrieval procedure

Given a route r and a node $n \notin r$, we denote by $\Omega_r(n)$ the set of nodes in r that are within the communication radius of n. Given a node n_i in the compromised route r_c, we say that an *identification route* for n_i, denoted $r_I(n_i)$, is a route that satisfies the following properties: (i) it starts with a prefix of a safe route r_s; (ii) it ends with a suffix of r_c; (iii) the two sub-routes are connected by a one-hop link (n_s, n_i), with $n_s \in r_s$

Node B constructs an *identification request* message m^{Id}_{REQ} containing the list of the nodes on r_c. Then, B sends m^{Id}_{REQ} to A along each safe route $r_s \in \mathcal{R}_S$. Each safe node n_s along each safe route r_s that forwards m^{Id}_{REQ} appends to this message the respective set of neighboring nodes $\Omega_{r_c}(n_s)$.

Once A receives m^{Id}_{REQ} and thus learns that r_c is a compromised route, A constructs identification routes for all nodes in r_c and initiates the identification procedure to pinpoint the compromised node n_c in r_c.[1]

Fig. 2 shows two examples of identification routes: $r_I(n_4)$, where r_s and r_c are connected by the one-hop link (n_1, n_4), and $r_I(n_6)$, where the link (n_3, n_6) connects the two routes.

IRP_{DR}: DRUID's enhanced procedure

$\mathsf{IRP}_{\mathrm{DR}}$ extends the previous one, starting with the following assumptions.

- The drones are stationary. This hypothesis aligns with several area coverage missions [20] (e.g., communication relay operations); the only expected movement is to establish new identification routes as detailed in the following.
- Each node n_i has access to (p^x_i, p^y_i), its current position (relative to the area of interest).
- Each node n_i is equipped with a proximity sensor of radius $R_p < R_c$ that measures distances to nearby nodes, without disclosing their identities. We denote with $\mathcal{P}_i = \{(d^x_{i_1}, d^y_{i_1}), \ldots, (d^x_{i_m}, d^y_{i_m})\}$ the set of vectors connecting n_i with such nodes.

The goal of $\mathsf{IRP}_{\mathrm{DR}}$ is to facilitate the execution of the Node Repositioning Procedure by drone A. To this end, B constructs an *identification and position request* message, denoted as $m^{Id,Pos}_{REQ}$. The message extends m^{Id}_{REQ} by requiring each node $n_i \in r_s$ forwarding $m^{Id,Pos}_{REQ}$ to A to append, on top of $\Omega_{r_c}(n_i)$, its coordinates (p^x_i, p^y_i) and its proximity data $\mathcal{P}_i$ (i.e., the set of distance vectors to any node within the proximity radius).

3.3. Identification Procedure (IDP)

The Identification Procedure (IDP) operates in a binary-search fashion, using the identification routes to progressively narrow down the set of suspect nodes until only the compromised node remains. First, we present $\mathsf{IDP}_{\mathrm{SM}}$, introduced by Zilberman et al. [11]. Next, we propose $\mathsf{IDP}_{\mathrm{DR}}$, which includes two upgrades that we show significantly improve the IDP module.

IDP_{SM}: SpiderMiners's identification procedure

The procedure operates in a binary-search fashion, using the identification routes to progressively narrow down the set of suspect nodes until only the compromised node remains. In detail, A executes the following steps:

1. Select a safe route r_s.
2. Initiate the list of suspect nodes as $r = r_c - \{A, B\}$
3. From $r = (n_1, \ldots, n_k)$, select n_p with $p = \lceil k/2 \rceil$.
4. If $p = k = 1$, then n_p is the adversary.
5. Select the identification route $r_I(n_p)$.
6. Generate 2 shares using Shamir's scheme.
7. Route one share on r_s and one on $r_I(n_p)$.
8. Receive along r_s a feedback (positive if the share routed on $r_I(n_p)$ arrived modified, negative otherwise).
9. Restrict r according to the feedback; if positive, $r = (n_p, \ldots, n_k)$, otherwise, $r = (n_1, \ldots, n_{p-1})$.
10. Go to Step 3

A drone in r_c may have no identification routes, meaning none of the drones in the safe routes are within its communication range. In such a case, the authors propose to resort to *boomerang messages*, messages sent by A along round-trip routes. However, this approach

[1] Later, we discuss the case when a node has no identification routes.

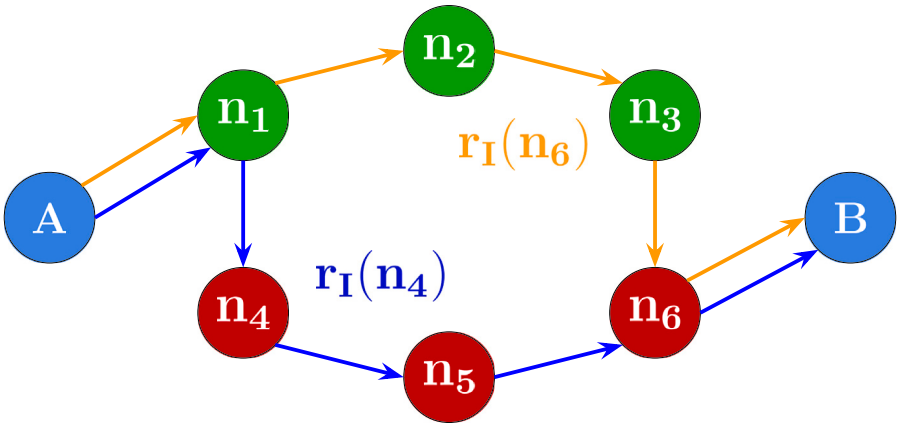


Fig. 2. Example scenario in which node A constructs *identification routes* (r_I) for n_4 and n_6 from the safe path (green nodes) and the compromised path (red path).

presents a fundamental drawback: the adversary can easily detect boomerang messages. Since it is unusual for a message to be routed on a path where the source matches the destination, the adversary could temporarily stop its disruptive actions or relocate upon observing a boomerang message.[2] We, thus, do not take into consideration the use of such boomerang messages.

IDP_{DR}: DRUID's enhanced procedure

$\mathsf{IDP}_{\mathrm{DR}}$ extends the previous version by integrating two key upgrades.

First, we notice that the only identification routes required to determine whether $n_k \in r_c = (A, \dots, n_k, \dots, B)$ is the adversary are $r_I(n_k)$ and $r_I(n_{k+1})$. When n_{k+1} corresponds to B, only one identification route can pinpoint the adversary: for example, if the adversary in the network illustrated in Fig. 2 is n_6, the identification $r_I(n_6)$ is sufficient to determine the compromised node. $\mathsf{IDP}_{\mathrm{DR}}$ maintains the binary search-like behavior while accounting for missing identification routes, thereby making use of all the available identification routes.

Second, we introduce *complementary identification routes*. Remind that, by definition, an identification route $r_I(n_i)$ is composed of three sub-routes: (i) the preamble of a safe route r_s, (ii) a 1-hop link (n_s, n_i), with $n_s \in r_s$ and $n_i \in r_c$, and (iii) the postamble of the compromised route r_c. We define a complementary identification route $r_I^C(n_i)$ as the path composed of (i) the preamble of the compromised route r_c, (ii) the 1-hop link from (n_i, n_s), and (iii) the postamble of a safe route r_s. The key difference is that, for a given node $n_i \in r_c$, the identification route $r_I(n_i)$ checks the integrity of shares routed by the nodes on the compromised path that follow n_i, whereas the complementary identification route $r_I^C(n_i)$ tests the nodes that precede n_i on the compromised path. When referring to Fig. 2, introducing complementary identification routes allows identifying the adversary in any case; from $r_I(n_6)$ we verify if n_6 is the adversary, whereas $r_I^C(n_4) = (A, n_4, n_1, n_2, n_3, B)$ can be used to check the integrity of shares routes through n_4 but not through n_5.

Since $\mathsf{IDP}_{\mathrm{DR}}$ extends $\mathsf{IDP}_{\mathrm{SM}}$ without altering the underlying logic identification algorithm, it preserves the deterministic characteristics of SpiderMiners [11]. Therefore, $\mathsf{IDP}_{\mathrm{DR}}$ (or more generally DRUID) does not produce false positives and a benign node can never be misclassified as the compromised drone.

3.4. Node Repositioning Procedure (NRP)

If the employed IDP fails to narrow the list of suspect nodes down to a single drone, A coordinates the following strategic repositioning of the drones along safe routes. That is, A carries out the following steps:

1. A uses a *heuristic* (Section 4.1) and the information in each $m_{REQ}^{Id,Pos}$ message to estimate the direction of the compromised path relative to each safe route.
2. A solves an optimization model (Section 4.2) that determines the new positions of each safe node.
3. A sends to B, along each safe route, the target positions that each node on that route should move to.
4. A waits for B to craft new $m_{REQ}^{Id,Pos}$ messages and routed them along each safe route.
5. If there are new identification routes, A employs $\mathsf{IDP}_{\mathrm{DR}}$. Otherwise, go to Step 2

Note that (i) this module requires $\mathsf{IRP}_{\mathrm{DR}}$, (ii) Step 1 is executed only once, (iii) at each iteration of Step 2, A solves a different instance of the same optimization model, and (iv) the optimization model may be unfeasible; in such a case, the instance is classified as a failure to identify the adversary.

Lastly, we remark that the three modules that compose DRUID are designed around the threat model of a single compromised node. Consequently, the secret sharing threshold value, t, can be set to 2, which ensures that the adversary cannot recover the secret message m while allowing B to correctly reconstruct m for $n > 2$. Extending this analysis to scenarios with multiple adversaries would require increasing t (ideally, at least the number of adversaries plus one). However, identifying appropriate threshold values and developing mechanisms to handle multiple—especially colluding—adversaries lies beyond the current scope of our work. Addressing such cases would likely demand substantially more sophisticated techniques (e.g., game-theoretic models, reputation systems, or new modules integrated into DRUID's pipeline). We therefore leave such generalizations as an important direction for future work.

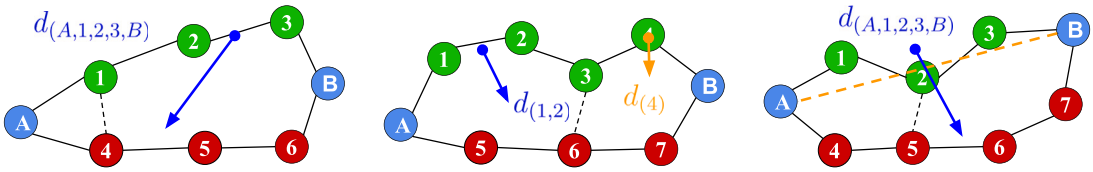
4. Reposition strategy

In this section, we outline in detail how the Node Repositioning Module operates.

4.1. Heuristics

We propose and evaluate three heuristics that estimate the best direction towards the compromised path with respect to subsets of adjacent safe nodes, while relying solely on information included in the $m_{REQ}^{Id,Pos}$ messages that A receives along the safe routes. For each safe route, a heuristic identifies one or more subsets of adjacent safe nodes, each of which may vary in length, and assigns them the general direction that those nodes should follow to get closer to the compromised route. For a given subset of safe nodes $\mathbf{n_i} = n_{i_1}, \dots, n_{i_k}$, the heuristics

[2] Recall that the entire route followed by the message is specified in the packet header.



(a) Heuristic 1 (b) Heuristic 2 (c) Heuristic 3

Fig. 3. Illustration of the three heuristics used to compute the direction vector(s) for repositioning safe nodes towards the compromised nodes.

calculate the vector $(d^x_{\mathbf{n_i}}, d^y_{\mathbf{n_i}}) \in \mathcal{D}$ of norm k. We denote with $\mathcal{D}$ the set of vectors that A calculates.

Notice that these approaches are simple heuristics, and the directions they provide are not guaranteed to be fully accurate; indeed, they do not rely on the positions of the nodes in the compromised path, which are supposed to be unknown or unreliable. In the remainder of the paper, we refer to our proposed approach combined with a specific heuristic as DRUID$_i$ where i represents one of the following three heuristics.

We remark that the heuristics are run once at the beginning of DRUID. We also explored an alternative approach in which we reevaluated the directions at each iteration; we may refer to these heuristics as *iterative heuristics*. This method showed marginal improvements with heuristics 1 and 3; however, both were still outperformed by heuristic 2, which yielded a worse identification rate in its iterative variant.

Heuristic 1. The first heuristic provides a direction for each safe route by evaluating the difference between the centroid (i.e., the average (x, y) of the coordinates) of safe nodes located within the communication range of the compromised path and the centroid of those outside. The result of this difference is a vector that we use to obtain the direction.

Heuristic 2. The second heuristic follows the same logic as Heuristic 1, but with a higher granularity. Specifically, the heuristic identifies tuples of adjacent safe nodes that are outside the communication range of the compromised path. Next, the heuristic assigns a direction to each of these tuples based on the difference between their centroid and the centroid of the two boundary nodes (which are, by construction, closer to the compromised route). Note that when the only safe nodes within the communication range of the compromised path are A and B, heuristics 1 and 2 yield the same direction.

Heuristic 3. The third heuristic only relies on the positions of the nodes. Specifically, given a safe route, it considers the line connecting A and B and counts the number of safe nodes that are on either side of this line. Next, the direction is obtained from the difference between the average coordinates of safe nodes on the side with the most safe nodes and the coordinates of the pair A, B. Like the first heuristic, the direction is assigned to an entire safe route.

Note that a heuristic may fail to provide a direction if it cannot compute any direction. Consider once again Fig. 2; if n_2 were within the communication range of n_5, heuristics 1 and 3 would be unable to provide a general direction of the compromised path, as they require at least one of n_1, n_2, and n_3 not to be in communication with a node of r_c.

Fig. 3 shows three examples of the direction vector(s) obtained according to the heuristics. In Fig. 3(a), Heuristic 1 returns a single vector norm 5 $d(A, 1, 2, 3, B)$, representing the repositioning all green nodes must undergo. It is calculated from the difference of the average (x, y) coordinates of nodes $A, 1, B$ (within communication range of r_c) and nodes 2,3 (not in communication with r_c). Fig. 3(b) exemplifies the operation of Heuristic 2. It returns separate vectors for two tuples of adjacent nodes: $(1, 2)$ and (4). For each of these, Heuristic 2 evaluates the direction vector from the difference between the average (x, y) coordinate of the boundary nodes (in communication with r_c) and the nodes in the tuple (not in communication with r_c). Specifically, $d_{(1,2)}$ is obtained using A and 3 as boundary nodes, whereas $d_{(4)}$ relies on 3 and B as boundary nodes. Lastly, Fig. 3(c) shows the direction $d_{(A,1,2,3,B)}$ returned by Heuristic 3. In this case, A counts the number of safe nodes on either side of the straight line that directly connects A and B, and returns the direction from the difference of the average (x, y) coordinate between 1,3 and A, B.

4.2. Optimization model

After obtaining the result of the aforementioned heuristics, DRUID solves a Mixed Integer Nonlinear Problem (MINLP) to determine the exact positions where nodes on the safe path(s) will move to. Note that the directions $\mathcal{D}$ are evaluated once, during the first iteration, whereas the following MINLP model is solved by A at each iteration.

$$\text{Minimize} \quad \sum_{i \in \mathcal{N}_S} (x_i - p_i^x)^2 + (y_i - p_i^y)^2 \tag{1}$$

Subject to

$$(x_i - x_j)^2 + (y_i - y_j)^2 \leq R_c^2,\ \forall (i,j) \in r_s, \forall r_s \in R_S \tag{2}$$

$$sgn(d^x_{\mathbf{n}}) \sum_{i \in \mathbf{n}} \left(x_i - p_i^x\right) \geq sgn(d^x_{\mathbf{n}})\, d^x_{\mathbf{n}},\ \forall d^x_{\mathbf{n}} \in \mathcal{D} \tag{3}$$

$$sgn(d^y_{\mathbf{n}}) \sum_{i \in \mathbf{n}} \left(y_i - p_i^y\right) \geq sgn(d^y_{\mathbf{n}})\, d^y_{\mathbf{n}},\ \forall d^y_{\mathbf{n}} \in \mathcal{D} \tag{4}$$

$$(x_i - p_i^x)^2 + (y_i - p_i^y)^2 \leq (0.75 R_p)^2, \forall i, j \in \mathcal{N}_S, i \neq j \tag{5}$$

$$(x_i - (p_i^x + d^x))^2 + (y_i - (p_i^y + d^y))^2 \geq 0.5^2, \forall i \in \mathcal{N}_S,\ \forall (d^x, d^y) \in \mathcal{P}_i \tag{6}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq (0.5)^2,\ \forall i, j \in \mathcal{N}_S, i \neq j \tag{7}$$

$$x_i, y_i \in \mathcal{W},\ \forall i \in \mathcal{N}_S \tag{8}$$

Eq. (1) represents the objective function, which aims at minimizing the overall movement of safe drones. Since secure feedback along safe routes is a fundamental requirement of the IDP, constraints (2) ensure that adjacent nodes in each safe path r_s stay within each other's communication radius, thus keeping safe routes intact. Constraints (3) and (4) direct each tuple of nodes $\mathbf{n}$ to follow the assigned direction $(d^x_{\mathbf{n}}, d^y_{\mathbf{n}})$. Without these constraints, the drones would not move at all, as they would prioritize optimizing the objective function Eq. (1). Since we coordinate the drones in a discrete, step-by-step manner, constraints (5) limit each drone to move no more than $0.75\, R_p$ at each iteration. This is necessary to avoid collisions with nodes outside the proximity range R_p and whose positions are unknown. Constraints (6) utilize the proximity information $\mathcal{P}_i$ of each safe node i to avoid collisions with nearby nodes. However, they cannot prevent collisions among safe nodes when two move towards each other, as their combined traveled distance may exceed the proximity radius R_p. To address this issue, constraints (7) ensure that each pair of safe nodes maintains a minimum distance of 0.5 m. Lastly, constraints (8) define the domain of decision variables x_i and y_i, which corresponds to the covered area $\mathcal{W}$.

The proposed model penalizes drone repositioning based on Euclidean distance (Eq. (1)). Moreover, we later introduce a movement budget to limit per-node travel (Section 6, Constraints (9)). While we do not implement such extensions in this work, a more energy-aware extension could instead account for UAVs energy consumption

in one of these two ways. First, the *identification and position request messages* ($m_{REQ}^{Id,Pos}$) could be extended so that each node i appends its current available energy, e_i. The model could then include additional constraints to ensure that only drones having $e_i \geq E_{min}$ shall reposition. Second, the cost function could be replaced (or augmented) with one that minimizes total drone energy use, differentiating between stationary hovering and horizontal movement [23]. Such a change implies introducing new decision variables that accurately measure the energy consumed by each drone during a given iteration. However, the actual energy consumption model heavily relies on the physical characteristics of the UAVs, the horizontal movement speed, and the weather conditions (e.g., wind) [24]. Both options allow, in different ways, to prioritize mission completion over adversary identification. These changes would preserve the mixed-integer formulation structure while improving DRUID's realism in energy-constrained scenarios.

5. Experimental setup

To assess the performance of DRUID, we perform a thorough evaluation by simulating different scenarios that differ in graph connectivity, drone communication radius, and path length. We evaluate three versions of DRUID that differ in the heuristics used. We use SpiderMiners, in its original and improved versions, as a baseline. In total, we evaluate five schemes and measure their performance in terms of identification rate (i.e., how often a scheme identifies the compromised node). Additionally, we analyze the cost associated with the node repositioning procedure in terms of computation time and drone movement.

5.1. Evaluated schemes

We evaluate DRUID in different versions to measure the performance of the three proposed heuristics. SpiderMiners serves as a baseline, representing a simpler alternative in which nodes are never relocated. Moreover, as DRUID employs an improved version of SpiderMiners's Identification Procedure, we evaluate the impact of this improvement individually. Overall, we consider the five schemes described in Table 1, detailing the modules used in each. Specifically, the five schemes are composed as follows: (i) SM is the original version of SpiderMiners and represents our initial baseline, (ii) SM+ employs $\mathsf{IDP}_{\mathrm{DR}}$, our improved version of the Identification Procedure module, (iii) DRUID_1, DRUID_2, and DRUID_3 represent the full pipeline we propose (Fig. 1) and only differ in the underlying heuristic used to estimate the direction to the compromised path.

5.2. Dataset

We employ a dataset comprising 10 000 graphs, each representing a network of 50 nodes randomly distributed within a 50 m × 50 m area (2500 m^2). Each node, representing a drone, has a default communication radius of $R_c = 10$ m and a proximity radius of $R_p = 2$ m. Each graph is built to be connected, guaranteeing the existence of a path between any pair of nodes within the network. Furthermore, to maintain spatial integrity and avoid overlap, we enforce a minimum separation of 4 m between every pair of nodes. Such networks effectively simulate realistic drone distributions in coverage-like missions, where maintaining connectivity and spatial separation is essential for efficient operation and coordination.

We carry out the experiments using a custom-made Python-based software that runs the required simulations. The optimization model is implemented using Gurobi 11, a widely used mathematical optimization solver. We set an instance-wise time limit of 30 s, meaning that the cumulative computation time of the NRP across iterations cannot exceed 30 s. For example, if Gurobi successfully solves the model in 5 s during the first iteration, then at the next iteration, if the IDP fails to identify the adversary, the remaining time available to Gurobi to solve the next model(s) would be 25 s.

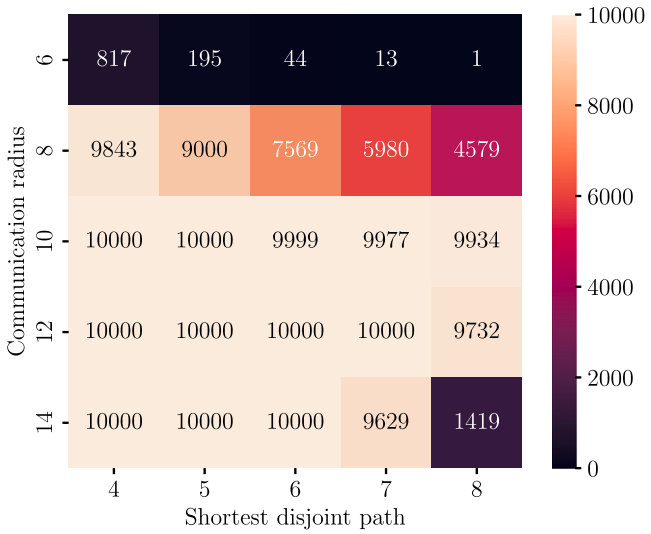


Fig. 4. Number of valid instances for each configuration.

Furthermore, we set the gap parameter to 25 %. This allows the Gurobi solver to terminate when a solution is found within 25 % of the closest lower bound, thereby reducing the computation time. We run our experiments on a remote virtual machine powered by an Intel Xeon Gold 6140 Processor and 376 GB of RAM.

To evaluate the impact of network characteristics on the performance of each scheme, we vary the connectivity of the graph and the distance between the nodes A and B. To vary the former, we consider the following values for the communication radius R_c: 6 m, 8 m, 10 m, 12 m and 14 m. Given that graphs are guaranteed to be connected for $R_c = 10$ m, we analyze the identification rate as the connectivity property of the graph is possibly lost ($R_c = 6$ m) or strengthened ($R_c = 14$ m). To measure the latter, the distance between the nodes A and B is quantified by the length of the *shortest disjoint path* among the node-disjoint paths identified using the *node_disjoint_paths* function from the NetworkX library [25]. We consider the shortest disjoint paths of length 4, 5, 6, 7 and 8, where the length of a path is given by the number of nodes it contains.

Given the 10 000 original graphs composing the dataset and the two varying parameters, we obtain 250 000 *instances*. However, we cannot operate in all instances due to the requirement of at least two node-disjoint routes combined with the shortest disjoint path parameter. Indeed, the more we decrease the communication radius, the more frequently A and B are connected by one or no path. Fig. 4 shows, out of the 10 000 graphs, the number of *valid instances* for each pair of communication radius and shortest path. In total, we obtain 178 731 valid instances. The instances where the communication radius is set to 6 meters are the least likely to present a pair of nodes satisfying the requirements of the shortest disjoint path and the existence of two or more node-disjoint paths. For $R_c = 10$ m, which grants connectivity in the graphs, for all the values of the shortest disjoint path parameter, more than 99 % of the instances are valid, highlighting the importance of maintaining network connectivity during operations to facilitate the usage of security protocols [26]. We observe an important drop in valid instances for $R_c = 14$ m due to how *node_disjoint_paths* operates.

All the experiments are run with the secret sharing threshold parameter, t, set to 2. This value ensures that a single compromised node cannot recover the secret message and aligns with the requirement of *at least* two node-disjoint paths between A and B. While higher threshold values are possible, they would require a larger minimum number of node-disjoint paths—further reducing the set of valid instances—and offer no practical advantage under the assumed threat model.

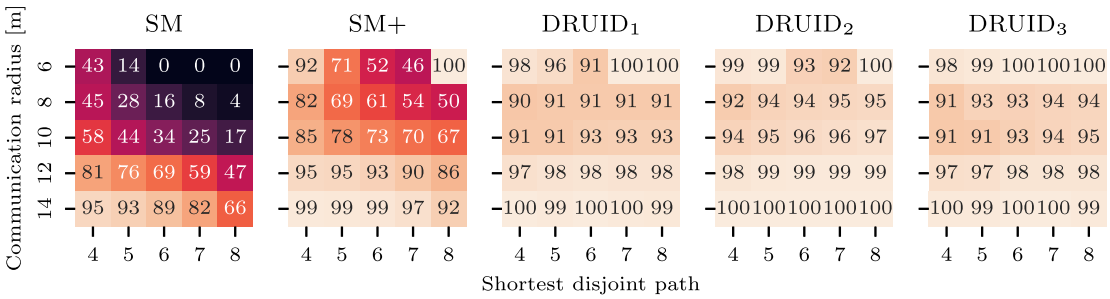


Fig. 5. Adversary identification rates achieved by each scheme.

Table 1
Adversary identification rate achieved by each scheme.

Scheme		Modules		Identification rate
SM	$\mathsf{IRP}_{\mathrm{SM}}$	$\mathsf{IDP}_{\mathrm{SM}}$	–	54.1 %
SM+	$\mathsf{IRP}_{\mathrm{SM}}$	$\mathsf{IDP}_{\mathrm{DR}}$	–	83.1 %
DRUID_1	$\mathsf{IRP}_{\mathrm{DR}}$	$\mathsf{IDP}_{\mathrm{DR}}$	$\mathsf{NRP}_{\mathrm{H1}}$	95.2 %
DRUID_2	$\mathsf{IRP}_{\mathrm{DR}}$	$\mathsf{IDP}_{\mathrm{DR}}$	$\mathsf{NRP}_{\mathrm{H2}}$	97.1 %
DRUID_3	$\mathsf{IRP}_{\mathrm{DR}}$	$\mathsf{IDP}_{\mathrm{DR}}$	$\mathsf{NRP}_{\mathrm{H3}}$	95.7 %

The values selected for parameters such as communication radius (R_c) and proximity radius (R_p) in our experimental setup reflect realistic configurations for medium-scale UAV network deployments. However, these parameters are not universal and should be tuned based on the intended deployment area, node density, and mission requirements. For instance, the communication radius R_c influences network connectivity and should be increased in larger or sparser deployments to maintain route availability. Conversely, smaller R_c values may help reduce interference and energy usage in dense or compact scenarios. The proximity radius R_p defines the spatial buffer for safe drone movement and represents a drone's collision avoidance capability. The values used in our simulations serve as representative baselines; however, practitioners should adjust them according to their real-world operational goals and hardware limitations.

6. DRUID performance

We first discuss the overall identification rate across the 178 731 instances, reported in the last column of Table 1. The baseline version of SpiderMiners (SM) achieves an identification rate of 54.1 %. The improved version SM+ achieves an identification rate of 83.1 %, meaning the upgrades introduced to $\mathsf{IDP}_{\mathrm{DR}}$ lead to a +29 % increment. Our proposed approach, DRUID, identifies the adversary in no less than 95.2 % of valid instances. Overall, DRUID_2 offers the best performance (97.1 %), meaning a near 80 % increment with respect to SM.

Fig. 5 breaks down the identification rate for each network *configuration*, i.e., for each pair of communication radius and shortest disjoint path values. DRUID achieves no less than 90 % identification rate for each configuration; the three heuristics show slight differences between them, with DRUID_2 being slightly better overall (as shown in Table 1). On the other hand, SM has a highly variable identification rate, going as high as 95 %, but also well below 20 % in some circumstances. Overall, it provides a lower identification rate when the connectivity is lower (i.e., $R_c \leq 8$ m) and with longer paths. This observation also holds for the updated version of SpiderMiners, although to a lesser extent: With SM+, the identification rate is also variable, but it only drops below 50 % once. Overall, these results show that DRUID outperforms SpiderMiners, especially in the situations where SpiderMiners struggles (i.e., with sparse graphs and long paths), proving the benefits of moving nodes to make the necessary missing connections. Moreover, in the remaining analyses (except for the one presented in the next paragraph), we focus on DRUID_2 since it yields the best overall results and differences with the other two variants in terms of performance are limited.

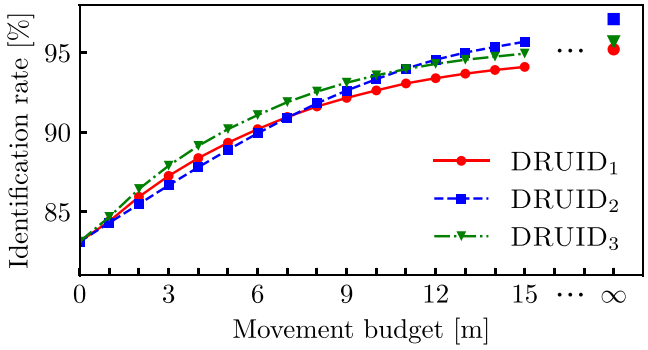


Fig. 6. Adversary identification rate for different movement budgets.

We next consider the case in which the network operator wants or needs to control the amount of movement allowed by drones. This is a constraint that may apply in certain use cases and practical applications. Indeed, there could be restrictions in node positioning (nodes cannot move too far from where they are supposed to operate) or energy consumption (moving a drone requires more energy than a stationary flight). To support such a case, we impose a *movement budget* in the optimization model. Specifically, we impose that a node cannot travel more than a specified distance b during the entire DRUID execution. This is achieved by adding constraints (9) to the model described in Section 4.2.

$$(x_i - p_i^x)^2 + (y_i - p_i^y)^2 \leq b - t_i,\ \forall i \in \mathcal{N}_S \tag{9}$$

t_i is the distance traveled by node n_i, and is updated after each iteration. Such constraints, which limit the cumulative repositioning distance per node, should reflect constraints related to energy availability and airspace restrictions.

We show the identification rate achieved by DRUID for movement budgets from 0 m (which corresponds to SM+ in Fig. 5) to 15 m in Fig. 6. The last values refer to the identification rate under no budget (i.e., $b \to \infty$), and correspond to the respective values in Table 1. We observe that DRUID_2, despite reaching the highest identification rate under no budget, yields the worst results (albeit by small margins) for the lowest budgets. In fact, DRUID_3 achieves the highest identification rates for $b \leq 10$ m conceding the lead to DRUID_2 for $b \geq 11$ m. For $b = 9$ m, the identification rate of DRUID_3 is 93.1 %, which is an additional 10 % identification rate over SM+. In summary, DRUID_2 provides the best results, albeit with marginal improvements over the other two; conversely, if a low movement budget is imposed, DRUID_3 may slightly outperform DRUID_2.

As discussed above, our model penalizes repositioning through a movement budget that constrains the cumulative distance traveled by each UAV. This abstraction reflects the fact that UAV endurance is primarily limited by battery capacity. In practice, the dominant share

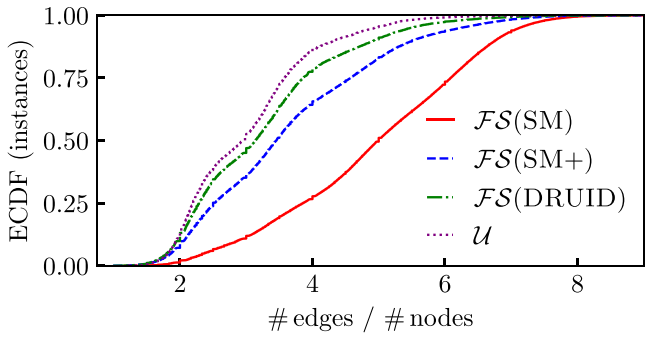


Fig. 7. Distribution of the connectivity across solved and unsolved instances.

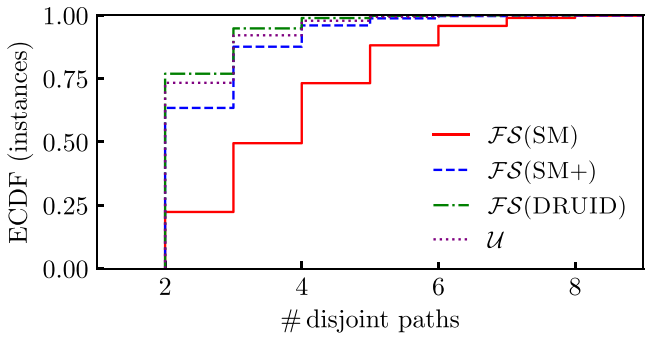


Fig. 8. Impact of the number of available disjoint paths on the adversary identification.

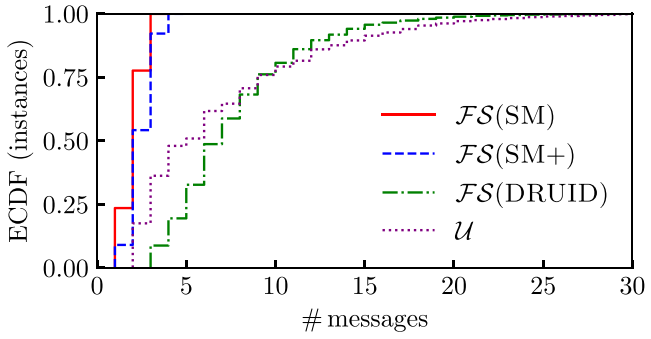


Fig. 9. Distribution of the number of messages sent by A during the Identification Procedure.

of the energy budget is consumed simply to maintain flight, while repositioning adds a smaller, albeit non-negligible, overhead. Experimental data from Mulgaonkar et al. [27] illustrates this for the AscTec Hummingbird: the platform requires ≈ 75 W in hover, with power rising to ≈ 95 W during horizontal motion and ≈ 128 W during vertical climbs, on a ≈ 23 Wh battery yielding 18 min endurance. Maintaining hover for 30 s consumes ≈ 2250 J, whereas a 30 m relocation at 1 m/s requires only ≈ 600 J extra (< 1 % of the total capacity). Thus, although repositioning incurs only a modest incremental cost compared to hovering, repeated maneuvers across multiple drones or iterations can accumulate into a significant share of the available energy, reinforcing the importance of incorporating movement constraints in our model. Hence, our movement budget abstraction already captures the limiting effect of finite energy and motivates minimizing repositioning, while incorporating platform-specific power models would be a valuable extension to more precisely quantify endurance impacts.

We now investigate more deeply how the characteristics of a network influence the identification of the malicious node for the different schemes. To better present the results, we define the concept of the *First Solver* for a given instance. We build this definition on the fact that if an instance is solved by SM, it is also solved by SM+ and DRUID. However, the same does not hold the other way around. A scheme is *First Solver* for an instance if that scheme is the first to identify the adversary. As such, we divide the instances in our dataset into four sets:

- $\mathcal{FS}$(SM): the 96 663 instances whose *First Solver* is SM, thus are solved by all schemes.
- $\mathcal{FS}$(SM+): the 51 881 instances solved by SM+ and DRUID, thus having SM+ as *First Solver*.
- $\mathcal{FS}$(DRUID): the 24 997 instances that are solved by DRUID_2 uniquely.[3]
- $\mathcal{U}$: the 5190 instances which are not solved by any scheme.

In Fig. 7, we present the distribution of connectivity across instances separately by *First Solver*. We measure connectivity as the ratio of edges to nodes within the subgraph of nodes involved in the Information Retrieval Procedure (IRP). As highlighted by the red line, SM tends to identify the adversary in cases with the highest connectivity. Specifically, 75 % of the instances SM solves have a connectivity above 3.8. From another perspective, SM solves 87 % of instances with connectivity above 6, but only 41 % of those with connectivity below 5. If we now focus on instances in $\mathcal{FS}$(SM+)—those solved by SM+ and DRUID, but not by SM—we observe a clear phenomenon: SM+ often succeeds where SM fails *and* connectivity is low. In fact, instances in $\mathcal{FS}$(SM+) have a median connectivity of 3.5 edges per node. The 24 997 instances in $\mathcal{FS}$(DRUID) (green line) have similar connectivity, suggesting that connectivity is not the factor driving DRUID success. Finally, unsolved instances have similar (though slightly lower) connectivity.

[3] In these and subsequent analyses, we only consider DRUID_2, omitting results for the other two heuristics.

Next, we analyze the effect of the number of disjoint paths available. Fig. 8 shows, separately for *First Solver*, the distribution of the number of paths that node A can use to communicate with B—which can never be less than 2. Notice that the step-like appearance of the CDFs in this figure is a natural consequence of the discrete nature of the underlying metric: the number of disjoint paths is an integer-valued quantity, so the cumulative distribution increases only at specific integer values. We first observe that SM—the base version of SpiderMiners—prefers cases with many disjoint paths, as shown by the solid red line referring to $\mathcal{FS}$(SM). In fact, 50 % of the cases solved by SM involve 4 or more disjoint paths (i.e., 3 or more safe routes that can be used to establish identification routes). In contrast, 63 % of the instances in $\mathcal{FS}$(SM+) (dashed blue line) have only 2 disjoint paths, i.e., only one safe path. Again, this shows that SM+ succeeds in the cases where SM struggles, i.e., those with only one safe path between nodes A and B. Similar to the distribution observed in Fig. 7, the differences between instances in $\mathcal{FS}$(SM+) and $\mathcal{FS}$(DRUID) are less pronounced. Although DRUID tends to succeed proportionally more often with two disjoint paths, the difference with unsolved instances $\mathcal{U}$ is small, suggesting that node repositioning is a winning strategy regardless of the graph number of disjoint paths—recall that $\mathcal{FS}$(DRUID) contains 14 % of instances, which could not be solved with SM+.

Lastly, we measure the communication costs associated with the IDP and compare them across the three schemes. In Fig. 9 we present the distribution of the number of messages sent by A (i.e., the node that orchestrates the IDP) during the IDP across different instances, one line for each *First Solver*. Again, the discrete nature of the metric leads to step-like CDFs. The figure shows a significant difference between the instances solved without the NRP (red and blue lines) and those in which A utilizes the NRP (green and purple lines). Such a difference is due to the number of IDP executions. Instances where the *First Solver* is either SM or SM+ (red and blue lines) require only a *single* execution of the IDP. Since the IDP operates in a binary search fashion, the number

of messages sent by A is bounded above by $\lceil \log_2(|r_c| - 2) \rceil$. In our experiments, we measured median values of 2 and 3 in the median, respectively, for SM and SM+. In contrast, instances *First Solved* by DRUID$_2$ involve *multiple* IDP executions, as A iteratively repositions nodes and the total number of messages sent by A accumulates possibly beyond the aforementioned bound. In our experiments, we measured a median value of 7 messages and a 90th percentile of 13 messages.

7. DRUID execution time

We now evaluate the DRUID execution time. Specifically, we study how long it takes for (i) executing the Identification Procedure, (ii) solving the optimization problem(s), and (iii) repositioning the drones. As discussed in Section 3, the drones are coordinated iteratively: A solves an optimization model, communicates the new positions to the safe nodes, and repeats this process until the IDP identifies the compromised drone. This raises an important question: How long does it take to identify the adversary? To answer the question, we must consider, for a given instance, both the time required to run the IDP, the time needed to solve all the optimization models (one at each iteration), and the time safe drones take to reposition themselves.

7.1. IRP And IDP

First, we consider the overhead introduced by the IRP and IDP modules through the number of messages sent by A and B.

B sends (i) a $m_{REQ}^{Id,Pos}$ message to A upon detecting a compromised node on the path r_c, (ii) one message that combines feedback (i.e., whether the share routed by A on the identification route was altered) and a new $m_{REQ}^{Id,Pos}$ message during each iteration of the IDP. Therefore, we estimate that B sends one message for each message sent by A.

From Fig. 9 we learn that, in the median, A sends 7 messages across all executions of the IDP. Assuming a Round Trip Time of 5 ms, we estimate that the combination of IRP and IDP introduces a time overhead of roughly 35 ms.

7.2. NRP

Next, we measure the primary component that builds the DRUID execution time: nodes relocation. For a given instance, we indirectly quantify the nodes' relocation time by evaluating the sum of the maximum movement at each iteration. Indeed, at each iteration, the relocation time is directly proportional to the largest distance traveled by any node. By design, each node during each iteration travels at most a distance equal to half of the drone's proximity radius R_p (Eq. (5)), which corresponds to 1.5 m in our experiments. Fig. 10 shows the distribution of the sum of maximum movement at each iteration over the instances (i.e., for each instance, we sum the maximum movement at each iteration). The figure distinguishes between solved instances (solid red line) and unsolved instances (dashed blue line). We observe a substantial difference between the two distributions: the median value of such a metric is 6.4 m across instances solved by DRUID$_2$ and 24.5 m across those that remain unsolved. To estimate the relocation time, we assume that the drones move at 1 m/s,[4] which enables a direct mapping of the cumulative maximum movement across iterations [m] to time [s]. We thus obtain a median relocation time of 6.35 s, with only the top-10 % taking more than 15 s. The important difference between the distributions of solved and unsolved instances further suggests the possibility of using the relocation time as a stopping criterion for DRUID.

As said, by design, the average drone movement is set to 1 m because of (i) constraints (3) and (4), and (ii) each direction has the same norm as the number of nodes that must move along it. We thus find that in 80 % of iterations across all instances, the max movement is ≈ 1 m. Consequently, Fig. 10 very closely resembles the distribution of the number of iterations (i.e., the number of times NRP is employed) of both solved (solid red line) and unsolved (dashed blue line) instances. Indeed, 97 % of solved instances require less than 20 iterations.

Lastly, we present in Fig. 11(a) the distribution of the cumulative computational time spent solving optimization models across the instances whose *First Solver* is DRUID$_2$. We define cumulative computational time as the sum of the execution time of our optimization models across all iterations needed to solve a given instance. We plot separate distributions for different communication radii as we observe a correlation between them and computation time. Notably, 95 % of instances are solved in less than 1 s for $R_c \leq 10$ m, and less than 3.4 s for $R_c = 14$ m. The median computation times range from 0.08 s (for $R_c = 6$ m) to 0.32 s (for $R_c = 14$ m), highlighting the low complexity of the proposed optimization model. Only 3 instances (out of 24 997) took,

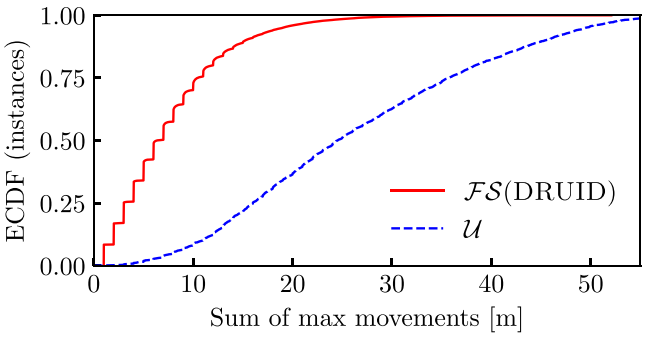
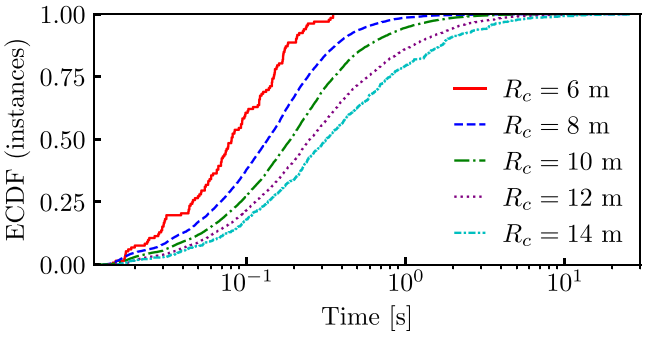
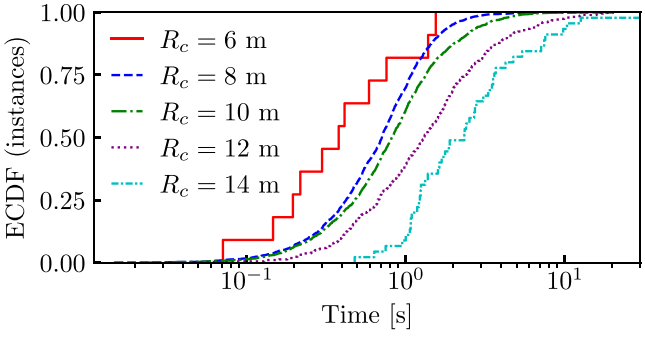


Fig. 10. Distribution of the sum of maximum movement at each iteration across instances solved (red line) and unsolved (blue line) by DRUID$_2$.



(a) Solved instances



(b) Unsolved instances

Fig. 11. Distribution across instances solved 11(a) and unsolved 11(b) by DRUID$_2$ of the cumulative computation time to solve the optimization models.

[4] Note that most commercial drones reach much higher lateral speeds.

Table 2
Comparison of the median execution time (in seconds) of the three variants of DRUID.

Scheme	Communication	Relocation	Model	Total
$DRUID_1$	0.03	5.00	0.24	5.27
$DRUID_2$	0.04	6.35	0.17	6.56
$DRUID_3$	0.03	4.50	0.24	4.77

overall, more than 15 seconds to be solved, justifying the cumulative time limit of 30 s.

For completeness, Fig. 11(b) shows the cumulative computation time among instances in $\mathcal{U}$ (those that remain unsolved as $DRUID_2$ fails to identify the compromised drone). Only 11 instances remain unsolved for $R_c = 6$ m, giving the respective CDF (red solid line) a step-like appearance. We observe that such instances tend to require roughly 5 times more computation time—notice the logarithmic x-axis. The median values range from 0.4 s to 2.3 s, whereas the 95th is reached in 3.2 s ($R_c = 10$ m) and 10.3 s ($R_c = 10$ m). Lastly, of the 3992 instances that remain unsolved, only 1 was stopped due to the 30 s time limit.

In short, by combining the communications time (derived from Fig. 9), the time required to relocate the drones (from the distances shown Fig. 10), and the model computation time (seen in Fig. 11(a)), we estimate a median time-to-identify of roughly 6.56 s, given our environment in terms of covered area and a drone speed of 1 m/s. However, in the case of different scenarios, our simulation frameworks allow easy estimation of the distribution of DRUID execution time.

Additionally, Table 2 compares the median time-to-identify of $DRUID_2$ with $DRUID_1$ and $DRUID_3$ (omitted in the Figs. 9, 10, 11). Most notably, the relocation procedure for $DRUID_3$ takes almost two seconds less than the one for $DRUID_2$, highlighting a tradeoff between identification rate (95.7 % vs. 97.1 %, see Table 1) and total execution time (4.77 s vs. 6.56 s). SM and SM+ are excluded from the table, as they do not employ NRP and thereby have no relocation time, which alone constitutes roughly 95 % of the execution time of DRUID. As a result, their execution time is in the order of milliseconds, but with lower identification rates.

8. Related work

Recent research has explored node relocation to restore connectivity and resilience in UAV networks, as well as intrusion detection techniques to identify and mitigate insider attacks. In the following, we summarize these two bodies of research.

8.1. Node relocation

Numerous studies in mobile ad hoc networks have investigated the possibility of repositioning mobile nodes to enhance both network connectivity and resilience against cyberattacks. Physical faults, battery exhaustion, environmental factors, and malicious attacks may result in node failures [28], which in turn could leave areas unmonitored or cause networks to split into disconnected sub-networks. Recently, [29] studied the issue of restoring connectivity between disconnected clusters of UAVs. To tackle this, they proposed a decentralized, leader–follower mechanism that relocates operational UAVs while avoiding collisions. For each cluster, the drone that is closest to the center of the initial swarm is selected as the leader, while the remaining UAVs become followers. As the leader flies towards the initial swarm center, followers need to maintain connectivity to avoid further network splits while avoiding collisions. Instead, [30,31] explored UAV-assisted approaches to detect network partitions, with [30] focusing on the deployment of new nodes and [31] utilizing UAVs as stationary relay nodes to restore connectivity among disjoint network partitions. In [32], the authors deal with the issue of coverage holes, areas left unmonitored due to node failures. They introduced a game-theoretic framework where nodes independently adjust their reposition and the sensing range, intending to minimize energy consumption. Further works studied the possibility of using position-aware UAVs as relays. Specifically, [33] proposed a multi-UAV relay system where nodes are optimally placed according to a Mixed Integer Linear Problem (MILP) model, which is then extended to account for mobile nodes in the ground networks and changes in the traffic patterns. Instead, [34] employed a single-UAV relay that adjusts its position—including altitude—to maintain optimal connectivity and allow communications among moving cars. Similarly, our work aims to enhance network connectivity and establish the missing identification routes, to identify a compromised drone disrupting the network operations and performance. However, since DRUID aims at a silent identification of the adversary, the exact positions of the nodes in the compromised path are unknown, and a general direction is estimated based only on information available to safe nodes. To the best of our knowledge, the only recent work in which mobile nodes are explicitly repositioned in response to a cyberattack is [35]. Specifically, this study examines a jamming device that isolates certain nodes, preventing any communication from and to these jammed devices. The proposed relocation strategies are based on the last known positions of jammed nodes.

Within the broader context of drone coordination, recent proposals focused on resource management in federated drone-based edge computing systems, specifically on how to optimize task scheduling, flight routes, and resource utilization across heterogeneous drone deployments [36,37]. In contrast, our work addresses a security challenge in UAV networks, introducing a decentralized scheme for identifying and isolating compromised drones using secret sharing, multipath routing, and strategic nodes repositioning. While [36,37] are concerned with performance and efficiency in task execution, our approach is focused on improving the resilience and trustworthiness of drone communications in the presence of an adversary.

8.2. Intrusion detection

UAV networks are subject to insider attacks due to their infrastructure-less nature. Timely identification of such an attacker is essential to the integrity of the network and the correct UAV operations, and researchers have explored this issue from multiple angles. In 2024, Zilberman et al. [11] presented SpiderMiners, a novel scheme that deterministically identifies a compromised node that selectively alters routed messages. Building upon previous work [12], SpiderMiners introduced an Information Retrieval Procedure (IRP) and an Identification Procedure (IDP) that pinpoints the adversary in a binary search fashion. However, as discussed in Section 3, when IDP fails to identify the compromised node due to a lack of identification routes, SpiderMiners resorts to *boomerang messages*, a fallback mechanism that is trivially detectable by an adversary, thereby alerting it about the ongoing identification procedure. In our work, we improve both IRP and IDP and introduce a Nodes Repositioning Procedure (NRP) that strategically relocates nodes to increase the silent identification rate of the adversary.

DRUID identifies a compromised drone by analyzing the integrity of messages routed through subsets of suspect nodes. Similarly, trust-based systems evaluate the historical behavior of nodes to either identify or mitigate the presence of malicious nodes [38,39]. For example, when nodes have access to trust scores, they can base their routing decisions on them, thereby mitigating multiple kinds of routing attacks [40,41]. Further works use context-awareness to distinguish between intentional and unintentional misbehaviors [42], or improve the trust model accuracy in outdoor environments, which are influenced by varying weather conditions [43].

Another important body of work is given by Intrusion Detection Systems (IDSs), tools that monitor the network traffic or node activity and provide alerts when they suspect intrusion or anomalies [5,6,44,45]. However, these systems predominantly rely on machine learning methods, especially deep learning, meaning that their performances strongly depend on the underlying training set [46–48].

9. Discussion and final remarks

In this article, we presented and evaluated DRUID, a fully decentralized scheme for the silent detection and identification of a compromised drone tampering with the communication between two UAVs. Building upon SpiderMiners [11], we first proposed $\mathsf{IDP}_{\mathrm{DR}}$, an enhanced Identification Procedure that significantly increases the identification rate (83.1 %) compared to our baseline SpiderMiners (54.1 %), without requiring any drone movement. We then introduced a Nodes Repositioning Procedure, where safe drones are iteratively relocated—according to a simple optimization model and lightweight heuristics—to establish the missing connectivity required for a silent identification, further elevating the identification rate to 97.1 %. Finally, we analyzed how two network metrics—overall connectivity (edge-to-node ratio) and the number of node-disjoint paths between the two nodes—affect the identification rate of each scheme, finding that the effectiveness of DRUID in sparsely connected graphs with fewer disjoint paths is much higher than current state-of-the-art approaches. In the remainder of this section, we discuss practical aspects related to the use of DRUID in a real-world scenario, and then, we discuss directions for future work.

9.1. Real-world deployment

While DRUID demonstrates strong performance in simulated environments, its real-world deployment might encounter some engineering challenges. First, GPS inaccuracies could affect position accuracy, thereby compromising the effectiveness of the repositioning strategy and leading to suboptimal identification rates. Second, communication delays and packet loss in wireless UAV networks, particularly in dynamic or noisy environments, might disrupt the reliable delivery of secret shares or repositioning commands, potentially reducing identification accuracy. Third, energy constraints are a critical challenge in physical deployments, as repeated repositioning consumes significant battery power and may shorten mission duration. To address these challenges, future work could explore the integration of robust control solutions to improve the accuracy and stability of drone movements under sensor noise and control delays. Moreover, incorporating energy-aware repositioning strategies and delay-tolerant communication protocols could enhance the resilience and practicality of DRUID in operational UAV networks. Finally, it should be emphasized that these challenges are largely orthogonal to DRUID, and are instead intrinsic to the specific problem and application domain considered.

9.2. Future directions

Throughout this work, we assumed a largely stationary network. This hypothesis aligns well with multiple area coverage missions; however, the continuous changes in routing patterns and the unscheduled movements from other drones could alarm the adversary, prompting it to adapt its behavior. To the best of our knowledge, no existing malicious UAV detection method explicitly models and counteracts a reactive adversary aware of the detection scheme [49]. To address this, future research could extend DRUID to explicitly model a reactive adversary, possibly via a game-theoretic formulation.

Moreover, we assumed at most one compromised node. While Wu and Wu's cheating detection scheme can identify multiple altered shares [16], extending DRUID to handle multiple colluding adversaries would require several nontrivial changes.

Firstly, the criteria for a valid instance would become stricter, requiring as many node-disjoint paths as there are compromised drones, along with additional safe paths to employ DRUID's procedures. Second, the secret-sharing threshold t would have to increase accordingly, to at least the number of adversaries plus one, to prevent any coalition of compromised drones from reconstructing the secret; however, the current scheme does not provide a priori knowledge of the number of adversaries or where they are positioned in the network. Lastly, the NRP would need new heuristics to estimate both the direction towards each compromised path and the most likely to be the closest.

Although these challenges are significant, we believe that DRUID provides a solid foundation for addressing them. Its modular structure lends itself to future extensions targeting more sophisticated attacker models, including reactive or colluding adversaries. Incorporating game-theoretic reasoning or trust-based mechanisms into DRUID would strengthen its ability to operate in more complex attack scenarios. We identify these directions as promising opportunities for future work.

CRediT authorship contribution statement

Mauro Farina: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Erica Salvato:** Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Martino Trevisan:** Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Alberto Bartoli:** Writing – original draft, Visualization, Validation, Supervision, Project administration, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Mauro Farina reports financial support was provided by European Union. Martino Trevisan reports financial support was provided by Government of Italy. Erica Salvato reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research leading to these results has been funded by the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union - NextGenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS 00000043) and by the COMPACT (Compressed features and representations for network traffic analysis in centralized and edge Internet architectures—2022M2Z728) project, funded by the European Union - NextGenerationEU within the PRIN 2022 program (D.D. 104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions. The Ministry cannot be considered responsible for them.

Data availability

Data will be made available on request.

References

[1] Z. Sun, P. Wang, M.C. Vuran, M.A. Al-Rodhaan, A.M. Al-Dhelaan, I.F. Akyildiz, BorderSense: Border patrol through advanced wireless sensor networks, Ad Hoc Networks 9 (3) (2011) 468–477.

[2] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, J.M. Alcaraz-Calero, Search and rescue operation using UAVs: A case study, Expert Syst. Appl. 178 (2021) 114937.

[3] J. Sánchez-García, D.G. Reina, S. Toral, A distributed PSO-based exploration algorithm for a UAV network assisting a disaster scenario, Future Gener. Comput. Syst. 90 (2019) 129–148.

[4] O.M. Bushnaq, A. Chaaban, T.Y. Al-Naffouri, The role of UAV-IoT networks in future wildfire detection, IEEE Internet Things J. 8 (23) (2021) 16984–16999.

[5] K. Khan, A. Mehmood, S. Khan, M.A. Khan, Z. Iqbal, W.K. Mashwani, A survey on intrusion detection and prevention in wireless ad-hoc networks, J. Syst. Archit. 105 (2020) 101701.
[6] Y. Mekdad, A. Aris, L. Babun, A. El Fergougui, M. Conti, R. Lazzeretti, A.S. Uluagac, A survey on security and privacy issues of UAVs, Comput. Netw. 224 (2023) 109626.
[7] F. Tlili, S. Ayed, L.C. Fourati, Advancing UAV security with artificial intelligence: A comprehensive survey of techniques and future directions, Internet Things (2024) 101281.
[8] M. Yahuza, M.Y.I. Idris, I.B. Ahmedy, A.W.A. Wahab, T. Nandy, N.M. Noor, A. Bala, Internet of drones security and privacy issues: Taxonomy and open challenges, IEEE Access 9 (2021) 57243–57270.
[9] H. Rezaee, E. Salvato, G. Fenu, T. Parisini, Resilient coverage by teams of quadrotor UAVs: Theory and experiments, IEEE Trans. Control Syst. Technol. (2024).
[10] K.-Y. Tsao, T. Girdler, V.G. Vassilakis, A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks, Ad Hoc Networks 133 (2022) 102894.
[11] A. Zilberman, A. Stulman, A. Dvir, Identifying a malicious node in a UAV network, IEEE Trans. Netw. Serv. Manag. 21 (1) (2024) 1226–1240.
[12] W. Lou, W. Liu, Y. Zhang, Y. Fang, SPREAD: Improving network security by multipath routing in mobile ad hoc networks, Wirel. Netw. 15 (3) (2009) 279–294.
[13] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[14] G.R. Blakley, Safeguarding cryptographic keys, in: Managing Requirements Knowledge, International Workshop on, IEEE Computer Society, 1979, 313–313.
[15] A.K. Chattopadhyay, S. Saha, A. Nag, S. Nandi, Secret sharing: A comprehensive survey, taxonomy and applications, Comput. Sci. Rev. 51 (2024) 100608.
[16] T.-C. Wu, T.-S. Wu, Cheating detection and cheater identification in secret sharing schemes, IEE Proc., Comput. Digit. Tech. 142 (5) (1995) 367–369.
[17] G. Bansal, B. Sikdar, Fault resilient authentication architecture for drone networks, in: 2022 IEEE International Conference on Communications Workshops (ICC Workshops), IEEE, 2022, pp. 866–871.
[18] G. Bansal, B. Sikdar, Achieving secure and reliable UAV authentication: A Shamir's secret sharing based approach, IEEE Trans. Netw. Sci. Eng. (2024).
[19] N. Shenets, Authentication in dynamic peer-to-peer networks based on homomorphic secret sharing, Autom. Control. Comput. Sci. 51 (2017) 936–946.
[20] L. Gupta, R. Jain, G. Vaszkun, Survey of important issues in UAV communication networks, IEEE Commun. Surv. & Tutorials 18 (2) (2015) 1123–1152.
[21] P. Papadimitratos, Z.J. Haas, Secure data transmission in mobile ad hoc networks, in: Proceedings of the 2nd ACM Workshop on Wireless Security, 2003, pp. 41–50.
[22] D.B. Johnson, D.A. Maltz, J. Broch, et al., DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks, Ad Hoc Netw. 5 (1) (2001) 139–172.
[23] X. Dai, B. Duo, X. Yuan, W. Tang, Energy-efficient UAV communications: A generalized propulsion energy consumption model, IEEE Wirel. Commun. Lett. 11 (10) (2022) 2150–2154.
[24] H. Gong, B. Huang, B. Jia, H. Dai, Modeling power consumptions for multirotor UAVs, IEEE Trans. Aerosp. Electron. Syst. 59 (6) (2023) 7409–7422.
[25] A.A. Hagberg, D.A. Schult, P.J. Swart, Exploring network structure, dynamics, and function using networkx, in: G. Varoquaux, T. Vaught, J. Millman (Eds.), Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, 2008, pp. 11–15.
[26] S. Čapkun, J.-P. Hubaux, L. Buttyan, Mobility helps security in ad hoc networks, in: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, 2003, pp. 46–56.
[27] Y. Mulgaonkar, M. Whitzer, B. Morgan, C.M. Kroninger, A.M. Harrington, V. Kumar, Power and weight considerations in small, agile quadrotors, in: Micro-and Nanotechnology Sensors, Systems, and Applications VI, vol. 9083, SPIE, 2014, pp. 376–391.
[28] M. ul Hassan, K. Mahmood, M.K. Saeed, S. Ali, S. Zaman, A. Al Awady, M. Saqib, Smart node relocation (SNR) and connectivity restoration mechanism for wireless sensor networks, EURASIP J. Wirel. Commun. Netw. 2021 (2021) 1–19.
[29] J. Li, P. Yi, T. Duan, Z. Zhang, J. Li, Y. Wang, J. Yu, Fast connectivity restoration of UAV communication networks based on distributed hybrid MADDPG and APF algorithm, Ad Hoc Networks (2025) 103785.
[30] A. Zear, V. Ranga, Uavs assisted network partition detection and connectivity restoration in wireless sensor and actor networks, Ad Hoc Networks 130 (2022) 102823.
[31] A. Zear, V. Ranga, K.K. Gola, Network partition detection and recovery with the integration of unmanned aerial vehicle, Concurr. Comput.: Pr. Exp. 36 (13) (2024) e8048.
[32] F. Hajjej, M. Hamdi, R. Ejbali, M. Zaied, A distributed coverage hole recovery approach based on reinforcement learning for wireless sensor networks, Ad Hoc Networks 101 (2020) 102082.
[33] A. Gholami, N. Torkzaban, J.S. Baras, C. Papagianni, Joint mobility-aware UAV placement and routing in multi-hop uav relaying systems, in: Ad Hoc Networks: 12th EAI International Conference, ADHOCNETS 2020, Paris, France, November 17, 2020, Proceedings 12, Springer, 2021, pp. 55–69.
[34] S.A. Hadiwardoyo, J.-M. Dricot, C.T. Calafate, J.-C. Cano, E. Hernández-Orallo, P. Manzoni, UAV mobility model for dynamic UAV-to-car communications in 3D environments, Ad Hoc Networks 107 (2020) 102193.
[35] M. De Liso, A. Di Maio, T. Braun, Throughput-and cost-aware node relocation for MANET resiliency under jamming attacks, in: 2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet), IEEE, 2024, pp. 1–10.
[36] U. Awada, J. Zhang, S. Chen, S. Li, AirEdge: A dependency-aware multi-task orchestration in federated aerial computing, IEEE Trans. Veh. Technol. 71 (1) (2021) 805–819.
[37] U. Awada, J. Zhang, S. Chen, S. Li, S. Yang, EdgeDrones: Co-scheduling of drones for multi-location aerial computing missions, J. Netw. Comput. Appl. 215 (2023) 103632.
[38] S. Benfriha, N. Labraoui, H.B. Salameh, H. Saidi, A survey on trust management in flying ad hoc networks: Challenges, classifications, and analysis, in: 2023 Tenth International Conference on Software Defined Systems, SDS, IEEE, 2023, pp. 107–114.
[39] J. Kundu, S. Alam, J.C. Das, A. Dey, D. De, Trust based flying ad-hoc network: A survey, IEEE Access (2024).
[40] Y. Inedjaren, M. Maachaoui, B. Zeddini, J.-P. Barbot, Blockchain-based distributed management system for trust in VANET, Veh. Commun. 30 (2021) 100350.
[41] V. Chandrasekar, V. Shanmugavalli, T. Mahesh, R. Shashikumar, N. Borah, V.V. Kumar, S. Guluwadi, Secure malicious node detection in flying ad-hoc networks using enhanced AODV algorithm, Sci. Rep. 14 (1) (2024) 7818.
[42] E. Barka, C.A. Kerrache, N. Lagraa, A. Lakas, C.T. Calafate, J.-C. Cano, UNION: a trust model distinguishing intentional and UNIntentional misbehavior in inter-UAV communication, J. Adv. Transp. 2018 (1) (2018) 7475357.
[43] S. Benfriha, N. Labraoui, R. Bensaid, H. Bany Salameh, H. Saidi, FUBA: A fuzzy-based unmanned aerial vehicle behaviour analytics for trust management in flying ad-hoc networks, IET Networks 13 (3) (2024) 208–220.
[44] K. Rahman, M.A. Aziz, A.U. Kashif, T.A. Cheema, Detection of security attacks using intrusion detection system for uav networks: A survey, in: Big Data Analytics and Computational Intelligence for Cybersecurity, Springer, 2022, pp. 109–123.
[45] A.B. Mohammed, L.C. Fourati, A.M. Fakhrudeen, Comprehensive systematic review of intelligent approaches in UAV-based intrusion detection, blockchain, and network security, Comput. Netw. 239 (2024) 110140.
[46] H.J. Hadi, Y. Cao, S. Li, Y. Hu, J. Wang, S. Wang, Real-time collaborative intrusion detection system in uav networks using deep learning, IEEE Internet Things J. (2024).
[47] F. Tlili, S. Ayed, L.C. Fourati, Exhaustive distributed intrusion detection system for UAVs attacks detection and security enforcement (E-DIDS), Comput. Secur. 142 (2024) 103878.
[48] X. He, Q. Chen, L. Tang, W. Wang, T. Liu, CGAN-based collaborative intrusion detection for UAV networks: A blockchain-empowered distributed federated learning approach, IEEE Internet Things J. 10 (1) (2022) 120–132.
[49] H. Shakhatreh, A.H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N.S. Othman, A. Khreishah, M. Guizani, Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges, Ieee Access 7 (2019) 48572–48634.