

Entrapping Adversaries for Source Protection in Sensor Networks *

Yi Ouyang^{1,3} Zhengyi Le^{1,3} Guanling Chen^{2,3} James Ford^{1,3} Fillia Makedon^{1,3}

¹Computer Science Department
Dartmouth College

{ouyang, zyle, jford, makedon}@cs.dartmouth.edu

²Computer Science Department
UMass Lowell

glchen@cs.uml.edu

³Institute for Security Technology Studies (ISTS), Dartmouth College

Abstract

Sensor networks are used in a variety of application areas for diverse problems from habitat monitoring to military tracking. Whenever they are used to monitor sensitive objects, the privacy of monitored objects' locations becomes an important concern. When a sensor reports a monitored object by sending a series of messages through the sensor network, the route these messages take in the network creates a trail leading back to their source. By eavesdropping on communications, an attacker may be able to move from node to node to follow this trail. Several approaches aimed at discouraging this kind of eavesdropping have been proposed, including mechanisms for constructing "phantom" routes and approaches that insert fake sources as background noise. A problem with existing approaches is that message latencies become larger and energy costs become higher as a result of introducing protections for the privacy of a source location. This paper proposes a new cyclic entrapment method (CEM) that protects source locations in sensor networks while adding a comparatively low cost in terms of additional message latency and energy.

*This research program is a part of the Institute for Security Technology Studies, supported under Award number 2000-DT-CX-K001 from the U.S. Department of Homeland Security, Science and Technology Directorate and Grant number 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance, a component of the Office of Justice Programs. Points of view or opinions in this document are those of the author(s) and do not represent the official position or policies of the U.S. Department of Homeland Security, the Science and Technology Directorate, or the United States Department of Justice.

1. Introduction

Sensor networks are used in many realtime applications for collecting information from monitored environments and objects, such as moving vehicle tracking, battlefield reconnaissance, and habitat monitoring. Following the increasingly wide deployment of sensor networks, privacy concerns have emerged as the main obstacle to success.

Privacy in sensor networks can be categorized into two classes: content-oriented privacy and contextual privacy [14]. *Content-oriented privacy* concerns the ability of adversaries to learn the content of transmissions in sensor networks [14]. If the data collected in sensor networks are used in statistical applications, there is no legitimate need to know any information about individuals monitored in the sensor network. However, even if content privacy is assured, adversaries may still be able to extract individual information. *Contextual privacy* considers the ability of adversaries to infer information from observations of sensors and communications without access to the content of messages. Sensors' behaviors, such as communication patterns and a message's routing path, can give adversaries clues for deducing information about the network, such as the location of a message's source or the location of the base station. This paper focuses on the problem of protecting the location of a source in sensor networks. The goal of our research is to minimize the chance of an adversary finding the source in a short time—in other words, we want to extend the time for an adversary to find a source, which is called the *safety period* [14].

Source location protection in sensor networks becomes a very important problem when a sensor network is used in monitoring valuable assets or the source is a sensitive object. For example, on a battlefield sensors can detect the movements of soldiers and report them to the headquarters;

an attacker may then be able to use intercepted sensor network communications to determine the exact location of opposing soldiers through traffic analysis. Advanced routing methods with contextual privacy protection for sources are needed for applications like this one. For different kinds of attackers, the methods used to keep sources safe may need to be different.

Attackers can be categorized into two general groups: mote-class attackers and laptop-class attackers [11]. *Mote-class attackers* have capabilities similar to those of a sensor node. The radius they can eavesdrop may be the same as, or perhaps up to twice as great as, a sensor node's communication radius. At any given time a mote-class attacker can thus only detect communication between a limited number of nodes. *Laptop-class attackers* have much stronger computational capabilities and longer radio range than mote-class attackers. If they are equipped with hardware powerful enough, the radius of their monitoring area could be equal to the entire network's size. Thus, we assume that laptop-class attackers could eavesdrop on all the communications in a sensor network. In this paper, we focus on mote-class attackers.

There has been some research on the source location protection problem. A Panda-Hunter model is described in [14]. In this model, a large array of sensor nodes are deployed to monitor the movement of pandas. When a panda is moving in the field, it will be detected by a nearby sensor, which will send a message back to the base station. If the panda remains nearby, the same sensor will keep sending messages back to the base station. If the panda moves, any sensors within range to its new location will send messages back to the base station. In a sensor network, each node has a very limited range due to size and power constraints. Thus, in order to transmit messages to a "base station" that integrates sensor data, each sensor must rely on its neighbors to retransmit its messages. A message generally traverses multiple sensor nodes along a predefined path to the base station. The Panda-Hunter model assumes there is a hunter who can eavesdrop on the communications between sensors, which are encrypted. Although the hunter can not analyze the content of messages to determine the source location, she or he can still find the source location by analyzing the routing path of the messages. If the hunter can observe only one sensor node, observing the communications between this node and its neighbors, then the hunter must guess which neighbor node is the most likely to be on the path from the source and move to it to continue monitoring. Using this "hop-by-hop" backtracking method, the hunter will ultimately find a generally immobile source. Routing strategies may be employed to defend against a hop-to-hop backtracking method, but will introduce varying performance penalties.

Kamat *et al.* [10] devised the phantom routing method

to increase the time needed by adversaries to find a source location. In this method, a message first is sent on a random walk through neighbor nodes, after which it is either broadcast or sent on the shortest path to the base station from the last node reached on the random walk. Because each message potentially traverses a different random walk path prior to the broadcast or being sent on the shortest path, adversaries will be confused by the appearance of different origination points, and as a result the true source will be hidden. Because every message needs to go through a random walk first, the routing path used may not be the shortest path; thus, the delivery time for each message is longer than it would be routing along the shortest path. However, the tradeoff is that shortest path routing has provably bad performance in protecting the location of a source [10].

We propose a new concept, the cyclic entrapment method (CEM), to preserve the performance advantage of shortest path routing while also protecting the location of a source. In this approach, several loops are generated after the deployment of the sensor network and before sources send any messages to the base station. Each loop consists of several sensor nodes. When a message is being routed along a path from the source to the base station and it encounters one of these pre-configured loops, the encountered loop will be activated and will begin cycling fake messages around the loop. When a mote-class attacker arrives at this spot, she will be unable to distinguish incoming messages correspond to to the source node she is seeking from those generated by the loop, and will be forced to choose her next node randomly. By ensuring that a message's path is likely to cross multiple loops, we can increase the expected time required for an adversary to locate a source node.

The rest of this paper is organized as follows. Section 2 discusses related work in more detail. Section 3 describes formal network models and threat models. Section 4 discusses CEM and its performance. Section 5 analyzes the privacy properties of CEM. Section 6 describes experiments and results applying CEM and analyzes its performance. Section 7 concludes this paper.

2. Related Work

Researchers are increasingly raising concerns about privacy issues in sensor networks, and also in related areas such as pervasive computing, data mining, and location-based services. Ozturk *et al.* [14] introduced the Panda-Hunter model to formalize the source location problem in sensor networks and also proposed the phantom flooding approach. Kamat *et al.* [10] extended the work of [14] and proposed phantom routing techniques based on both flooding and single-path routing. Deng *et al.* [3, 4] investigated several countermeasures against traffic analysis that aimed to protect the location of a base station. Chaum's mixing

approach [2] was shown to provide anonymous connections that protected against traffic analysis and were useful in an onion routing mechanism [15]. Kong *et al.* [12] proposed an anonymous on-demand routing protocol for mobile ad hoc networks and addressed two problems: route anonymity and location privacy.

Duri *et al.* [5] proposed a framework for telematics data protection that enables users to define flexible data models. Gruteser *et al.* [7] presented a methodology for identifying and assessing location privacy risks in mobile computing and proposed to use anonymity-based mechanisms to address them. Gruteser *et al.* introduced an adaptive algorithm to adjust the resolution of information in order to protect individual privacy when supplying location-based services [6, 8].

Secure routing in sensor networks is another related research area. Karlof *et al.* [11] showed how attacks in ad-hoc networks can be adapted as attacks against sensor networks and suggested countermeasures and design considerations. Hu *et al.* [9] presented Ariadne, which prevents attackers from tampering with uncompromised routes consisting of uncompromised nodes. Several other secure routing protocols have been proposed, such as [1], [13], and [16].

3. Sensor Network and Threat Models

This section describes the formal models and assumptions that will be used in this paper.

3.1. Sensor networks models

Sensor networks consist of many sensor nodes deployed in an area that needs to be monitored. Every sensor can send messages to neighboring nodes that are within its limited radio range. Figure 1 shows an example of a sensor network.

Sensors collect information from the environment and send messages to a base station, which is a node with greater computational capabilities that functions as the interface between the sensor network and applications. The base station processes the data received from all sensors and answers queries from applications. When an object appears at a location monitored by a sensor node, the node will send a message destined for the base station. The routing path of a message is determined by the routing strategies adopted by the sensor networks, such as shortest path routing and broadcast routing. The messages will continue to be sent periodically while the object is present, and will stop when the object leaves the sensor's monitoring area. When the object moves to a new location, it may trigger another sensor node to send messages. The object may be detected by multiple nodes or may be out of range of all the nodes. Every node will send a message as soon as it detects the object.

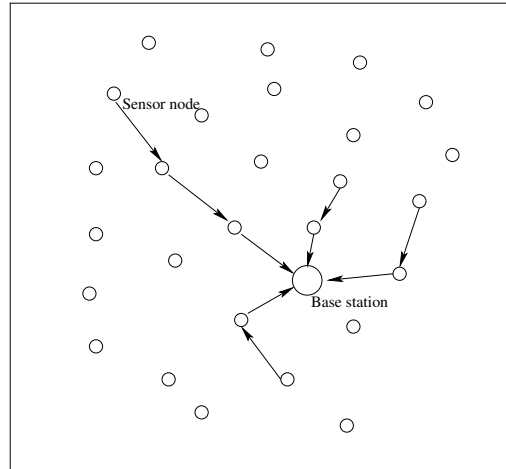


Figure 1. Sensor Networks

3.2. Threat models

Adversaries can eavesdrop on the local traffic between nearby sensor nodes to try to determine the source node's location. All messages are assumed to be encrypted with secret keys, and thus attackers cannot determine the content of messages even if they intercept or eavesdrop on communications. However, an attacker may use RF localization techniques to trace back hop-by-hop to a source's location [10]. We assume that an attacker can determine the sender of each message it intercepts, and that adversaries have the capability of storing an arbitrary number of messages. We assume the adversaries use the *patient adversary* model, as described in [10] and shown in Algorithm 1.

Algorithm 1 Patient adversary

- 1: Start from base station.
 - 2: **while** Does not arrive at the source location **do**
 - 3: Eavesdrops on the communications between the current node and its neighboring nodes
 - 4: Receives a new message M_{new}
 - 5: Determines the sender S_{new} of M_{new}
 - 6: Moves to the new location S_{new}
 - 7: **end while**
-

A patient adversary always selects the latest message's sender S_{new} as the next node to move to. Assuming a sensor node will not forward the same message twice and that the sensor network is using broadcast routing, S_{new} should be the nearest node to the source's location among all the neighbors of the current node. If this were not the case, and another node S' were nearer to the source than S_{new} , a message newer than M_{new} should have already been passed along by S' and arrived at the current node prior to M_{new} .

Thus, under the assumptions of this model, moving to S_{new} every time one detects a message will move one closer and closer to the source, eventually reaching it.

3.3. Countermeasures

To protect the privacy of a source location, the sensor network needs to prevent an attacker from finding the path of messages from the source to the base station. Two different approaches have been suggested, both of which fool adversaries into making an incorrect decision when selecting the next node to move to.

First, a sensor network can route the same source’s messages on different paths. In other words, each message sent through the network is routed on a different *logical topology* than the others. Figure 2 is an example of two different logical topologies for a source. We can formally define the notion of a *logical topology* as follows in order to describe the paths of messages:

Definition A *logical topology* for a sensor network is a tree covering all the nodes, where the root is the source and the base station is one of the leaves.

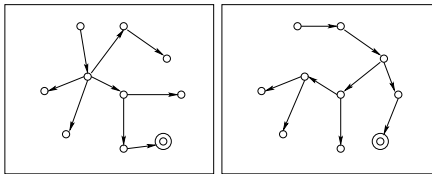


Figure 2. Two different logical topologies

For practical purposes, some logical topologies may be unusable because the distance between linked nodes will be too high for communication. At a node S , the adversary selects the next node depending on which node sent out the most recent message. Because the logical topology for each message can be different, the adversary may be drawn in different directions by different messages. The aim is to cause this to happen even when the messages have the same source node. Thus, the adversary will spend time on some nodes which are not on the correct shortest path to the source, and the time required to find the source will be prolonged. One example of this approach is phantom routing using a single path [10].

An alternative to changing the path used by messages each time is for the sensor network to create and send fake messages specifically to mislead adversaries. In this paper, we focus on this approach. Work in the literature has proposed several different ways to send these fake messages. The sensor network can generate several fake sources that keep sending fake messages throughout the whole network.

However, if the fake sources’ locations are fixed, they can be located and recorded by adversaries—and for this reason one fake source can’t be expected to confuse adversaries for a long time. Increasing the number of fake sources or making them dynamic may solve this problem, but this means more overhead in terms of energy cost and node capabilities. The new concept of “cyclic entrapment” proposed in this paper provides a new means for sending fake messages with lower energy cost and also improved privacy.

4. The Cyclic Entrapment Method

This section describes the details of the cyclic entrapment method. Using CEM, loops are configured immediately following the deployment of a sensor network. Each loop originates with a sensor node and consists of an ordered sequence of nodes that are sequentially in range. By traversing the members of the loop, a message can cycle indefinitely along the loop. Attackers who are enticed to trace a loop message back through its loop are thus “trapped”, at least until they discover they are in a cycle. Traffic in these loops is triggered by the passage of sensor messages, and can coexist with these true messages. When an adversary is trying to analyze the traffic and trace the message’s path back to the source, if it encounters a node that is a common node of both a loop and a true path, it will need to select a direction to go on, and in doing so it may make a wrong decision and be drawn into this loop. Because there is no way for an attacker to determine that they have left the correct path until they complete a cycle, the expected time for an adversary to find the correct path is increased.

4.1. Loop generation

To use CEM, the critical step is to create loops in sensor networks. We propose a simple method that can be used immediately after the deployment of a sensor network. After the sensor network is deployed, every node will decide whether it will generate a loop with a probability p . A node A will initiate a message to create a loop with this probability; otherwise, it will do nothing. If A decides to generate a loop, it will first detect its neighbors and collect all of its neighbors’ IDs. For instance, let nodes B , C , and D be A ’s neighbors in a sensor network. Suppose the sensor network uses routing protocol R . A randomly selects two nodes B and C from its neighbors, creates a message with destination “ B ”, and then sends this to C . The message is forced to go through a random walk first, and then go to the destination B using the original routing protocol R . The message includes a value that specifies the number of hops for the random walk, and it also records the IDs of all nodes it has been to. The format of this message is shown in Figure 3. Messages of this type are called *REQ_LOOP* messages.

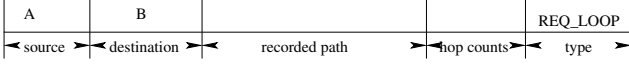


Figure 3. The format of a message with type *REQ_LOOP*

When this message reaches its destination *B*, *B* can extract the path information from this message. From the path extracted from the message, *B* will know the IDs of an ordered list of nodes that can form a loop. *B* then sends *A* a new message, which it assigns a special routing path that includes all the nodes in the loop. After this second message goes through all the nodes in the loop, the loop has been created. Messages of this type are called *CON_LOOP* messages. The format of this second message is shown in Figure 4.

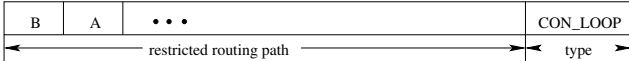


Figure 4. The format of a message with type *CON_LOOP*

A single node could be in five situations as follows. 1. It generates a loop itself. 2. It is selected by another node as a destination when creating a loop. 3. It is in the path of one loop. 4. It is in the paths of multiple loops. 5. It is not in any loop. Note that a node may generate a loop while one of its neighbors is also generating a loop; loop generation by two neighboring nodes is independent, and their activities will not interfere with each other, even if they generate loops using some of the same nodes. The algorithm for each node's actions is shown in Algorithm 2; here, *p* is a pre-defined threshold.

4.2. Loop activation

It is not necessary to let all the loops in the sensor network keep working all the time. To save energy, the sensor network only needs to activate loops that overlap with the routing paths of current messages. When a source begins to send messages, it will select a path to the base station using some specific routing mechanism. Here we use a single path routing mechanism as an illustration. After a source selects a path to the base station, the messages from this source will go through the sequence of nodes on the path. Each node receiving the message will check if it is on a loop as it forwards the message to the next node on the path; if it is on a loop, it will activate it by sending a fake message along the loop. The node that activated a loop is called the

activation node of that loop. The activation node of a loop is not necessary the initiator of this loop: every node in a loop can activate it. If an activation node is in multiple loops, all these loops will be activated (to avoid severe throughput degradation of a node that have many loops to activate, it can optionally select only one or some of the loops to activate). If the routing path hits multiple nodes in the same loop, these nodes will activate the same loop multiple times, thus increasing the probability of keeping the adversary in the loop.

Since the source will keep sending messages along the chosen routing path, an activation node will randomly generate a new fake message and send it along a loop each time it receives a new authentic message from the source. In this way, the fake loop messages will have the same frequency as the real source messages. Fake messages will be created to have the same length as the trigger message, so that since all messages are encrypted there is no way for an adversary to distinguish fake and true messages. Also, once a loop is activated its activation node may be set to continue creating and sending new randomly generate messages to the next node on the loop at the same frequency as the real source is sending messages for a predetermined amount of time. Figure 5 shows an example of the cyclic entrapment method. Activation node *A*'s actions are detailed in Algorithm 3.

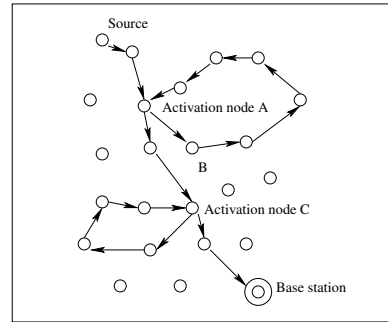


Figure 5. Cyclic entrapment method

4.3. Loop deactivation

By default, sensor nodes that are in activated loops will keep sending messages along the loops. Thus, it seems clear that loops need to be deactivated when they are no longer useful; otherwise, it is possible that sensor nodes in loops will be depleted much faster than other nodes. In our method, this deactivation is handled automatically without any special mechanism.

A loop should be deactivated when the triggering source's messages cease going through any loop nodes. If messages from the source no longer go through the activation node that triggered a loop, then this node is no longer

Algorithm 2 Algorithm for a node creating a loop

```
1: Generates a random number  $q$ 
2: if  $q < p$  then
3:   Sends messages to all of its neighbors and gets
   replies;
4:   if the number of neighbors  $\geq 2$  then
5:     Selects two neighbors  $B$  and  $C$ ;
6:     Generates a message with type  $REQ\_LOOP$  and
     destination  $B$ ;
7:     Sends the message to  $C$ ;
8:   end if
9: end if
10: while Receives a message  $M$  from a neighbor do
11:   if  $M$  has type  $REQ\_LOOP$  then
12:     if  $M$ 's destination == current node's ID then
13:       Gets the source of this message  $A$ ;
14:       Gets the routing path  $L$  from this message,  $L$  is
       a loop;
15:       Sends  $A$  a message with type  $CON\_LOOP$ 
       and this message includes  $L$ ;
16:     else
17:       Adds its ID to the path included in this message;
18:       if  $M$ 's hopcount  $> 0$  then
19:         Forwards this message to a randomly selected
         neighbor other than the node sending the
         message;
20:       else
21:         Sends this message to its destination using
         the original routing protocol of the network;
22:       end if
23:     end if
24:   else
25:     if  $M$  has type  $CON\_LOOP$  then
26:       Extracts the loop information  $L$  from the mes-
       sage and stores it locally.
27:       Forwards this message to the next node accord-
       ing to the routing path assigned to it;
28:     end if
29:   end if
30: end while
```

Algorithm 3 Algorithm for an activation node A

```
1: while Receive a message from one of the neighbors do
2:   if A is a member node of a loop then
3:     Transmit a randomly generated message to the
     next node  $B$  in the loop.
4:   end if
5:   Forward the message to the next node in the path
6: end while
```

on the routing path, and the adversary will not trace back to this activation node; thus, the loop activated by this node is no longer required and it can be canceled. In Algorithm 3, if an activation node A doesn't continue to receive messages from its neighbors, it will not generate any new fake messages. If there is only one activation node A in the loop, the messages transmitted in the loop are all generated by the activation node A . When A stops sending fake messages, there will be no messages transmitted in the loop. Thus, the loop is deactivated automatically. If there are multiple activation nodes in the loop, when A stops sending fake messages, other activation nodes in the loop will continue sending messages to maintain the loop until they don't receive any real messages. Thus, a loop is deactivated only when all the activation nodes in the loop cancel their activation.

4.4. Probabilistic loop activation

To further mitigate the problem of extra energy costs for sensor nodes in activated loops, we consider a modification of the basic cyclic entrapment method aimed at saving energy. In the network initialization phase, every node gets a random number before the deployment in the range $1-k$. When a sensor node sends messages, it will generate a range and include it in the messages, e.g. 31–60. A node in the path of a message will activate loops using Algorithm 3 only if it has a number in the range specified by the message. Because the numbers assigned to sensor nodes are uniformly generated, the expected number of nodes that activate loops will be decreased by a selected fraction: in the case of the preceding example range, and a k of 100, roughly 1/3 of the number in the standard cyclic entrapment method. Note that in addition to lowering power consumption, this probabilistic approach may also make it harder for an attacker to discover and map all the loops in a network. The modified procedure for an activation node is shown in Algorithm 4.

Algorithm 4 Modified procedure for activation node A

```
1: while Receive a message from a neighbor do
2:   if A is a member node of a loop then
3:     Check the range  $R$  included in the message
4:     if  $A.rand \in R$  then
5:       Transmit a randomly generated message to the
       next node  $B$  in the loop.
6:     end if
7:   end if
8:   Forward the message to the next node in the path
9: end while
```

5. Safety Period Analysis

Research on the problem of source protection in sensor networks has not yet produced a way to absolutely prevent an adversary from finding a source; instead the protection a method provides is typically measured and compared according to the expected safety period (the time for an adversary to reach the source). In this section, we calculate the expected safety period that results from using CEM in terms of the expected number of hops an adversary must make from the base station.

The nodes on the path of a message from a source to the base station may or may not be members of loops. Those nodes that are in loops are distraction locations for an adversary—an adversary reaching one of these nodes will be confronted with a choice, one where it cannot distinguish the correct choice from the false one(s) and must guess. Let S denotes the set of s nodes that are in loops in a message's path. The probability of the adversary selecting correctly is $\frac{1}{k_i+1}$, where k_i is the number of loops activated by node i of the s nodes in S . Thus, the probability of an adversary always selecting the correct direction to the source is

$$P_{correct} = \prod_S \frac{1}{k_i + 1} \leq \frac{1}{2^s}.$$

If the adversary selects a direction which is a loop, then after it goes through all the nodes in the loop it will encounter the activation node that it started with again. Assume the adversary has enough capability to store any information it wishes to when it travels in a sensor network. When the adversary visits this node a second time and faces the same situation as before, it can realize that it has been in a loop and the direction it selected last time is incorrect. This time it can select a new direction among the remaining choices. Assume the length of one loop is l nodes (and thus l hops for an adversary to traverse), and let h_i denote the number of additional hops an adversary traverses at an activation node i . Then the expected additional number of hops for an adversary to successfully go through an activation node and emerge in the right direction is

$$\begin{aligned} E(h_i) &= \sum_{1 \leq j \leq k_i} p_j \times l \times (j - 1) \\ &= \sum_{1 \leq j \leq k_i} \left(\frac{1}{k_i + 1 - (j - 1)} \times l \times (j - 1) \right) \\ &\quad \times \prod_{1 \leq x \leq j-1} \frac{k_i - (x - 1)}{k_i + 1 - (x - 1)} \\ &= \sum_{1 \leq j \leq k_i} \frac{(j - 1)l}{k_i + 1}, \end{aligned}$$

where p_j is the probability of an adversary not selecting the correct direction until its j th visit (meaning that the previous $j - 1$ visits resulted in being delayed along loops), and l is the number of steps an adversary needs to go through for each loop¹. Thus the expected steps for an adversary going from the base station to the source is

$$\begin{aligned} E(N_{ad}) &= n + \sum_{1 \leq i \leq s} E(h_i) \\ &= n + \sum_{1 \leq i \leq s} \sum_{1 \leq j \leq k_i} \frac{(j - 1)l}{k_i + 1}, \end{aligned}$$

where n is the length of the real message's path and s is the number of nodes that are in loops in this path. If we let $k_i = 1$ for all i , which means there is exactly one loop activated by each node, then $E(N_{ad}) = n + \sum_{1 \leq i \leq s} \frac{l}{2}$.

In estimating the value of s , the probability of activating a loop is vital. We will analyze the relation between the probability p of one node initiating a loop and how many loops a path will go through. If the probability p is larger, the number of loops in the whole sensor network will become larger, and thus the number of loops activated by a message's path will be larger.

First, we need to compute the probability of one node being in a loop. Note that here we refer to the process of loop creation and not loop activation. A node can be a loop initiator (which we can refer to as "the first node" of the loop), and it also can be the second node, the third node, and so on. Assume P denotes the probability of one node being in a loop, and P_i denotes the probability of a node being the i th node in a loop. Thus, the probability P of a node being in a loop is $P_1 + P_2 + P_3 + \dots$, where P_i is the probability of the node being the i th node. Here, $P_1 = p$, as defined in Section 4.1 on loop creation.

For the sake of simplicity, we assume every node has m neighbors. Assume A and B are neighbors. The probability of A being a loop initiator is p . The probability P_{AB} of A selecting B as a second node in the loop is $p \times \frac{1}{m}$. Because B has m neighbors, each other neighbor can also select B as a second node in their loops if they are initiators. Thus the probability P_2 that B is not an initiator $(1 - p)$ but is the second node in a loop is $(1 - p) \times \binom{m}{1} \times P_{AB} = (1 - p) \times m \times p \times \frac{1}{m} = (1 - p)p$. For similar reasons, the probability P_3 , that a node is the third node in a loop, is $(1 - p)^2 p$. Assume that the average length of a loop in the sensor network is l ; then, $P = P_1 + P_2 + P_3 + \dots + P_l = p + (1 - p)p + (1 - p)^2 p + \dots + (1 - p)^{l-1} p = 1 - (1 - p)^l$. If a message's path length is n , the expected number of activation nodes that are in this path is $s = n \times (1 - (1 - p)^l)$.

From the analysis above, we can conclude that when each activation node on the path activates at most one loop

¹Assuming that a visitor remembers previous exits from a node, the probability at each departure will be $\frac{1}{\text{remaining possibilities}}$.

(i.e. ignoring the possibility of multiple loops), the expected number of steps for an adversary is

$$\begin{aligned} E(N_{ad}) &= n + \frac{n(1 - (1 - p)^l)l}{2} \\ &= \frac{n(2 + l - l(1 - p)^l)}{2}. \end{aligned}$$

For different p and different l , the expected value of the safety period is shown in Figure 6.

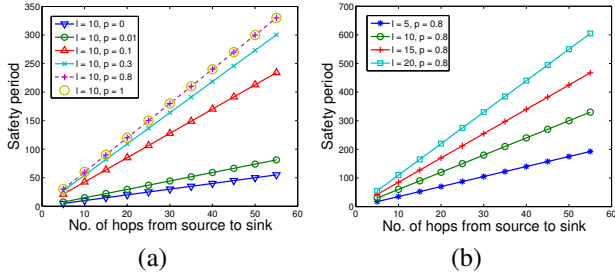


Figure 6. Theoretical safety period for different loop initiating probability p and different loop length l

6. Experiments and Performance Analysis

In this section, we will compare the performance of CEM with other existing methods on different aspects including safety period, delivery quality, energy consumption, and message latency. The other source location protection methods tested are the phantom routing method, the fake source scheme, probabilistic forwarding, and some basic routing mechanisms.

6.1. Experiment environment

This section describes the simulation environment used to evaluate the performance of CEM. The simulation consists of 5000 sensor nodes, the locations of which are randomly generated. All the nodes are deployed in a $5000\text{m} \times 5000\text{m}$ area. Assume the radio range of a node is 100m. For the initialization of the sensor network, there are three steps. First, each node discovers all the neighbors in its radio range. Second, the nodes in the network will generate loops by sending request messages to each other using Algorithm 2. Third, the base station will send a broadcast message to the whole network. After nodes receive copies of this broadcast message and compare the route information attached to the messages they receive, they will know the shortest path to the base station from their location.

6.2. Performance analysis

6.2.1 Safety period

The safety period, measured as the number of steps required to reach the source from the base station, can be used to evaluate and compare methods. We implemented phantom flooding, probabilistic flooding, and CEM in our simulation environment and compare the safety period calculated for each method in Figure 7. In the simulation, the length of the random walk in phantom flooding is 15. Larger random walk length may improve the phantom flooding method’s safety period, but the message latency will be higher. For every method, 30 random nodes for each hop are selected and the reported value is the average of their safety periods. In this figure, “ $l = 10, p = 0.8$ ” means that the minimum length of loops is 10, and the probability of one node to initiate a loop is 0.8. Depending on the parameters used, CEM can get a better safety period than the other two methods. Figure 8 shows that the choice of parameters for CEM is critical for its safety period. It is interesting that the result of “ $l = 10, p = 0.8$ ” is better than “ $l = 20, p = 0.8$ ”. An explanation for this is that longer loops are more difficult to generate, and thus the number of loops generated is smaller when the required length is longer. Figure 9 shows that when only the probability of one node to initiate a loop is changed, and the minimum loop length and hops are fixed, the safety period is not monotonic as expected from the analysis of Section 5 and Figure 4. An explanation for this is that in the simulation, we only activate one loop at every activation node. When more nodes initiate loops, there are more short loops generated and they more likely to be activated instead of longer loops.

6.2.2 Delivery reliability

The flooding and shortest path methods can achieve a 100% delivery ratio; probabilistic forwarding, on the other hand, does not necessarily deliver every message to the base station. CEM can be combined with any routing protocol, and in this paper we use the shortest path routing scheme, where messages follow a shortest path from the source to the base station, and loops are activated along the path. Since CEM doesn’t affect the delivery quality of the routing protocol, its delivery ratio is 100% when using this method.

6.2.3 Energy consumption

Energy consumption is a significant concern in sensor networks research. There is a tradeoff between energy consumption and protection for source location: if every sensor node were to send messages all the time, there would be no way for an adversary to determine which ones are reporting real events. Although the location of the source is

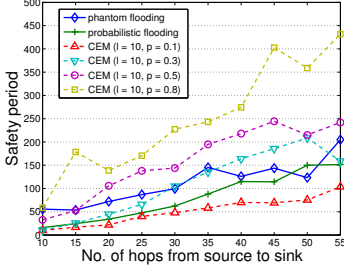


Figure 7. Safety period of phantom flooding, probabilistic flooding, CEM

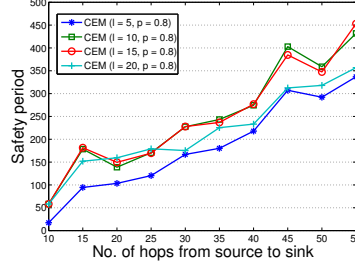


Figure 8. Safety period of CEM with different minimum loop length (l = 5, 10, 15, 20)

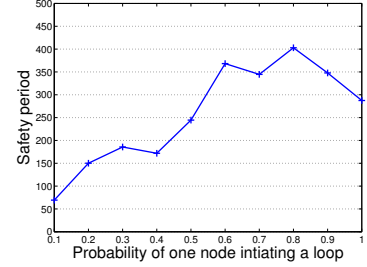


Figure 9. Safety period of CEM with different loop initiating probability, l = 10, hops = 45

perfectly protected in this situation, the energy cost is very high and the lifetime of this kind of a sensor network (using current technology) would be very short. In contrast, if messages always follow a fixed single path to the base station, the energy cost is the minimum possible but source locations are relatively unprotected, since an adversary can trace back along this fixed path very easily. In this section, we will show that CEM adds a modest energy increase to gain a relatively high degree of additional protection for source locations.

For any routing protocols using broadcast as a component, the energy cost will be high. Regardless of which node a broadcast message originates from, it would be transmitted through the entire sensor network until every node in the network including the base station receives it. Because of the broadcast nature of wireless communications, all the nodes in the vicinity of a sender receive each packet it broadcasts. Thus, every node in the network will receive a message at least once and send it once. The total energy cost of broadcasting is at least $(E_{send} + E_{receive})N$, where E_{send} is the energy needed for one node to send a message, $E_{receive}$ is the energy needed for one node to receive a message, and N is the number of nodes in the sensor network. Note that if the sensor network is dense, a node located in the vicinity of multiple neighbors may actually receive the same message multiple times. Since E_{send} is typically greater than $E_{receive}$, we do not consider this extra cost here. The broadcast path of a message can be viewed as a spanning tree on all the sensor nodes.

In the phantom flooding protocol, a message will go through a random walk before it is broadcast to the network. The nodes in the random walk path will not forward this broadcast message again since they have already processed it, and the broadcast will let every other node in the network including the base station receive the message. Thus, the energy cost of phantom flooding is the same as that of the flooding protocol [10], which is $(E_{send} + E_{receive})N$

(again ignoring the possibility of multiple receipts). In the phantom single path protocol, a message will be sent on a random walk and then will select the shortest path to the base station; thus, the energy cost is $(E_{send} + E_{receive}) \times w$, where w is the number of nodes on the path. The length of the path w is at most $2l_{walk} + l_{shortest}$, where l_{walk} denotes the length of the random walk, and $l_{shortest}$ denotes the length of the shortest path between the source and the base station.

In CEM, when a message is sent to the base station, it will activate loops along the path. An activation node will generate a fake message and send it along the loop while forwarding the original message to the next node on the shortest path to the base station. While loops are designed to continue to remain active while messages continue to arrive from the source (Section 4.3), here we assume a single source message for comparison purposes. If a message activates k loops along the path, and the maximum perimeter of the loops is l_{loop} , then the transmission caused by a message is less than $l_{shortest} + k \cdot l_{loop}$. From the analysis in Section 5, we know that the expected number of loops activated by a message is $n \times (1 - (1-p)^{l_{loop}})$, where n is the length of the message's path and p is the probability of one node creating a loop during network initialization. Here the path of a message is the shortest path, so $n = l_{shortest}$. Thus, the number of transmissions using CEM is $l_{shortest} + l_{shortest}l_{loop}(1 - (1-p)^{l_{loop}}) = l_{shortest}(1 + l_{loop}(1 - (1-p)^{l_{loop}})) < l_{shortest}(1 + l_{loop})$. The energy cost of CEM thus must be less than $l_{shortest}(1 + l_{loop})(E_{send} + E_{receive})$. With an appropriate value of l_{loop} , the energy cost of CEM will be less than both the phantom flooding protocol and the broadcast protocol.

From the analysis above, we can see that as the length of loops increases, the energy cost of CEM will be higher. Thus, loop length is an important consideration for saving energy as well as for protecting source locations.

6.2.4 Message latency

The message latency is the time between a message's departure from the source and its arrival on the base station. In different routing protocols, messages will go through different paths to the base station, and thus will have different message latencies. CEM can be combined with other routing protocols, such as shortest path routing. The message latency will depend on the performance of the routing protocol it is combined with. A strong advantage of CEM is that it can supply good protection for source locations while still retaining a minimal message latency.

7. Summary and Conclusions

In this paper, we have studied the problem of protecting source locations in sensor networks. Whenever sensor networks are used in sensitive applications such as monitoring and battlefield reconnaissance, the protection of source locations must be addressed. We proposed a new approach, cyclic entrapment, to lead adversaries into traffic loops in a sensor network. A comparison of CEM with existing methods shows that it can get a comparable source location protection while adding a comparatively low cost in terms of message latency and energy usage. A significant advantage of CEM over existing techniques is that it can protect a source's location while allowing for an optimal routing time for messages from that source. As future work, we will investigate the impact of source mobility, multiple sources, and message rate from the source on this problem and our approach.

References

- [1] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. In *Proceedings of IEEE Global Telecommunications Conference*, 2003.
- [2] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.
- [3] J. Deng, R. Han, and S. Mishra. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *DSN'04: International Conference on Dependable Systems and Networks*, 2004.
- [4] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *SecureComm'05: Proceedings of 1st IEEE Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2005.
- [5] S. Duri, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J.-M. Tang. Framework for security and privacy in automotive telematics. In *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*, pages 25–32, New York, NY, USA, 2002. ACM Press.
- [6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys'03: Proceedings of the International Conference on Mobile Systems, Applications, and Services*, 2003.
- [7] M. Gruteser and D. Grunwald. A methodological assessment of location privacy risks in wireless hotspot networks. In *Security in Pervasive Computing, First International Conference, Boppard, Germany*, pages 10–24, March 2003.
- [8] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy-aware location sensor networks. In *HotOS IX: 9th USENIX Workshop on Hot Topics in Operating Systems*, 2003.
- [9] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *MOBI-COM '02: Proceedings of the 8th ACM International Conference on Mobile Computing and Networking*, pages 12–23, 2002.
- [10] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *ICDCS '05: Proceedings of the 25th International Conference on Distributed Computing Systems*, 2005.
- [11] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.
- [12] 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.
- [13] B. Lu and U. W. Pooch. Cooperative security-enforcement routing in mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Mobile and Wireless Communications Networks*, 2002.
- [14] C. Ozturk, Y. Zhang, and W. Trappe. Source-location privacy in energy-constrained sensor network routing. In *SASN '04: Proceedings of 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2004.
- [15] 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.
- [16] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 299–302. ACM Press, 2001.