Protecting Location Privacy in Sensor Networks against a Global Eavesdropper

Kiran Mehta, Donggang Liu, *Member*, *IEEE*, and Matthew Wright, *Member*, *IEEE*

Abstract—While many protocols for sensor network security provide confidentiality for the content of messages, contextual information usually remains exposed. Such contextual information can be exploited by an adversary to derive sensitive information such as the locations of monitored objects and data sinks in the field. Attacks on these components can significantly undermine any network application. Existing techniques defend the leakage of location information from a limited adversary who can only observe network traffic in a small region. However, a stronger adversary, the *global eavesdropper*, is realistic and can defeat these existing techniques. This paper first formalizes the location privacy issues in sensor networks under this strong adversary model and computes a lower bound on the communication overhead needed for achieving a given level of location privacy. The paper then proposes two techniques to provide location privacy to monitored objects (source-location privacy)—*periodic collection* and *source simulation*—and two techniques to provide location privacy to data sinks (sink-location privacy)—*sink simulation* and *backbone flooding*. These techniques provide trade-offs between privacy, communication cost, and latency. Through analysis and simulation, we demonstrate that the proposed techniques are efficient and effective for source and sink-location privacy in sensor networks.

Index Terms—Sensor networks, location privacy.

1 Introduction

A wireless sensor network (WSN) typically consists of a large number of small, multifunctional, and resource-constrained sensors that are self-organized as an ad hoc network to monitor the physical world [1]. Sensor networks are often used in applications where it is difficult or infeasible to set up wired networks. Examples include wildlife habitat monitoring, security and military surveillance, and target tracking.

For applications like military surveillance, adversaries have strong incentives to eavesdrop on network traffic to obtain valuable intelligence. Abuse of such information can cause monetary losses or endanger human lives. To protect such information, researchers in sensor network security have focused considerable effort on finding ways to provide classic security services such as confidentiality, authentication, integrity, and availability. Though these are critical security requirements, they are insufficient in many applications. The communication patterns of sensors can, by themselves, reveal a great deal of *contextual information*, which can disclose the location information of critical components in a sensor network. For example, in the Panda-Hunter scenario [15], a sensor network is deployed to track endangered giant pandas in a bamboo forest. Each panda has an electronic tag that emits a signal that can be detected by the sensors in the network. A sensor that detects this signal, the *source* sensor, then sends the location of pandas to a data sink (destination) with help of intermediate sensors. An adversary (the hunter) may use the communication between sensors and the data sinks to locate and then capture the monitored pandas. In general, any target-tracking sensor network is vulnerable to such attacks. As another example, in military applications, the enemy can observe the communications and locate all data sinks (e.g., base stations) in the field. Disclosing the locations of the sinks during their communication with sensors may allow the enemy to precisely launch attacks against them and thereby disable the network.

Location privacy is, thus, very important, especially in hostile environments. Failure to protect such information can completely subvert the intended purposes of sensor network applications. Location privacy measures, thus, need to be developed to prevent the adversary from determining the physical locations of *source sensors* and *sinks*. Due to the limited energy lifetime of battery-powered sensor nodes, these methods have to be energy efficient. Since communication in sensor networks is much more expensive than computation [23], we use communication cost to measure the energy consumption of our protocols.

Providing location privacy in a sensor network is challenging. First, an adversary can easily intercept network traffic due to the use of a broadcast medium for routing packets. He can use information like packet transmission time and frequency to perform traffic analysis and infer the locations of monitored objects and data sinks. Second, sensors usually have limited processing speed and energy supplies. It is very expensive to apply traditional anonymous communication techniques for hiding the communication between sensor nodes and sinks. We need to find alternative means to provide location privacy that accounts for the resource limitations of sensor nodes.

Recently, a number of privacy-preserving routing techniques have been developed for sensor networks. However, most of them are designed to protect against an adversary

• *The authors are with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76019. E-mail: kiranmehta1981@yahoo.com, dliu@uta.edu, mwright@cse.uta.edu.*

Manuscript received 5 Apr. 2010; revised 16 Aug. 2010; accepted 21 Dec. 2010; published online 9 Feb. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-04-0158. Digital Object Identifier no. 10.1109/TMC.2011.32.

only capable of eavesdropping on a limited portion of the network at a time. A highly motivated adversary can easily eavesdrop on the entire network and defeat these schemes. For example, the adversary could deploy his own set of sensor nodes to monitor the communications in the target network [17]. This is especially true in a military or industrial spying context, where the adversary has strong, potentially life-or-death, incentives to gain as much information as possible from observing the traffic in the target network. Given a global view of the network traffic, the adversary can easily infer the locations of monitored objects and sinks. For example, a region in the network with high activity should be close to a sink, while a region where the packets originate should be close to a monitored object.

In this paper, we focus on privacy-preserving communication methods in the presence of a *global eavesdropper* who has a complete view of the network traffic. The contributions in this paper are twofold.

- We point out that the assumption of a global eavesdropper who can monitor the entire network traffic is often realistic for highly motivated adversaries. We then formalize the location privacy issues under such an assumption and apply an analysis based on Steiner trees to estimate the minimum communication cost required to achieve a given level of privacy.
- We provide the first formal study of how to quantitatively measure location privacy in sensor networks. We then apply the results of this study to evaluate our proposed techniques for location privacy in sensor networks. These include two techniques that hide the locations of monitored objects—*periodic collection* and *source simulation*—and two techniques that provide location privacy to data sinks—*sink simulation* and *backbone flooding*. Our analysis and simulation studies show that these approaches are effective and efficient.

The rest of the paper is organized as follows: Section 2 reviews existing approaches for providing location privacy in sensor networks. Section 3 presents our network and adversary models. In Section 4, we formalize the location privacy issues in sensor networks and provide a model for evaluating the location privacy. Section 5 discusses the proposed techniques for source and sink-location privacy. Section 6 evaluates the proposed techniques via simulation study. Finally, Section 7 concludes this paper and points out some directions for future research.

2 Existing Approaches

Location privacy has been an active area of research in recent years. In location-based services, a user may want to retrieve location-based data without revealing her location. Techniques such as k-anonymity [2] and private information retrieval [10] have been developed for this purpose. In pervasive computing, users' location privacy can be compromised by observing the wireless signals from user devices [24], [27]. Random delay and dummy traffic have been suggested to mitigate these problems. Location privacy in sensor networks also falls under the general framework of location privacy. The adversary monitors the wireless transmissions to infer locations of critical infrastructure. However, there are some challenges unique to sensor networks. First, sensor nodes are usually battery powered, which limits their functional lifetime. Second, a sensor network is often significantly larger than the network in smart home or assisted living applications.

Source-location privacy. Prior work in protecting the location of monitored objects sought to increase the *safety period*, i.e., the number of messages sent by the source before the object is located by the attacker [15]. The *flooding technique* [20] has the source node send each packet through numerous paths to a sink, making it difficult for an adversary to trace the source. *Fake packet generation* [15] creates fake sources whenever a sender notifies the sink that it has real data to send. The fake senders are away from the real source and approximately at the same distance from the sink as the real sender. *Phantom single-path routing* [15] achieves location privacy by making every packet walk along a random path before being delivered to the sink. *Cyclic entrapment* [19] creates looping paths at various places in the network to fool the adversary into following these loops repeatedly and thereby increase the safety period. However, all these techniques assume a local eavesdropper who is only capable of eavesdropping on a small region. A global eavesdropper can easily defeat these schemes by locating the first node initiating the communication with the base station.

Recently, several techniques have been proposed to deal with global eavesdroppers. Yang et al. propose to use proxies to shape the network traffic such that global eavesdroppers cannot infer the locations of monitored objects [29]. Shao et al. propose to reduce the latency of real events without reducing the location privacy under a global eavesdropper [26]. This technique ensures that the adversary cannot determine the real traffic from statistical analysis.

Sink-location privacy. In [6], Deng et al. described a technique to protect the locations of sinks from a local eavesdropper by hashing the ID field in the packet header. In [8], it was shown that an adversary can track sinks by carrying out *time correlation* and *rate monitoring* attacks. To mitigate these two kinds of attacks, Deng et al. introduced a *multiple-parent routing* scheme, a *controlled random walk* scheme, a *random fake path* scheme, and a *hot spots* scheme [8]. In [13], redundant hops and fake packets are added to provide privacy when data are sent to the sink. However, these techniques all assume that the adversary is a local eavesdropper. A global eavesdropper can easily defeat these schemes. For example, the global eavesdropper only needs to identify the region exhibiting a high number of transmissions to locate the sink. We, thus, focus on privacy-preserving techniques designed to defend against a global eavesdropper.

3 Network and Adversary Model

Sensor networks are a relatively recent innovation. There are a number of different types of sensor nodes that have been and continue to be developed [12]. These range from very small, inexpensive, and resource-poor sensors such as SmartDust up to PDA-equivalent sensors with ample power and processing capabilities such as Stargate. Applications

for networks of these devices include many forms of monitoring, such as environmental and structural monitoring or military and security surveillance.

In this paper, we consider a *homogeneous network model*. In the homogeneous network model, all sensors have roughly the same computing capabilities, power sources, and expected lifetimes. This is a common network architecture for many applications today and will likely continue to be popular moving forward. It is well studied and provides relatively straightforward analysis in research as well as simple deployment and maintenance in the field.

Although our research can be applied to a variety of sensor platforms, most sensors run off battery power, especially in the kinds of potentially hostile environments that we are studying. Given this, each sensor has a limited lifespan and the network must be designed to preserve the sensors' power reserves. It has been demonstrated that sensors use far more battery power transmitting and receiving wireless communications than any other type of operation [23]. Thus, we focus our evaluation on the amount of communication overhead incurred by our protocols.

For the kinds of sensor networks that we envision, we expect highly motivated and well-funded attackers whose objective is to learn sensitive information such as the locations of monitored objects and sinks.

The objects monitored by the network can be critical. Such objects could be soldiers, vehicles, or robots in a combat zone, security guards in a protected facility, or endangered animals in the wild. If the locations of these objects were known to an adversary, the endangered animals could be captured for profit, security guards could be evaded to enable theft of valuable property, and military targets could be captured or killed. Sinks are also critical components of sensor networks. In most applications, sinks act as gateways between the multihop network of sensor nodes and the wired network or a repository where the sensed information is analyzed. Unlike the failure of a subset of the sensors, the failure of a sink can create permanent damage to sensor network applications. Compromise of a sink will allow an adversary to access and manipulate all the information gathered by the sensor network, because in most applications, data are not encrypted after they reach a sink. In some military applications, an adversary could locate sinks and make the sensor network nonfunctional by destroying them. Thus, it is often critical to the mission of the sensor network to protect the location information of monitored objects as well as data sinks.

In this paper, we consider *global eavesdroppers*. For a motivated attacker, eavesdropping on the entire network is a fast and effective way to locate monitored objects and sinks. There are two realistic options for the attacker to achieve this. The first option is to deploy his own snooping sensor network to snoop on the target network. Note that, at the current price for a BlueRadios SMT Module at $25, the attacker needs only $25,000 to build a network of 1,000 nodes [3]. Thus, for even moderately valuable location information, this can be worth the cost and trouble. Another option is to deploy a few powerful nodes to snoop on the network. However, due to the short radio ranges of typical sensor platforms, the snooping nodes still need to be deployed densely enough to sense the radio signals from all sensor nodes. Thus, in practice, we may not be able to reduce the number of snooping nodes significantly by using powerful devices. Overall, we consider the first option as more practical for the adversary.

It is certainly possible that an adversary deploys sensors to directly sense the objects of his interest, instead of collecting and analyzing the traffic in the original network. However, directly recognizing an object is a very challenging problem in practice due to the difficulty of distinguishing the physical features of the objects from background noises. For example, recognizing a panda is much harder than detecting a packet and estimating some physical features (e.g., RSSI) from this packet. In most scenarios, such sensing problem is simply avoided by installing a small device (e.g., a sensor node) on each object; these small devices emit signals from time to time so that we can sense them accurately. Thus, locating objects by monitoring the traffic in the original network becomes much more attractive to the adversary. We consider our defense a success if the adversary is forced to launch the direct sensing attack.

Although such an eavesdropping sensor network would face some system issues in being able to report the precise timing and location of each target network event, we do not believe that these would keep the attackers from learning more approximate data values. This kind of attacker would be able to query his own network to determine the locations of observed communications. He could have appropriate sensors which send signals that could then be physically located. He could equip his sensors with GPS to get locations or use localization algorithms to avoid the cost of GPS [25], [18]. We do not assume that the adversary has to precisely locate each node in the target network. In most cases, a rough idea about where the critical events occurred would be sufficient for the adversary.

It should, thus, be feasible to monitor the communication patterns and locations of events in a sensor network via global eavesdropping. An attacker with this capability poses a significant threat to location privacy in these networks. We, therefore, focus our attention to this type of attacker.

4 Privacy Evaluation Model

In this section, we describe a model for evaluating the location privacy of critical components in sensor networks. In this model, the adversary deploys a *snooping network* to monitor the wireless transmission activity in the target network. We consider a scenario in which an adversary can monitor all transmissions of the sensors in the network. In practice, the adversary does not need to know exactly where a packet is sent or the exact location of the sensor node that sends the packet. A rough estimate of the location will be good enough for the attacker to conduct traffic analysis. However, in this section, we assume the worst case scenario: for every observed packet, the adversary knows where it is sent or which sensor node sends the packet. This indicates that each sensor i is an *observation point*, and a tuple (i, t, e) is available to the adversary by observing each packet e send by node i at time t. We assume that all

transmissions are encrypted, and hence, the actual useful information available to the adversary is (i, t). We assume that the network begins operations at time $t = 0$.

The attacker's objective is to locate the source or the sink by snooping on the wireless transmissions. The main observation used by the global adversary is: *there must be a sequence of spatial-temporal correlated packets involved in each communication from the source to the sink*. As long as the adversary knows the routing protocol, he can easily identify all these sequences from the traffic and determine the set of possible sources and sinks. Intuitively, the defender has to create dummy sequences in the network to confuse the attacker; such dummy sequences usually require the addition of dummy traffic into the network, leading to more communication overhead. Clearly, there is a trade-off between the location privacy and the communication overhead. In this section, we develop a theoretical study of this trade-off.

4.1 Assumptions

In some sensor network applications, sensor nodes may not be easily accessible to the adversary. In other applications, the adversary may have physical access to sensor nodes, which makes such applications vulnerable to node compromise. Having a set of compromised sensors in the network will provide an advantage to the adversary. In this paper, however, we assume that an adversary does not compromise sensor nodes. We will seek solutions to the problem of providing location privacy despite nodes being compromised in future work. We assume that we can protect the monitored objects if we can prevent the leakage of the locations of source sensors. *We use the terms objects and sources interchangeably in this paper*.

In this paper, we investigate both source-location privacy and sink-location privacy. When we study source-location privacy, we assume that the sink-locations are known to the adversary via eavesdropping on the network. Similarly, when we study sink-location privacy, we assume that the source locations are known to the adversary. This is especially true when an adversary tries to locate sinks by triggering event detection in the network and then monitoring the ensuing communication pattern. We call the set of components (e.g., the source nodes in source-location privacy) whose location information needs to be protected the `protected set`, and we call the set of components (e.g., the source nodes in sink-location privacy) whose location information is available to the adversary the `available set`. We assume that all sinks and sensors are stationary, but monitored objects can be mobile. We concentrate on sinks receiving data only. We plan to study how to provide location privacy for broadcasting nodes in future work.

4.2 The Attacker

We now describe our privacy model in detail. We will first describe a privacy model for source-location privacy and then extend it to include sink-location privacy. Table 1 lists some notations frequently used in this paper; their specific meanings will be further explained in our discussion below.

A sensor network deployed for an application can be viewed as a graph $G = \{V, E\}$ where the set of vertices V is the union of the set I of sensor nodes and the set of sinks. A small subset of the sensors will be the source nodes. The set E of edges includes all direct communication links between sensor nodes physically close to each other. At any point in time, from the global eavesdropper's point of view, the network can be considered to include a set S_P (i.e., the protected sources), a set S_A (i.e., the sinks where the data is sent), and a set of sensors that transfer data between sources and sinks.

TABLE 1
Frequently Used Notations in the Paper

b	Level of privacy in terms of bits
T	Time period that the network has been operating
S_P	The `protected set`
S_A	The `available set`
S_T	Set of sensors in whose range the adversary expects to find protected components at time T
I	Set of all sensor IDs in the network
N	Number of sensor nodes in the network
n_b	Average number of neighbors of each sensor
L	Number of fake sinks selected
P	Probability of a fake object behaving like a real one

The adversary eavesdrops on the entire network with the intention of physically locating objects. We model each *observation* of the adversary as the tuple (i, t), representing the fact that a packet has been emitted by a node i and observed by the adversary at time t. Let $O_{i,T}$ be the set of all observations collected by the adversary about node i by time T. Thus, at time T, the knowledge that an attacker can obtain from eavesdropping on the entire target network is

$$O_T = \bigcup_{i \in I} O_{i,T}.$$

The objective of the adversary is to identify a set $S_T \subset I$ of possible sources, i.e., sensor nodes in whose range the adversary expects to find objects at time T. Informally, the attacker will not believe that a lone observation (i, t) indicates the presence of an object. The presence of an object should generate a *trace*, which is a set of observations over the lifetime of the network up to time T. This means that there will be a communication path between each possible source and at least one sink. More precisely, for each source $i \in S_T$, there must exist a subset of sinks $K \subseteq S_A$ and a set of observations $A_{i,K} \subset O_T$ that could be exactly generated due to the communication from node i (based on observing an object) to the sinks in K. We call each such set of observations a *candidate trace*. In other words, a candidate trace is any subset of the attacker's observations that could be the result of a source sensor reporting to the base station.

For the attacker to check whether a set of observations is a candidate trace, we define a *pattern analysis* function

$$f : 2^{\ddot{O}_T} \rightarrow I \cup \{\perp\},$$

where $\ddot{O}_T$ is the set of all possible observations, i.e., $\ddot{O}_T = \{(i, t)\}_{i \in I, 0 \leq t \leq T}$. The *pattern analysis* function outputs the ID of the possible source sensor, if the set of observations is a candidate trace, and returns $\perp$ otherwise.

For simplicity, we assume that the pattern analysis does not return fractional values, e.g., a probabilistic measure of the chance that a trace is a candidate trace or not. We say that a pattern analysis function is *perfect* if it can identify all candidate traces without error, i.e., without false positives or false negatives. In this paper, we consider a strong adversary who uses a perfect pattern analysis function. Let f_p be such a perfect pattern analysis function. We have

$$S_T = \{i \mid \exists A_{i,K} \subseteq O_T, K \subseteq S_A, (i = f_p(A_{i,K})) \neq \perp\}.$$

4.3 Measuring Privacy

An intuitive way of measuring location privacy is to evaluate the attacker's accuracy in locating sources. Note that the adversary will need to identify the areas in which the objects of his interest can be found. We assume that the attacker knows the routing protocol and does not miss any real sources. In other words, the real sources can always be found in the sensing range of the possible source sensors identified by the adversary (S_T). We could measure the total area covered by these sensors' sensing range as a metric of how much area the attacker would need to search to find the sources. However, since all sensors have the same sensing radius in our model, we simplify this by just taking the size of the set S_T. Intuitively, the larger the size of S_T, the more uncertainty the adversary will have about the locations of real sources. We assume that the sensors in S_T are equally likely to be source sensors. The probability of any sensor node in S_T being a source sensor can, thus, be estimated by $\frac{|S_P|}{|S_T|}$. Hence, we formally define the location privacy of our system as

$$b = \sum_{|S_T|} -\frac{1}{|S_T|} \log_2 \frac{|S_P|}{|S_T|} = \log_2 \frac{|S_T|}{|S_P|}.$$

We can use this notion to define the optimal privacy. Let $\ddot{S}_T$ represent the set of all possible locations for the real object at time T based on the set of all possible observations $\ddot{O}_T$, i.e.,

$$\ddot{S}_T = \{i \mid K \subseteq S_A, A_{i,K} \subseteq \ddot{O}_T, (i = f_p(A_{i,K})) \neq \perp\}.$$

For simplicity, we always assume that an object can be anywhere in the deployment field at time T. We, thus, have $|\ddot{S}_T| = N$. We then define set $S_T^* = I$ for optimal privacy, representing the case that $\forall i \in I$, there exist $K \subseteq S_A$ and $A_{i,K} \subseteq O_T$ such that $(i = f_p(A_{i,K})) \neq \perp$. In other words, there is a subset of observations in O_T (the set of observations made by the adversary by time T) that supports the case for an object in the range of any sensor i at time T. We, thus, have the optimal location privacy as

$$b \leq \log_2 \frac{|S_T^*|}{|S_P|} = \log_2 \frac{N}{|S_P|}.$$

The level of location privacy is measured in terms of bits of information. Depending on the users and applications, this can be easily modified to support different kinds of privacy measurement models. For example, we can define high, medium, and low privacy levels by using appropriate values of b.

We note that the traffic in the network can cause the level of privacy to vary. The privacy would go lower if the attacker ascertains that a particular trace is no longer a candidate trace. If a candidate trace splits into two candidate traces, then the level of privacy goes up because S_T grows. The interpretation of this depends on the sensor network application and the attacker model considered. For example, if the attacker seeks to physically destroy the object being observed with a missile (instantaneous attack), then the privacy should be taken as the minimum at any time before T. In cases where the attacker must spend time to investigate the candidate locations, then the average privacy over time is adequate. We provide a snapshot of the privacy at any given time, which can be used for either purpose.

4.4 Privacy and Communication Cost

In the following, we explore the relationship between the level of privacy and the amount of communication overhead. To minimize communication overhead, we should minimize the communication required to produce all the candidate traces in the network.

We now simplify our model to understand the effect of different policies on the overheads and location privacy in the network. We model communication in sensor networks as a discrete time system with a granularity of Δ. In particular, the time line is divided into discrete time intervals with equal length of Δ. The communication between sensor nodes happens at the end of each time interval, i.e., at time $\{\Delta, 2\Delta \ldots, i \times \Delta, \ldots\}$. A sensor node can receive all the packets targeted to itself and will send or forward no more than one packet in any time interval. When a sensor node receives multiple dummy packets during a given time interval, it only needs to forward one of them to save communication costs. Intuitively, the larger the value of Δ, the greater the communication cost we can save. Assume that on average an object in the network will trigger one event every α time intervals. There will be an *event reporting round* between the corresponding source and sinks every α intervals.

Next, we estimate a lower bound on the communication cost needed to achieve a certain level of location privacy. Note that, for each source sensor, there will be a sequence of packets from it to all sinks it is communicating with during an event reporting round. As a result, we will have a set of routing paths that connect all nodes in S_T and all communicating sinks together. Every node in the routing paths will transmit at least one packet. Thus, the minimum communication cost in each event reporting round can be achieved by constructing a graph of shortest length that connects all source sensors and all communicating sinks, where the length is the sum of the lengths of all edges and the length of an edge is the hop distance between two nodes. The problem of constructing such a graph is actually the well-known Steiner tree problem [4].

The Steiner tree problem has been studied extensively in the past; it is similar to the *minimum spanning tree problem*, i.e., finding a network of shortest length to connect a given set of vertices, where the length (or *weight*) is the sum of the lengths of all edges. The main difference is that, in the Steiner tree problem, additional intermediate vertices, called *Steiner points*, may be added into the network to reduce the weight of the spanning tree. For example, given three vertices that form a triangle, the cost of connecting them can be reduced by adding a Steiner point at the

centroid of the triangle and connecting the three vertices through this centroid. The Steiner tree problem is an NP-hard problem and many approximation algorithms have been proposed to construct near-optimal trees. In this paper, we use the simple approximation algorithm in [28] for our purpose. Since sensor networks are usually dense enough, we assume that any location where a Steiner point could possibly be added has a sensor node in place. Let K_{davg} denote the average number of real sinks a source communicates with. We have the following theorem.

Theorem 4.1. *To achieve b bits of source-location privacy, the communication cost is at least*

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha\times\Delta}} M_s(i),$$

where $M_s(i)$ is the weight of a Steiner tree connecting the given $2^b \times |S_P| + K_{davg}$ nodes in the network during the ith round of event reporting.

Proof. Let $S_T = \{f_1, \ldots, f_l\}$ be the candidate set identified by the adversary. Since $b = \log_2 \frac{l}{|S_P|}$, we have $l = 2^b \times |S_P|$. We need to minimize the communication cost required for these l candidate traces in the network. In other words, we need to find a Steiner tree that connects these given $2^b \times |S_P|$ sensors and K_{davg} sinks. The weight of this Steiner tree would give us the minimum communication cost for sending packets from sources to sinks. Packets transmitted by sources would travel to sinks via edges in the tree. If a node in the tree receives multiple fake packets in one time interval, it can drop as many as possible to save energy. Thus, the communication cost needed to achieve b bits of source-location privacy is at least

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha\times\Delta}} M_s(i).$$

□

We have mentioned details about communication cost for protecting the locations of monitored objects. A similar argument can be made for finding the communication cost for protecting the locations of sinks in a network. The set of available locations S_A would be the set of source sensors, the set of components to be protected S_P would be the sinks, and S_T would be the set of sensors identified by the adversary. The objective of the adversary would be to find the sinks instead of the objects. As locations of all objects in the network are known to the adversary, fake sources are of no use. In addition, generating dummy packets will be of little use for sink-location privacy since communication only happens when there are real monitored events, which we assume can be detected by adversaries as well. Since an intermediate node should not drop packets received from different sources, we can consider traffic from each object separately from others.

Since the adversary is aware of the locations of all objects in the network, each source has to communicate with at least 2^b fake sinks in addition to each real one to achieve b bits of privacy. Consider the last sensor node in the communication path from the source to the sink. Any sensor node that is within the signal range of this node can be used as a fake sink since it can also receive any packet

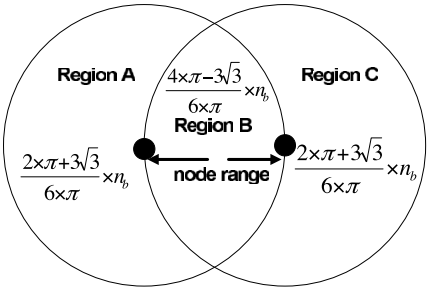

Fig. 1. The overlap of signal ranges.

that the real sink receives. Thus, to minimize communication costs, we can create 2^b fake sinks around each real sink. Essentially, we *locally flood* every event report around the real sink such that at least 2^b nodes can hear the report. The minimum cost of this local flooding technique can be estimated as follows: let n_b be the average number of neighbors for a sensor. As shown in Fig. 1, based on simple geometry rules, we can calculate the minimum size of the intersection between the radio range of two neighboring nodes as $\frac{4\times\pi-3\sqrt{3}}{6} \times r^2$. All sensor nodes in this area can hear rebroadcasted messages from both nodes. As a result, the maximum number of new sensor nodes we can reach by rebroadcasting an event packet is the number of nodes outside of the intersection, i.e., $\frac{2\pi+3\sqrt{3}}{6\pi} \times n_b$. Hence, the minimum number of rebroadcasting we need to reach 2^b sensors can be estimated by

$$\gamma = \left\lfloor \left(\frac{2^b - \frac{4\times\pi-3\sqrt{3}}{6\times\pi} \times n_b - 1}{\frac{(2\times\pi+3\sqrt{3})}{(6\times\pi)} \times n_b} \right) \right\rfloor + 1.$$

At least γ sensors would need to rebroadcast a given event so that at least 2^b sensors and a real sink can receive it. The adversary would then believe that this sink could be any of these 2^b sensors. For the sake of presentation, we call these γ sensors the *local flooding set*. We now present the following theorem:

Theorem 4.2. *To achieve b bits of sink-location privacy, the communication cost is at least*

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha\times\Delta}} M_d(i) \times |S_A|,$$

where $M_d(i)$ is the weight of a Steiner tree connecting a source and K_{davg} local flooding sets in the ith round of event reporting.

Proof. Considering a graph that represents our network and the knowledge that a source communicates with K_{avg} real sinks on average, we need to find a Steiner tree that connects a source and all the nodes in those K_{davg} local flooding sets. As weight of the tree represents the minimum number of transmissions required to send an event report from a source to the sinks, for $|S_A|$ objects we get

$$\omega_T = \sum_{i=1}^{\frac{T}{\alpha\times\Delta}} M_d(i) \times |S_A|.$$

□

Though constructing the actual Steiner tree is an NP-hard problem [4], it gives us a benchmark to compare the efficiency of source and sink-location privacy techniques. We calculate the average weight of a Steiner tree by using a simple approximation algorithm proposed in [28] whose cost ratio is not more than twice the optimal. This algorithm starts from a graph $G = \{V, E\}$ and an input set $S \subseteq V$ of nodes to be connected. A node is randomly selected from S and connected to the nearest other node in S via the shortest path. This forms a tree containing at least two nodes from S. The next node from S is selected such that it is not part of the tree constructed so far but is the closest one to the tree. This process continues until all the nodes in S are in the tree. In each step to connect a new node to the existing tree, we use Dijkstra's single source shortest path algorithm, which calculates the shortest distance from the new node to the existing tree.

5 Privacy-Preserving Routing

In this section, we present the proposed privacy-preserving techniques for protecting the location information of monitored objects and data sinks. We assume that all communications between sensor nodes in the network are encrypted so that the contents of packets appear random to the global eavesdropper. Many key predistribution protocols can be used for this purpose [9], [5], [16].

5.1 Source-Location Privacy Techniques

In this section, we present two techniques to provide location privacy to monitored objects in sensor networks, *periodic collection* and *source simulation*. The periodic collection method achieves the optimal privacy but can only be applied to applications that collect data at a low rate and do not have strict requirements on the data delivery latency. The source simulation method provides practical trade-offs between privacy, communication overhead, and latency.

5.1.1 Periodic Collection

As described in Section 2, previous schemes fail against a global eavesdropper. The primary reason is that the presence of a real object will change the traffic pattern at the place where the object resides. This allows the global eavesdropper to easily find out where the change happens. An intuitive solution is to make the traffic pattern independent of the presence of real objects. To achieve this, we have every sensor node independently and periodically send packets at a reasonable frequency regardless of whether there are real data to send or not.

Specifically, each sensor node has a timer that triggers an event every Δ seconds, as well as a first-in-first-out (FIFO) queue of size q for buffering received packets that carry real data reports. When the timer fires, the node checks if it has any packets in its queue. If so, it dequeues the first packet, encrypts it with the pairwise key it shares with the next hop, and forwards it to that node. Otherwise, it sends a *dummy packet* with a random payload that will not correctly authenticate at the next hop. Since every sensor node only accepts the packets that correctly authenticate, dummy packets do not enter the receiver's queue. When the queue at a sensor node is full, it will stop accepting new packets.

Privacy. The periodic collection method provides the optimal location privacy one can ever achieve in the network since the traffic pattern is entirely independent of the activity of real objects. Specifically, since the object can be anywhere in the field at time T, we know that for any $i \in I$, there exists a candidate trace $A_{i,K} \subset \ddot{O}_T$ with $f_p(A_{i,K}) = i$ for $K \subseteq S_A$. This indicates that $S_T = I$. Hence, we have

$$b = \log_2 \frac{|S_T|}{|S_P|} = \log_2 \frac{N}{|S_P|}.$$

Energy consumption. It is generally believed that communication in sensor networks is much more expensive than computation [23]. For a privacy-preserving routing technique, its energy consumption can, thus, be measured by the additional communication used for hiding the traffic carrying real data.

Since the network starts operation at time 0, the total number of data packets transmitted in the network can be estimated by $(T \times N)/\Delta$. Certainly, a small Δ indicates a large amount of additional traffic for our periodic collection method. This means that this method cannot handle real-time applications very well. However, we do believe that this method is practical for applications where Δ can be set large enough for a reasonable amount of covering traffic.

We now estimate the minimum communication overhead needed to achieve the optimal privacy using Theorem 4.1 discussed in Section 4. The problem is to find the Steiner tree that connects all N nodes with the sinks they communicate with. In this case, the Steiner tree problem is reduced to finding the weight of a minimum spanning tree of the graph [4]. The weight of a minimum spanning tree for a graph of N nodes and one or more sinks is at least N. Thus, the minimum communication cost by the end of time T would be $\omega_T = (T \times N)/(\alpha \times \Delta)$.

Assume that Δ is set properly such that an object will trigger an event to the sink for each interval Δ, i.e., $\alpha = 1$. We can see that the performance of the periodic collection method is close to the optimal solution in terms of the communication overhead needed to achieve the optimal privacy.

Latency. Sensor networks can support a wide range of applications. Different applications have different requirements that may affect the usage of the periodic collection method in real-world scenarios. Example of these requirements include *the latency of a real event being reported to the sink* and *the network lifetime*.

There is a trade-off between energy consumption and latency depending on the value of Δ. Since the setting of Δ determines the amount of wireless communication and corresponding energy consumption in the network, it also determines how long the sensors' batteries will last. In effect, it determines the lifetime of the network. Therefore, Δ should, therefore, be set as large as possible. On the other hand, it needs to be set low enough to allow the network to meet the latency requirements of the application.

The queue size q is the number of real packets that a sensor node can buffer. This will affect how well the periodic collection method can handle situations in which real events are frequently observed and reported by the sensors. Increasing the value of q will allow the queuing of more real packets and, thus, will help the network in

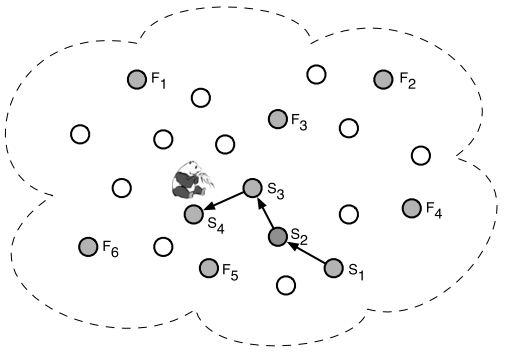

Fig. 2. Movement pattern leaks location information.

forwarding more information about real objects to the sink. In other words, the number of packets dropped in transit can be reduced. However, a large value of q may increase the average latency of a real packet reaching the sink. This occurs because a newly received packet that carries real data may need to wait for a long time before getting forwarded in the case of a large queue. We give a detailed simulation study of the effects of different values for parameter q in Section 6.

5.1.2 Source Simulation

Though the periodic collection method provides optimal location privacy, it consumes a substantial amount of energy for applications that have strict latency requirements. It is clearly not well suited for real-time applications.

In the periodic collection method, every sensor node is a potential source node. To reduce energy consumption, we choose to reduce the number of potential sources in the network. In other words, we will trade off privacy with communication overhead. For this purpose, we propose to *create multiple candidate traces in the network to hide the traffic generated by real objects*. How to determine the number of candidate traces is application dependent. In general, we expect that this number is much smaller than the size of I.

Creating candidate traces in the field is quite challenging in practice. The main problem lies in the difficulty of modeling the behavior of a real object in the field (e.g., the behavior of a panda). A poorly designed model will likely fail in providing privacy protection. For example, as shown in Fig. 2, the behavior of fake objects is modeled *inaccurately* as remaining in one location all the time. Based on this model, the candidate traces are created at locations $\{F_1, F_2, \ldots, F_6\}$. Sensors at each of these locations will send fake traffic to the sink, simulating a real object. However, the adversary can simply notice that the object moves around in the field along the path $\{S_1, S_2, S_3, S_4\}$ and use this extra knowledge to distinguish real objects from fake ones. In general, the attacker may be able to distinguish the movement patterns of real objects from fake ones, even if we have the fake ones move.

Fortunately, in most cases, an adversary will not have substantially more knowledge about the behavior of real objects than the defender. Even if the attacker learns about the object behavior over time, the defender can observe and learn the same behavior and can broadcast occasional updates to the object movement model. Thus, it is often reasonable to assume that the adversary and the defender have similar knowledge about the behavior of real objects. We can then create more useful candidate traces in the field to hide real objects. Though it is challenging to model real objects, research on learning and modeling behavior of objects is quite active. We believe that it will not be a very difficult problem to obtain a reasonable behavior model for the object in the field. Such modeling is beyond the scope of this paper.

Protocol description. In the source simulation approach, a set of *fake objects* will be simulated in the field. Each of them generates a traffic pattern similar to that of a real object to confuse the adversary.

Source simulation works as follows: before deployment, we randomly select a set L of sensor nodes and preload each of them with a different *token*. Every token has a unique ID. These tokens will be passed around between sensor nodes to simulate the behavior of real objects. For convenience, we call the node holding a token the *token node*. We also assume that the profile for the behavior of real objects is available for us to create candidate traces.

After deployment, every token node will emit a signal mimicking the signal used by real objects for event detection. This will trigger event detection in the local area and generate traffic as if a real event was detected. The token node will then determine who in its neighborhood (including itself) should run the next round of source simulation based on the behavior profile of real objects. The token will then be passed to the selected node. The delivery of the token between sensor nodes will always be protected by the pairwise key established between them.

Note that the simulation requests create additional messages that can help the attacker distinguish real objects from fake ones. We, thus, require that nodes that detect the real object also send an extra message each round. Alternatively, we can attach the requests to the messages to the sink, given that all messages would be received by neighbors due to the broadcast nature of the wireless medium.

Privacy. Assume that the defender can build a model for the behavior of real objects that can always create a useful candidate trace in the network with probability P. In other words, any candidate trace created by the defender in the network will be considered as a valid candidate trace by the attacker with probability P. Let $|S_P|$ be the number of real objects in the network. We can see that the set of candidate locations S_T includes an average of $|S_P| + |L| \times P$ node IDs. As a result, the privacy provided by the source simulation approach can be estimated by

$$b = \log_2 \frac{|S_P| + |L| \times P}{|S_P|} = \log_2 \left(1 + \frac{|L| \times P}{|S_P|}\right).$$

Since we assume that both the adversary and the defender have similar knowledge about the behavior of real objects, we will usually have $P \approx 1$. In this case, $b \approx \log_2(1 + \frac{|L|}{|S_P|})$.

Energy consumption. The source simulation method can be effectively used for protecting location privacy in applications with real-time data delivery requirements. In such applications, sensor nodes should deliver their sensing reports as fast as possible. We, thus, assume that a sensor node will immediately forward a report whenever the channel is free. Hence, Δ is very small and there is no

time to wait to queue multiple fake packets. Letting ϵ be the average number of packets required to send an event from a source to sinks that receive it, we get

$$\omega_T \approx \epsilon \times (|L| + |S_P|) \times \frac{T}{\alpha \times \Delta}.$$

This shows that the average communication overhead is approximately increased by a factor of $\frac{|L|+|S_P|}{|S_P|}$ to protect location privacy. This technique features optimal latencies.

5.2 Sink-Location Privacy Techniques

This section presents two privacy-preserving routing techniques for sink-location privacy in sensor networks: *sink simulation* and *backbone flooding*. The sink simulation method achieves location privacy by simulating sinks at specified locations, and the backbone flooding method provides location privacy by flooding the event reports in a backbone network that covers the data sinks. Both techniques provide trade-offs between privacy, communication cost, and latency. In this work, we focus on protection of passive sinks that only receive data from sensors. We will consider location privacy for sinks that broadcast packets in future work.

5.2.1 Sink Simulation

Similar to source simulation, an intuitive solution for sink-location privacy would be to confuse the adversary by creating virtual sinks. For this purpose, we propose to create multiple candidate traces toward the fake sinks in the network to hide the traffic between real objects and real sinks.

Protocol description. In sink simulation, *fake sinks* will be simulated in the field. Each of them will receive traffic similar to the traffic received by a real sink. To achieve this, we make no distinction between fake and real sinks when sensors send packets.

During deployment, we place real sinks and select locations where fake sinks are to be simulated. A subset of the sensors will be used as fake sinks. It is also required that each real sink have a fake sink simulated in its communication range. We will only send messages to the fake sinks and have all fake sinks perform a one-hop broadcast of the message, ensuring full concealment of the real sink locations.

Note that our privacy model in Section 4 does not consider the distance between the fake sinks in S_T. However, in practice, selecting fake sinks that are too close to each other may reduce the location privacy drastically. An attack on one of them can destroy others as well. For example, if an adversary needs to destroy sinks using missile, nearby sinks may be destroyed in one attack. Similarly, the adversary can physically check and locate nearby sinks with little additional effort. Thus, the locations of fake sinks should be made as far way from each other as possible.

Once fake sinks are selected after deployment, the sensors should have routing paths to send data to places where fake sinks are simulated. Many routing protocols can be used for this purpose [14], [22]. During network operations, whenever a source node senses an event, a report will be sent to all fake sinks. Whenever a fake sink receives a packet, it broadcasts it locally so that the adversary would believe that a real sink could be in its range.

Privacy. Let L be the set of fake sinks *other than those in the neighborhood of real sinks*. Let n_b be the average number of neighbors of a sensor. Since each fake sink broadcasts every received packet locally in its range, any node in the communication range of a fake sink can be a real sink. Assuming that fake sinks are simulated at a distance of at least two hops from each other, we can see that the set S_T of candidate nodes includes $(|S_P| + |L|) \times n_b$ node IDs on average. As a result, the location privacy provided by the sink simulation approach can be estimated by

$$b = \log_2 \frac{(|S_P| + |L|) \times n_b}{|S_P|} = \log_2 \left(1 + \frac{|L|}{|S_P|} \times n_b\right).$$

Energy consumption. Since there are $|L|$ places other than those in the neighborhood of real sinks where the fake sinks are present in the network, the communication overhead will be approximately increased by a factor of $\frac{|L|+|S_P|}{|S_P|}$. In the sink simulation approach, we select fake sinks randomly in the network. When we need to create a large number of fake sinks to meet high location privacy requirements, the sink simulation approach can be very expensive. The reason is that a lot of extra communication is wasted during the routing of packets to *randomly selected* fake sinks. To address this problem, we describe a *backbone flooding* technique in the next section.

5.2.2 Backbone Flooding

In *backbone flooding*, we send packets to a *connected* portion of the network, the *backbone*, instead of sending them directly to a few sinks. The packets are only flooded among the *backbone members*, the sensors that belong to this backbone. As long as the real sinks are located in the communication range of at least one backbone member, they can receive packets from any source in the field. Clearly, for a global eavesdropper, the sink could be anywhere near the backbone. We assume that the backbone is created soon after the network is deployed and that the adversary does not eavesdrop until the backbone is created.

The main component of backbone flooding is the construction of the backbone. Existing studies have focused on finding the minimal number of sensors that are needed to flood a packet so that the entire network can receive it [11], [21]. In our case, we need to flood the packets to cover an area large enough to achieve the desired level of location privacy.

Protocol description. In backbone flooding, we create a backbone consisting of $|L|$ members, such that each sink is within the range of at least one backbone member. Given $|L|$, the backbone formed should cover as large an area as possible for maximum location privacy. However, finding optimal solutions has been shown to be NP-hard [21], [11].

We present an approximation algorithm for this problem. When we say that a sensor v *covers* some other sensors, we mean that v is responsible for directly delivering packets to these sensors via local broadcast. The backbone formation will terminate when the backbone members cover the required number of sensors for the desired level of location privacy.

We first consider the backbone formation algorithm in the case where there is only one real sink in the network and then present a simple extension for the case of multiple sinks. The pseudocode of the algorithm is presented in

Algorithm 1. Let m be the *mincover*, which is the minimum number of uncovered neighbors that a sensor should cover to be eligible to become a backbone member. The inputs to the backbone formation algorithm are m, the mincover, and b, the amount of privacy required. The main operation of the algorithm is to identify sensors that have many neighbors to cover and add them to the backbone until the privacy requirement is met.

Algorithm 1. Backbone Construction
Require: Each node has list of its neighbors

```
1: procedure BACKBONE(b, m)
2:    TotalCoverage ← 1                ▷ first sensor in the set L
3:    Id ← GetMyId()
4:    Leader ← −1
5:    LocalCoverage ← GetNeighborCnt()
6:    while true do
7:       if TotalCoverage ≥ 2^b then
8:          EXIT
9:       end if
10:      Msg ← GetNextMsgFromQueue()
11:      if MsgType = NewMemberSelection then
12:         if CheckNewMemberId(Msg) = Id then
13:            DestId ← GetDestId(Msg)          ▷ Identification of sink
14:            SendElectionMsg(Id, DestId)
15:            CollectVotes(Id, DestId)
16:            CollectCoverageInfo(Id, DestId)
17:            (ResultId, Coverage) ← Max_Id(m)
18:            if Valid(ResultId) = true then
19:               TotalCoverage ← TotalCoverage + Coverage
20:            else
21:               (ResultId, Coverage) ← Backtrack(Coverage, ResultId, m)
22:               if Valid(ResultId) = true then
23:                  TotalCoverage ← TotalCoverage + Coverage
24:               else
25:                  EXIT ▷ Cannot find more sensors to cover
26:               end if
27:            end if
28:            NotifyNewMember(ResultId, TotalCoverage)
29:         end if
30:      else if MsgType(Msg) = ElectionRequest then
31:         if Leader = −1 then
32:            SenderId ← GetSenderId(Msg)
33:            SendVote(SenderId)
34:            Leader = SenderId
35:         end if
36:      else if MsgType(Msg) = CoverageInfoRequest then
37:         SendCoverageInfo(LocalCoverage)
38:      else if MsgType(Msg) = AcceptMessage then
39:         LocalCoverage ← LocalCoverage − 1
40:      end if
41:   end while
42: end procedure
```

For a given sensor i, let $max_i(m)$ be a function that examines i's neighbors and outputs the ID of the neighbor that covers the maximum number of uncovered sensors. It also returns the number of sensors that the newly identified sensor covers. If this maximum number is less than m, the algorithm instead outputs $\perp$. The algorithm produces a connected backbone network at each step.

We assume that each sensor has the list of its neighbors. Let L be the set of IDs of the backbone members. L is initially empty. We begin by adding a sensor that has the real sink in its range. We now describe this algorithm from perspective of a sensor that sends an *election* message and then from the perspective of a sensor that receives this message.

Every new sensor v added to the set L will send an *election* message to find the number of uncovered sensors each neighbor can cover. If $max_v(m)$ outputs a valid sensor ID and the coverage of this node, the ID of this node will be added to L. The newly added sensor will then execute the same algorithm. If $max_v(m) = \perp$, v will collaborate with existing nearby backbone members to find a usable sensor using Algorithm 2. The backbone is a tree structure with backbone members as the tree nodes. During the collaboration, the sensor will need to get information from its parent to find a node that can cover at least m nodes. This collaboration could continue to next level of ancestors if such a node is not known to the immediate parent. This *backtracking process* continues until a sensor meeting the required constraints is found. If a sensor that can cover at least m uncovered sensors is unavailable, a sensor that covers the maximum number of uncovered sensors is used.

Algorithm 2. Backtrack Procedure

```
1: procedure BACKTRACK(Coverage, Id, m)
2:    ResultId ← Id
3:    Max ← Coverage
4:    LocalMaxId ← −1
5:    CollectCoverageInfo(GetMyId(), NULL)
6:    (LocalMaxId, Max) ← Max_Id(m)
7:    if Max ≥ m then
         return LocalMaxId, Max
8:    else if Max < Coverage then
9:       ResultId = LocalMaxId
10:      Max = Coverage
11:   end if
12:   for EachUnvisitedNeighborBKMember do
13:      (Id, Coverage) = Backtrack(Max, ResultId, m)
14:      if Coverage ≥ m then
15:         ResultId = Id
16:         Max = Coverage
17:         break
18:      else if Coverage > Max then
19:         Max = Coverage
20:         ResultId = Id
21:      end if
22:   end for
        return ResultId, Max
23: end procedure
```

The beginning and termination of the backtracking process depends on the value of m. A value of $m \leq 1$ would mean that backtracking would start only if v cannot find any

neighbor that can cover at least one uncovered sensor. If m has a value greater than the number of neighbors any sensor could have, then the backtracking process would ensure that the sensor that covers the maximum number of uncovered sensors is selected. Intuitively, this would mean that an increase of m would help in covering more sensors with the help of fewer backbone members. However, more energy will be consumed to form a backbone for a large m due to more backtracking steps.

If an uncovered sensor receives an election message, it will send an *accept* message to the sender. The sender then becomes the node's parent. When these accept messages are being successfully sent, all other sensors that can overhear these messages reduce their count of number of sensors they can cover by the number of unique accept messages heard. After this is done, each sensor sends to its parent the number of sensors it can cover in a *coverage* packet. Note that we start the backbone formation from one of the neighbors of the real sink. To prevent the sink from being located at the end points of the backbone, the algorithm should be executed twice from the same starting point. The input parameter for the backbone size to these two executions should add up to original input value. For example, if we need b bits of privacy, we randomly choose $\{b_1, b_2\}$ such that $b_1 + b_2 = b$. The backbone formation algorithm is then executed twice from the same neighbor of the sink, one with input b_1 and the other with input b_2.

Once backbone construction is complete, the source sensor just needs to communicate with the nearest member of the backbone. After a member receives the data, it will flood the data in the backbone. The real sink can always receive every packet. Based on the backbone creation algorithm, we can see that from the attacker's point of view, the real sink could be anywhere in the communication range of the backbone.

Let us now turn our attention to the case where there are multiple real sinks. To protect the location of multiple sinks, we simply generate a set of random numbers $b_1, b_2, b_2, \ldots, b_c$ such that $b = b_1 + b_2 + \cdots + b_c$ where each b_i is an input that corresponds to a sink $dest_i$. The backbone creation process is executed in sequence for each sink $dest_i$ with the corresponding parameter b_i.

Privacy. The algorithm for constructing the backbone takes b as one of the inputs and the covers required number sensors to achieve b bits of privacy. The achieved privacy is, thus, at least b bits.

Energy consumption. Let h be the average hop distance from a source to the nearest backbone member. Since there are $|L|$ backbone members, every event received by the backbone will be rebroadcasted by $|L|$ sensors. Thus, the communication overhead for delivering one event will be $h + |L|$ packets. Since there are $|S_A|$ sources and each source will trigger an event report every α time intervals, the overall communication cost can be estimated by

$$\omega_T = (h + |L|) \times |S_A| \times \frac{T}{\alpha \times \Delta}.$$

Efficiency of backbone creation. In the proposed algorithm, a newly elected backbone member will send an election message and select a neighbor that can cover the maximum number of uncovered neighbors. When $m = 1$, it will backtrack if and only if none of its neighbors can cover at least one uncovered node. Hence, when $m = 1$, the algorithm will rarely need to backtrack and require only $O(|L|)$ messages.

The algorithm will require $O(|L|^2)$ messages when $m \gg n_b$. This is because a newly elected backbone member will need to collaborate with all other backbone members to find a sensor (the new backbone member) that can cover the maximum uncovered sensors. Detailed simulation studies are done in Section 6 to show the cost for different values of m.

5.2.3 Discussion

In our sink simulation protocol, the simulated sinks are static. As a result, if the real sinks are mobile, then the attacker will be able to directly distinguish the simulated sinks from the real ones in the field. There are two options available to deal with mobile sinks. The first option is to create a movement model for simulating the sinks, similar to what we proposed for simulating the sources in the source-location privacy. Another option is to use the backbone flooding idea, where the mobile sink will be able to receive the packets as long as it is within the communication range of at least one backbone node.

In our backbone flooding protocol, the backbone is static. Since the backbone nodes will need to forward more packets than other nodes in the network, they may run out of power quickly. We, thus, need to distribute the backbone function evenly across the network. We can adopt one or more of the following options. The first option is to periodically rebuild the flooding backbone for balancing the load in the network. During backbone reconstruction, we will need to consider the remaining energy at each sensor node. The second option is to construct multiple backbones at the beginning such that each node belongs to approximately the same number of backbones. Each event packet will be delivered to sinks using a randomly selected backbone. In this way, the sensor nodes in the network will roughly forward the same number of event packets. We will investigate these issues in future work.

6 Simulation Evaluation

In this section, we use simulation to evaluate our techniques in terms of energy consumption and latency.

The Panda-Hunter example was introduced in [15], and we will use the terminology from this example to describe our simulation. In this application, a sensor network is deployed to track endangered pandas in a bamboo forest. Each panda has an electronic tag that emits a signal that can be detected by the sensors in the network. We include 5,093 sensor nodes distributed randomly in a square field of $1{,}000 \times 1{,}000$ square meters to monitor the pandas. The *base station* is the *sink* for all real data traffic. Each sensor node can communicate with other sensor nodes in a radius of 50 meters, while an electronic tag attached to a panda can emit radio signals that can reach sensor nodes within 25 meters. We noticed that, on average, each sensor node has 40 neighbors and that the presence of any panda will be detected by 10 sensor nodes. For source-location privacy

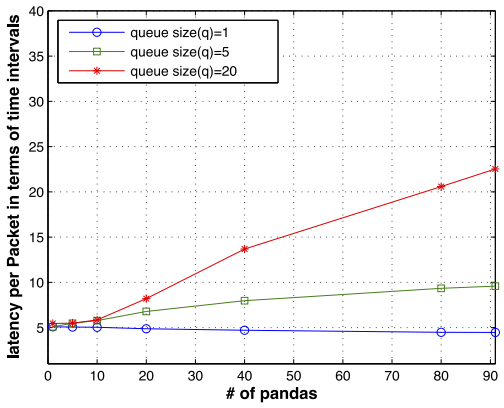

Fig. 3. Latency versus #pandas (periodic collection).

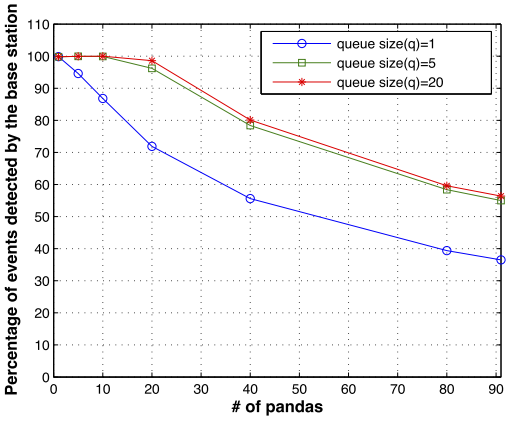

Fig. 4. Event detection versus #pandas (periodic collection).

techniques, we assume that the base station is located at the center of this field. For sink-location privacy techniques, we randomly choose the locations of fake base stations in the field.

The proposed techniques assume a routing protocol for sensor networks, though the choice of routing protocol does not affect our results. For simplicity, we adopt a simple and widely used routing method used in many studies [7]. In this method, the routing paths are constructed by a beacon packet from the base station. Each node, on receiving the beacon packet for the first time, sets the sender of the beacon packet as its parent. In this way, each node will likely select a parent that is closest to the base station.

For the purpose of simulation, we assume that the network application only needs to detect the locations of pandas and always wants to know the most recent locations. We, thus, have every sensor node drop a new packet if it has already queued an identical packet that was generated from the same event.

In our simulation, we assume that the adversary has deployed a network to monitor the traffic in the target network. Specifically, he is able to locate every sensor node in the target network and eavesdrop on every packet this node delivers. Though the adversary may face some engineering problems in developing methods to collect the observations from its network, we do not believe that this will be a very difficult issue to address. For simplicity, we assume that the adversary can always reliably collect all the observations.

Each simulation in our experiment lasts for 6,000 intervals of τ seconds each. The initial locations for pandas are randomly selected. In the experiments, the tag attached to a panda emits a signal for detection at a rate of one per $10 \times \tau$ seconds. In addition, every panda moves from its current location (x, y) to a random location $(x \pm a_1, y \pm a_2)$ every $10 \times \tau$ seconds, where a_1 and a_2 are two random values uniformly selected between 0 and 60.

We compare our techniques with the optimal technique that follows a Steiner tree to route packets to the base stations. We use the approximation algorithm from [28] for approximating the construction of Steiner trees. Section 4 includes a brief description of this algorithm. We also compare our source privacy techniques with the Proxy-based Filter Scheme (PFS) [29]. Dividing the field into square cells of length 17.68 meters gives us 3,249 cells in the entire field. Each of these cells has a candidate proxy. Finding the close to optimal number of proxies and their locations in the network using the proxy placement algorithm $(O(n^7))$ [29] will take significant time. We, therefore, divide the field into square cells of 100 meters so that the number of cells reduces to 100. For simulation of PFS, proxy nodes emit one packet every interval and other sensor nodes generate traffic with interpacket delays following an exponential distribution with a mean of 10 (intervals).

6.1 Source-Location Privacy

6.1.1 Periodic Collection

The analysis in Section 5 shows that the periodic collection method achieves optimal location privacy. In addition, the communication overhead in the network remains constant and is independent of both the number of pandas and their patterns of movement. Hence, the focus of our simulation evaluation is on the latency and the packet drop rate when there are multiple pandas in the field. We set the time interval for periodic collection as $\Delta = \tau$.

Fig. 3 shows the latency of packet delivery when there are multiple pandas. We can see that as the number of pandas increases, the latency increases. This is because the nodes close to the base station receive multiple reports at the same time, which requires them to buffer the packets. When the number of pandas grows too large, the buffered packets start being dropped due to the limited size of the queue, and the latency of the packets that do arrive at the base station becomes stable after a certain point. When the queue size q decreases, packets traveling long distances have a high probability of getting dropped, making the latency of the packets that do arrive at the base station smaller. This can be seen by a drop in the latency for smaller values of q in the figure.

Fig. 4 shows the percentage of the detected events received by the base station. We can see that the percentage of events received decreases when there are more pandas in the field. Increasing q will certainly increase the percentage of the events forwarded to the base station. However, after a certain point, increasing q will not substantially raise the packet drop rate, as seen by the small difference from when $q = 5$ to $q = 20$. On the other hand, we see from Fig. 3 that increasing q will significantly increase the latency of packet

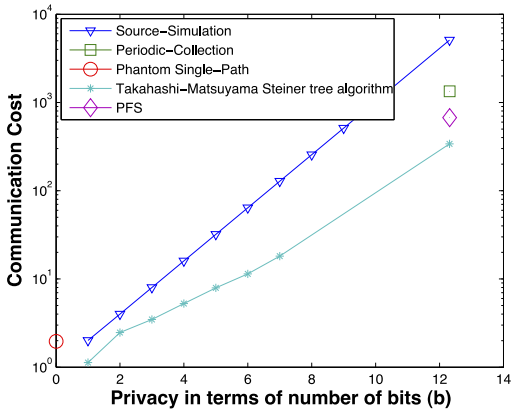

Fig. 5. Comparison of different source-location privacy schemes in terms of communication overhead.

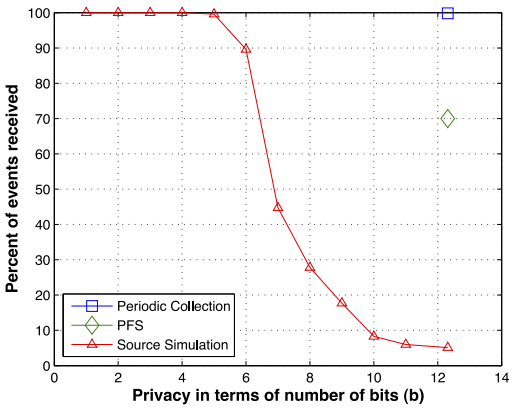

Fig. 6. Comparison of different source-location privacy schemes in terms of event detection rates.

delivery. Thus, fairly small values of q will usually present the best trade-off point between packet drops and latency. Overall, the results in Figs. 3 and 4 give a guideline for configuring the queue size q to meet various requirements.

6.1.2 Source Simulation

According to the analysis in Section 5, the location privacy achieved by the source simulation approach is determined by the number of virtual sources simulated in the network. Thus, the focus of our simulation evaluation is on how much communication cost we have to pay to achieve a given level of location privacy. We use these results to illustrate the efficiency of the proposed technique.

During the simulation, we assume that there is only one panda in the network. Multiple fake pandas are created and simulated in the field. The initial positions of the fake pandas are randomly selected. In addition, we assume that the sensor network is deployed to handle real-time applications. In other words, whenever a sensor node receives a packet, it will forward it to the next hop as soon as possible. Thus, while we set the time interval for periodic collection as $\Delta = \tau$, we set it to $\Delta = \frac{\tau}{10}$ for source simulation. In other words, in source simulation, nodes will forward packets ten times faster than in the periodic collection method. We set P to 1, which means that the adversary has the same knowledge about the panda behavior as the defender and thus cannot distinguish between fake pandas and real pandas based on the observed behavior.

Fig. 5 shows the communication overhead involved in our source simulation method to achieve a given level of privacy. We can see that the communication overhead increases as the location privacy requirement increases. This figure also includes the performance of other approaches for further comparison, which we will explain in Section 6.1.3.

6.1.3 Comparison

We now compare the proposed source-location privacy approaches in this paper with two other privacy-preserving techniques: phantom single-path routing [15] and proxy-based filtering [29]. We focus on the location privacy achieved and the communication overhead introduced in the following comparison. The result of our simulations is shown in Fig. 5. The overhead of the phantom single-path routing scheme is represented by a single point at the bottom-left corner of the figure, and overheads of the periodic collection and the proxy based filtering techniques are represented by points on the right part of the figure.

In terms of privacy, we have already shown that none of the previous methods (including phantom single-path routing) can provide location privacy under the assumption of a global eavesdropper. In contrast, both of our methods provide location privacy against a global eavesdropper. The periodic collection method provides the highest level of privacy and is suitable for applications that collect data at a low rate and do not require real-time data delivery, while the source simulation method can support real-time applications with practical trade-offs between privacy, communication overhead, and latency.

Fig. 5 also shows the communication costs involved in different methods. The simulation results are as we would predict from intuition. The phantom single-path routing technique introduces relatively little communication overhead, while the periodic collection method involves significant but constant communication cost for a given period of time. The source simulation method provides increasing levels of privacy at the cost of more communication. We notice that in the figure, the periodic collection method requires less communication overhead to achieve privacy of around $b = 12$ bits when compared with the source simulation method. The reason is that the source simulation method is configured to support real-time applications with a time interval Δ one-tenth the length of that used in the periodic collection method.

From Fig. 5, we notice that the cost of the proxy-based filtering (PFS) technique [29] lies between the costs of the periodic collection technique and the (theoretical) Steiner tree-based technique. However, both of our methods also have advantages over PFS. First, during simulation of PFS technique, we noticed that around 70 percent of events were received by the base station. However, for the periodic collection method, the detection rate can be as high as 99 percent. The results are shown in Fig. 6. Second, the source simulation scheme can provide practical trade-offs between location privacy and communication cost. In addition, based on Fig. 6, we can clearly see that the source simulation idea can achieve a better detection rate when the privacy requirement is $b = 6$ or fewer bits.

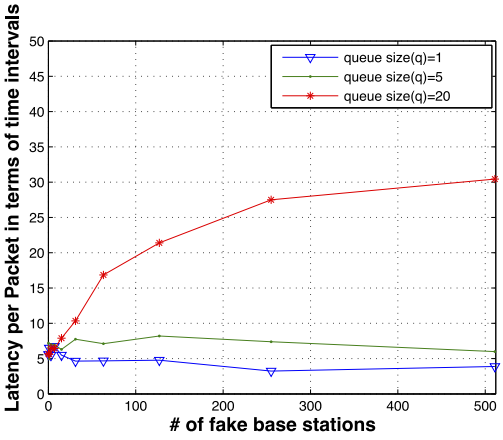

Fig. 7. Effect of the number of fake base stations on latency (sink simulation scheme).

We can also see the performance of these techniques in comparison to the approximate Steiner tree algorithm. For achieving the maximum privacy, the periodic collection technique consumes more energy than the approximate Steiner tree algorithm. The reason is that, in the periodic collection scheme, each sensor emits a packet every $\Delta = \tau$ seconds, while in the approximate Steiner tree algorithm, each sensor emits a packet once every $10 \times \Delta = 10 \times \tau$ seconds, as is the case with a real source ($\alpha = 10$).

6.2 Sink-Location Privacy

6.2.1 Sink Simulation

The analysis in Section 5 shows that the location privacy achieved and the amount of energy consumed by the sink simulation scheme depend on the number of fake base stations simulated in the network. The packets generated by the sources are sent to all fake and real base stations. Hence, the focus of our simulation evaluation is on the latency and the packet drop rate when there are multiple base stations in the field.

Fig. 7 shows the latency of packet delivery when there are multiple fake base stations in the field. We can see that as the number of fake base stations increases, thereby providing more location privacy, the latency increases. This is because having more base stations causes more traffic in the network and thus more packets to be buffered. When the number of fake base stations grows too large, the buffered packets start being dropped due to nodes' limited queue sizes, while the latency of the packets that do arrive at the base station becomes stable after a certain point. When the queue size q decreases, packets traveling long distances have a high probability of getting dropped, making the latency of the packets that do arrive at the real base station smaller. This can be seen by a drop in the latency for smaller values of q.

Fig. 8 shows the percentage of detected events received by the real base station. We see that the percentage of events received decreases when there are more fake base stations in the field. Figs. 8 and 7 give guidelines for configuring the queue size q and the number of fake base stations to meet various requirements.

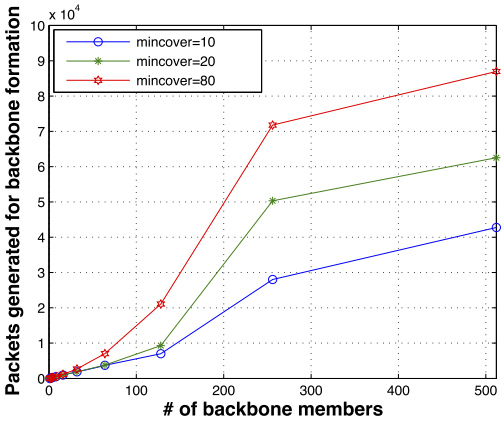

Fig. 9. Energy consumed for creation of backbone.

6.2.2 Backbone Flooding

From the analysis in Section 5, we see that the location privacy achieved by the backbone flooding approach increases with the number of backbone members. Packets generated by a source are sent to all backbone members. Hence, the focus of our simulation evaluation is on the delivery latency, the packet drop rate, and the energy required for backbone creation.

Fig. 9 shows that increasing the backbone size will cause more energy to be consumed. We also see that an increase in the parameter m, the mincover, will result in more backtracking in the backbone creation and hence consume more energy.

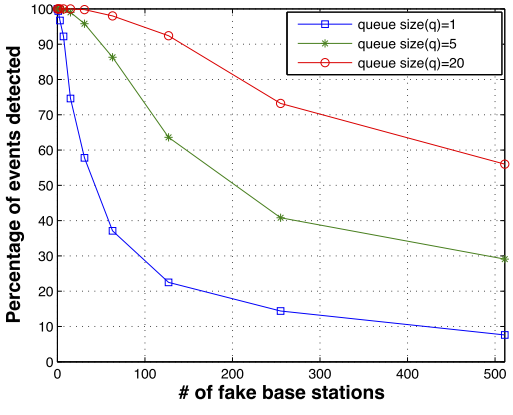

Fig. 8. Effect of the number of fake base station on the percentage of events detected by the base station (sink simulation scheme).

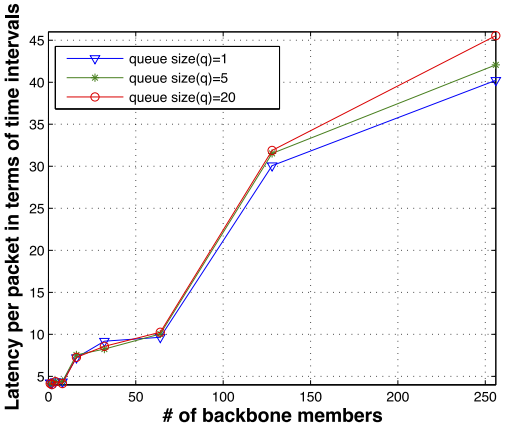

Fig. 10. Effect of backbone size on latency.

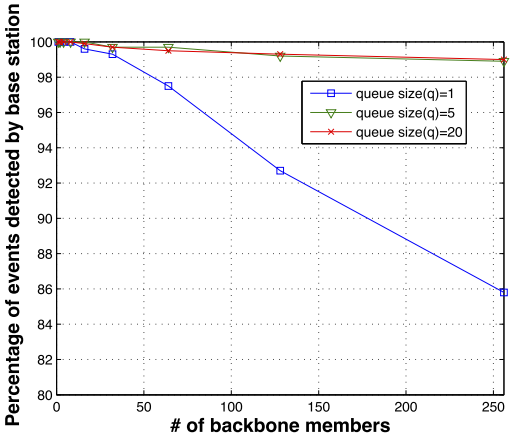

Fig. 11. Effect of backbone size on percentage of events detected by the base station.

Fig. 10 shows that the latency of packet delivery increases as the size of the backbone increases. This is because an increase in the backbone size will cause an increase in the number of packets in the network, causing buffering of more packets and a corresponding increase in latency.

Fig. 11 shows the percentage of the detected events received by the base station. We can see that the percentage of events received decreases when there are more backbone members in the field. We have to make trade-offs between the latency and the packet drop rate to meet various requirements.

6.2.3 Comparison

We now evaluate the proposed sink-location privacy approaches described in this paper. We focus on the location privacy achieved and the communication overhead introduced by each technique. The simulation results are shown in Fig. 12.

In terms of privacy, we have already shown that none of the previous methods can provide location privacy under the assumption of a global eavesdropper. In contrast, both of our methods provide sink-location privacy against a global eavesdropper.

We compare the communication overheads through simulation. Fig. 12 shows the communication costs involved in different methods. Both techniques can provide practical

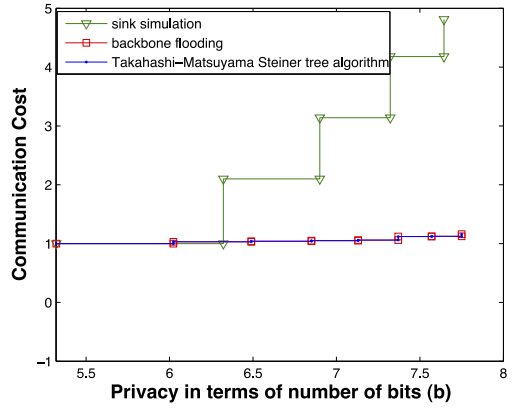

Fig. 12. Comparison of sink-location privacy techniques in terms of communication cost.

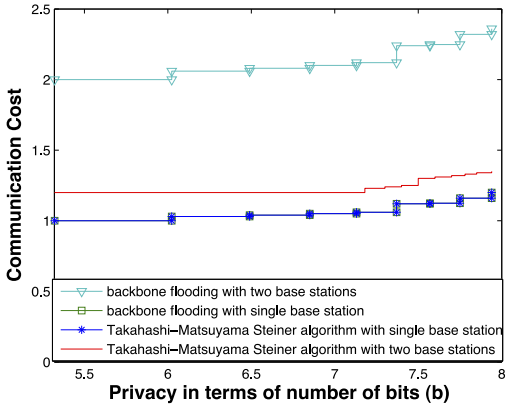

Fig. 13. Effect of multiple base station on backbone flooding and the approximate Steiner tree technique.

trade-offs between privacy and communication cost. We note that backbone flooding consumes less energy. The reason is that this method does not incur much cost to generate traffic toward the fake base stations. A single rebroadcast of packets in the backbone effectively creates many fake base stations. We note that both the approximate Steiner tree and backbone flooding techniques are stair curves because a single packet transmission can be received by all neighbors of the sender. All of the neighbors will be considered by the adversary to be equally likely to be a real base station. Hence, the energy consumption will remain the same for privacy in the range of $(1, log_2(n_b))$ bits.

In Fig. 13, we see the effect of multiple real base stations on communication cost for the desired level of location privacy. Each source sends every packet to both base stations. We randomly placed the two base stations in the network. The communication cost of backbone flooding doubles when the number of base stations doubles. This is because, by design, the source communicates with each backbone independently. However, the Steiner tree algorithm only incurs a small increase in communication cost. We can see that when we build the approximate Steiner in the case of multiple base stations, the communication cost remains constant until the privacy requirement grows above 7 bits. This is because the packets from a source will always go through the same 10 hops and these 10 hops cover as many sensors as required for about 7 bits of privacy.

6.3 Discussion on Using the Proposed Techniques

The proposed location privacy techniques in this paper have advantages and disadvantages when compared with each other. We now briefly summarize our understanding of which solutions should be used for different applications. The periodic collection and source simulation methods can be used for providing source-location privacy. The periodic collection method provides the highest location privacy and is hence useful when we are monitoring highly valuable objects. Additionally, the communication cost—though high—does not increase with the number of monitored objects. Thus, it is suitable for applications that collect data at a low rate from the network about many objects. The source simulation method provides a trade-off between privacy and communication costs. It is suitable for scenarios

where 1) the object movement pattern can be properly modeled and 2) we need to collect real-time data from the network about the objects.

The sink simulation and backbone flooding methods can provide location privacy for the sinks. The backbone flooding method is clearly more suitable for the cases where a high level of location privacy is needed, as we can see from Fig. 12. However, when the required level of location privacy is below a certain threshold (e.g., 6.4 bits as shown in Fig. 12), the sink simulation method becomes more attractive, since it is more robust to node failure in the network. In the backbone flooding idea, we need to always keep the backbone connected and rebuild the backbone from time to time to balance the communication costs between nodes.

7 Conclusions

Prior work on location privacy in sensor networks assumed a local eavesdropper. This assumption is unrealistic given a well-funded, highly motivated attacker. In this paper, we formalized the location privacy issues under a global eavesdropper and estimated the minimum average communication overhead needed to achieve a given level of privacy. We also presented techniques to provide location privacy to objects and sinks against a global eavesdropper. We used analysis and simulation to show how well these techniques perform in dealing with a global eavesdropper.

There are a number of directions that worth studying in the future. First, in this paper, we assume that the global eavesdropper does not compromise sensor nodes. However, in practice, the global eavesdropper may be able to compromise a subset of the sensor nodes in the field and perform traffic analysis with additional knowledge from insiders. This presents interesting challenges to our methods. Second, it takes time for the observations made by the adversarial network to reach the adversary for analysis and reaction. Studying the impact of such "delayed" analysis and reaction will be another interesting research direction.

Acknowledgments

This work was supported by the US National Science Foundation under grant CNS-0916221. The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks,* vol. 38, no. 4, pp. 393-422, 2002.

[2] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting Anonymous Location Queries in Mobile Environments with Privacygrid," *Proc. Int'l Conf. World Wide Web (WWW '08),* 2008.

[3] BlueRadios Inc., "Order and Price Info," http://www.blueradios.com/orderinfo.htm, Feb. 2006.

[4] B. Bollobas, D. Gamarnik, O. Riordan, and B. Sudakov, "On the Value of a Random Minimum Weight Steiner Tree," *Combinatorica,* vol. 24, no. 2, pp. 187-207, 2004.

[5] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy (S&P '03),* pp. 197-213, May 2003.

[6] J. Deng, R. Han, and S. Mishra, "Enhancing Base Station Security in Wireless Sensor Networks," Technical Report CU-CS-951-03, Univ. of Colorado, Dept. of Computer Science, 2003.

[7] J. Deng, R. Han, and S. Mishra, "Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '04),* June 2004.

[8] J. Deng, R. Han, and S. Mishra, "Decorrelating Wireless Sensor Network Traffic to Inhibit Traffic Analysis Attacks," *Pervasive and Mobile Computing J.,* Special Issue on Security in Wireless Mobile Computing Systems, vol. 2, pp. 159-186, Apr. 2006.

[9] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. ACM Conf. Computer and Comm. Security (CCS '02),* Nov. 2002.

[10] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.L. Tan, "Private Queries in Location Based Services: Anonymizers are not Necessary," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08),* 2008.

[11] H. Gupta, Z. Zhou, S. Das, and Q. Gu, "Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution," *IEEE/ACM Trans. Networking,* vol. 14, no. 1, pp. 55-67, Feb. 2006.

[12] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The Platforms Enabling Wireless Sensor Networks," *Comm. ACM,* vol. 47, no. 6, pp. 41-46, 2004.

[13] Y. Jian, S. Chen, Z. Zhang, and L. Zhang, "Protecting Receiver-Location Privacy in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1955-1963, May 2007.

[14] D.B. Johnson, D.A. Maltz, Y. Hu, and J.G. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF Internet draft, Feb. 2002.

[15] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing Source-Location Privacy in Sensor Network Routing," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '05),* June 2005.

[16] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. ACM Conf. Computer and Comm. Security (CCS '03),* Oct. 2003.

[17] K. Mehta, D. Liu, and M. Wright, "Location Privacy in Sensor Networks against a Global Eavesdropper," *Proc. IEEE Int'l Conf. Network Protocols (ICNP '07),* 2007.

[18] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) Using AoA," *Proc. IEEE INFOCOM,* pp. 1734-1743, Apr. 2003.

[19] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon, "Entrapping Adversaries for Source Protection in Sensor Networks," *Proc. Int'l Conf. World of Wireless, Mobile, and Multimedia Networking (WoWMoM '06),* June 2006.

[20] C. Ozturk, Y. Zhang, and W. Trappe, "Source-Location Privacy in Energy-Constrained Sensor Network Routing," *Proc. Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* Oct. 2004.

[21] V. Paruchuri, A. Duressi, M. Duressi, and L. Barolli, "Routing through Backbone Structures in Sensor Networks," *Proc. 11th Int'l Conf. Parallel and Distributed Systems (ICPADS '05),* 2005.

[22] C.E. Perkins, E.M. Belding-Royer, and S.R. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," IETF Internet draft, Feb. 2003.

[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. ACM MobiCom,* July 2001.

[24] T.S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno, "Devices that Tell on You: Privacy Trends in Consumer Ubiquitous Computing," *Proc. USENIX Security Symp.,* 2007.

[25] A. Savvides, C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proc. ACM MobiCom,* July 2001.

[26] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards Statistically Strong Source Anonymity for Sensor Networks," *Proc. IEEE INFOCOM,* 2008.

[27] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting Your Daily In-Home Activity Information from a Wireless Snooping Attack," *Proc. Int'l Conf. Ubiquitous Computing (UbiComp '08),* 2008.

[28] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Math. Japonica,* vol. 24, pp. 573-577, 1980.

[29] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards Event Source Unobservability with Minimum Network Traffic in Sensor Networks," *Proc. ACM Conf. Wireless Network Security (WiSec '08),* 2008.

Kiran Mehta received the BS and MS degrees in computer science from the University of Pune, India, in 2002 and 2004, respectively. He also received the MS degree in computer science from the University of Texas at Arlington in 2008. He is currently working toward the PhD degree in the Department of Computer Science, The George Washington University. From 2004 to 2006, he was with Symantec Corporation, Pune, in the Cluster Server Group. His research interests include wireless security and database security.

Donggang Liu received the BS degree from the Department of Computer Science, Beijing Institute of Technology, China, in 1998, the MS degree from the Institute of Computing Technology, Chinese Academy of Science, China, in 2001, and the PhD degree from the Department of Computer Science, North Carolina State University, in 2005. He is an assistant professor in the Department of Computer Science and Engineering, University of Texas at Arlington. His research interests include network and distributed system security. Currently, he focuses on wireless security and software system security. He is a member of the IEEE.

Matthew Wright received the BS degree in computer science from Harvey Mudd College. He received the MS and PhD degrees from the Department of Computer Science, University of Massachusetts, in 2002 and 2005, respectively. His dissertation work addresses the robustness of anonymous communications. He is an assistant professor at the University of Texas at Arlington. His other interests include intrusion detection, security and privacy in mobile and ubiquitous systems, and the application of incentives to security and privacy problems. He is a recipient of the US National Science Foundation CAREER Award and the Outstanding Paper Award from the 2002 Symposium on Network and Distributed System Security. He is a member of the IEEE.