A Probabilistic Framework for Localization of Attackers in MANETs*

Massimiliano Albanese[1], Alessandra De Benedictis[2], Sushil Jajodia[1], and Paulo Shakarian[3]

[1] Center for Secure Information Systems
George Mason University, Fairfax, VA, USA
{malbanes,jajodia}@gmu.edu
[2] Department of Computer Science
University of Naples "Federico II", Naples, Italy
alessandra.debenedictis@unina.it
[3] Department of Electrical Engineering and Computer Science
United States Military Academy, West Point, NY, USA
paulo@shakarian.net

Abstract. Mobile Ad Hoc Networks (MANETs) represent an attractive and cost effective solution for providing connectivity in areas where a fixed infrastructure is not available or not a viable option. However, given their wireless nature and the lack of a stable infrastructure, MANETs are susceptible to a wide range of attacks waged by malicious nodes physically located within the transmission range of legitimate nodes. Whilst most research has focused on methods for detecting attacks, we propose a novel probabilistic framework for estimating – independently of the type of attack – the physical location of attackers, based on the location of nodes that have detected malicious activity in their neighborhood. We assume that certain countermeasures can be deployed to capture or isolate malicious nodes, and they can provide feedback on whether an attacker is actually present in a target region. We are interested in (i) estimating the minimum number of countermeasures that need to be deployed to isolate all attackers, and (ii) finding the deployment that maximizes either the expected number of attackers in the target regions or the expected number of alerts *explained* by the solution, subject to a constraint on the number of countermeasures. We show that these problems are NP-hard, and propose two polynomial time heuristic algorithms to find approximate solutions. The feedback provided by deployed countermeasures is taken into account to iteratively re-deploy them until all attackers are captured. Experiments using the network simulator NS-2 show that our approach works well in practice, and both algorithms can capture over 80% of the attackers within a few deployment cycles.

Keywords: Attacker localization, MANET, probabilistic framework.

* This research was funded in part by the US Army Research Office under MURI grant W911NF-09-1-0525 and DURIP grant W911NF-11-1-0340. Part of the work was performed while Sushil Jajodia was a Visiting Researcher at the US Army Research Laboratory.

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 145–162, 2012.
© Springer-Verlag Berlin Heidelberg 2012

1 Introduction

Mobile Ad Hoc Networks (MANETs) connect mobile devices without an underlying fixed infrastructure. The topology of the network keeps changing over time as nodes move or leave, and new nodes join the network. These intrinsic features make MANETs an attractive and cost effective solution for providing connectivity in areas where a fixed infrastructure is not available or deploying one is not a technically or financially viable alternative [4]. Current applications of ad hoc networks cover a variety of areas, ranging from tactical networks for military communications to smart sensor networks for environmental monitoring. For instance, in the scenario depicted in Fig. 1(a), troops deployed on a battlefield are equipped with mobile devices forming a tactical MANET that enables them to communicate with commanders at the headquarters.

On the downside, their wireless nature and the lack of a stable infrastructure make MANETs vulnerable to a wide range of attacks, both active and passive, that can be waged by malicious nodes physically located within the transmission range of one or more legitimate nodes. In the scenario of Fig. 1(a), enemies equipped with wireless devices may be hiding in the field (e.g., attackers A_1 and A_2), or they may have simply planted wireless sensors at certain locations.

In the last couple of decades, considerable research effort has been devoted to the problem of detecting various types of attacks against wireless networks. However, despite an increasing interest in attacker localization, in both wireless sensor networks and ad-hoc mobile networks, no general solution has been devised yet. Instead, ad-hoc solutions based on the specific nature of certain attacks have been widely investigated. In particular, many different jammer localization approaches have been proposed in recent years [3,9,10].

In order to address this important problem, we propose a novel and more general approach to attacker localization in MANETs, based on a probabilistic model of the attacker's location. Specifically, we can estimate the location of

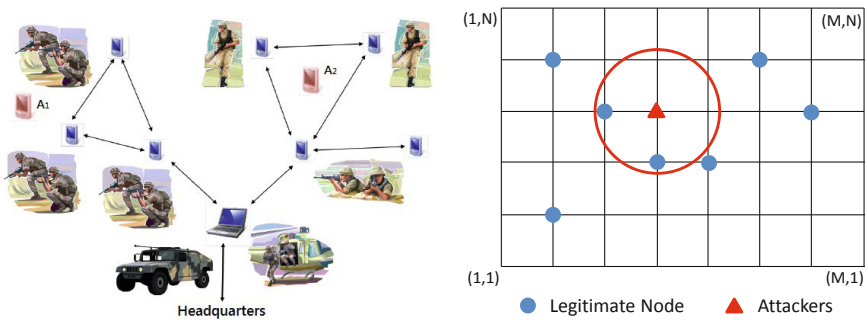

Fig. 1. Examples of (a) tactical MANETs, and (b) nodes in a discrete space

malicious nodes based on the location of all attacks detected in the network, or, more generally, the location of legitimate nodes that have detected malicious activity in their neighborhood. For the purpose of our analysis, we assume that alerts are given. Additionally, our model assumes that malicious nodes are in the neighborhood of nodes generating an alert. This assumption is justified by extensive work in the area of activity detection for MANETs. For instance, Watchdog [11] is an intrusion detection system running on each node of the network. By listening to its neighbors, each node can detect routing misbehavior. In [12], Patwardhan et al. present a similar technique. The main difference is that in Watchdog a node monitors the traffic it sends to its neighbors, whereas in [12] a node monitors the traffic between neighbors which are in range of each other.

The proposed framework does not rely on attack-specific assumptions, such as those typically used in the literature on jammer localization, namely, static wireless nodes and single attacker scenario [3]. For ease of presentation, we assume that attackers are static, but extending our framework to take mobility of malicious nodes into account is quite straightforward, and it is part of our furure plans. Additionally, we assume that a number of countermeasures are available to capture or isolate malicious nodes. Such countermeasures might include sending a patrol to physically capture malicious nodes, or running a specialized algorithm to identify and isolate malicious nodes in the regions selected by our framework. We are interested in optimally deploying available countermeasures. Specifically, we are interested in addressing the following classes of problems:

(i) Estimating the minimum number of countermeasures that need to be deployed to capture all attackers.
(ii) Finding the deployment that maximizes either the expected number of attackers in the target regions or the expected number of alerts *explained* by the solution, subject to a constraint on the number of countermeasures.

We show that these problems are NP-hard, and propose two polynomial time heuristic algorithms to find approximate solutions. Experiments show that our approach works well in practice, and both algorithms can capture over 80% of the attackers within 10-12 deployment cycles, in most scenarios.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the proposed framework, and provides a formal statement of the problems addressed in our work. Heuristic algorithms are presented in Section 4, and experimental results are reported in Section 5. Finally, Section 6 gives concluding remarks, and indicates future research directions.

2 Related Work

In recent years, there has been increasing interest in the localization of attackers in both wireless sensor networks and ad-hoc mobile networks. The vast majority of current approaches focus on specific types of attacks, most notably jamming attacks [3,9,10]. Cheng et al. [3] offer a comprehensive study of the jammer localization problem, and propose a simple yet effective algorithm called *Double*

Circle Localization (DCL). They assume all nodes in the network (i) are deployed randomly; (ii) are static; (iii) have the same capability (e.g. transmission power); (iv) know their own location; and (v) can recognize whether they are jammed. They consider a single-jammer scenario, and apply the free-space propagation model, according to which jamming signals attenuate with distance. The proposed algorithm calculates the minimum bounding circle and the maximum inscribed circle of the convex hull of the set of jammed nodes, and combines their centers to estimate the location of the jammer. They show that their algorithm outperforms three existing geometry-based algorithms, namely, Centroid Localization (CL), Weighted Centroid Localization (WCL), Virtual Force Iterative Localization (VFIL) [10]. The CL algorithm estimates the position of the jammer by simply averaging the coordinates of all jammed nodes. The WCL algorithm [2] weights the contribution of each jammed node when computing the centroid. One way of assigning weights is based on the distance between the jammer and the affected node, which can be estimated by measuring the strength of the incoming radio signal. VFIL tries to improve CL by adjusting its estimation based on the distribution of jammed nodes.

Other approaches focus on identifying nodes or subnetworks that are affected by attacks, but they provide a very coarse grained estimate of the attacker's position. Kim and Song [8] present a simple approach for fast detection of attacks using CCA (Clear Channel Assessment) values – a measure of the availability of a communication medium. If a node tries to send a message and finds the channel busy, its CCA value is increased and the node will retry to retransmit later. If the CCA value of a node exceeds a given threshold, this mechanism judges that the node is attacked, or otherwise affected by nearby attacks.

Han et al. [6] address the problem of attackers intentionally hiding or falsifying their position in order to decrease the accuracy of the localization process, which is traditionally executed by multiple observers (usually Access Points) which can simultaneously observe the intruder's transmissions and use time delays, angle of arrival, or signal strength information to localize the intruder. They propose a proactive technique, named Access Point Coordinated Localization (APCL), that forces the attacker to reveal undistorted signal features unintentionally, in order to subsequently use traditional localization techniques.

Our approach significantly differs from existing literature, in that it seeks a more general solution to the problem of attacker localization in MANETs. Research in this area has mostly focused on specific types of attacks, requiring several simplifying assumptions. We drop most such assumptions, and show that our framework can deal with different types of attacks, including jamming.

3 Probabilistic Framework

In this section, we present our probabilistic framework for attacker localization in MANETs. We first provide some technical preliminaries in Section 3.1, and then present the framework in detail (Section 3.2). We conclude in Section 3.3 by providing a formal statement of the problems addressed in this paper.

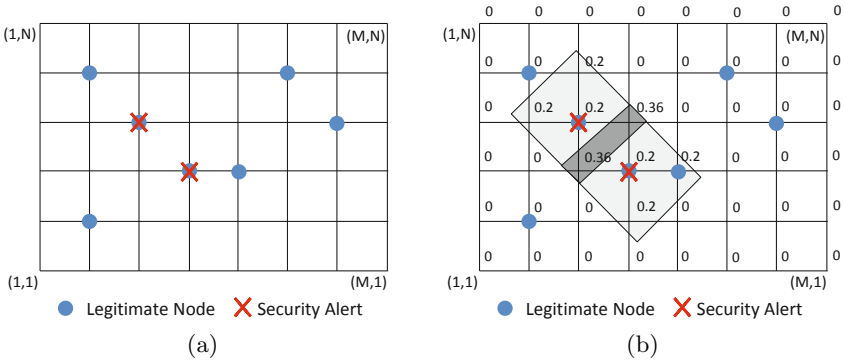

Fig. 2. Examples of (a) security alerts, and (b) computation of $\Pr(attacker(p))$

3.1 Technical Preliminaries

Without loss of generality we assume a discrete notion of space, as formalized by Definition 1, and assume that, at any time, both legitimate and malicious nodes are at one of a finite number of discrete locations.

Definition 1 (Space). *Given two integers* $M, N \in \mathbb{N}$*, a* space $\mathcal{S} = \{1, \ldots, M\} \times \{1, \ldots, N\}$ *is a finite subset of points of* $\mathbb{N}^2$*.*

For ease of modeling, the above definition assumes that a space is a rectangular region within $\mathbb{N}^2$. Fig. 1(b) shows a discrete space for $M = 8$ and $N = 6$. In this example, legitimate nodes are located at points $(2, 2)$, $(2, 5)$, $(3, 4)$, $(4, 3)$, $(5, 3)$, $(6, 5)$, and $(7, 4)$, and an attacker is located at point $(4, 4)$.

Associated with the space is a distance function $dist : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ that satisfies the normal distance axioms:

- Positive definiteness: $\forall p_1, p_2\ dist(p_1, p_2) \geq 0$ and $dist(p_1, p_2) = 0 \Leftrightarrow p_1 = p_2$
- Symmetry: $\forall p_1, p_2\ dist(p_1, p_2) = dist(p_2, p_1)$
- Triangle inequality: $\forall p_1, p_2, p_3\ dist(p_1, p_2) + dist(p_2, p_3) \geq dist(p_1, p_3)$

Given the above notion of space, a MANET $\mathcal{M}$ at time t can be represented, for the purpose of our analysis, as a subset of $\mathcal{S}$ including all the points where a mobile node is deployed at time t. A set $\mathcal{A}$ of alerts can be represented as a set of pairs $a = (p, t)$, where t is the time at which an alert a was generated and p is the location at time t of the node triggering the alert.

3.2 Framework

Given an alert, we first need to define the probability that an attacker located within range of the alert's location is responsible for causing the alert. We use the binary predicate $causes : \mathcal{S} \times \mathcal{A} \rightarrow \{true, false\}$ to specify if there is an attacker at point $p \in \mathcal{S}$ causing an alert $a \in \mathcal{A}$. We assume that the transmission range $r \in \mathbb{R}$ is fixed and equal for all legitimate nodes and attackers.

Definition 2 (Attacker's Probability Distribution). *Let $a \in \mathcal{A}$ be an alert. The attacker's probability distribution for a, denoted θ_a, is a probability distribution over $\mathcal{S}$ defined as follows:*

$$\theta_a(p) = \Pr(causes(p, a)) \tag{1}$$

s.t.

$$\theta_a(p) \begin{cases} \geq 0, \text{ if } dist(p, a) <= r \\ = 0, \text{ if } dist(p, a) > r \end{cases} \tag{2}$$

where $dist : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ is the distance associated with the space $\mathcal{S}$.

Note that, for all $a \in \mathcal{A}$, $\sum_{p \in \mathcal{S}} \theta_a(p) = 1$, i.e., the attacker who caused a must be in $\mathcal{S}$. Intuitively, if a node in the MANET has been attacked, this node must be within an attacker's transmission range r. Given an alert $a \in \mathcal{A}$, we use S_a to denote the set of points $S_a = \{p \in \mathcal{S} \mid \theta_a(p) \geq 0\}$. In other words, the attacker who caused a must in S_a. We do not assume a specific distribution $\theta_a(p)$. Any distribution can be used in our framework, as long as the properties described by Equation 2 are satisfied. The choice of a specific distribution depends on a number of factors, including the radio propagation model, and the information available to nodes triggering an alert. In the simplest case, we can assume that the attacker's probability is uniformly distributed in a circular region of radius r centered at the alert's location. If more information is available, we can add constraints to possible attacker locations. For instance, when signals propagate according to the free space model and receivers can measure the received signal power the attacker's probability is uniformly distributed in an annulus of radius $d \pm \epsilon$, where d is the estimated distance of the attacker and ϵ is a tolerance parameter. The free space propagation model assumes the ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver. In [5], H. T. Friis presented the following equation to calculate the received signal power in free space at distance d from the transmitter.

$$P_r(d) = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4 \cdot \pi)^2 \cdot d^2 \cdot L} \tag{3}$$

Where P_t is the transmitted signal power, G_t and G_r are the antenna gains of transmitter and receiver respectively, L is the system loss, and λ is the wavelength. The free space model basically represents the communication range as a circle around the transmitter.

We now introduce the notion of an *explanation*. Intuitively, an explanation is a set of points such that the presence of an attacker at each of these points would *explain* all the alerts that were generated.

Definition 3 (Explanation). *Let $\mathcal{S}$ be a space, and let $\mathcal{A}$ be a set of alerts triggered by nodes of a MANET $\mathcal{M}$ deployed over $\mathcal{S}$. An explanation E for $\mathcal{A}$ is a subset of $\mathcal{S}$ s.t. for all $a \in \mathcal{A}$, $E \cap S_a \neq \emptyset$. We use $\mathcal{E}$ to denote the set of all possible explanations. An explanation is said to be* minimal *iff $\nexists E' \in \mathcal{E}$* s.t. $|E'| < |E|$.

In Definition 2, we introduced the probability $\theta_a(p)$ that an attacker in p is responsible for a single alert a. Given a set of alerts, we are interested in finding the probability that any given point $p \in \mathcal{S}$ hosts an attacker. We use the unary predicate $attacker : \mathcal{S} \rightarrow \{true, false\}$ to specify if there is an attacker at a point $p \in \mathcal{S}$. If we assume that, given any two points $p_1, p_2 \in \mathcal{S}$ and any two alerts $a_1, a_2 \in \mathcal{A}$, $causes(p_1, a_1)$ and $causes(p_2, a_2)$ are independent, then the following result can be proved. We refer to this assumption as *causality independence*.

Proposition 1. *Given a space $\mathcal{S}$, and a set of alerts $\mathcal{A}$, the following property holds under causality independence:*

$$(\forall p \in \mathcal{S}) \left(\Pr(attacker(p)) = 1 - \prod_{a \in \mathcal{A}} (1 - \theta_a(p)) \right) \tag{4}$$

Example 1. Consider the scenario of Fig. 2(a), and assume that $r = 1.1$ and θ_a is uniformly distributed over $S_a = \{p \in \mathcal{S} \mid dist(p, a) \leq r\}$. By computing the value of $\Pr(attacker(p))$ according to Equation 4, we obtain the result shown in Fig. 2(b). The light-shaded region comprises all points $p \in \mathcal{S}$ s.t. $\Pr(attacker(p))$ is greater than 0. In other words, the attacker(s) must be in that region. The dark-shaded region only contains points with $\Pr(attacker(p)) > 0.3$.

Definition 4 (Expected Number of Attackers). *Given a set of points $D \subseteq \mathcal{S}$, the number of attackers in D is a random variable N_a^D which can assume value $|E \cap D|$ for any $E \in \mathcal{E}$. The* expected number of attackers *in D is*

$$\mathrm{Ex}\left[N_a^D\right] = \sum_{E \in \mathcal{E}} \Pr(E) \cdot |E \cap D| \tag{5}$$

where $\Pr(E)$ is the probability of explanation E.

Intuitively, the expected number of attackers is the weighted average, over all possible explanations, of the number of attackers in an explanation, where the weight of an explanation is its probability. The expected numbers of attackers in $\mathcal{S}$ is $\mathrm{Ex}\left[N_a^{\mathcal{S}}\right] = \sum_{E \in \mathcal{E}} \Pr(E) \cdot |E|$. In the example of Fig. 2(b), $E\left[N_a^{\mathcal{S}}\right] = 1.92$, meaning that, although most explanations include two attackers, single-attacker explanations also exist (e.g., $E_1 = \{3, 3\}$). The following result shows that, under *causality independence*, calculating $\mathrm{Ex}\left[N_a^D\right]$ is computable in polynomial time.

Proposition 2. *Let $D \subseteq \mathcal{S}$ be a set of points in $\mathcal{S}$ and let N_a^D denote the number of attackers in D. The following property holds.*

$$\mathrm{Ex}\left[N_a^D\right] = \sum_{E \in \mathcal{E}} \Pr(E) \cdot |E \cap D| = \sum_{p \in D} \Pr(attacker(p)) \tag{6}$$

3.3 Problem Statement

Given an ad-hoc network deployed over a space $\mathcal{S}$, and a set of alerts $\mathcal{A}$, we are interested in optimally deploying a limited number of resources in order

to capture the attackers responsible for the alerts. We assume that a deployed resource can capture all malicious nodes within a capture range defined as a circle of radius l – with $l \ll r$ – centered at the location of the resource. We use the term *deployment* to refer to a set of points in $\mathcal{S}$ where resources are deployed. We can define the following three optimization problems.

Problem 1 (Minimize deployment size). Given a space $\mathcal{S}$, a set of alerts $\mathcal{A}$ over $\mathcal{S}$, and a probability threshold $\tau \in [0, 1]$, find a deployment $D \subseteq \mathcal{S}$ of minimum size that *sufficiently* explains all the alerts in $\mathcal{A}$.

$$\begin{aligned} &\textbf{minimize}_{D \in 2^{\mathcal{S}}} |D| \\ &\quad \textbf{subject to} \\ &\qquad (\forall a \in \mathcal{A}) \ \textstyle\sum_{p \in D} \theta_a(p) \geq \tau \end{aligned} \tag{7}$$

In this optimization problem, the constraints require that D be an explanation (see Definition 3), and each alert be explained with probability equal to or greater than a threshold τ. We wish to minimize the number of resources deployed.

Problem 2 (Maximize expected number of attackers). Given a space $\mathcal{S}$, a set of alerts $\mathcal{A}$ over $\mathcal{S}$, a positive integer $k \in \mathbb{N}^+$, and a probability threshold $\tau \in [0, 1]$, find a deployment $D \subseteq \mathcal{S}$ of size k or less that *sufficiently* explains all the alerts in $\mathcal{A}$, and maximizes the expected number of attackers in D under causality independence.

$$\begin{aligned} &\textbf{maximize}_{D \in 2^{\mathcal{S}}} \textstyle\sum_{p \in D} \Pr(\mathit{attacker}(p)) \\ &\quad \textbf{subject to} \\ &\qquad |D| \leq k \\ &\qquad (\forall a \in \mathcal{A}) \ \textstyle\sum_{p \in D} \theta_a(p) \geq \tau \end{aligned} \tag{8}$$

In this problem, the first constraint limits the number of resources that can be deployed, whereas the second set of constraints, similarly to Problem 1, require that D be an explanation (see Definition 3), and each alert be explained with probability equal to or greater than a threshold τ. The objective function is the expected number of attackers in D, which, based on Proposition 2, can be computed as the sum over $p \in D$ of $\Pr(\mathit{attacker}(p))$.

Problem 3 (Maximize expected number of explained alerts). Given a space $\mathcal{S}$, a set of alerts $\mathcal{A}$ over $\mathcal{S}$, a positive integer $k \in \mathbb{N}^+$, and a probability threshold $\tau \in [0, 1]$, find a deployment $D \subseteq \mathcal{S}$ of size k or less that *sufficiently* explains all the alerts in $\mathcal{A}$, and maximizes the expected number of alerts that would be explained by attackers located at each point in D.

$$\begin{aligned} &\textbf{maximize}_{D \in 2^{\mathcal{S}}} \textstyle\sum_{a \in \mathcal{A}} \sum_{p \in D} \theta_a(p) \\ &\quad \textbf{subject to} \\ &\qquad |D| \leq k \\ &\qquad (\forall a \in \mathcal{A}) \ \textstyle\sum_{p \in D} \theta_a(p) \geq \tau \end{aligned} \tag{9}$$

In this problem, the constraints are the same as in the previous problem, but the objective is to maximize the expected number of alerts that the presence of an attacker in each point of D would explain. Note that, unlike Problem 2, Problem 3 does not use any independence assumptions.

Algorithm 1. MIN-K$(S, \mathcal{A}, \tau, l)$

Input: set of points S, set of alerts $\mathcal{A}$, threshold τ, and capture radius l
Output: deployment $D \subseteq S$

1: $D \leftarrow \emptyset$
2: $\mathcal{A}^* \leftarrow \emptyset$ // Alerts covered
3: **while** $\mathcal{A}^* \neq \mathcal{A} \wedge S \neq \emptyset$ **do**
4: $S^{'} \leftarrow \{p_i \in S \setminus D \mid |\mathcal{A}_i^* \setminus \mathcal{A}^*| \text{ is maximum}\}$ // $\mathcal{A}_i^*$: set of alerts covered by p_i
5: $S^{''} \leftarrow \{p_j \in S^{'} \mid \Pr(attacker(p_j)) \text{ is maximum}\}$
6: $p \leftarrow$ randomly selected point from $S^{''}$
7: $D \leftarrow D \cup \{p\}$
8: $S \leftarrow S \setminus \{p_k \in S \mid dist(p_k, p) \leq l\}$
9: $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{a_k \in \mathcal{A} \mid \sum_{q \in D} \Pr(causes(q, a_k)) \geq \tau\}$
10: **end while**
11: **return** D

4 Algorithms

In this section, we first show that all the three problems defined in the previous section are NP-Hard, and then presents two polynomial heuristic algorithms that offer good approximation guarantees. The first of these two algorithm solves Problem 1, whereas the second algorithm solves both Problem 2 and Problem 3.

Theorem 1. *Problems 1, 2, and 3 are NP-Hard.*

Proof. Problem 1 can be shown to be NP-Hard by reduction from the set cover problem, which is known to be NP-Hard. Specifically, the universe in the set cover problem can be treated as the set of alerts $\mathcal{A}$ to be covered (*explained*) and the several subsets of the universe can be treated as the candidate locations explaining subsets of $\mathcal{A}$. As Problem 1 is NP-Hard, the corresponding decision problem, where there is some cardinality constraint of k on the solution, is also NP-Hard (and easily shown to be NP-Complete). Therefore, finding any solution that meets the constraints of Problems 2 and 3 is NP-Hard as well.

4.1 Algorithm MIN-K

Algorithm MIN-K (Algorithm 1) approximates Problem 1, and it is inspired by the heuristic algorithm for solving the set covering problem [7]. Given a set of elements, called the universe, and n sets whose union comprises the universe, the set cover problem is to identify the smallest number of sets whose union still contains all elements in the universe. In our case, given a set of alerts, we are interested in identifying the smallest number of locations that can explain all the alerts, where each location explains one or more alerts.

The algorithm takes as input a set $S \subseteq \mathcal{S}$ of candidate points, a set of alerts $\mathcal{A}$, a threshold τ, and a capture radius l, and returns a deployment $D \subseteq S$. Lines 4-9 are iterated until either all the alerts are covered ($\mathcal{A}^* = \mathcal{A}$) or there

Algorithm 2. MULT-UPD$(S, \mathcal{A}, k, \tau, l)$

Input: set of points S, set of alerts $\mathcal{A}$, integer k, threshold τ, and capture radius l
Output: deployment $D \subseteq S$

1: $D \leftarrow \emptyset$
2: $\lambda \leftarrow e^{k-\tau} \cdot (1 + |\mathcal{A}|)$
3: $\mathcal{A}^* \leftarrow \emptyset$ // Alerts covered
4: **for all** $a_i \in \mathcal{A}$ **do**
5: $w_i \leftarrow \frac{1}{(k-\tau)}$
6: **end for**
7: **while** $|D| < k \wedge \mathcal{A}^* \neq \mathcal{A} \wedge S \neq \emptyset$ **do**
8: $S' \leftarrow \{p_i \in S \setminus D \mid \frac{\sum_{a_i \in \mathcal{A}}(w_i - w_i \cdot \theta_{a_i}(p_j))}{f(D \cup \{p_j\}) - f(D)}$ is minimum$\}$
9: $p \leftarrow$ randomly selected point from S'
10: $D \leftarrow D \cup \{p\}$
11: $S \leftarrow S \setminus \{p_k \in S \mid dist(p_k, p) \leq l\}$
12: $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{a_k \in \mathcal{A} \mid \Pr(causes(p, a_k)) \geq \tau\}$
13: **for all** $a_i \in \mathcal{A}$ **do**
14: $w_i \leftarrow w_i \cdot \lambda^{(1-\theta_{a_i}(p_j))/(k-\tau)}$
15: **end for**
16: **end while**
17: **return** D

are no more candidate points to examine ($S = \emptyset$). The algorithm first considers the set $S' \subseteq S$ such that the number of additional alerts covered by each point $p_i \in S'$ – w.r.t. the set $\mathcal{A}^*$ of alerts covered so far – is maximum (Line 4). Then, a point p is randomly selected from the set S'' of points p_j in S' having maximum value of $\Pr(attacker(p_j))$ (Lines 5-6). Finally, p is added to the solution and all points with a radius l from p are excluded from further consideration (Lines 7-8). All the alerts a_k that are sufficiently explained by D are added to $\mathcal{A}^*$ (Line 9).

Proposition 3. *MIN-K runs in $O(r^2 \cdot |\mathcal{A}|^2)$ time.*

Proof. The outer loop of the algorithm takes no more than $|\mathcal{A}|$ steps. The bound on the inner loop is $O(r^2 \cdot |\mathcal{A}|)$ iterations, as the number of points to consider at each step is proportional to a node's transmission area, that is $\pi \cdot r^2$.

4.2 Algorithm MULT-UPD

Algorithm MULT-UPD (Algorithm 2) is a multiplicative-updates algorithm that can be used to approximate solutions to Problems 2 and 3. It is based on the multiplicative-updates algorithm of [1], which is designed to find approximate solutions to the maximization of a submodular function with respect to packing constraints. We show in Proposition 5 that this algorithm runs in polynomial time, though we do not guarantee it provides optimal solutions (Theorem 1 suggests that an efficient polynomial algorithm that provides optimal solutions is unlikely). We apply the algorithm of [1] by embedding Problem 2 or 3 into a packing problem. As this algorithm is used for both problems, we will use the

notation $f : 2^{\mathcal{S}} \rightarrow \mathbb{R}$ to denote a generic objective function. The definition of f depends on which problem is being solved. For Problem 2, it is:

$$f(D) = \sum_{p \in D} \Pr(attacker(p)) \tag{10}$$

For Problem 3, it is defined as follows:

$$f(D) = \sum_{a \in \mathcal{A}} \sum_{p \in D} \theta_a(p) \tag{11}$$

As stated earlier, the algorithm of [1] is designed to find a solution to maximize a *submodular* function. As f above is additive under either problem, submodularity follows trivially.[1] What remains to be shown is that our problems can be re-written as packing problems. We prove this in the following proposition.

Proposition 4. *The constraints of Problems 2 and 3 can be re-written as:*

$$\begin{array}{l} |D| = k \\ (\forall a \in \mathcal{A}) \ \sum_{p \in D}(1 - \theta_a(p)) \leq k - \tau \end{array} \tag{12}$$

Proof. For any $a \in \mathcal{A}$, we can re-write the original constraint as $k - \sum_{p \in D} \theta_a(p) \leq k - \tau$. We also note that the size of D must be k (except in a degenerate case). Therefore, we can re-write the constraint again as $\sum_{p \in D}(1 - \theta_a(p)) \leq k - \tau$.

With this embedding in mind, the algorithm functions by associating a weight with each alert (corresponding to the constraint that the alert must be explained with probability τ). The algorithm then proceeds in a generally greedy fashion – but every time an element is added to D, the weights for all constraints that are not met increase. When the algorithm makes a greedy selection, these weights are considered in addition to the increase experienced by the objective function. Though the algorithm of [1] provides an approximation ratio, this ratio does not apply to our embedding as the original algorithm allows solutions of less than size k (this is because Proposition 4 does not necessarily hold for approximate solutions). As a consequence, some of the alerts are not explained within probability τ. These alerts can be thought of as "difficult to explain" given the resource constraint (k). However, this is acceptable for our application as we look to provide iterative deployments of the resources (see algorithm ITER-DEP in the next section) so unexplained alerts will likely be covered in a subsequent deployment. We show that this algorithm runs in polynomial time.

Proposition 5. *MULT-UPD runs in $O(k \cdot r^2 \cdot |\mathcal{A}|^2)$ time.*

Proof. The outer loop of the algorithm takes no more than k steps. The bound on the inner loop is $O(r^2 \cdot |\mathcal{A}|)$ iterations, as the number of points to consider for each alert is proportional to the a node's transmission area, that is $\pi \cdot r^2$, and the calculation at Line 8 requires $O(|\mathcal{A}|)$ time.

[1] As does some other requirements such as monotonicity and that $f(\emptyset) = 0$.

Algorithm 3. ITER-DEP$(\mathcal{S}, \mathcal{A}, M, \tau, l)$

Input: space $\mathcal{S}$, set of alerts $\mathcal{A}$, max number of deployment cycles M, threshold τ, and capture radius l

```
 1: A' ← A                    // Alerts to explain
 2: count ← 0                 // Iteration counter
 3: for all p ∈ S do
 4:    compute Pr(attacker(p))
 5: end for
 6: S ← {p ∈ S | Pr(attacker(p)) > 0} = ⋃_{a∈A} S_a
 7: while S ≠ ∅ ∧ count < M ∧ A' ≠ ∅ do
 8:    D ← computeDeployment(S, A', k, τ, l)
 9:    for all p ∈ D s.t. p is a hit do
10:       for all q ∈ S s.t. dist(p, q) < l and q is an attacker do
11:          for all a ∈ A' s.t. Pr(causes(q, a)) ≥ τ do
12:             S ← S \ {s ∈ S | Pr(causes(s, a)) ≥ 0}
13:          end for
14:          A' ← A' \ {a ∈ A' | Pr(causes(q, a)) ≥ τ}
15:       end for
16:    end for
17:    for all p ∈ S do
18:       compute Pr(attacker(p))
19:    end for
20: end while
```

4.3 Algorithm ITER-DEP

Algorithms MIN-K and MULT-UPD both compute a single deployment of countermeasures. As the number of countermeasures deployable at each step may be limited (Problems 2 and 3), and some of the deployed countermeasures may result in false positives, it may not be possible in practice to capture all malicious nodes in a single deployment.

Algorithm ITER-DEP (Algorithm 3) takes as input a space $\mathcal{S}$, a set of alerts $\mathcal{A}$, the maximum number of deployment cycles M, a threshold τ, and a capture radius l, and iteratively redeploys countermeasures taking into account feedback from countermeasures deployed in the previous cycle. The algorithm first computes the initial values of $\Pr(attacker(p))$ under causality independence (Lines 3-5), and, based on those, a set S of candidate locations (Line 6). It then iterates until there are no more locations to consider, or all the alerts have been explained, or the maximum number of iterations has been reached (Lines 7-20). During each iteration, a deployment is computed using any of the algorithms presented earlier (Line 8). For each true positive, the set of alerts is updated by removing those that are sufficiently explained by the captured attacker, and the set of candidate points is updated accordingly (Lines 9-16). Lastly, the values of $\Pr(attacker(p))$ are updated.

Proposition 6. *ITER-DEP runs in $O(M \cdot r^2 \cdot |\mathcal{A}|^2)$ time when MIN-K is used, and $O(M \cdot k \cdot r^2 \cdot |\mathcal{A}|^2)$ time when MULT-UPD is used.*

Proof. The outer loop of the algorithm (Lines 7-20) takes no more than M steps. The complexity of Lines 8-19 is dominated by the complexity of Line 8, that is the complexity of the deployment algorithm – $O(r^2 \cdot |\mathcal{A}|^2)$ and $O(k \cdot r^2 \cdot |\mathcal{A}|^2)$ for MIN-K and MULT-UPD respectively. In fact, the bound on the loop at Lines 9-16 is $O(r^2 \cdot |\mathcal{A}|^2)$, as Line 13 is executed $|\mathcal{A}| \cdot \pi \cdot r^2 \cdot |\mathcal{A}|$ times, in the worst case.

5 Experiments

This section reports on the experiments we conducted to validate our framework. Additional experiments, showing how our approach can be used to localize jammers, are reported in Appendix A. We used NS-2 to simulate different network scenarios, with nodes moving according to a Random Way Point model[2], and attackers randomly choosing one or more of their neighbors as their targets.

We implemented a prototype of the proposed framework as a Java application that takes as input log files generated by NS-2 and containing detailed information about all the alerts. We studied algorithms MIN-K and MULT-UPD in terms of (i) number of deployment cycles needed to capture all the attackers, and (ii) time to compute a deployment as the number of alerts increases.

5.1 Experimental Setup

In our experiments, we considered a $20km \times 20km$ field, using a 10-meter granularity, and deployed 4,000 network nodes and about 600 attackers, both uniformly distributed in this area. Overall, attackers triggered more than 1,000 alerts. For the purpose of these experiments, we assumed that deployable countermeasures are physical resources, such as patrols, that can capture attackers within a 30-meter radius l. After a resource has been deployed, points within a radius l are assumed to be free of malicious nodes. With this assumption, all resources can be reused in the next deployment cycle.

As discussed in Section 3.2, several models can be use to compute the *attacker's probability distribution* of Definition 2. In our experiments, we assume all nodes are compatible with the free space radio propagation model and have a transmission range of 250 meters. Assuming that we can link an alert to a specific communication attempt from the attacker node, and that attackers are not able to falsify their beam direction and radio parameters to distort signal features, we can use the received signal power to estimate the distance d between the attacker and the victim, based on the selected radio propagation model. In this case, the probability distribution θ_a for an alert a can be assumed to be uniform in a annulus of radius $d \pm \epsilon$ centered at the alert's location (we set $\epsilon = 0.1 \cdot r$, where r is the transmission range). Fig. 3 shows the result of computing $\Pr(attacker(p))$ for a simple scenario, based on assumptions discussed above. Different colors denote different levels of probability values, with darkest areas being more likely to contain attackers. Points in high probability areas may explain multiple alerts.

[2] However, any mobility and radio propagation model can be used in our framework.

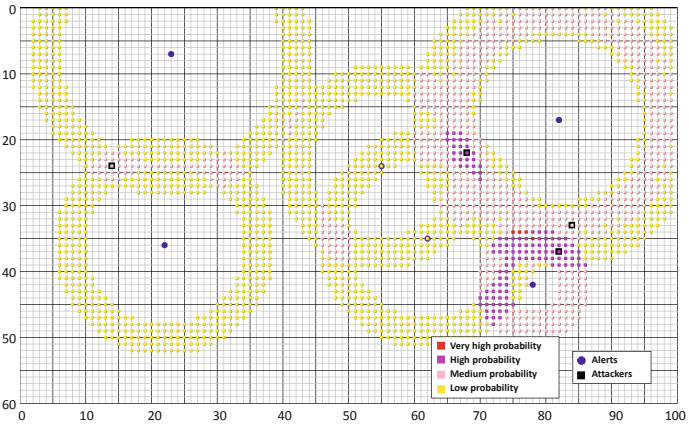

Fig. 3. Values of Pr($attacker(p)$) for a simple scenario

In order to compare the two algorithms, we first run MIN-K and found the minimum number of resources needed to cover all the alerts in a single deployment cycle, and then used this number as the value of k in MULT-UPD. To cover all the alerts in the scenario described above, we need about 500 resources.

5.2 Experimental Results

We analyzed the convergence of the proposed approach with respect to *recall*[3], which we measured as the fraction of attackers *captured* in each deployment cycle. Specifically, we use a notion of *cumulative recall*, as for each deployment cycle we count the total number of attackers captured since start.

Fig. 4 shows the cumulative recall function for the algorithms proposed: MIN-K, and the two versions of MULT-UPD maximizing expected number of attackers (MULT-UPD $v1$ in figure) and expected number of explained alerts (MULT-UPD $v2$) respectively. As shown, all the algorithms can capture 80% of the attackers in a very few deployment cycles, even in unfavorable scenarios, such as low network density. Indeed, convergence is influenced by network configuration and density: if an attacker triggers more alerts, it will be localized in a higher probability area and will be captured faster.

We also studied how the time for computing a single deployment varies when the number of alerts increases. As shown in Fig. 5, MIN-K performs significantly better than the two versions of MULT-UPD: it takes about 200 seconds to process around 1,000 alerts, while both versions of MULT-UPD take more then 700 seconds (about 12 minutes). However, both algorithms run in time quadratic in the number of alerts, as shown by the trend lines in Fig. 5. This confirms the theoretical results presented earlier in Section 4.

[3] Recall is a widely adopted measure in the information retrieval and pattern recognition fields, indicating the fraction of relevant instances retrieved by an algorithm.

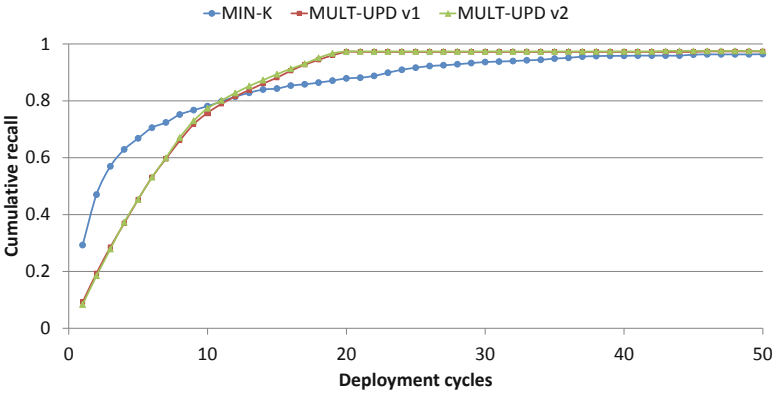

Fig. 4. Cumulative recall over subsequent deployment cycles

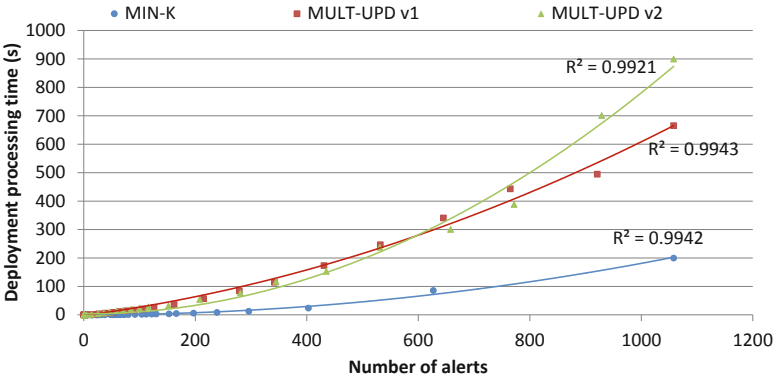

Fig. 5. Deployment processing time vs. number of alerts

6 Conclusions

In this paper, we have presented a probabilistic framework for the localization of attackers in Mobile Ad Hoc Networks (MANETs). Prior to our work, no general solution was devised to address this important problem, and most proposed approaches focused on specific types of attacks, most notably jammer attacks. The proposed framework can estimate the physical location of attackers, based on the location of nodes that have detected malicious activity in their neighborhood.

We assume that certain countermeasures can be deployed to capture or isolate malicious nodes, and they can provide feedback about the actual presence of an attacker in the target regions. We presented different variants of the localization problem, and we showed that all of them are NP-hard. We then proposed two polynomial heuristic algorithms that can compute approximate solutions. The feedback provided by deployed countermeasures is taken into account to

iteratively re-deploy countermeasures until all attackers are captured. Experiments showed that our approach works well in practice, and both algorithms can capture over 80% of the attackers within a few deployment cycles.

Our future plans include removing the assumption that attackers are static, and extending our framework to be able to track moving attackers.

References

1. Azar, Y., Gamzu, I.: Efficient submodular function maximization under linear packing constraints. The Computing Research Repository (CoRR) (July 2010)
2. Blumenthal, J., Grossmann, R., Golatowski, F., Timmermann, D.: Weighted centroid localization in zigbee-based sensor networks. In: Proceedings of the IEEE International Symposium on Intelligent Signal Processing (WISP 2007) (October 2007)
3. Cheng, T., Li, P., Zhu, S.: An algorithm for jammer localization in wireless sensor networks. In: Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA), Fukuoka, Japan (March 2012)
4. Datta, R., Marchang, N.: Security for Mobile Ad Hoc Networks. In: Handbook on Securing Cyber-Physical Critical Infrastructure, pp. 147–190. Morgan Kaufmann (January 2012)
5. Friis, H.T.: A note on a simple transmission formula. Proceedings of the IRE 34(5), 254–256 (1946)
6. Han, C., Zhan, S., Yang, Y.: Proactive attacker localization in wireless LAN. SIGCOMM Computer Communication Review 39(2), 27–33 (2009)
7. Johnson, D.S.: Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9, 256–278 (1974)
8. Kim, Y.-J., Song, S.: The Feasibility Study of Attacker Localization in Wireless Sensor Networks. In: Kim, T.-h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) UCMA 2011, Part II. CCIS, vol. 151, pp. 180–190. Springer, Heidelberg (2011)
9. Liu, H., Liu, Z., Chen, Y., Xu, W.: Determining the position of a jammer using a virtual-force iterative approach. Wireless Networks 17(2), 531–547 (2011)
10. Liu, H., Xu, W., Chen, Y., Liu, Z.: Localizing jammers in wireless networks. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom 2009), Galveston, TX, USA (March 2009)
11. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000), Boston, MA, USA, pp. 255–265 (August 2000)
12. Patwardhan, A., Parker, J., Iorga, M., Karygiannis, T.: Secure routing and intrusion detection in ad hoc networks. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), Kauai Island, HI, USA, pp. 191–199 (March 2005)

A Localizing Jammers

In this appendix, we study the performance of our framework with respect to jamming attacks. Although many countermeasures have been proposed against

jamming attacks, mostly based on frequency manipulation, it would be desirable to capture and deactivate jammers upon detection, as their activity heavily affects power consumption.

In the following, we first compare MIN-K and MULT-UPD w.r.t. localization error for a complex scenario involving multiple jammers, and show that MIN-K guarantees lower error. We then compare MIN-K with the Double Circle Localization (DCL) algorithm [3], and show that MIN-K offers better performance and lower sensitiveness to network density.

We considered a $20km \times 20km$ field and randomly placed 185 jammers, which jammed 445 nodes; we run MIN-K and MULT-UPD 100 times and recorded, for each jammer, the minimum distance from a deployed resource in the first deployment cycle. The cumulative distribution function of the error is shown in Fig. 6. As expected, the localization error of MIN-K is much smaller than MULT-UPD. In fact, the mechanism used by MIN-K to choose deployment points, unlike MULT-UPD, is aimed at covering all the alerts: this means that each resource will be deployed within distance r from an alert. As the attacker is also be within distance r from the alert, the maximum possible error is twice the transmission range. When running MULT-UPD, an alert might not be covered in the first deployment cycle, so the responsible jammer could be very far from any deployed resource.

We now compare MIN-K with the Double Circle Localization (DCL) algorithm [3]. The authors of DCL considered a square field of $100m^2$ with uniformly distributed nodes having a transmission range of 10 meters; they placed the jammer at the center of the field and evaluated the accuracy of jammer localization for two different network densities, 1 and 3 nodes per m^2 respectively, showing that the algorithm is able to achieve a very small error. In fact, 100% of the results were computed with an error smaller than 10 meters for the lowest density

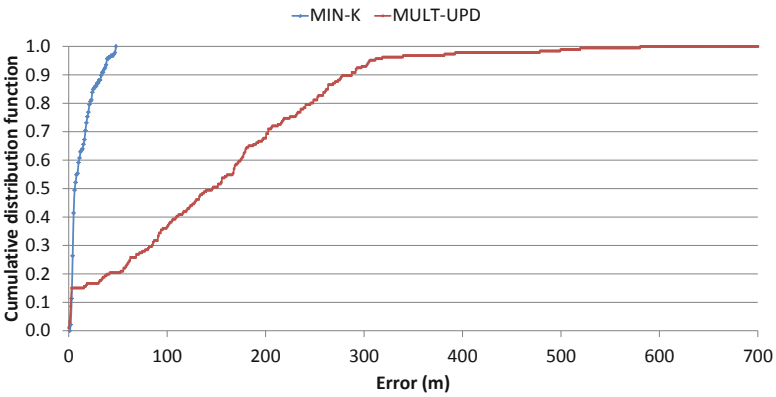

Fig. 6. CDF of the localization error for MIN-K and MULT-UPD

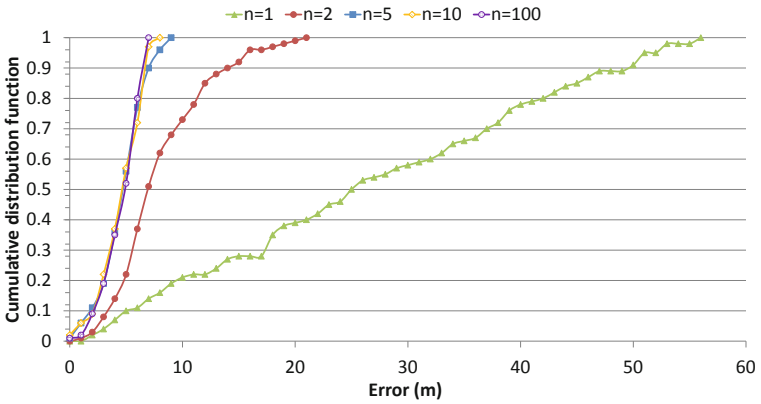

Fig. 7. CDF of the localization error for MIN-K for several node densities

and smaller than 5 meters for the highest density. It should be noted that this algorithm – similarly to other algorithms based on geometric considerations – achieves good accuracy only when network density is sufficiently high. Moreover, DCL, in its current form, can be applied only to one-jammer scenarios. Not only our framework is able to deal with multiple jammers, but it is also less sensitive to network density, as we show next.

We first considered the same network scenario used in [3], and run both MIN-K and MULT-UPD. Due to the high node density of this scenario and the resulting values of $\Pr(attacker(p))$, our framework – using a 1-meter resolution – was always able to find the correct position of the jammer in the first deployment cycle. To evaluate our framework's performance for jamming attacks in more general scenarios, we considered a $100m \times 100m$ field, with the jammer placed at the center of the field, and deployed networks having different densities. Specifically, we deployed 1, 2, 5, 10, and 100 nodes respectively, with a 71 meter transmission range (the whole field is jammed), and evaluated the accuracy for MIN-K over 1,000 independent runs. Fig. 7 shows the cumulative distribution function of the error. When considering a single node in the jammed area, the maximum error is high. As network density slightly increases, performance dramatically increases: it takes only a few nodes in the jammed area to bound the error within 10% of the transmission range. At this point, further increasing node density has a very small impact on localization error.