Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 6354514, 10 pages
http://dx.doi.org/10.1155/2016/6354514

Research Article

Sink Location Protection Protocols Based on Packet Sending Rate Adjustment

Juan Chen,[1] Zhengkui Lin,[1] Yan Liu,[2] Ying Hu,[1] and Xiaojiang Du[3]

[1]*Department of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*
[2]*Department of Automation, Harbin University of Science and Technology, Harbin 150080, China*
[3]*Department of Computer and Information Science, Temple University, Philadelphia, PA 19122, USA*

Correspondence should be addressed to Xiaojiang Du; dux@temple.edu

Received 25 August 2015; Revised 18 December 2015; Accepted 20 December 2015

Academic Editor: Andrei Gurtov

Sink location protection is critical to the viability of sensor networks as the central point of failure. Most existing work related to sink location protection focuses on local traffic analysis attack. In this paper, we study the sink location protection problem under a more powerful type of attack, the global traffic analysis attack. In order to hide the sink location, a protocol based on packet sending rate adjustment (SRA) is proposed. By controlling the packet sending rate of each node according to the current number of source nodes, SRA conceals the real traffic volume generated by source nodes and hence disguises the location of the sink. For further reducing the communication cost, we propose a light weight SRA protocol (L-SRA), which protects the sink location while significantly decreasing the communication cost. Performance of both SRA and L-SRA has been validated by theoretical analysis and simulation results.

1. Introduction

Wireless sensor networks (WSNs), which feature information sensing, data processing, and wireless communication, have been widely used in military and civilization sectors [1, 2]. A typical WSN is composed of hundreds of sensor nodes and one sink. Once a sensor node detects an abnormal event, it acts as the source node (or source) and sends several event (or real) packets periodically to the sink. Then, the sink collects these packets and sends them to the network manager. Such many-to-one communication pattern makes the sink the central point of failure [3, 4]. An attacker could destroy the sink physically after tracing and locating it and hence paralyze the whole sensor network. Therefore, preserving the sink location is of great importance, and sink damage can cause the whole network to become useless.

Two types of sink location attacks can be used to determine the location of the sink, the global traffic analysis attack (GTA) [5–9] and the local traffic analysis attack (LTA) [10–14]. Existing sink location protection protocols mostly consider the local traffic analysis attack. The schemes against the local attack [10–14] become ineffective in the presence of a global attacker, since a global attacker attempts to locate the sink by identifying the region exhibiting a high traffic sending rate. For example, the global attacker can deduce the location of the sink by monitoring the high volume of transmissions caused by the appearance of a new source (or several new sources). We thus focus on the privacy preserving techniques that defend against a global attacker.

There are several solutions proposed to defend against global attackers [5–9]. However, some of them generated high traffic volumes around the sink [5, 6, 9]. Consequently, the sink cannot be well concealed. Some other solutions are based on several restrict assumptions which were not suitable to most applications [7]. A simple solution proposed by [8] tried to control the packet sending rate of each node in such a way that every node sent packets at the same rate. However, if sensor nodes send packets at a low rate, the real packets must be delayed seriously. On the contrary, if sensor nodes send packets at a high rate, the communication cost is high. To address above problems, in this paper, we propose a sink location protection protocol based on packet sending rate adjustment (SRA) under the global attack. SRA sets the packet sending rate of each node according to the

current number of sources in WSNs. (Note that the packet sending rate of each node is the sum of the event packet sending rate and the fake packet sending rate.) With uniform packet sending rate across the entire sensor network, SRA can defend against global attack effectively without incurring extra forwarding latency (the latency from event packet receiving to event packet forwarding by a node). Based on SRA, we further find that the communication cost can be significantly reduced if the packet sending rate is set according to the maximum traffic of the network. So we continue to propose a light weight SRA protocol (L-SRA), which protects the sink location successfully while decreasing the communication cost significantly. Particularly, L-SRA does not increase forwarding latency. Performance of SRA and L-SRA has been validated by analysis and simulation results.

The rest of the paper is organized as follows. Section 2 introduces the related work. We then present our network and attack models in Section 3. Sections 4 and 5 propose our new protocols, SRA and L-SRA, respectively, for sink location protection against the global attack. The performance analysis of our scheme is given in Section 6. Section 7 presents the performance evaluation through simulations. Finally, we conclude the paper with future work in Section 8.

2. Related Work

A variety of approaches have been used to protect the sink location, such as fake message injection, randomization of forwarding delay, and the use of fake sinks in order to hide the real sinks' positions [8]. Generally, existing related work can be classified into sink location protection against a local eavesdropper (LTA) [10–14] and sink location protection against a global eavesdropper (GTA) [5–9].

In order to defend against the local eavesdropper, Deng et al. [10] introduced a technique to protect sink location by changing the ID field from time to time. In [11], it was shown that, by carrying out rate monitoring and time correlation attacks, an attacker can trace sinks. To mitigate these two kinds of attacks, Deng et al. presented four schemes including the multiple-parent routing scheme, the controlled random walk scheme, the random fake path scheme, and the hot spots scheme [11]. In [12], fake packets and redundant hops are considered to misguide the local attacker when real packets are sent to the sink. In order to balance between the packet delivery latency and the sink's location privacy, Li et al. [13] proposed an intelligent fake packet injection scheme based on the random walk. Ebrahimi and Younis [14] protected the sink's location by improving the traffic volumes in low-traffic-activity areas. So the local attacker will be distracted to these areas other than the sink. However, these techniques cannot resist passive traffic analysis attacks under a global attacker. A global eavesdropper can easily defeat these schemes. For example, the global eavesdropper only has to find the area with a high number of transmissions to locate the sink.

For a motivated attacker, eavesdropping on the whole network is a fast and effective way to locate the sink. In [6], fake sinks were designed to confuse a global attacker. Although fake sinks can protect the sink from local attacker, it is not effective in the case of a global attacker. This is because global traffic analysis can identify all fake and real sinks according to the high traffic volume compared with other areas. Acharya and Younis [5] consider the attackers with both local and global views. Since nodes near the sink send more packets than other nodes, the global attacker attempts to find the area with high traffic volume and then locate the sink. BAR was proposed to hide the sink location by making the sink selectively transmit packets to random sensor nodes in its vicinity. Relaying these packets away from the sink can confuse a local attacker. However, BAR is efficient to defend against a global attacker. This is because all these packets will definitely travel through the nodes near the sink and then form a higher traffic volume near the sink. Thus, the global attacker can locate the sink by traffic analysis. Ying et al. [7] introduced a concealing sink location (CSL) protocol. Each node in CSL generated the same traffic volume as the sink's neighboring nodes by transmitting a number of fake packets. Thus CSL can defend against traffic analysis attack launched by a global attacker. However, the design of CSL protocol is based on the following restrict assumptions. (a) The sink is located at the center sensor network. (b) Sensors are deployed within a circular area. (c) The deployment is done according to a uniform distribution. Such restrict assumptions do not apply to many real WSN deployments. Mehta et al. [9] introduced fake sinks and backbone flooding to defend against global eavesdropper. In the scheme using fake sinks, each source redundantly sent packets to several fake and real sink nodes. In the backbone flooding scheme, each source sent packets to one of the backbone nodes which then flooded these packets to other backbone nodes. The sink is a neighbor of a backbone node which can overhear the flooding. However, in both schemes, all traffics will meet at either dummy backbone nodes (near the real sink) or the real sink. A simple solution proposed by [8] tried to control the packet sending rate of each node in such a way that every node sent packets at the same rate. However, how to choose an appropriate value for the packet sending rate has not been solved yet.

3. System Model

3.1. Network Model. We assume N evenly distributed sensor nodes and one sink in the whole network. The sink node and other sensor nodes have the same appearance. The sink constructs the network topology (e.g., build the broadcast tree) by one-time broadcast over the entire network [9]. After that, sensor nodes can send packets hop by hop to the sink by parent node of each node [9]. For example, given a node u, its parent set PS_u is composed of all u' neighboring nodes that have a shorter hop count (the hop count from the node to the sink) than u. Furthermore, we assume clock synchronization of the nodes. At any time, there are m ($N \geq m \geq 0$) sources in the network and the event packet (or real packet) sending rate of each source is R ($R \geq 0$), where N denotes the number of nodes in the WSN.

3.2. Attack Model. Different from sensor nodes, the attackers have faster computational ability and more storage space

and can communicate with each other in a larger range. Several attackers are deployed in the network to launch collusion attack. Specifically, their attacking abilities are listed as follows:

(i) *Passive Traffic Monitoring*: the attacker is able to eavesdrop the packet transmissions in a range but is unable to decipher packets.

(ii) *Ability of Collusion*: Several attackers monitor their local traffic separately for a period of time and then move close to share their information. At last, they can infer and obtain the whole network traffic pattern.

4. Packet Sending Rate Adjustment Protocol

In order to defend against the global traffic attacker, we propose an efficient sink location protection protocol based on packet sending rate adjustment (SRA). SRA first investigates what value should be assigned to the packet sending rate of each node so that low communication cost and low forwarding latency can be achieved (e.g., in an extreme case, if all event packets are transmitted by one node, the node cannot transmit all these event packets immediately unless its packet sending rate is high enough). Then, SRA creates a uniform packet sending rate, thus preventing the attackers with a global monitoring ability from tracing the sink with extra low forwarding latency. Specifically, the procedures of SRA include network initialization phase and packet sending rate setting phase.

4.1. Network Initialization. In this phase, each node, say u, initializes a list T_u including elements in the form of $\langle event\ type, number\ of\ packets\rangle$, where $T_u[event\ type] \cdot number\ of\ packets$ is the number of event packets that must be sent from source to the sink once a node detects an event and becomes the source. As the source sends event packets periodically, $T_u[event\ type] \cdot number\ of\ packets$ measures the duration from sending the first to the last event packet by the source. For instance, temperature and humidity stand for different events. When u detects a sudden change of temperature or humidity, the number of packets sent from u to the sink turns different. Each node and the sink are also preloaded two queues T_n and A_n. T_n and A_n are used to record the subintervals and the number of sources for each subinterval.

4.2. Packet Sending Rate Setting Based on Number of Sources. SRA protects the sink location against global traffic analysis attack by creating a uniform packet sending rate. One important related question is how large the packet sending rate should be. The high or low packet sending rate can result in high communication cost or forwarding latency. Figure 1 shows an example, where three sources (s_1, s_2, and s_3) in a moment and all event packets from these three sources should be transmitted by node u'. If the packet sending rate is set less than $3R$, some event packets must be delayed at u' and hence the forwarding latency increases. Theorem 1 proves that, given m sources in a network at a moment, if the packet sending rate is set to $m * R$, low communication cost and forwarding latency can be guaranteed.

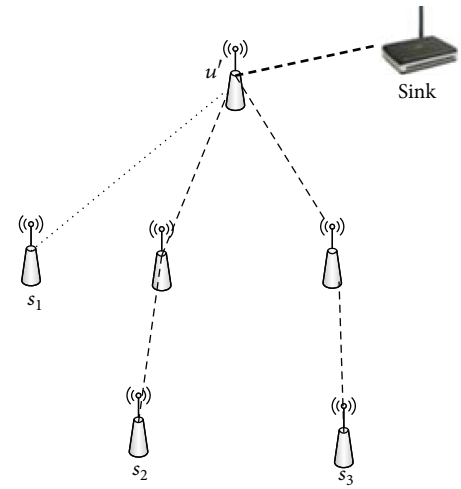

Figure 1: Event packets forwarding at node u'.

Theorem 1. *If there are m sources, for any sensor, say u, in order to forward the event packets immediately while incurring low communication cost, r_u should be set as $m * R$, where r_u represents the packet sending rate of u.*

Proof. Given m sources and the fact that the event packet sending rate of each source is R, if u must transmit event packets from m_u ($0 \le m_u \le m$) sources, u will not delay the forwarding of each packet on condition that $r_u \ge r_u * R$. Considering the extreme case, if event packets from all sources must be forwarded by u, then we have $m_u = m$. Thus, only if r_u is set equal to or larger than $m * R$, u can forward all event packets immediately. However, the communication cost increases as r_u increases. Therefore, if $r_u = m * R$, u guarantees the immediate packet forwarding with low communication cost. □

According to Theorem 1, in order to set an appropriate packet sending rate for different subinterval, the sink must obtain the number of sources during the subinterval. As source nodes appear and disappear randomly, once a node, say u, becomes a source, the sink divides (t_s, t_e) into several subintervals, in each of which the number of sources is different. Specifically, the sink proceeds following three steps.

Step 1 ((t_s, t_e) computation). The sink computes the time interval, known as (t_s, t_e), such that u remains a source.

Once a source, say u, appears, u starts a broadcast M_a to inform the whole network about its appearance. As soon as receiving M_a, the sink computes the time interval, say (t_s, t_e), for source u according to (1) and (2). Parameters including t_{start} and T_{ime} stand for the time of receiving M_a at the sink and the duration that u keeps generating event packets (i.e., $(T_u[event\ type] \cdot number\ of\ packets - 1)/R$), respectively. Equation (1) shows that after all nodes receive M_a, u starts to send the first event packet to the sink. Equation (2) shows

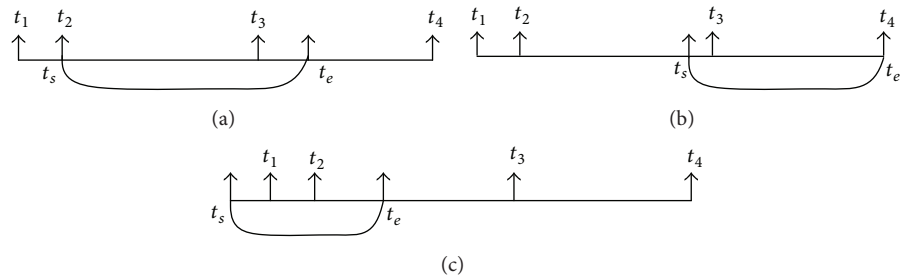

Figure 2: Subinterval divivison for a new source.

that the source is considered to disappear after it has sent its last event packet. After that u becomes a normal sensor which only forwards event packets instead of generating and sending event packets. Here, our "disappearance" is different from a conventional "nonexistence." A node may become source again and again since it may detect events occasionally, so it is possible for it to go through the process from source appearance to source disappearance now and then. Consider

$$t_s = t_{\text{start}}, \tag{1}$$

$$t_e = t_s + T_{ime} + \partial. \tag{2}$$

Step 2 (subintervals division). If a new source u is the only source during (t_s, t_e), we have the subinterval (t_s, t_e) for which the number of source is 1. If there is any other source except u during (t_s, t_e), SRA divides (t_s, t_e) into several subintervals to satisfy the fact that the number of sources is different in each subinterval by Subinterval Partition Algorithm Based on Number of Sources (SPAN), as is shown in Algorithm 1. For example, Figure 2(a) shows 2, 3, and 4 sources during subintervals (t_1, t_2), (t_2, t_3), and (t_3, t_4). Then, u becomes a source during (t_s, t_e) which will be divided into subintervals (t_s, t_3) and (t_3, t_e). This is because the number of sources in (t_s, t_3) increases from 3 to 4. And the number of sources in (t_3, t_e) increases from 4 to 5. After that, we obtain four subintervals including (t_1, t_s), (t_s, t_3), (t_3, t_e), and (t_e, t_4).

In Algorithm 1, before u appears, the subintervals and the number of sources are recorded in queues T_s and A_s, respectively, where $T_n = \{t_1, t_2, \ldots\}$ and $A_n = \{a_1, a_2, \ldots\}$. The number of sources for subinterval (t_i, t_{i+1}) is a_i, where $t_i < t_{(i+1)}$. (t_s, t_e) will be divided according to the following conditions:

(i) If $\exists t_j \in T_n$ which satisfies the fact that $t_j == t_s$ (e.g., $t_2 == t_s$ in Figure 2(a)), then we have $a_j{+}{+}$ and insert a_j into A_n.

(ii) If $\nexists t_j \in T_n$ which satisfies the fact that $t_j == t_s$, then insert t_s and a_s into T_n and A_n, respectively. Two conditions are to be further considered as follows:

(a) If t_s is the first element in T_n as is shown in Figure 2(c), then $a_s = 1$.

(b) If t_s is not the first element in T_n as is shown in Figure 2(b), then we have $a_s = a_{(s-1)} + 1$.

Input: $T_n = \{t_1, t_2, \ldots\}$, $A_n = \{a_1, a_2, \ldots\}$, (a_s, t_e)
Output: T_n, A_n
tag1 = FALSE, tag2 = FALSE;
For *any element* in T_n **do**
  if $\exists t_j \in T_n$ *which satisfies that* $t_j == t_s$ **then**
    a_j + +;
    Insert a_j into A_n;
  end
  if $t_j \in T_n$ *which satisfies that* $t_s < t_j < t_e$ **then**
    a_j++;
  end
  if $t_j \in T_n$ *which satisfies that* $t_j == t_s$ **then**
    tag1 = TRUE;
  end
  if $t_j \in T_n$ *which satisfies that* $t_j == t_e$ **then**
    tag2 = TRUE;
  end
end
if *tag1 == FALSE* **then**
  Insert t_s and a_s into T_n and A_n;
  if t_s *is the first element in* T_n **then**
    $a_s = 1$;
  end
  if t_s *is not the first element in* T_n **then**
    $a_s = a_{(s-1)} + 1$;
  end
end
If *tag2 == FALSE* **then**
  $a_e = a_{(e-1)} - 1$;
  Insert t_e and a_e into T_n and A_n;
end

Algorithm 1: SPAN algorithm.

(iii) For $\forall t_j \in T_n$ which satisfies the fact that $t_s < t_j < t_e$ (e.g., $t_s < t_j < t_e$ in Figure 2(a)), then we have a_j + +.

(iv) If $\nexists t_j \in T_n$ which satisfies the fact that $t_j == t_e$ as is shown in Figure 2(a) or Figure 2(c), then insert t_e and a_e into T_n and A_n, respectively. And we have $a_e = a_{(e-1)} - 1$.

Step 3 (uniform packet sending rate creation). After obtaining the subintervals in Step 2, SRA sets the packet sending rate of each node according to the number of sources at each subinterval. For example, if there are m' sources in a

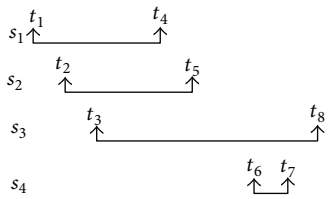

Figure 3: Duration of event packet sending.

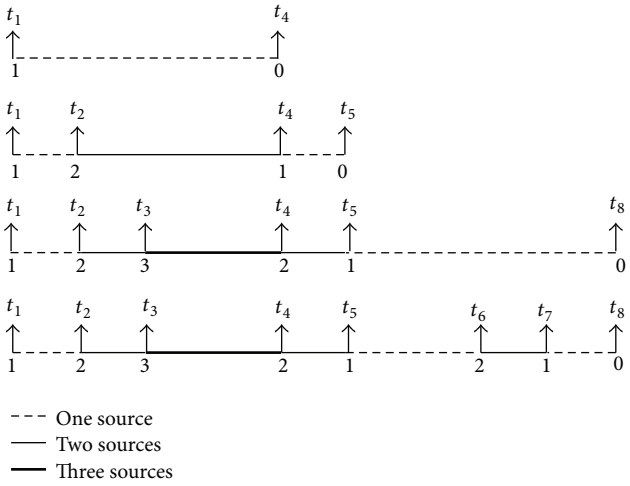

Figure 4: Subinterval division for four sources.

subinterval, each node sends packets with the rate $m' * R$. Specifically, the process of packet sending rate setting is as follows.

The sink broadcasts M_b (i.e., the rate adjustment broadcast packet) which includes T_n and A_n. Node v may send a number of fake packets if there is no enough event packets to be transmitted, so that the predefined packet sending rate can be achieved.

For instance, Figure 4 shows how SRA adjusts the packet sending rate of each node when four sources including s_1, s_2, s_3, and s_4 appear one after another. Figure 3 shows the duration in which each source appears. Figure 4 shows the subinterval division process by SPAN when four sources appear one by one. More specifically, when source s_1 appears, there is only one source and one subinterval (t_1, t_4) as can be seen in Figure 4. After that, s_2 detects an event and becomes a source which sends event packets during (t_2, t_5). Then, the sink divides (t_2, t_5) into two subintervals: (t_2, t_4) and (t_4, t_5). This is because the number of sources has changed into two and one during (t_2, t_4) and (t_4, t_5). Similarly, when s_4 appears, seven subintervals have been obtained by Algorithm 1 as shown in Figure 4. As a result, the packet sending rate is set to R, $2R$, $3R$, $2R$, R, $2R$, R and 0 at $t_1, t_2, t_3, t_4, t_5, t_6, t_7$, and t_8.

5. Lightweight Packet Sending Rate Adjustment Protocol

In SRA, the packet sending rate of each node for the current subinterval is set to "*the current number of sources*" $*$ R.

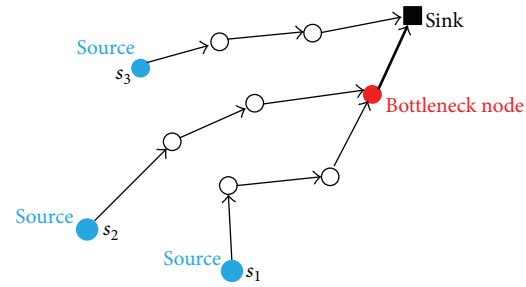

Figure 5: The traffic of the bottleneck node.

Table 1: The average bottleneck degrees varying with different number of sources.

Number of sources	5	10	15	20	25
D	1	1	2	2	3

This is because in the worst case all real traffics (traffic formed by event packets other than fake packets) go through a single node. In order to hide the real traffic, SRA adjusts the packet sending rate of each node to "*the current number of sources*" $*$ R. However, in real cases, the possibility that all real traffics go through one node is low. As is illustrated in Figure 5, there are three sources including s_1, s_2, and s_3 during the current subinterval. Each node transmits event packets from no more than two sources. We can also observe that only the bottleneck node (see Definition 4) transmits event packets generated by two sources; thus the bottleneck degree is 2 (see Definition 4). Therefore, to further decrease the communication cost, it is better to set the packet sending rate to "*the bottleneck degree*" $*$ $R(2R)$ other than $3R$. According to Definition 5, given that the bottleneck degree for the current subinterval is D, obviously we have $D \leq$ "*the number of sources*". Table 1 shows the average bottleneck degrees when the number of sources varies from 5 to 25 based on simulation results. The network used here includes 1024 nodes uniformly distributed, each of which has an average of 8 neighbors. For different number of sources, simulation is repeated 100 times. Apparently, we observe that $D \leq$ "*the number of sources*", which means the communication cost can be highly reduced if the packet sending rate is set to $D * R$.

Definition 2. D_v^σ is the traffic degree of node v for subinterval σ. Specifically, D_v^σ is the number of sources from which node v transmits event packets during σ.

Definition 3. $\Psi(\sigma)$ is the traffic degree set including the traffic degree of each node for subinterval σ. Specifically, $\Psi(\sigma) = \{i \in N \mid D_i^\sigma\}$.

Definition 4. BTN(σ) is the bottleneck node who has the maximum traffic degree during subinterval σ.

Definition 5. BT(σ) is the bottleneck degree for subinterval σ. Specifically, BT(σ) is the traffic degree of BTN(σ).

```
Input: {σ_1, σ_2, ..., σ_a}, {λ_1, λ_2, ..., λ_b}
// {λ_1, λ_2, ..., λ_b} is the sub-intervals before the new source appears
Output: {σ_1, σ_2, ..., σ_a}
For i = 1; i ≤ a; i++ do
  for j = 1; j ≤ b; j++ do
    if σ_i ⊆ λ_j then
      // update the traffic degree of each node for σ_i
      Ψ(σ_i) = Ψ(λ_j);
      // update the bottleneck degree for σ_i
      BT(σ_i) = BT(λ_j);
    end
  end
end
```

Algorithm 2: Traffic degree and bottleneck degree update.

In view of the above discussion, we further propose a Light weight SRA protocol, entitled L-SRA, which attempts to further reduce the communication cost of the network and balance the real traffic while achieving the sink location security requirement.

5.1. Overview of L-SRA. When a new source u appears, L-SRA first partitions (t_s, t_e) into several subintervals by Algorithm 1 (see Section 4.2) and updates the traffic degree of each node for each newly obtained subinterval. Then, L-SRA chooses shortest path from the source to the sink carefully for each subinterval divided from (t_s, t_e). Specifically, in order to balance the real traffic, the shortest path is constructed by the Simple Greedy Algorithm Based on Node Traffic Degree proposed by us. Theorem 6 proves that if the real traffic is more uniform, the bottleneck degree is smaller, thereby resulting in less communication cost. Therefore, the communication cost can be further reduced.

L-SRA includes four major steps: (t_s, t_e) computation, subintervals division, path construction for traffic balance, and packet sending rate adjustment. The first step of L-SRA is the same as that of SRA. All the steps other than the the first step will be detailed later.

Theorem 6. *The more uniform the real traffic is, the less the bottleneck degree for a subinterval σ is.*

Proof. We use entropy to measure the randomness of real traffic. Entropy is a mathematical measure of information uncertainty. Entropy of a random variable X with a probability function $p(x)$ is defined as

$$-\sum p(x) \log_2 p(x). \tag{3}$$

Suppose that the numbers of event packets transmitted by node v and all nodes in the network are p_v and Θ during σ. We thus define the network entropy according to (4) and use it to estimate the randomness of real traffic during σ. Consider

$$H(N) = -\sum_{v \in N} \left(\frac{p_v}{\Theta}\right) \log_2 \left(\frac{p_v}{\Theta}\right). \tag{4}$$

Obviously, a higher value of $H(N)$ implies that the traffic pattern of N nodes is more random. In an extreme case, $H(N)$ achieves the maximum value $\log_2 N$ when each node sends Θ/N packets during σ. □

5.2. Subintervals Division. L-SRA first divides (t_s, t_e) into several subintervals, say $\{\sigma_1, \sigma_2, \ldots, \sigma_a\}$, according to Algorithm 1 in Section 4.2. Then, L-SRA updates the traffic degree of each node and the bottleneck degree by Algorithm 2.

5.3. Path Construction for Traffic Balance. Here, L-SRA finds shortest path from u to the sink while balancing real traffic distribution. For $\forall \sigma_i \in \{\sigma_1, \sigma_2, \ldots\}$, the sink constructs shortest path for each source. The real traffic distribution is determined by these paths. Since more than one shortest path exist for each source, if the paths for different sources during σ_i can be carefully chosen, the real traffic can be well distributed. According to Theorem 6, the more uniform the network traffic is, the smaller the bottleneck degree is. And hence, the communication cost is less. So, in order to balance the network traffic, when the new source u appears, L-SRA constructs shortest path for u which tries to bypass the node with a high traffic degree. To do this L-SRA selects nodes for the path one by one by a Simple Greedy Algorithm Based on Node Traffic Degree (see Algorithm 3).

Specifically, the path construction process is as follows. Source u is first selected as the current node; L-SRA then chooses the next node from u's parent set whose traffic degree is minimum during σ_i. After that, the "next node" becomes the current node and repeats the node selection process as u does until the sink is chosen.

5.4. Packet Sending Rate Adjustment. In this step, different from SRA, the sink broadcasts not only T_n and A_n but also the shortest path obtained in Section 5.3 at the same time in L-SRA. Thus, for any subinterval, L-SRA adjusts the node packet sending rate to $R *$ "*the bottleneck degree of the sub-interval*". As "*the bottleneck degree of the subinterval*" $\leq$ "*the number of sources*," the communication cost can be significantly reduced in L-SRA.

```
Input: Ψ(σ_i)
Output: P_ath
c = u;
P_ath = ∅;
While the sink is not the current node c do
   add c to P_ath;
   v = the node whose traffic degree is minimum from c's parent set;
   c = v;
end
for j = 1; j ≤ N; j++ do
   // increase the traffic degree of each node on the path for sub-interval σ_i
   if j ∈ P_ath then
      Ψ_j^{σ_i}++;
      if BT(σ_i) < Ψ_j^{σ_i} then
         // update the bottleneck degree for sub-interval σ_i
         BT(σ_i) = Ψ_j^{σ_i};
      end
   end
end
```

Algorithm 3: Simple Greedy Algorithm Based on Node Traffic Degree.

6. Performance Analysis

The performance of SRA and L-SRA will be analyzed from the following three aspects.

6.1. Security Performance. A global attacker can ascertain which nodes send packets, their sending rate, and the number of packets. Then the attacker can infer the location of a sink, because the neighbors of the sink node tend to send more packets than other nodes. Our SRA and SRA-L make every node send the same number of encrypted events or fake packets at a constant rate. Therefore, the global attacker cannot determine the sink from the number and rate of packets.

Specifically, we define and use the entropy of the network traffic $H(N)$ by (4) in Section 5.1 to measure the security performance. Suppose that the numbers of event packets transmitted by node v and all nodes in the network are p_v and Θ during σ. Thus we have $H(N) = -\sum_{v \in N}(p_v/\Theta)\log_2(p_v/\Theta)$. In the following, we give the security performance analysis of SRA and SRA-L, respectively.

Security Performance for SRA. SRA includes two phases: network initialization phase and the packet sending rate setting phase. Without loss of generality, we just analyze the security performance for the latter here because the former is one time and thus the sink is considered to be safe.

In the packet sending rate setting phase, network traffic appears in the following three conditions according to Section 4.2:

(a) The source broadcasts the packet M_a. It is obvious that this broadcast process will not reveal the sink location. This is because each sensor transmits the packet once and thus the traffic is well distributed. The entropy of the network traffic is $H(N) = -\log_2 N$.

(b) The sink broadcasts the packet M_b. The same as (a), we have that the entropy of the network traffic $H(N) = -\log_2 N$ and the traffic is well distributed.

(c) Each node sends packets with the same rate. The entropy of the network traffic is $H(N) = -\log_2 N$. With evenly traffic distribution, the sink location is well preserved.

Security Performance for L-SRA. Similar to SRA, the network traffic in L-SRA appears in three conditions. Different from SRA, in L-SRA, the sink broadcasts two packets, one being M_b and the other including the shortest path introduced in Section 5.4. As a result, the traffic is evenly distributed in the three conditions and L-SRA achieves good security performance.

6.2. Communication Cost. The communication cost of SRA and L-SRA for a subinterval, say σ, is the total number of packets transmitted for the three conditions described in Section 6.1. There are S sources and S' ($S'leS$) out of S sources are new appearance sources.

The communication cost of SRA for subinterval σ resulted by two broadcasts and sending packets with the same rate, say $R * S$. Two broadcasts (one started from the source and the other started from the sink) require $2S' * N$ packet transmissions for S' new sources. The uniform packet sending requires $R * S * N * \sigma$ packet transmissions. Thus, the communication cost of SRA is $2S' * N * \sigma + R * S * N * \sigma$.

For L-SRA, extra communication cost is added due to the broadcast of the shortest path compared with SRA. And the packet sending rate is different for L-SRA and SRA. Specifically, each node sends packets at the rate of *"the bottleneck degree of the sub-interval"* $* R$ other than $S * R$ in L-SRA. Since *"the bottleneck degree of the sub-interval"* $\leq S$, the communication cost of L-SRA is

obviously lower than that of SRA. The maximum and minimum bottle degrees for σ are S and S/m, respectively. Therefore, we have that the maximum communication cost for L-SRA is $(3S' + S * \sigma * R) * N$. The minimum communication cost for L-SRA is $(3S' + \lceil S/m \rceil * \sigma * R) * N$, where m is the average number of neighbors for each node.

6.3. End-to-End Latency. We use the number of hops that an event packet takes from the source to the sink on average to measure the end-to-end latency. Take note that the computation time that the sink needs is very short and hence has not been considered for two reasons: firstly, we assume that the sink has powerful computational ability; secondly, the computation process is simple for SRA and L-SRA and hence the computation time can be ignored.

The end-to-end latency for SRA and L-SRA includes the latency incurred by packet transmission and broadcasts from the source and sink. The packet transmission takes $h_{u,\text{sink}}$ hops from the source u to the sink. The latency for the broadcast process started from the source and the sink is $2h_{\max}$, where $h_{\max}$ is the furthest distance measured by hops between any two nodes in the network. Note that the broadcast process is once for a new source. Thus, the end-to-end latency for SRA and L-SRA is $h_{u,\text{sink}} + 2h_{\max}/T_u[\textit{event type}] \cdot \textit{number of packets}$.

7. Simulation Results

We evaluate the performance of SRA and L-SRA in view of communication cost and end-to-end latency in this section. There are several solutions proposed to defend against the global attackers [5–9]. However, some of them generated high traffic volumes around the sink [5, 6, 9]. So the sink cannot be well concealed. Some of them are based on several restrict assumptions which were not suitable to most applications [7]. The solution proposed by [8] tried to hide the sink location by balancing the traffic distribution without introducing strict assumptions. Thus, we compare our SRA and L-SRA with the *"Baseline"* scheme proposed by [8]. All the simulations have been performed based on OMNET++. The sink is placed randomly. Each source sends 5 event packets to the sink at the rate of $R = 1$ packet/second. The predefined packet sending rate for *"Baseline"* was $S_b * R$. S_b was set high enough to achieve better sink location protection and low end-to-end latency in [8]. However, as the communication cost for *"Baseline"* grows as S_b increases, S_b is set to $S + 1$ here. In order to provide an accurate result, each combination of the simulation parameters has been repeated 50 times.

Figure 6 shows how the communication cost changes as the average number of neighbors, say m, increases for SRA, L-SRA, and *"Baseline."* The parameters are set as follows: $S = 5$, $S' = 1$, and $N = 1024$. There are S' new sources out of S sources. It is observed that the communication cost for L-SRA is significantly lower than the other two protocols. This is because L-SRA adjusts the packet sending rate of each node dynamically according to the bottleneck degree of the current subinterval, thereby reducing its communication cost obviously. Figure 6 further shows that the communication cost for SRA is lower than *"Baseline."* This is because the packet sending rate of SRA is set according to the current number of sources which is lower than S'. It is also observed that an increase in m reduces the communication cost of L-SRA. Since there are more neighbors to forward packet, the bottleneck degree decreases and hence the communication cost can be reduced. As SRA and *"Baseline"* adjust their packet sending rate according to the current number of sources and $S+1$, respectively, both of which are irrelevant to m, it follows that their communication cost remains consistent across all simulations.

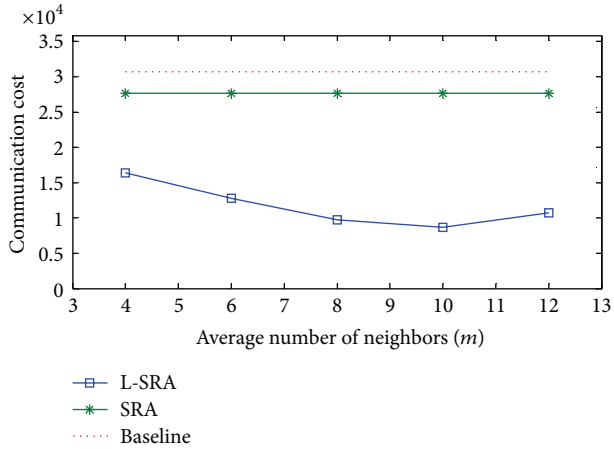

FIGURE 6: End-to-end latency under different protocols.

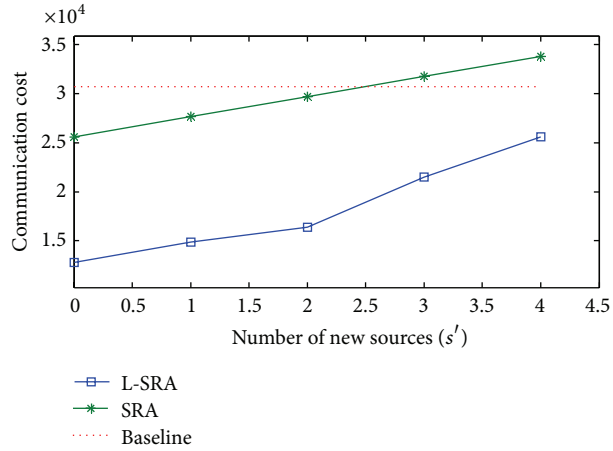

FIGURE 7: End-to-end latency under different protocols.

Next, in order to investigate how S and S' affect the communication cost of L-SRA, SRA, and *"Baseline,"* we first set a fixed value 5 to S and change S' from 0 to 4 and then set a fixed value 1 to S' and change S from 1 to 5. Other parameter settings are $N = 1024$, $\sigma = 5$, and $m = 4$. Take note that we are looking at almost the worst case here for L-SRA since m is set to a very small value which will result in an increase in its communication cost as Figure 6 shows. Figures 7 and 8 show the simulation results, respectively.

We can observe in Figure 7 that L-SRA performs significantly better than the other two. The communication cost for SRA is generally lower than *"Baseline."* Nevertheless,

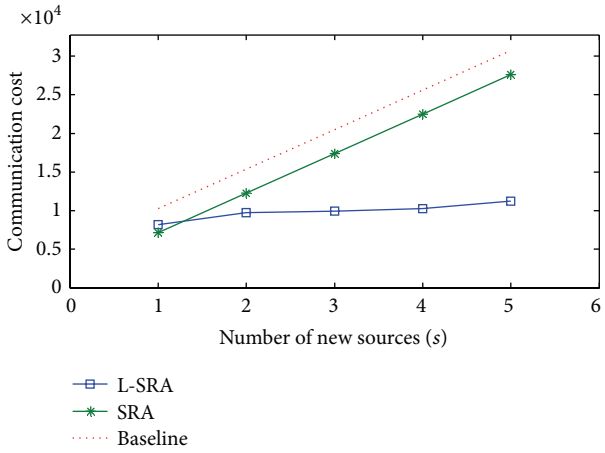

Figure 8: End-to-end latency under different protocols.

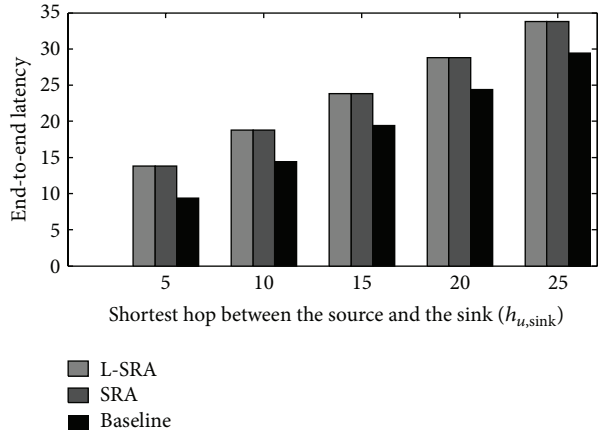

Figure 9: Communication cost comparison among different protocols.

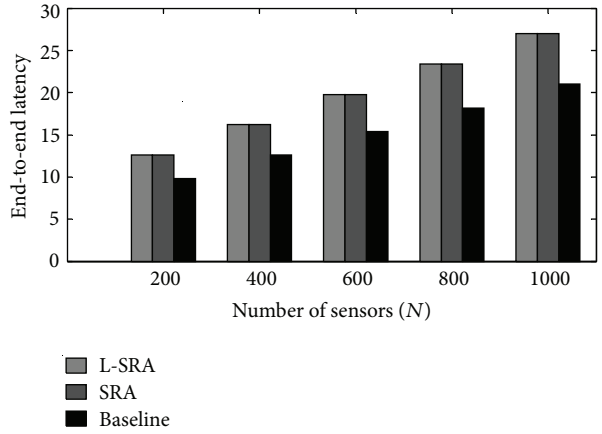

Figure 10: End-to-end latency under different protocols.

when the value of S' approaches its upper limit S, it is not the case. We attribute this to two things. Firstly, the packet sending rate of *"Baseline,"* say $S_b * R$, is set to a lower and also an almost ideal value, say $S + 1$ here, thereby reducing its communication cost obviously. However, in real cases, without knowing the actual number of sources, $S_b * R$ should be set high enough to assure less end-to-end latency for event packets. Secondly, S' approaches S which means nearly all sources are new appearance ones. According to Section 4.2, for SRA, each new source will bring about two broadcasts started by itself and the sink, respectively. Hence, in rare cases, when the number of new sources approaches its upper limit, the communication cost for SRA may be a bit higher than *"Baseline"* as more broadcasts are needed. Based on the discussion above, SRA performs better than *"Baseline"* in general.

As expected, L-SRA performs best and SRA performs better than *"Baseline"* as shown in Figure 8. Besides, we further observe that the communication cost of both SRA and *"Baseline"* grows quickly, while that of L-SRA grows slowly as S increases. This is because in most cases the bottleneck degree not only is far less than S but also increases less with the growth of S. Therefore, L-SRA can reduce its communication cost significantly. Take note that when $S = 1$, the communication cost for L-SRA is a little higher than SRA. When compared with SRA, extra communication cost is brought about for L-SRA by the broadcast of shortest path as introduced in Section 5.4. Since there is only one existent source, the bottleneck degree is also the number of sources. Thus, the packet sending rates for L-SRA and SRA are the same. So, under this condition, L-SRA does not benefit from its packet sending rate setting based on bottleneck degree. However, there is no doubt that L-SRA performs better than SRA in most cases.

Next, we examine the end-to-end latency of real packets for different $h_{u,\text{sink}}$ in Figure 9, where $h_{u,\text{sink}}$ represents the hops between the source u and the sink. There are $N = 500$ nodes in the network. It can be seen that the end-to-end latency for L-SRA, SRA, and *"Baseline"* increases with the growth of $h_{u,\text{sink}}$. We can further observe that the end-to-end latency of L-SRA and SRA is almost the same and it is a little more than that of *Baseline*. This is because in order to reduce the packet sending rate of each node, thereby reducing the communication cost, both SRA and L-SRA bring about extra end-to-end latency which resulted by sink broadcast as introduced in Sections 4.2 and 5.4. However, the packet sending rate of *"Baseline"* is predefined and is usually set to a high value to reduce the end-to-end latency as far as possible. As a result, its communication cost is significantly increased and hence sensors can easily and soon be exhausted. Consequently, disastrous security problems such as network outage can arise.

Figure 10 shows the end-to-end latency with different network scale N for L-SRA, SRA, and *"Baseline."* It is observed that the end-to-end latency of them all increases with the increasing of N. This is because the latency which resulted by broadcast (initiated from the source or sink) increases with the growth of $h_{\max}$ which grows as N increases. Similar to the result shown in Figure 9, *"Baseline"* achieves least end-to-end latency at the cost of very high communication cost compared with SRA and L-SRA. Though we can see that the

end-to-end latency of L-SRA and SRA is a little higher than that of *Baseline*, it is acceptable.

8. Conclusion

In order to defend against the global traffic attack, we first propose a sink location protection protocol based on packet sending rate adjustment (SRA). By controlling the packet sending rate of each node dynamically, SRA hides the location of the sink successfully. For further communication cost reduction, we then propose a light weight SRA protocol (L-SRA), which protects the sink location while reducing the communication cost obviously. Future work will focus on sink location protection against the global attacks in mobile sensor networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Authors' Contribution

Juan Chen, Zhengkui Lin, Yan Liu, and Ying Hu contributed equally to this work. Xiaojiang Du provided the preliminary idea. Juan Chen improved the idea and finally wrote the paper. Zhengkui Lin gave the theory analysis. Ying Hu and Yan Liu designed and performed the simulation.

Acknowledgments

This research is supported in part by the Natural Science Foundation of China under Grants nos. 61300188, 61301131, 61301132, and 61203082, by Liaoning Province Science and Technology Plan Program no. 2011402003, by National Key Technology R&D Program no. 2012BAF09B01, and by the US National Science Foundation under Grant no. 1065444.

References

[1] E. C.-H. Ngai and I. Rodhe, "On providing location privacy for mobile sinks in wireless sensor networks," *Wireless Networks*, vol. 19, no. 1, pp. 115–130, 2013.

[2] H. Chen and W. Lou, "On protecting end-to-end location privacy against local eavesdropper in wireless sensor networks," *Pervasive and Mobile Computing*, vol. 16, pp. 36–50, 2015.

[3] J. R. Ward and M. Younis, "Increasing base station anonymity using distributed beamforming," *Ad Hoc Networks*, vol. 32, pp. 53–80, 2015.

[4] X. Lin, R. Lu, X. Liang, and X. S. Shen, "STAP: a social-tier-assisted packet forwarding protocol for achieving receiver-location privacy preservation in VANETs," in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM '11)*, pp. 2147–2155, IEEE, Shanghai, China, April 2011.

[5] U. Acharya and M. Younis, "Increasing base-station anonymity in wireless sensor networks," *Ad Hoc Networks*, vol. 8, no. 8, pp. 791–809, 2010.

[6] X. Wu, J. Liu, X. Hong, and E. Bertino, "Achieving anonymity in mobile ad hoc networks using fuzzy position information," in *Mobile Ad-Hoc and Sensor Networks*, vol. 4325 of *Lecture Notes in Computer Science*, pp. 461–472, Springer, Berlin, Germany, 2006.

[7] B. Ying, J. R. Gallardo, D. Makrakis, and H. T. Mouftah, "Concealing of the sink location in wsns by artificially homogenizing traffic intensity," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '11)*, pp. 988–993, IEEE, Shanghai, China, April 2011.

[8] X. Lin, R. Lu, X. Shen, Y. Nemoto, and N. Kato, "Sage: a strong privacy-preserving scheme against global eavesdropping for ehealth systems," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 365–378, 2009.

[9] K. Mehta, D. Liu, and M. Wright, "Protecting location privacy in sensor networks against a global eavesdropper," *IEEE Transactions on Mobile Computing*, vol. 11, no. 2, pp. 320–336, 2012.

[10] J. Deng, R. Han, and S. Mishra, "Enhancing base station security in wireless sensor networks," Tech. Rep., Department of Computer Science, University of Colorado, 2003.

[11] J. Deng, R. Han, and S. Mishra, "Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks," *Pervasive and Mobile Computing*, vol. 2, no. 2, pp. 159–186, 2006.

[12] Y. Jian, S. Chen, Z. Zhang, and L. Zhang, "Protecting receiver-location privacy in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1955–1963, IEEE, Anchorage, Alaska, USA, May 2007.

[13] X. Li, X. Wang, N. Zheng, Z. Wan, and M. Gu, "Enhanced location privacy protection of base station in wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks (MSN '09)*, pp. 457–464, Fujian, China, December 2009.

[14] Y. Ebrahimi and M. Younis, "Using deceptive packets to increase base-station anonymity in wireless sensor network," in *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC '11)*, pp. 842–847, IEEE, Istanbul, Turkey, July 2011.