

Source Location Privacy against Laptop-Class Attacks in Sensor Networks

Yi Ouyang
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, TX
yi.ouyang.a@gmail.com

Zhengyi Le
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, TX
zyle@uta.edu

Donggang Liu
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, TX
dliu@uta.edu

James Ford
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, TX

Fillia Makedon
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, TX
makedon@uta.edu

ABSTRACT

Sensor networks may be used in many monitoring applications where the locations of the monitored objects are quite sensitive and need to be protected. Previous research mainly focuses on protecting source location against *mote-class attackers* who only have a local view of the network traffic. In this paper, we focus on how to protect the source location against *laptop-class attackers* who have a global view of the network traffic. This paper proposes four schemes—naive, global, greedy, and probabilistic—to deal with laptop-class attacks. The naive solution uses maintenance messages sent periodically to hide real event reports. The global and greedy solutions improve the naive solution by reducing the latency of event delivery without increasing communication overhead. The probabilistic solution further improves the performance by reducing communication overhead without sacrificing location privacy. Experiments show that the probabilistic solution is practical for providing source location privacy against a laptop-class attacker.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—Security and protection

General Terms

Design, Security, Performance.

Keywords

Source location privacy, Sensor networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SecureComm 2008, September 22 - 25, 2008, Istanbul, Turkey
Copyright 2008 ACM ISBN # 978-1-60558-241-2 ...\$5.00.

1. INTRODUCTION

In recent years, sensor networks have been developed and applied in various applications. In our view, there are three main research areas for sensor networks: the first is to develop more sophisticated and accurate sensors; the second is to bring together all necessary information quickly and effectively; and the third is to guarantee that messages are tamper-proof and that there is no unnecessary information leakage in sensor networks. This paper focuses on the third area, and in particular on information leakage.

Information may leak under many circumstances even when data have been encrypted during transmission; one example is the so-called *source location protection problem*, in which attackers attempt to discover the source node emitting messages that are relayed through the network to their final destination(s). Attackers take advantage of the fact that when a sensor senses an event, it will send messages towards a base station to report this event. Note that an attacker can easily detect the transmission of messages in sensor networks. Given enough time, he/she can determine the source location(s) originating messages by analyzing the traffic patterns even if the communications are all encrypted. Since it is then possible for them to interfere with the phenomena being sensed or even mount physical attacks on the monitored objects, the exposure of the source location can be quite dangerous.

Previous work (summarized in the following section) has divided attackers into two types: *mote-class* and *laptop-class* [7]. *Mote-class attackers* are assumed to be limited to small physical devices with capabilities similar to sensor nodes. Thus, at any given time, a *mote-class* attacker can only monitor communications between a limited number of nodes. In contrast, *laptop-class attackers* are assumed to have much stronger computational capabilities and longer radio range—if they are equipped with hardware powerful enough, the radius of their monitoring area could even cover the entire network. In this work, we assume that *laptop-class* attackers can eavesdrop on all communications in a sensor network. Existing research has focused on countering *mote-class* attacks, for example using phantom routing

techniques [6, 11], generating loops to mislead attackers [10], countermeasures for traffic analysis [2], and anonymous approaches [1, 3, 8]; however, there has been little work addressing the problem of laptop-class attackers. This paper addresses laptop-class attacks in sensor networks.

We assume that a laptop-class attacker can monitor the entire network traffic, but is not able to decrypt messages. Monitoring the entire network traffic allows the attacker to identify the source locations of messages using timing analysis. For example, a sensor node that initiates a particular communication should be close to the monitored object it reports on. The essence of our approach for laptop-class attackers is to have every sensor node generate *maintenance* (dummy) messages and make event messages appear indistinguishable from the maintenance messages that do not involve important events.

In this paper, we propose four schemes—naive, global, greedy, and probabilistic. In the naive approach, every node periodically sends a maintenance message and replaces it with a real event report whenever an event is detected or a real event report is received. The time interval of maintenance messages will be usually quite long to lower the communication cost. Therefore, this approach often leads to long delay for event messages to reach the base station.

The global and greedy approaches improve the naive approach by reducing the delivery latency without increasing the communication cost. These two approaches take advantage of the fact that there will be multiple routing paths to the base station and these paths will result in different delivery latencies (to the base station). The global approach assumes a knowledge of global network topology and transmission schedules of all sensor nodes and can always discover the routing path that leads to the shortest delivery latency. The greedy approach is a heuristic algorithm that tries to accomplish the same objective as the global approach but only requires the knowledge of local network topology and transmission schedules.

The probabilistic approach further improves performance by reducing communication overhead without sacrificing location privacy. This approach is based on the fact that it is very difficult for an adversary to tell the exact source of a maintenance message that do not include any information about the source node. Hence, for a given sensor node, the maintenance messages sent by nearby sensor nodes can also be used to hide its real event messages. As a result, we have every node *probabilistically* send out maintenance messages to guarantee that every node can hear a maintenance message during any fixed period of time. This makes it difficult for the adversary to tell whether a message is originated or forwarded by a given node. This approach can significantly increase the performance when the sensor network is densely deployed, since every maintenance message can thus be used to hide the real event messages at many other nodes. In addition, we also analyze the requirements of privacy and the minimum probability of a node sending maintenance messages and give quantitative analyses for the proposed approaches. Simulation results show that the probabilistic approach is a practical solution in dealing with laptop-class attackers.

Section 2 summarizes related work on location privacy. In Section 3, we discuss the background, attack models, and design goals. Our approaches are presented in Sections 4, 5, 6, and 7. Experimental results are shown in Section 8.

Finally, Section 9 gives our conclusions and lists some areas for future research.

2. RELATED WORK

There has been significant past research on source location privacy in wireless networks. Chaum’s mixing approach [1] was shown to provide anonymous connections that protected against traffic analysis and were useful in an onion routing mechanism [12]. Kong *et al.* [8] proposed an anonymous on-demand routing protocol for mobile ad hoc networks and addressed two problems: route anonymity and location privacy. Gruteser *et al.* [3, 4] introduced an adaptive algorithm to adjust the resolution of information in order to protect individual privacy when supplying location-based services.

Existing research on source location privacy in sensor networks has focused on mote-class attackers. Ozturk *et al.* [11] introduced the Panda-Hunter model to formalize the source location problem in sensor networks and also proposed the phantom flooding approach. Kamat *et al.* [6] extended the work of [11] and proposed phantom routing techniques based on both flooding and single-path routing. Deng *et al.* [2] investigated several countermeasures against traffic analysis that aimed to protect the location of a base station. Ouyang *et al.* [10] proposed a cyclic entrapment method to protect source locations. Kamat *et al.* [5] analyzed temporal privacy in delay tolerant sensor networks and proposed adaptive buffering at intermediate nodes. Mehta *et al.* [9] proposed periodic collection and source simulation techniques to prevent the leakage of location information. Shao *et al.* [13] discussed a scheme called FitProbRate, which realizes statistically strong source anonymity for sensor networks. Yang *et al.* [14] proposed to introduce carefully chosen dummy traffic to hide the real source and select some sensors as proxies that proactively filter dummy messages on their way to the base station.

3. PROBLEM DEFINITION

This section gives a more detailed background on sensor networks, the capabilities of a laptop class attacker, and then defines the goals of our work.

3.1 Network Model

A sensor network consists of a base station and potentially a large number of sensor nodes. The base station can send commands to and collect messages from sensor nodes in its network through multi-hop routing. The base station may also act as a gateway to connect a sensor network with traditional wired/wireless networks. A sensor node can sense events, generate *event messages*, and send them to its neighbor nodes within its limited radio range. Sensor nodes can also forward messages received from their neighbors toward the base station according to a pre-set routing protocol.

3.2 Attack Model

Here we give the properties of our laptop class attack model. The attacker knows the location of every sensor node because of his powerful detection radius and has a global view of the sensor network. The attacker can always hear all on-going communications in the sensor networks through powerful equipment. The attacker has sufficient off-line storage to retain all data he gains by eavesdropping, allowing subsequent analysis on the data. The attacker is

able to sense, store, and analyze all transmitted data, but is not able to understand them (based on the assumption that data can be safely encrypted for transmission). The attacker only eavesdrops on communications between sensor nodes and will not physically compromise sensor nodes. The attacker's goal is to *find the location of the source node who is originating an event message*.

3.3 Goals

If there is only one sensor node sending messages in the network, the laptop class attacker can very easily locate the sensor node because he has a global view of the sensor network. In this research, our goals are as follows.

1. *Protecting the source location against laptop-class monitoring*: even if the hiding algorithm itself is public and thus available to attackers, the attacker cannot determine the source location through traffic monitoring and analysis.
2. *Lowering the computation and transmission cost of hiding sources*: we want to minimize the cost for our solutions in terms of the resources (such as energy and storage) required in the sensor network.

3.4 Metrics

Here we define the metrics used to evaluate the methods for source location protection methods.

- *Security*: The probability that an attacker successfully identifies the source node originating a real event message.
- *Delivery time*: The time for an event message traveling from the source to the base station.
- *Energy cost*: The energy cost of sensor nodes consists of computational cost and communication cost. Communication is much more expensive than computation in terms of energy. To simplify the analysis, we focus on the communication cost of the network. In this paper, we use the number of messages transmitted in the network to measure the energy cost.

4. A NAIVE ALGORITHM

We now consider a simple but incomplete solution as a way to motivate the methods introduced later in the paper. In standard sensor networks as described in Section 3.1, a sensor node sends event messages immediately when it senses events. When an attacker notices that a sensor node has sent a message, he knows that an event happened in the location of the sender. A naive solution is to use dummy messages to hide the real event messages. In this idea, we let each node broadcast a dummy message to its neighbors at the end of each fixed time period. In this paper, we call these dummy messages *maintenance messages* and the periods *maintenance periods*. The maintenance period is usually made long enough to make sure that the sensor nodes do not run out of battery quickly due to maintenance messages. From the view of an attacker, all nodes in the network are sending messages at a fixed rate.

When a source wants to send an event message, the event message can masquerade as a maintenance message, *i.e.* by delaying this event message and replacing the next maintenance message with it. In this way, when a source is sending

a real event message instead of a maintenance message, the attacker cannot distinguish between them and will still see that every node sends messages at a fixed rate. The receiver (sensor node) of this event message needs to wait until the end of its current maintenance period and forward the message to the next node along the routing path to the base station.

4.1 Security

From the view of the attacker, every node is sending messages at a fixed rate no matter whether there is a source sending an event message to the base station. Thus, the probability of a successful compromise of source location is not larger than randomly selecting a node in the network as a possible source. Thus, the compromise probability of this naive solution is $1/n$, where n is the number of nodes in the network.

4.2 Delivery time

Because the event message is always forwarded by the nodes along the routing path to the base station masquerading as maintenance messages, the event message needs to stay at every intermediate node until the end of the current fixed maintenance period. The expected waiting time on every node is $t_m/2$, where t_m is the time for the fixed maintenance period. Thus, the delivery time for this message will be $T_d = t_w + (t_m/2 + t_{trans}) \times h$, where t_w is the time from when an event happens to the end of current maintenance period ($t_w < t_m$), t_{trans} is the time for transmitting a message between two nodes, and h is the number of hops from the source to the base station along the routing path. If every node forwards the event message as soon as it receives the message, the delivery time is $t_{trans} \times h$. To save energy in sensor nodes, generally $t_m \gg t_{trans}$. Thus, an event message will take much more time to arrive at the base station.

4.3 Energy cost

The energy cost for all the sensor nodes in this naive algorithm depends on how frequently messages are sent. Since every sensor node sends messages according to a fixed maintenance period t_m , in time t , the expected number of periods is t/t_m . Thus the expected number of messages sent in time t is nt/t_m , where n is the number of nodes in the network. The energy cost of the network will increase if the size of the network becomes larger or the maintenance period becomes shorter.

In this naive solution, it is impossible for the attacker to compromise the source's location since there is no difference from the view of the attacker before and after the events, however, the delivery time is long because of the waiting time on every maintenance period. In the following sections, we propose several methods to decrease the delivery time and the energy cost while maintaining $1/n$ probability for source locations being compromised.

5. A GLOBALLY OPTIMAL ALGORITHM

The naive approach uses the underlying routing protocol to find the path to the base station. However, there are many paths to the base station, and the one used by the naive solution does not necessary result in the shortest delivery time. The reason is that an event report at any sensor node

has to wait until the end of the maintenance period, and the waiting time is usually different for different sensor nodes.

In the globally optimal approach, the duration of maintenance periods will no longer be fixed; they are actually determined by a Pseudo Random Number Generator (PRNG). The random maintenance period for a node could be a PRNG output in a range $[a, b]$ ($a < b$) so that every node at least waits for time a before sending another maintenance message. As a result, when an event message arrives at a sensor node, it needs to wait for time $\frac{a^2+b^2+ab}{3(a+b)}$ on average (see Lemma 1 in Section 6). Let h be the number of hops from the source to the base station, the delivery time would be $\frac{(a^2+b^2+ab) \times h}{3(a+b)}$ on average. However, as we will show next, the globally optimal solution can achieve much better performance since it can easily compute the path with the shortest overall waiting time, which may be different from the one identified in the naive approach.

Since we use a PRNG, it is possible (given the random seeds for the generator) for the transmitting (source) node to predict not only the forthcoming pseudo-random numbers for itself, but also those of all nodes in the network. Given this information, it can calculate the fastest routing path leading to the shortest delivery time for an outgoing event message when the global network topology is available and all sensor nodes are well-synchronized in time.

Algorithm 1 determines the fastest path from the source to the base station. Here, G is an undirected sensor network graph, vector p is the source node, and $base$ is the base station. Each node i has the following fields:

$i.no$	Index of this node
$i.t$	Length of next period (the default invalid value is -1)
$i.previous$	Pointer to the previous node on the shortest path
$i.neighbour$	Pointer to the first neighbor node
$i.nextNeighbour$	Pointer to the next neighbor node

Algorithm 1 is globally optimized. It requires every sensor node to store the global topology of the sensor network. It runs an exhaustive breadth-first search and dynamically adjusts the path from the source to the base station until finally the fastest path is found. Note that this algorithm is distinct from the Dijkstra shortest path algorithm, which requires that every edge has a fixed weight (the weight, i.e., the waiting time, in our case is changing randomly with time). In Algorithm 1, $nextP(index, time)$ takes the index of a node and a specific time point as input and outputs the starting time point of the next period according to the agreed PRNG. The output of the PRNG depends only on its random seed and is within the range of $[a, b]$, where a and b are the minimum and maximum lengths of the time period and are usually set by users. Algorithm 2 gives an example implementation for $nextP(index, time)$.

After the source node determines the path with the shortest overall waiting time, the source node can replace the next maintenance message with the real event message and send it to the next node along this path. The event message will go through the path and every forwarding node along the path will replace its next maintenance message with this event message. This event message will have the minimum delivery time.

5.1 Security

In this globally optimal algorithm, the message sender only replaces a maintenance message with a real event mes-

sage. All the messages are still sent using the same time sequence as before. Thus, the attacker cannot distinguish event messages from maintenance messages and the compromise probability is $1/n$.

5.2 Delivery time

The globally optimal algorithm finds the minimum delivery time for a message given the topology of the sensor network. To analyze the performance of this algorithm, we implemented the global algorithm and the naive algorithm in the same simulation environment, with 20,000 sensor nodes in a $5,000m \times 5,000m$ area. The radio range of a sensor node is 100m and the transmission time of a message from one node to its neighbors is 10 ms. Every sensor node sends a message after a random maintenance period that in the range of $a-b$. The average waiting time for a message at every nodes is $\frac{a+b}{2}$. Figure 1 shows the comparison of delivery time between the naive method and the globally optimal algorithm. We can see that the delivery time of the globally optimal solution is much better than the naive solution. Thus, when using pseudo random numbers as waiting time, a sensor node can use brute force search to get a path with minimum delivery time. However, a sensor node needs to know the topology of all the nodes to compute the optimal solution.

```

Input: Graph G; Vector p; Vector base
Result: A routing path with shortest delivery time
createQ(Q);
p.t ← t_start; //p is the source node
enterQ(p, Q); //enter as a tail
repeat
  i ← frontQ(Q);
  if i = base then
    | movetoTail(i, Q);
    | i ← frontQ(Q);
  end
  delHeadQ(Q);
  q ← i.neighbor;
  while (q ≠ nil) ∧ (i ≠ base) do
    if q.t = -1 then
      | q.t ← nextP(q.no, i.t);
      | q.previous ← i;
      | enterQ(q, Q);
    else
      | if q.t > nextP(q.no, i.t) then
        | | q.t ← nextP(q.no, i.t);
        | | q.previous ← i;
      end
    end
  end
  q ← i.nextNeighbour;
end
until i = base;

```

Algorithm 1: Global optimal

```

Input: Index no; Time t
Result: Start time point for the next period
if no = base.no then
  | nextP ← ∞;
  | return nextP;
end
nextP ← system initial time;
repeat
  | nextP ← nextP + PRNG(no, nextP);
until nextP > t;
return nextP;

```

Algorithm 2: Computing nextP()

5.3 Energy cost

Since each node keeps sending messages according to its random maintenance period in the range $a-b$, in a time period t the expected number of maintenance period is $\frac{2t}{a+b}$. Thus, the expected number of messages sent in the network during time t is $N_m = \frac{2tn}{a+b}$. In the naive method, the energy cost is $\frac{nt}{t_m}$. Thus, if $t_m = \frac{a+b}{2}$, the energy cost of the global algorithm is as same as the naive algorithm.

Although the global algorithm has the minimum delivery time, it requires that the source knows the pseudo random

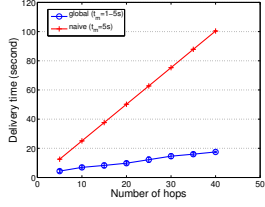


Figure 1: Global algorithm (Random period: 1-5s) vs. Naive method (Maintenance period: 3s)

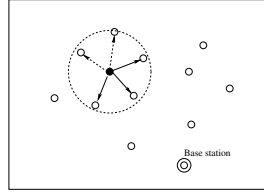


Figure 2: Selecting the next node using greedy algorithm

numbers used by all the sensor nodes in the network to compute the path with shortest delivery time. Because of sensor nodes' limited capabilities, the global algorithm is not very suitable for real applications. To mitigate the storage and computation constraints of global algorithm, we propose a greedy algorithm in the next section.

6. A HEURISTIC GREEDY ALGORITHM

Our globally optimal algorithm is an optimal solution for finding the fastest path given pseudo-random maintenance periods, but it is costly in terms of both computation and storage. It will be impractical for every node to store the global topology of the network and to compute the fastest path. In order to reduce the extra costs in computation, storage, and communication, we propose a heuristic greedy algorithm that only require a local view of the network topology.

The greedy approach simply has every node select the next node towards the base station only based on its neighbors' PRNGs and their distance to the base station. In this way, each node does not need to know the global topology and store all the PRNGs for all the nodes; it only needs to store the PRNGs of all its neighbors.

The main issue of the greedy approach is how to select the next node towards the base station. In order to send the event message back to the base station faster, those neighbor nodes that are closer to the end of the maintenance periods are preferable because the waiting time on these nodes will be shorter. Since a sensor node can compute the end of maintenance periods for all of its neighbors using their random seeds, it can certainly select the one with the minimum waiting time as the next node in the routing path. However, this is not always the best option since it is possible that the one picked is further away from the base station. Picking a node further away from the base station will likely increase the delivery time since the message will likely go through more intermediate nodes.

In Figure 2, the nodes pointed to by the dashed arrows are those we will not choose as the next hop. In order to select the right node, we have alternatives such as the physical distance, the angle between the base station and the neighbor node, and the minimum number of hops. Because sensor nodes are discretely distributed, the fact that two nodes are physically further apart does not necessarily mean the minimum number of hops between them is increased; a similar observation applies for angles. Thus, using the minimum number of hops to the base station as a measure is

preferable. Therefore, the greedy approach can choose the neighbor that has the shortest waiting time and is closer (in terms of the hop distance) to the base station as the next hop towards the base station. The hop distance is defined as the number of hops on the shortest path to the base station, which can be obtained in the system initialization phase.

```

Input: PRNGs of all the neighbors of node  $i$ 
Result:  $i.next, t$ 
 $i.t \leftarrow nextP(i.no, t_c)$ ;
 $i.next \leftarrow i$ ;
 $q \leftarrow i.neighbor$ ;
 $t \leftarrow nextP(q.no, i.t)$ ;
 $i.next \leftarrow q$ ;
while  $q \neq nil$  do
   $q \leftarrow i.nextNeighbor$ ;
   $t_q \leftarrow nextP(q.no, i.t)$ ;
  if  $(q.h < i.h) \wedge (t_q < t)$  then
     $t \leftarrow t_q$ ;
     $i.next \leftarrow q$ ;
  end
end

```

Algorithm 3: Greedy algorithm

The detail of the greedy algorithm is described in Algorithm 3. The value $q.h$ is the hop distance from node q to the base station, $i.next$ is a pointer to the next sensor node that node i will send the message to, and t_c is the current system time. This approach avoids significant overhead but does introduce a loss of delivery efficiency; the exact value of this loss will be evaluated later.

6.1 Security

In this heuristic greedy algorithm, the event messages are still only replacing the maintenance messages. From the view of an adversary, the traffic in the network is as same as in the naive algorithm and the globally optimal algorithm. Thus, the compromise probability of this heuristic greedy algorithm is still $1/n$.

6.2 Delivery time

The expected delivery time for a message $T_d = E_Y(Y) \times N_h$, where $E_Y(Y)$ is the expected waiting time at one hop, and N_h is the number of hops in the routing path. Since in the greedy algorithm, the neighbor node that is in the direction to the base station and has the minimum waiting time is selected, $Y = MIN\{X_1, X_2, X_3, \dots, X_j\}$, where j is the number of neighbor nodes which are in the direction to the base station, and X denotes the waiting time between any two nodes. Lemma 1 computes the expected waiting time $E(X)$ between any two neighbor nodes. Theorem 1 computes the probability density function of Y and can be used to compute the expected waiting time at each hop, $E_Y(Y)$.

LEMMA 1. *Assume that the PRNG is uniformly distributed between $[a, b]$ and that events are uniformly distributed during the lifetime of a sensor node. Suppose random variable X is the random waiting time between any two neighbor nodes. Then its probability density function $p(x)$ is*

$$p(x) = \begin{cases} \frac{2}{a+b}, & 0 \leq x < a \\ 2\frac{x-b}{a^2-b^2}, & a \leq x \leq b, \end{cases} \quad (1)$$

and its expected waiting time is $\frac{a^2+b^2+ab}{3(a+b)}$.

PROOF. In order to obtain the expectation of X , It is necessary to find its probability density function, $p(x)$. There are two cases for X :

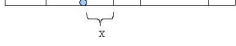


Figure 3: Waiting time in a source node

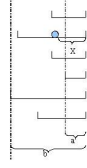


Figure 4: Alternate view of Figure 3

1. the message is waiting in a source node, or
2. the message is waiting in an intermediate node.

For the first case, we create the following model to study the distribution of X . First, we consider a line segment whose total length is the lifetime of a sensor node. This line segment is decomposable into many shorter line sub-segments, the lengths of which are random between the range of $[a, b]$. Suppose events are uniformly distributed at random onto the entire line segment. In this model, X is the length from the center of an event to the left start point of the next segment (refer to Figure 3). It is easy to verify that the X in this model is the same as the waiting time when a sensor node senses an event in our sensor network.

In order to solve this problem more intuitively, we reorganize these line sub-segments in another way (Figure 4). The minimum length of these sub-segments is a and the maximum is b . The center of an event may fall into any point of any sub-segment with the same probability. From Figure 4, we can tell that $P\{X = x | x \leq a\}$ is a fixed number, c , and that $P\{X = x | a < x \leq b\}$ is inversely proportional to x , since the lengths of the segments are uniformly distributed between $[a, b]$. Thus, the probability function $p(x)$ is

$$p(x) = \begin{cases} c & 0 \leq x < a \\ \frac{c}{a-b}(x-b) & a \leq x \leq b, \end{cases} \quad (2)$$

as shown in Figure 5.

In order to compute c , we use the following property of $p(x)$, $\int_0^b p(x)dx = 1 \Leftrightarrow \int_0^a c dx + \int_a^b \frac{c}{a-b}x - \frac{bc}{a-b} dx = 1 \Leftrightarrow c = \frac{2}{a+b}$.

Now we can derive $p(x)$ in (1). The experiment below is given to support our result. The length of the entire line segment time is set to $4 * 10^5$, and the length of the line sub-segments is within the range of $[10, 30]$, i.e. $a = 10$ and $b = 30$. We uniformly distribute 10^4 events onto the entire line segment at random and count the X s. Figure 6 shows that our experimental results match our $p(x)$ very well.

For the second case, in which the transmitting node is not the source, we create another model to study the distribution of X . In this model, there are two line segments, line 1 and line 2, which stand for two sensor nodes that are next to each other. As above these lines are composed of line sub-segments, and their lengths represent the lifetime of sensor nodes. Suppose the lengths of these line sub-segments are random between the range of $[a, b]$, and that X is the length between the ending time of each line sub-segment in line 1 and the starting time of the next line sub-segment in line 2 (see Figure 7). It is harder to determine the distribution of X analytically in this case; here, we use experiments to

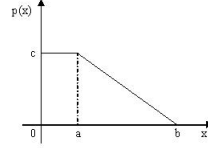


Figure 5: $p(x)$ when waiting in a source node

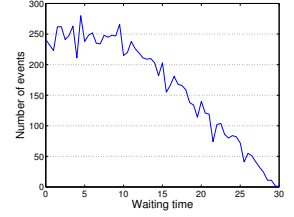


Figure 6: Waiting in a source node



Figure 7: Waiting time in an intermediate node

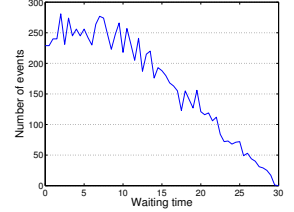


Figure 8: Waiting in an intermediate node

calculate the probability density of X (see Figure 8). These results show that the probability density of X for case 2 is almost the same as that for case 1, and thus, we use (1) as the probability density of X for case 2.

Thus, (1) is the probability density function, $p(x)$, for the entire sample space of X . The expectation of X can be derived as follows: $E(X) = \int_0^b x \cdot p(x)dx = \int_0^a \frac{2}{a+b}x dx + \int_a^b 2 \frac{x-b}{a^2-b^2} x dx = \frac{a^2+b^2+ab}{3(a+b)}$ \square

Knowing the expected waiting time between any two neighbor nodes, we can compute the expected waiting time Y for a message at one hop. Theorem 1 computes the probability density function of Y .

THEOREM 1. Assume $Y = \text{Min}\{X_1, X_2, \dots, X_j\}$, where j is the number of neighbor nodes which are in the direction to the base station. Then, the probability density function, $p_Y(y)$, is as follows,

$$p_Y(y) = \begin{cases} \frac{2j}{(a+b)^j} (a+b-2y)^{j-1}, & 0 \leq y < a \\ \frac{-2j}{(b^2-a^2)^j} (y-b)^{2j-1}, & a \leq y \leq b, \end{cases} \quad (3)$$

PROOF. We already have $p(x)$ in (1) from Lemma 1. The probability distribution function for X is $D(x) = \int_0^x p(x)dx$. So, we have

$$D(x) = \begin{cases} \frac{2}{a+b}x, & 0 \leq x < a \\ \frac{(x-b)^2}{a^2-b^2} + 1, & a \leq x \leq b \\ 1, & x > b. \end{cases} \quad (4)$$

Y follows the minimum value distribution of j X s. From probability theory, we have

$$D_Y(y) = 1 - (1 - D(y))^j \quad (5)$$

Since $p_Y(y) = D'_Y(y)$, the following equation can be derived.

$$p_Y(y) = D'_Y(y) = j(1 - D(y))^{j-1} D'(y) \quad (6)$$

From (4) and (6), we can derive (3). \square

Our goal is to obtain the expected value of waiting time Y for the greedy algorithm. We already have its probability density function $p_Y(y)$ in (3). Thus, the expected waiting time at every hop is

$$\begin{aligned} E_Y(Y) &= \int_0^b y \cdot p_Y(y) dy \\ &= \int_0^a \frac{2yj}{(a+b)^j} (a+b-2y)^{j-1} dy \\ &\quad + \int_a^b \frac{-2yj}{(b^2-a^2)^j} (y-b)^{2j-1} dy \\ &= \frac{a+b}{2(j+1)} + \frac{(b-a)^{j+1}}{2(b+a)^j(2j+1)(j+1)} \end{aligned}$$

Let $a = xb$, $E_Y(Y) = \frac{(1+x)^{j+1}(2j+1)+(1-x)^{j+1}}{2(j+1)(2j+1)(1+x)^j} \times b$. It can be easily proved that when b and j are fixed, $E_Y(Y)$ is a monotonically increasing function of x . Thus, when a is smaller, the expected waiting time will be shorter. Table 1 shows some examples of values of $E_Y(Y)$. Note that when $a = b$, our scheme is actually a greedy algorithm without using PRNGs.

Now, we can use $E_Y(Y)$ to estimate how much delivery time can be saved using our global algorithm relative to the naive solution. Since we always select the neighbor closer to the base station, the number of hops a message goes through in the greedy algorithm is the same as in the shortest path routing algorithm. Thus, the delivery time for a message originated from node i in the greedy algorithm is $T_d = E_Y(Y) \times i.h$, where $i.h$ denotes the minimum number of hops from node i to the base station. In the greedy algorithm, the PRNG is uniformly distributed between $[a, b]$, so for any sensor node, the average interval is $\frac{a+b}{2}$. To compare the performance of the greedy and naive algorithms, we let the fixed maintenance period in the naive algorithm $t_m = \frac{a+b}{2}$. In this way, they have the same traffic load.

COROLLARY 1. *When traffic loads (i.e., number of messages sent in a fixed period) are the same, the greedy solution can improve the delivery time faster than the naive solution with the percentage of 20% when $j = 2$ and $a = 0$, 33.33% when $j = 2$ and $a = b$, 42.86% when $j = 3$ and $a = 0$, and 50% when $j = 3$ and $a = b$.*

PROOF. Assume the source node is i . When $j = 2$ and $a = 0$, $t_m = \frac{b}{2}$, and $E_Y(Y) = \frac{b}{5}$. The expected delivery time in the greedy solution is $\frac{b}{5} \cdot i.h$ while the delivery time in the naive solution is $\frac{b}{4} \cdot i.h$. The delivery time in the greedy solution is faster than that in the naive solution by the following percentage: $(\frac{b}{4} \cdot i.h - \frac{b}{5} \cdot i.h) / (\frac{b}{4} \cdot i.h) = \frac{1}{5}$.

When $j = 2$ and $a = b$, $t_m = b$ and $E_Y(Y) = \frac{b}{3}$. The delivery time in the greedy solution is faster than that in the naive solution by the following percentage: $(\frac{b}{2} \cdot i.h - \frac{b}{3} \cdot i.h) / (\frac{b}{2} \cdot i.h) = \frac{1}{3}$.

Using the same method, we can get the results for other cases. \square

When the number of neighbors of a node increases, in other words, the density of sensor network increases, the greedy algorithm will have a much better performance than the naive algorithm. The reason is that it has more options on selecting the next node.

Table 1: Values of $E_Y(Y)$

	$a = 0$	$a = b$
$j = 1$	$b/3$	$b/2$
$j = 2$	$b/5$	$b/3$
$j = 3$	$b/7$	$b/4$

Table 2: P_{th} values for different random period ranges

Random period range	P_{th}
0.05-5s	0.21
2.5-5s	0.31
4-5s	0.31
2.5-3.5s	0.31

6.3 Energy cost

Each sensor node keeps sending messages according to its random maintenance period in the range $a-b$. In a time period t , the expected number of messages sent by one node is $\frac{2t}{a+b}$. Thus, the number of messages sent by all the nodes in the network is $N_m = \frac{2tn}{a+b}$. This is the same as in the global algorithm, the reason being that while the routing paths of the event messages are different, every node still sends messages according to the same frequency.

From the analysis of the delivery time of the greedy algorithm, we know that when the average frequencies of messages being sent in the network are the same, the greedy algorithm will have a shorter delivery time than the naive algorithm. However, the overall energy cost is still high, since every node will send maintenance messages after a random period. Can we save some energy by decreasing the number of maintenance messages and still have very low probability of compromise close to $1/n$? Can we find a balance between energy cost and privacy? We proposed a probabilistic algorithm in the following section to answer these questions.

7. A PROBABILISTIC ALGORITHM

The probabilistic source location protection algorithm is based on the following observation. An attacker can track the delivery of a message and its source location when he observes a series of neighboring nodes sending messages one by one that form a path from the source node to the base station. The reason that we can hide the delivery paths and the sources of messages in previous algorithms is that when every node sends messages periodically, it is impossible for the attacker to determine any link between messages in successive intervals. This leads to the following question: "is it possible to further reduce the communication overhead without disclosing the delivery path and increasing the delivery time?"

We assume that the attacker cannot determine the exact location of a message's sender by eavesdropping on a communication. This assumption is often reasonable, since there are many nodes sending out messages simultaneously, and it is not practical for the attacker to determine the exact location of the sender of every message in the network. The attacker can only do so by having many nodes monitor different areas in the network. Hence, *in addition to its own maintenance messages, a sensor node can use the maintenance messages sent by its nearby neighbors to hide its own real event messages.* This means that we can reduce communication overhead without compromising privacy and increasing delivery time.

Specifically, if the attacker always hears at least one message at any location and any given time period, it will be very difficult for him to identify the delivery path of any

event message being sent in the network. Indeed, when the attacker tries to correlate messages to identify the delivery path, there will always be many choices at every point in the path since the message observed at a given sensor node can originate from any of its neighbors. Thus, we do not need to let every node frequently send maintenance messages, especially in a high density sensor network. A small number of nodes doing so is enough to hide the delivery paths of messages as long as they make the attacker hear messages at any location at any given period of time. Thus, we need to have a method in which only some of the sensor nodes will be selected to send maintenance messages after every random period, and their combined radio ranges can cover the whole network.

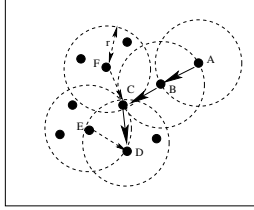


Figure 9: Using fewer nodes sending maintenance messages to hide the real routing path

Figure 9 shows an example of maintenance messages from different sensor nodes being used to hide a real event message from a particular sensor node. The delivery path of an event message is A, B, C, D . When there are many sensor nodes around in neighboring areas, we only need to let E and F send maintenance messages. At location D , the attacker cannot determine which one of E and C sent a real event message. The maintenance message from E hides the real event message at C . Similarly, at location C , the attacker cannot determine the sender of the real event message as well.

Based on the above observation, we proposed the probabilistic method. In this method, if a node originates or receives an event message, it will select the next node according to the greedy algorithm described in Section 6 and forward the message to this node at the end of the current maintenance period. If a node is not in the delivery path of any real event message, it will send a maintenance message at the end of every maintenance period with a probability of P_{th} .

```

while true do
  if receives a message from a neighbor then determine
    the next node based on greedy algorithm;
    wait for the next sending time;
    forward the message;
  else
    generate a random number  $0 < p < 1$ ;
    if  $p > P_{th}$  then wait for the next sending time;
    else
      wait for the next sending time;
      send out a maintenance message;
    end
  end
end
end

```

Algorithm 4: Probabilistic algorithm

Clearly, for a small P_{th} , the number of maintenance messages will be less, and it is possible that the maintenance messages cannot hide the real event messages very well. For a large P_{th} , there will be more maintenance messages, and the event messages are more secure. In general, we want to

find the smallest P_{th} such that the network generates minimum maintenance messages and still protects the source location of real event messages. The detailed process is described in Algorithm 4.

7.1 Security

In our greedy algorithm, we let every node send a message after every random period, which were in the range $a-b$. Our probabilistic algorithm is based on the greedy algorithm, so at any location, as long as the attacker can hear one message in any period duration b , the attacker cannot determine if that message is an event message or a maintenance message. Let P denotes the probability that at least one node sends a message during the period b in an area S_a , then $P = 1 - P_i^d$, where d is the number of nodes in area S_a , and P_i denotes the probability that node i doesn't send any messages during this period b . If $P = 1$, then the event messages are perfectly hidden by maintenance messages. A node i waits a random period $x(a \leq x \leq b)$ to send the next message according to a probability P_{th} . Thus, in a period b , the probability that node i doesn't send any messages is $P_i = \sum_{k=1}^{\infty} p_k (1 - P_{th})^k$, where p_k denotes the probability that node i has k chances to send a message in period b . From probability theory, the Poisson distribution is used to model the number of events occurring within a given time interval, so we assume that p_j complies with Poisson distribution. Since in period b , node i always has at least one chance to send a message, we assume $p_k = \frac{\lambda^{k-1}}{(k-1)!} e^{-\lambda} (k \geq 1)$. Thus,

$$\begin{aligned}
 P_i &= \sum_{k=1}^{\infty} \frac{\lambda^{k-1}}{(k-1)!} e^{-\lambda} (1 - P_{th})^k \\
 &= (1 - P_{th}) e^{-\lambda} \sum_{k=1}^{\infty} \frac{(\lambda(1 - P_{th}))^{k-1}}{(k-1)!}
 \end{aligned}$$

By the exponential function's definition $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, $P_i = (1 - P_{th}) e^{-\lambda} e^{\lambda(1 - P_{th})} = (1 - P_{th}) e^{-\lambda P_{th}}$. Thus, $P = 1 - P_i^d = 1 - (1 - P_{th})^d e^{-\lambda d P_{th}}$

The value of λ is determined by the relation between a and b . To estimate the value of λ , we use MATLAB (The MathWorks, Inc. Version 7.2.0.283) to plot the distribution of p_k and compare it to a Poisson distribution with different λ . When $a = \frac{b}{100}$, $\lambda \approx 0.7$; for $a = \frac{b}{10}$, $\lambda \approx 0.5$; for $a = \frac{b}{2}$, $\lambda \approx 0.02$; and, for $a = \frac{4b}{5}$, $\lambda \approx 0.01$.

The number of nodes d in area s depends on the density of nodes in the sensor network: $d = N \times S_a / S = \pi r^2 N / S$, where S denotes the area of the whole network. Assuming there are 20,000 nodes in an area 5,000m \times 5,000m and the radio range of a sensor node is 100m, $d \approx 25$. Thus, when $a = \frac{b}{100}$, we have $\lambda \approx 0.7$ and $P = 1 - (1 - P_{th})^{25} e^{-17.5 P_{th}}$. So, when $P_{th} = 0.1$, $P = 0.9875$, when $P_{th} = 0.21$, $P = 0.9999$. In other words, in the case of $a = \frac{b}{100}$, when every node sends a message with probability 0.21 after every random period, the probability of the event messages being discovered by an attacker is close to $1/n$.

7.2 Delivery time

The probabilistic algorithm doesn't change how the messages are routed in the greedy algorithm. Thus, the delivery time for a message is the same as in the greedy algorithm.

7.3 Energy cost

Compared to the greedy algorithm, not every node sends a message after every random period in the probabilistic algorithm. Thus, the energy cost is much less than in the greedy algorithm. In time t , the expected number of messages sent by one node is $\frac{2tP_{th}}{a+b}$. Thus, the total number of messages sent in time t is $N_m = \frac{2tnP_{th}}{a+b} < \frac{2tn}{a+b}$. The energy cost of the probabilistic algorithm is much less than the greedy algorithm.

8. EXPERIMENTS

This section describes simulation experiments on all proposed solutions (naive, global, greedy, and probabilistic) in this paper. Our results show that the global solution has the fastest delivery time; however, since the global algorithm needs every node to know the topology of the sensor network, it is not very practical in real applications. The greedy and probabilistic algorithms are more practical and they have shorter delivery times than the other methods. The probabilistic algorithm also has the lowest energy cost among all the algorithms.

In our simulations, we distribute $n = 20,000$ sensor nodes uniformly in an rectangular area $s = 5,000m \times 5,000m$. We set the radio range $r = 100m$ of each sensor node, and assume the transmission time between two sensors is $10ms$. Since the security performance is the same in all the proposed methods and the compromise probability is uniformly $1/n$, we focus on their performance in terms of delivery time and energy cost. The algorithms are compared on different random period ranges. For the probabilistic algorithm, based on the analysis in Section 7.1, we have P_{th} values as shown in Table 2, e.g. when the random period range is 2.5–5s and the probability threshold of every node sending messages is 0.31, then the probability of an event message being compromised is close to $1/n$.

8.1 Delivery time

In Figures 10(a) and (b), we compare the different algorithms when the random periods are 0.05–5s and 4–5s, respectively. The algorithms show similar performance relations in both settings, and the naive method has the longest delivery time. When the random period is 4–5s, the shortest path algorithm performs similarly to the naive algorithm, which is expected since the average waiting time of shortest path algorithm is close to the naive algorithm. The delivery time of the greedy algorithm is always shorter than that of the shortest path algorithm. When the number of hops increases, the difference between the shortest path and greedy algorithms becomes larger, and the greedy algorithm performs better. The probabilistic algorithm's delivery time is as same as the greedy algorithm's because the routing for an event message is the same. Since the global algorithm requires that every node has full knowledge of the topology of the sensor network, it is not practical in most applications. Thus, overall the greedy and probabilistic algorithms are the best choices for practical usage.

In Figure 11, to understand how the range of random period affects the performance of our algorithms, different random periods are used in the probabilistic algorithm and delivery times are then compared. The performance of range 0.05–5s is better than 2.5–5s and 4–5s. Thus, when the random period range is a – b and b is fixed, the performance is

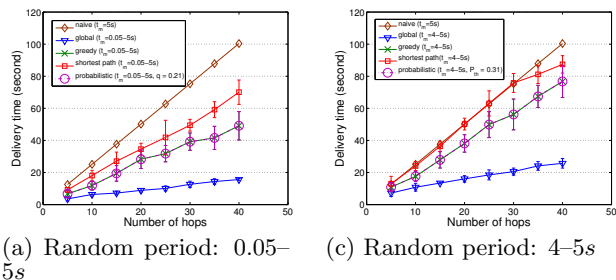


Figure 10: Comparison on delivery time of global algorithm, greedy algorithm, shortest path routing, and probabilistic algorithm

better if a is smaller, which matches our analysis in Section 6.2. The performance for 0.05–5s and 2.5–3.5s is close; this means the delivery time mainly depends on the average value of the random period $\frac{a+b}{2}$. So, if we want to decrease the delivery time, we should decrease the average value of the random period.

8.2 Energy cost

In Figure 12, the energy cost of the greedy algorithm and the probabilistic algorithm are compared when the random period is 0.05–5s. From the analysis for energy costs in previous sections, we know that the global and greedy algorithms have the same energy cost if they are using the same random periods. Thus, we only compare the greedy and probabilistic algorithms here. We use the number of messages sent in the network as the metric for energy cost. From Figure 12, we can see that the probabilistic algorithm's energy cost is significantly less than the greedy algorithm's. Thus, using the probabilistic algorithm can minimize the energy cost while having a short delivery time, and it is the most energy efficient solution overall.

In Figure 13, the energy cost of the probabilistic algorithm with different random periods is compared. The energy cost for random period 4–5s is less than for 0.05–5s and 2.5–5s; the reason is that random period 4–5s has a longer average waiting time on each node. The energy costs of 0.05–5s and 2.5–5s are similar, although 2.5–5s has a longer average waiting time. The reason for this is that the threshold probability of sending messages after every period is higher in 2.5–5s, as shown in Table 2. As a result, the energy costs of the two are similar.

9. CONCLUSIONS AND FUTURE WORK

This paper proposed four algorithms to protect source locations in sensor networks. In all algorithms, event messages are transmitted identically to maintenance messages so that a laptop-class attacker cannot identify them even though he can monitor the traffic of the entire network. Thus, we can provide almost perfect security for the locations of sources. The globally optimal algorithm has the fastest delivery time, but it requires significant storage and computation for each sensor node. The greedy algorithm greatly mitigates the requirements for storage and computation and has a shorter delivery time than the naive method. The probabilistic algorithm further decreases the energy cost by letting sensor nodes send messages probabilistically. Thus, the probabilis-

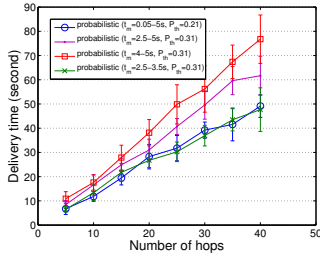


Figure 11: Comparison on delivery time of probabilistic algorithm with different random periods

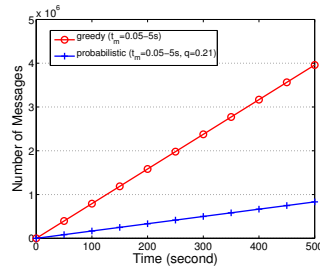


Figure 12: Comparison on energy cost of greedy algorithm and probabilistic algorithm

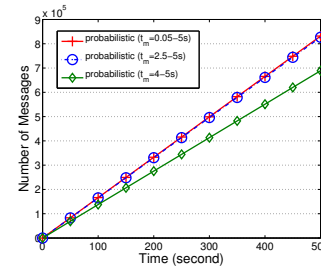


Figure 13: Comparison on energy cost of probabilistic algorithm with different random periods

tic algorithm is a good and practical solution for source location protection against laptop-class attackers.

To further decrease the delivery time for event messages, the global algorithm may be used as a base to design practical solutions since it is the fastest algorithm. A possible future direction would be to let sensor nodes work collaboratively to find a globally optimal path. Another future direction is to search for a better heuristic method to improve on the greedy algorithm's performance.

10. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation under award number CT-ISG 0716261. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

11. REFERENCES

- [1] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.
- [2] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *SecureComm '05: Proceedings of the first IEEE/CerataNet Conference on Security and Privacy in Communication Networks*, pages 113–124, 2005.
- [3] M. Gruteser and D. Grunwald. Anonymous usage of location based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the First International Conference on Mobile Systems, Applications, and Services*, 2003.
- [4] M. Gruteser and D. Grunwald. A methodological assessment of location privacy risks in wireless hotspot networks. In *SPC'03: First International Conference on Security in Pervasive Computing, Boppard, Germany*, volume 2802, pages 10–24, March 12–14 2003.
- [5] P. Kamat, W. Xu, W. Trappe, and Y. Zhang. Temporal privacy in wireless sensor networks. In *Proceedings of 27th IEEE International Conference on Distributed Computing Systems (ICDCS 2007), June 25–29, 2007, Toronto, Ontario, Canada*, 2007.
- [6] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 599–608, 2005.
- [7] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.
- [8] J. Kong and X. Hong. Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 291–302, 2003.
- [9] K. Mehta, D. Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *ICNP 2007: Proceedings of the IEEE International Conference on Network Protocols*, pages 314–323, 2007.
- [10] Y. Ouyang, Z. Le, G. Chen, J. Ford, and F. Makedon. Entrapping adversaries for source protection in sensor networks. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 23–34, 2006.
- [11] C. Ozturk, Y. Zhang, and W. Trappe. Source-location privacy in energy-constrained sensor network routing. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 88–93, 2004.
- [12] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.
- [13] M. Shao, Y. Yang, S. Zhu, and G. Cao. Towards statistically strong source anonymity for sensor networks. In *INFOCOM 2008: Proceedings of the 27th IEEE Conference on Computer Communications*, pages 51–55, 2008.
- [14] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao. Towards event source unobservability with minimum network traffic in sensor networks. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 77–88, 2008.