Elsevier Pervasive and Mobile Computing Journal, Special Issue on Security in Wireless Mobile Computing Systems, vol 2, issue 2, April 2006, pp. 159-186

Decorrelating Wireless Sensor Network Traffic To Inhibit Traffic Analysis Attacks

Jing Deng Richard Han Shivakant Mishra
Department of Computer Science
University of Colorado at Boulder
Boulder, Colorado, USA
{jing,rhan,mishras}@cs.colorado.edu

Abstract

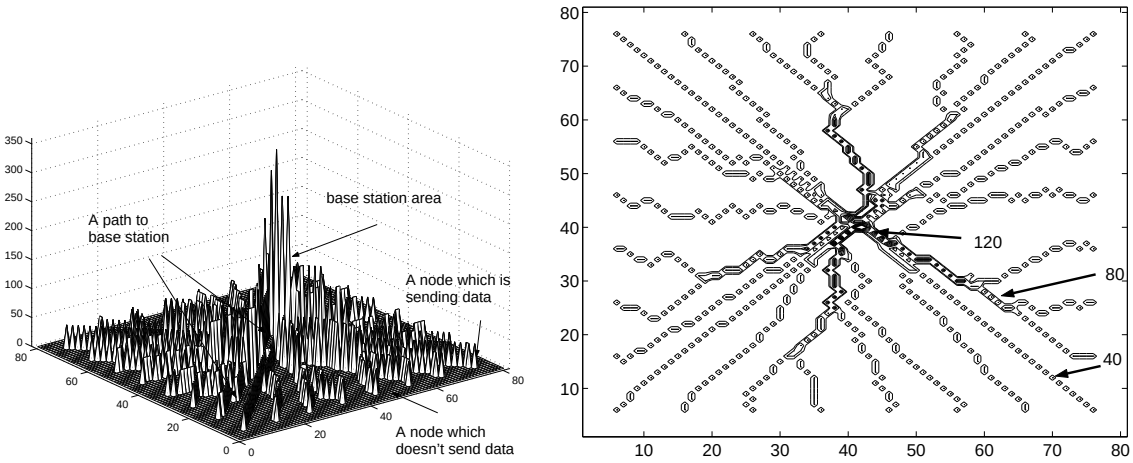
Typical packet traffic in a sensor network reveals pronounced patterns that allow an adversary analyzing packet traffic to deduce the location of a base station. Once discovered, the base station can be destroyed, rendering the entire sensor network inoperative, since a base station is a central point of data collection and hence failure. This paper investigates a suite of decorrelation countermeasures aimed at disguising the location of a base station against traffic analysis attacks. A set of basic countermeasures is described, including hop-by-hop reencryption of the packet to change its appearance, imposition of a uniform packet sending rate, and removal of correlation between a packet's receipt time and its forwarding time. More sophisticated countermeasures are described that introduce randomness into the path taken by a packet. Packets may also fork into multiple fake paths to further confuse an adversary. A technique is introduced to create multiple random areas of high communication activity called hot spots to deceive an adversary as to the true location of the base station. The effectiveness of these countermeasures against traffic analysis attacks is demonstrated analytically and via simulation using three evaluation criteria: total entropy of the network, total overhead/energy consumed, and the ability to frustrate heuristic-based search techniques to locate a base station.

Keywords: Sensor Network Security, Traffic Analysis

1 Introduction

In wireless sensor networks, sensor data is typically routed along relatively fixed paths from sensor nodes towards the base station. This produces quite pronounced traffic patterns that reveal the direction towards and hence the location of the base station. Figure 1 illustrates the packet traffic volume forwarded by each node in the network with the shortest path routing scheme (which we term the SP scheme). The nodes near the base station clearly forward a significantly greater volume of packets than nodes further away from the base station, in the same manner that a river grows wider as it collects more water from its tributaries. Aggregator nodes that compress the data from multiple child nodes before forwarding upstream towards the base station can mitigate the pronounced increase in traffic volume towards the base station. However, the data traffic still accumulates towards the base station, if the aggregators send their data through multiple hops.

An adversary can analyze the traffic patterns revealed in Figure 1 to deduce the location of the base station within the WSN's topology. For example,



(a) 3-D graph of data traffic. (b) Contour map of data traffic.

Figure 1: Pronounced data traffic patterns in a WSN using SP routing scheme reveal the location of the base station.

1. If the contents of a packet being transmitted are in plain text, an adversary can determine which packets are being forwarded towards the base station. This allows the adversary to follow the direction of these packets towards the base station.

2. If there is a correlation in time between the instant a node X receives a packet (a neighbor transmits that packet to X) and when node X forwards that packet, an adversary can use this time correlation to identify the same packet as it is relayed hop by hop, and thereby trace the direction towards the base station.

3. Given that there is higher communication activity near the base station, an adversary can move closer to the base station by moving towards areas of higher packet traffic.

Since the base station is a central point of failure, once the location of the base station is discovered, an adversary can disable or destroy the base station, thereby rendering ineffective the data-gathering duties of the entire sensor network.

A simple defense against plaintext observation is to encrypt each packet. However, if data packets are encrypted, but do not change hop by hop, then an adversary can still follow a given encrypted packet pattern towards its destination, which will often wind up at the base station. Following the path of encrypted packets can be defeated if each data packet is reencrypted at each hop, thereby changing the appearance of each packet at each hop, e.g. by employing pair-wise key schemes [8, 3, 7, 14, 25].

Even with hop-by-hop reencrypted packets, an adversary can still deduce significant information that can reveal the base station's location by monitoring traffic volume, or by looking at time correlations. The act of transmitting itself reveals information to the attacker, regardless of whether packet contents can be inspected. In the case of rate monitoring, the volume of transmissions can be exploited. In the case of time correlation, an adversary can listen to a transmission and also the next-hop forwarding transmission along a relay path and infer some path relationship between two neighboring nodes regardless of whether the packet is redisguised at each hop.

We therefore identify two classes of traffic analysis attacks in wireless sensor networks, a *rate monitoring*

attack and a *time correlation* attack. In a *rate monitoring* attack, an adversary monitors the packet sending rate of nodes near the adversary, and moves closer to the nodes that have a higher packet sending rate. In a *time correlation* attack, an adversary observes the correlation in sending time between a node and its neighboring node that is assumed to be forwarding the same packet, and infers the path by following the "sound" of each forwarding operation as the packet propagates towards the base station.

In this paper, we focus on developing countermeasures against traffic analysis attacks that seek to locate the base station, particularly against the rate monitoring and time correlation attacks. Given an adversary who is analyzing packet transmissions within its range, the overall objective is to significantly delay an adversary from locating a base station. In particular, our goals are:

- An adversary cannot determine a packet destination by inspecting the contents of the packet.
- An adversary cannot find the data flow direction by analyzing the time correlation between the packets sent by children nodes and packets sent by their parent nodes.
- An adversary cannot find the data transmission direction by employing statistical analysis of the packet transmission rate of every node within its range.

One way to defend against traffic analysis is to control the packet sending rate of every node in the network in such a way that every node sends packets with the same rate. In Section 3, we describe two methods to control the packet sending rate and packet sending time of each sensor node. These two methods can be used to defend against the rate monitoring and time correlation attacks. However, there are some limitations to these rate control methods. For example, they may delay data reports, or introduce too much traffic to the network.

To address these limitations, we propose four improved techniques in Section 4 that introduce randomized traffic volumes throughout the sensor network to deceive or misdirect an adversary from discovering the true location of the base station. First, a multiple parent routing scheme is introduced that allows a sensor node to forward a packet to one of its parents. This makes the patterns less pronounced in terms of routing packets towards the base station. Second, a controlled random walk is introduced into the multi-hop path traversed by a packet through the WSN towards the base station. This distributes packet traffic, thereby rendering less effective rate monitoring attacks. Third, random fake paths are introduced to confuse an adversary from tracking a packet as it moves towards a base station. This mitigates the effectiveness of time correlation attacks. Finally, multiple, random areas of high communication activity are created to deceive an adversary as to the true location of the base station, which further increases the difficulty of rate monitoring attacks.

In Section 5, we analyze our decorrelation techniques to determine their effectiveness in thwarting rate monitoring and time correlation attacks, and assess the costs of these countermeasures. The general issue is whether employing the set of decorrelation techniques outlined above is worth the cost incurred in additional messaging and hence energy expenditure, e.g. for multiple random fake paths. Consider that the lifetime of the network is actually the *minimum* of two quantities: the time to destroy the base station, which will disable the WSN; and the time until most nodes in the sensor network are depleted of energy, which will also disable the WSN. In the absence of any randomization, let us define the time to destroy the base station as T_b, the time that most leaf sensor nodes will be exhausted by typical duty cycling to be T_n, and the lifetime of the network $T = min(T_b, T_n)$. With randomization, define $T_b^{'}$ as the time to destroy the base station where $T_b^{'} > T_b$, and $T_n^{'}$ as the reduced energy lifetime where $T_n^{'} < T_n$, and $T^{'} = min(T_b^{'}, T_n^{'})$. In general, randomization is worth it only if we can increase the lifetime of the network, i.e. only if $T^{'} > T$.

Let us consider a simple example to provide an idea of the tradeoffs in employing randomization. The energy lifetime of each MICA2 sensor mote is typically $T_n \approx$ (2 to 3 months) assuming two AA batteries and a low duty cycle. T_b is approximately the average number of search steps by an adversary multiplied by the average time/search step. As shown later, under certain assumptions like shortest path routing, no randomizing defenses, and a specific search method, the number of search steps to find the base station is on the order of tens of steps. If the sensor reporting rate is sufficiently high, and each step takes say ten minutes to physically move from one hop to the next, then $T_b \approx$ (2 to 3 hours). The unprotected WSN will only survive two to three hours when under a concerted traffic analysis attack, so we desire the introduction of randomization techniques that will improve the network's survival time. As shown later, under certain assumptions, our results demonstrate that our randomization techniques significantly delay (up to 19X) an adversary from finding a base station, at the cost of introducing additional overhead and energy consumption (2-3X). In this case, $T_b^{'} \approx 19 \times T_b \approx$ 2 to 3 days, while $T_n^{'} \approx (\frac{1}{2}$ to $\frac{1}{3}) \times T_n \approx$ 1 month. In this simple example, randomization is able to extend the lifetime of the WSN by a factor of 19, while keeping the energy/overhead costs affordable since $T_n^{'} > T_b^{'}$.

A natural extension of this approach is to broadcast every packet, which achieves maximum decorrelation at maximum cost. The methods proposed in this paper, e.g. DEFP defined later, achieve close to broadcast's maximal decorrelation, as signified by maximizing the number of search steps by an adversary, at a fraction of the cost, namely about two orders of magnitude less overhead than flooding.

The proposed countermeasures are specially adapted for wireless sensor networks and exhibit several desirable properties. First, all four techniques are distributed in nature. There is no single initialization or coordination point involved to setup these mechanisms. Second, memory and computation requirements in each sensor node are relatively low, and can easily be met by modern sensors such as the MICA2 mote. Third, any compromise of one or a small number of sensor nodes by an adversary is easily tolerated. If an adversary compromises some nodes, the damage it can inflict upon the WSN is limited. Fourth, our techniques don't require a node to delay sending packets, as would be the case in the simple decorrelation approaches discussed in Section 3. A node can send/forward its packet as soon as it is ready. This aids in reducing the time delay introduced by countermeasures against traffic analysis attacks. Finally, the cost of these techniques is moderate and the techniques are applicable to large sensor networks. This is confirmed by simulation results presented in Section 5.

2 Network Traffic and Threat Model

We assume a standard wireless sensor network consisting of at least one base station collecting sensor data over multiple hops from a wireless network of sensor nodes. These nodes form a tree-structured WSN routing topology rooted in the base station, forwarding data to the base station thereby creating the pronounced traffic patterns in Figure 1. The nodes are severely resource-constrained in terms of limited memory, bandwidth, CPU, and energy. Aggregator nodes are permitted and process data from their local sensor nodes before sending the aggregated result to the base station through multiple hops. We assume that the number of base stations in a WSN is relatively small, so that the pronounced data traffic pattern shown in Figure 1 is not likely to be mitigated in any significant way by introducing multiple base stations. So, even if there are multiple base stations, an adversary can employ the same traffic analysis techniques to locate and destroy each base station one by one. We assume that the data reporting rate of the nodes generates sufficient packet traffic such that an adversary has some advantage in employing traffic analysis to discover the base station, i.e. traffic analysis gives an adversary an opportunity to reduce the network lifetime rather than waiting for energy exhaustion of

the WSN.

For the capabilities of an adversary, we assume that an adversary can monitor network traffic, and perform a *rate monitoring attack* and a *time correlation attack*. An adversary can capture sensor nodes, compromise them and obtain all information, e.g. encryption keys and routing tables, inside a node. A compromised node can be reprogrammed and converted into a malicious node. However, we assume that the adversary requires some non-trivial amount of time to compromise a node. We also assume that an adversary can physically move from one location to another in the network. However, the adversary doesn't have global information about the whole network, and cannot jam the entire network. Our solutions are designed for a large sensor network, in which an adversary cannot find the base station unless it is close by. We term the area within which an adversary can immediately identify a base station as the *base station area*. The threat of traffic analysis from the adversary is assumed to be imminent, i.e. the adversary will be present some time during the network's lifetime. In other scenarios, attacks may be infrequent or rare, so that traffic analysis countermeasures, rather than extending the lifetime, could decrease the lifetime when the probability of an attacker being present is small. We do not consider such scenarios.

We assume that sensor nodes use the key framework proposed in LEAP [25] to protect hop-by-hop communication. Nodes can set up pair-wise keys using existing protocols [8, 3, 7, 14, 25]. Every node can also set up a single cluster key [25] shared with all of its neighboring nodes. As described in [5], when a node sends a packet, it protects and encrypts the packet with its cluster key. An adversary who has not obtained the cluster key via compromise cannot decrypt the contents of a packet. At the same time, other nodes in the cluster can easily understand the type of packet and process it accordingly.

3 Basic Decorrelation Countermeasures

3.1 Hidden Packet Destination Address

The first countermeasure is to ensure that the external appearance of a packet changes as it moves forward through a multi-hop sensor network. To do this, a cluster key is established among each set of neighboring nodes. The packet destination address, packet type, and packet contents are encrypted by a node using its cluster key. As a packet moves forward, each node first decrypts the packet and then reencrypts it using the cluster key. The current sender's address remains in plaintext so that the receiver can choose the correct cluster key to decrypt the packet. The format of a packet is

$$ID_{src}||E_{KC_{src}}(type||ID_{dst}||data)$$

When a node receives this packet, it checks ID_{src} and decides which cluster key to use to decrypt the packet. After decrypting the rest of the packet, a node checks if it is the destination of the packet. Integrity can be added via a MAC based on the cluster key.

The net effect is that the packet's entire appearance is transformed at every hop along its path, making it difficult for an eavesdropper to trace the path of the packet. Hop-by-hop reencryption spatially decorrelates the packet's appearance. Unless an attacker can compromise a sender's neighboring node and obtain the cluster key, it won't know the contents of the packet. If an attacker compromises a node s and obtains all the keys inside the node, it will be able to decrypt the packets sent by s's parent node, and can then track two hops towards the base station, but cannot track beyond that.

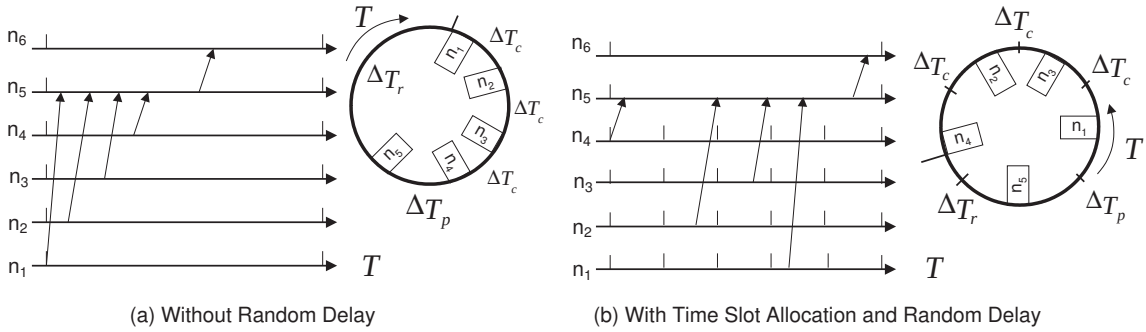


Figure 2: Decorrelating packet send times via random delays.

3.2 Decorrelating Packet Sending Times

Packet encryption can hide a packet destination, but cannot hide its sender. By carefully monitoring the packet sending time of every node, an adversary may obtain some information about data traffic flows. For example, if a parent node s receives a packet from its child node c and forwards that packet immediately, an adversary can observe the short time interval between s and c and eventually infer the parent-child hierarchy given sufficiently long observations.

To prevent this, we decorrelate the packet sending times between a parent node and its child nodes. Here we only consider the situation that every node sends data at the same rate. This situation occurs when every node regularly aggregates data from its children nodes and sends a result to its parent node. Suppose all child nodes and parent nodes report their data during time period T. Let's denote the time interval between two child nodes sending packets as Δt_c (we assume sensor nodes use a MAC layer protocol to avoid packet collisions), the time interval from the last child node sending data to the parent node sending data as Δt_p, and the time between a parent node sending data and its grandparent forwarding data as Δt_r. We denote $\overline{\Delta t_c}$, $\overline{\Delta t_p}$, $\overline{\Delta t_r}$ as the average value of Δt_c, Δt_p, and Δt_r. If the differences between $\overline{\Delta t_c}$, $\overline{\Delta t_p}$ and $\overline{\Delta t_r}$ are observable, an adversary may be able to extract which node is the parent node after monitoring the network for an extended period of time.

If the parent node and child nodes send packets with the same rate, sensor nodes can introduce random delay between packet sending times. This makes the differences between $\overline{\Delta t_c}$, $\overline{\Delta t_p}$ and $\overline{\Delta t_r}$ unobservable. To do this, first the time period T is divided into m slots, if there are $m-1$ child nodes and 1 parent node. Every node is assigned a slot and randomly chooses a time within its slot to send its packet. For example, in Figure 2, the time slot assignment algorithm is centered at the parent node. The parent node informs each child node of its time slot with a secure unicast message. Nodes n_1 to n_4 are n_5's child nodes, and n_6 is n_5's parent node. Figure 2(a) shows every node sends its packet as soon as it can. The differences between Δt_c, Δt_p and Δt_r are correlated. Figure 2(b) shows that n_1 to n_5 occupy different time slots and each node sends its packet randomly within its time slot. The differences between Δt_c, Δt_p and Δt_r are indistinguishable. Experiments show that a sensor node only spends about 40 to 50 milliseconds to send a 36-byte packet. Normally, a sensor reports data once per minute or once per tens of seconds. In a connected sensor network, a sensor node may have 10 to 20 neighboring nodes. So the time slot is big enough for a sensor node to successfully send its packet.

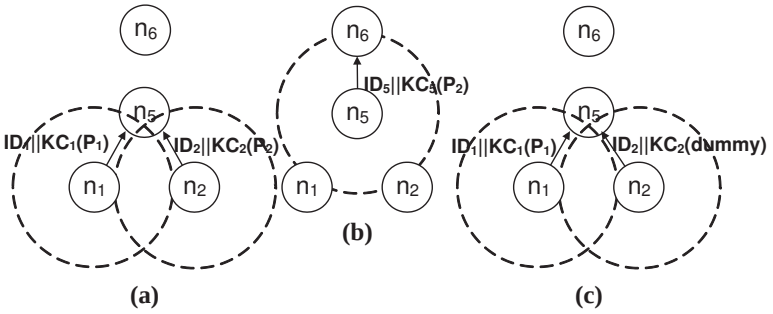


Figure 3: Rate control scheme. (a) node n_1 and n_2 send their to parent node n_5; (b) node n_5 forwards n_2's packet; (c) node n_1 continues to send its packet, and n_2 sends a dummy packet to n_5.

3.3 Controlling Packet Sending Rates

In the previous subsection, we assumed that every node sends packets at the same rate. However, in some cases, different sensor nodes may send packets with different rates. For example, the base station may require that each sensor node sends its neighborhood information (which contains the IDs of its identified neighboring nodes) back to the base station. We call this a *topology report*. The topology report helps a base station to update its complete network topology picture. The end user can use this information to learn what sensor nodes and base stations are unreachable. For the topology report messages, a parent node has to forward every message from its children nodes, and aggregation is avoided. If every node sends a packet with the same rate, then nodes closer to the base station will sustain larger sending rates. By monitoring packet sending rates, an adversary can track the base station.

Our solution is to set the packet sending rate control between a parent node and its children nodes. That creates a uniform sending rate across the entire sensor network, so that every node behaves like every other node in terms of traffic volume. When a parent node has a packet in its buffer to send, it won't accept any packet from its children nodes. When the parent has sent out its packet, it accepts one packet from its children nodes and saves that packet into its buffer. All children nodes are monitoring the packet sent out by their parent node, because they have the parent's cluster key. If a child node finds that its packet was just transmitted by its parent node (that means its parent node has received its packet), or if it finds its parent beginning to send dummy packets (that means the parent node has an empty buffer), then the child begins to accept a new packet from its children nodes. Otherwise it will continue to send the same packet to its parent node. If a node doesn't have any packet to send, it just injects a dummy packet to its parent, until the whole topology reporting process stops. The base station can send a broadcast message to start and stop the topology reporting process. This rate control scheme is depicted in Figure 3; Figure 4 describes the algorithm. This algorithm implements rate control, and it is robust, handling the case when the child node fails to hear the parent node forward its packet.

3.4 Limitations

There are some limitations to the prior methods. First, they confine the data sending pattern of the network. They require every node to aggregate data from its downstream nodes. Second, these methods do not work if there is urgent data that needs to be sent to the base station as quickly as possible. Third, imposing a uniform sending rate results in dummy packets being sent for those nodes that have less data to contribute than the global sending rate. The dummy packets increase the energy expenditure of such nodes. Finally,

```
while (1) {
   send P_s to parent node
   listen to packet sending of neighboring nodes
   if receive packet p
   if (p.sender == parent_node) {
      if ((p == P_s)||(p == dummy)){
         P_s ← dummy
      }
   } else if (p.sender ∈ s.children) {
      if (p ≠ dummy&&P_s == dummy) {
         P_s ← p
      }
   }
   wait for next time slot
}
```

Figure 4: Algorithm for packet sending control. (P_s is the packet to send)

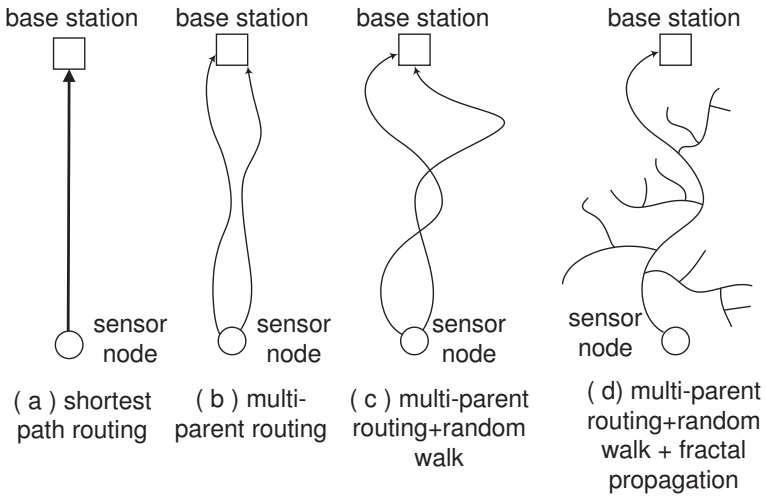


Figure 5: Techniques to counter traffic analysis.

they introduce extra delays in forwarding packets at each hop, cumulatively increasing the time to deliver data from source nodes to the base station. In the next section, we introduce more advanced schemes to defend against traffic analysis attacks.

4 Inhibiting Traffic Analysis Attacks With Randomized Traffic

This section introduces a suite of randomized network traffic techniques that improve upon Section 3's basic traffic analysis countermeasures. These techniques do not impose onerous limitations such as a high uniform sending rate throughout the network or enforced aggregation at every hop, as in the previous section. Figure 5 summarizes each of the major randomization techniques that we introduce in this section.

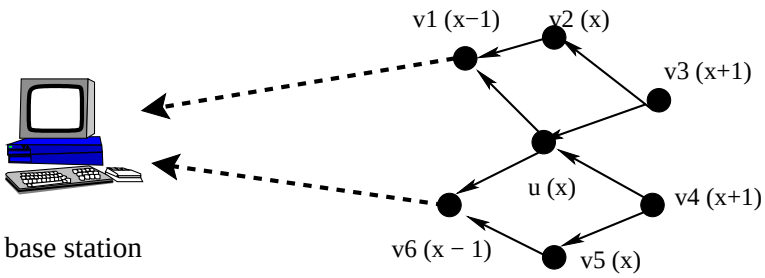


Figure 6: Neighbors and parents of node u. Figure shows node ID and its $level$ value. In SP, node u has one parent node v_1. In MPR, node u has two parent nodes, v_1 and v_6. In RW, u forwards packets to v_1 with probability p_r or v_6 with probability $1 - p_r$.

4.1 Multi-parent routing scheme

To reduce the starkness of pronounced paths caused by shortest path (SP) routing, as shown by the direct node-to-base station route in Figure 5(a), we require each node to randomly select one of multiple parent nodes to route data to the base station, as shown in Figure 5(b). When a node needs to forward a packet, the node randomly selects one of its parent nodes to forward the packet. We call this scheme multi-parent routing (MPR). We propose two methods for setting up multiple parents for each node. In the first method, as shown in Figure 6, whenever a base station sets up a routing structure, it broadcasts or floods a message called the beacon. The beacon message contains a *level* field. The base station sets the value of *level* to 0. When a node forwards a beacon message, it increments it by 1. So the value of *level* represents the number of hops that a node is from the base station along a particular path. A sensor node s selects all neighboring nodes whose *level* value is less than s's *level* value as its parent nodes. In the second method, a node monitors all beacon messages it receives before forwarding the first beacon message. Since a node s has to wait for some amount of time before forwarding a beacon message (waiting time in MAC layer), it selects all nodes from whom it receives a beacon message while waiting to forward the first received beacon message as its parent nodes.

An adversary has several ways to attack these multi-parent routing setup schemes. For example, a malicious node can claim a low *level* value to attract traffic from other nodes, or it can use unfair media access control mechanisms to occupy the wireless channel. Protecting routing schemes from such attacks is beyond the scope of this paper. Here we assume that the routing set up scheme is relatively fast, so an adversary doesn't have enough time to attack the routing set up process. Several mechanisms [13, 5] have already been proposed to protect against attacks during the routing setup. Notice that the *level* information will be erased by every node after the routing paths are set up. So even if a node is compromised, an adversary won't know the distance to the base station because the *level* information of the compromised node won't be available.

4.2 Random Walk

To further diversify routing paths and mitigate rate monitoring attacks, we propose a random walk (RW) routing scheme. In RW, when a node receives a packet, it forwards the packet to one of its parent nodes with probability p_r. However, it uses a random forwarding algorithm with probability $1 - p_r$. In the random forwarding algorithm, the node forwards the packet to one of its neighboring nodes with equal probability. Like [12] and [23], MPR and RW use probabilistic routing. However, [12] and [23] use probabilistic routing for reliable data transmission in sensor networks, while we use probabilistic routing to defend against the *rate monitoring attack*.

The RW technique results in some packets traversing a longer path to reach the base station than the shortest available path, as shown in Figure 5(c). This implies that RW will consume more energy per node on an average. To estimate how much extra energy is consumed by RW, we calculate the cost C of RW, where cost is defined as [5]: $C = \frac{M'}{M}$. Here, M' is the average number of hops a packet takes to reach the base station from an aggregator node in RW, and M is the number of hops a packet takes to reach the base station from the same aggregator node in SP. Clearly, M' depends on the several factors related to network topology, e.g. how many neighbors a sensor node has, how far the base station is from a sensor node or from one of its neighboring nodes, and so on. We calculate the value of C by making the following simplifying assumption. Suppose a node u randomly selects a neighboring node v to forward a packet, and the distance (number of hops along the shortest path) between v and the base station is d, while the distance between u and base station is d'. We assume that the probability that $d > d'$ is the same as the probability that $d < d'$. So on average, when u forwards a packet to v, the distance from the base station doesn't change. Only when u forwards the packet to its parent node, the distance is reduced by 1. We denote n as the number of hops from the aggregator to the base station in SP, and n' as the number of average hops in RW. We have $n' \times p_r = n$. This implies $C = \frac{M'}{M} = \frac{1}{p_r}$.

In addition, a packet will take a longer time to reach the base station in RW. In fact, the extra time delay is directly proportional (linear) to the extra hops used for forwarding the packet. So, the time cost for each packet to reach the base station is roughly $\frac{1}{p_r}$ in RW.

4.3 Fractal Propagation

MPR and RW spread out data traffic and make it difficult to use a rate monitoring attack. However, RW is still vulnerable to the *time correlation* attack. Usually, for a node s, the number of parent nodes is less than half of s's neighboring nodes, and for energy and efficiency considerations, $p_r > 0.5$ typically. As a result, the possibility that a node forwards a packet to its parent node is higher than the possibility it forwards the packet to any one of its other neighbors. An adversary can exploit this to launch a time correlation attack, either by injecting abnormal report data or monitoring over a long period of time.

To address the shortcomings of MPR and RW, we propose a new technique called *fractal propagation*. In this technique, several *fake* packets are created and propagated in the network to introduce more randomness in the communication pattern. When a node hears that its neighboring node is forwarding a packet to the base station, the node generates a fake packet with probability p_c, and forwards it to one of its neighboring nodes. To control the propagation range of the fake packet, each newly generated fake packet contains a *length* parameter with value K. K is a constant that is known to all nodes, so an adversary cannot flood the whole network by sending fake packets with *length* parameter higher than K. When a node receives a fake packet, it decrements *length* by 1. If the value of *length* is greater than zero, the node forwards the fake packet to one of its neighboring nodes (not necessarily in the direction of the base station). If the value of *length* is zero, a node stops forwarding the fake packet. In addition, when a node hears that its neighboring node is forwarding a fake packet to someone else with *length* value k ($k < K$), it generates and forwards another fake packet with probability p_c and *length* value $k - 1$.

These fake packets spread out in the network and their transmission paths form a tree (see Figure 5(d)). In particular, the communication traffic is much more spread out than RW. So even if an adversary can track a packet using time-correlation, she cannot track where the real (as opposed to fake) packet is going. This is because she cannot differentiate between a real and a fake packet without knowing the encryption key.

Suppose a node has x neighboring nodes on average. Let $p_f = p_c \times x$ and $f(K)$ represents the total length of

a fake tree that originated with *length* value K. We have

$$f(K) = p_f \times f(K-1) + f(K-1) + 1$$

Solving this recursive equation, we get

$$f(K) = \sum_{i=0}^{K-1} (p_f + 1)^i = \begin{cases} \frac{(p_f+1)^K - 1}{p_f} & \text{if } p_f > 0 \\ K & \text{otherwise} \end{cases}$$

Suppose the length of real path from the aggregator node to the base station is n. The cost is

$$C = \frac{M'}{M} = \frac{n + n \times p_f \times f(K)}{n} = \frac{n + n \times p_f \times \frac{(p_f+1)^K - 1}{p_f}}{n} = (p_f + 1)^K$$

If we combine RW and the fractal idea, the total cost is

$$C = \frac{(p_f + 1)^K}{p_r}$$

If we use fixed values of p_r, p_f and K, the average cost is a fixed value that is independent of the size of the network.

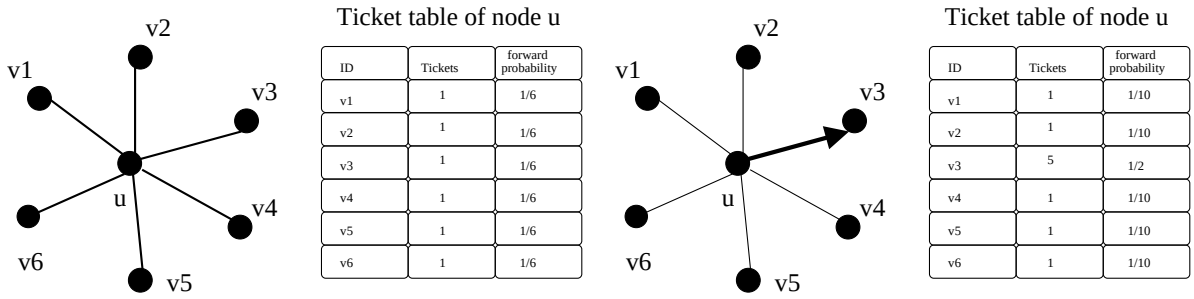
4.3.1 Fractal propagation with different forking probabilities

One problem with simple fractal propagation is that it generates a large amount of traffic near the base station. This will potentially increase the packet collision rate and packet loss rate.

To address this problem, nodes can use different probabilities to generate fake packets. When a node forwards packets more frequently, it sets a lower probability for creating new fake packets. This technique is called Differential Fractal Propagation (DFP). The algorithm for setting this probability is as follows. When the packet forwarding rate r at a node is lower than a threshold h, the node generates new fake packets with probability p. When the packet forwarding rate is higher than h, the node generates new fake packets with probability $p' = p \times (h/r)^2$; h can be chosen as the packet sending rate of the aggregator node.

4.3.2 Enforced fractal propagation

The idea of fractal propagation aids significantly in spreading out the communication traffic evenly over the network and obfuscating any paths to the base station. To make matters worse for an adversary, we generate local high data sending rate areas, called *hot spots*, in the network. An adversary may be trapped in those areas and not be able to determine the correct path to the base station. This routing technique is called Differential Enforced Fractal Propagation (DEFP). The challenge here is how to create hot spots that are evenly spread



(a) before node u forwards any fake packet (b) after u forwards a fake packet to node v_3

Figure 7: Ticket table of node u

out in the network, such that only a minimum (preferably zero) amount of extra communication/coordination among the sensor nodes is needed.

DEFP is a simple distributed algorithm based on DFP. The key idea is to let the nodes that forwarded fake packets earlier have a higher chance to forward fake packets in the future. In particular, if a node u forwarded a fake packet to another node v in the past, then it forwards the next fake packet received to v with a higher probability. The node uses a *lottery scheduling* algorithm [22] to choose the next node to forward the fake packet to. In this algorithm (see Figure 7), a node assigns tickets to each of its neighboring nodes. It chooses the next node to forward a fake packet to based on the number of tickets assigned to the neighboring nodes. A neighboring node with more tickets assigned has the higher probability of being chosen. In the beginning, all neighboring nodes are assigned one ticket. When the node chooses a neighboring node as the next node for forwarding a fake packet, it increments that node's tickets by k. This way, after a node has forwarded a fake packet to one of its neighboring nodes, it will continue to forward other fake packets to the same neighboring node with higher and higher probability. If an area of nodes receive fake packets, they are more likely to process more and more fake packets in the future. This will turn that area into a hot spot. It is also very easy to destroy current hot spots and reconstruct new hot spots at different places. For example, sensor nodes just reset the value of tickets to 1 when they receive a broadcast message from the base station, and then start to build hot spots from scratch. A patient attacker can wait at a hot spot until the communication pattern changes. While this will allow the attacker to determine that he was at a fake hot spot, it does not provide any other information about the possible location of the base station. Furthermore, waiting for a long time at a fake hot spot will add more delay to finding the location of the base station.

4.4 Node Compromises

If an adversary compromises a node, she can normally find out the identity of its parent nodes, and read the contents of all packets passing through this node. In addition, by monitoring the traffic for some sufficiently long period of time, she can obtain distribution information about all the ancestor nodes within her activity range. However, with this knowledge, she cannot determine the location of the base station, and cannot block communication between an aggregator node and the base station. To determine the location of the base station, the adversary will have to compromise a large number of nodes along the path to the base station.

To further minimize the damage of node compromise, we propose the *directional pairwise ID* mechanism such that every node has different IDs to its children nodes and parent nodes. When a node u is compromised, an adversary knows its parent node p's ID. However, the parent p only uses that ID to accept packets, and doesn't use that ID to send packets. So the adversary cannot know which node is p by listening to packets.

Hops	Average Number of Compromised Nodes
5	53.7
10	87.1
15	120.8
20	152.6
25	181.2

Table 1: Average number of nodes that need to be compromised to trace back to the base station given directional pairwise ID's.

She has to compromise u's neighboring nodes one by one, until capturing p.

Concretely, every node shares a unique pair of IDs with each of its parent nodes. For example, suppose node u has parent node p_1 and p_2. Then u has an ID ID_{up_1} to p_1, and p_1 has an ID ID_{p_1u} to u. Thus u and p_2 shares another pair of IDs ID_{up_2} and ID_{p_2u}. When u sends a packet to p_1, it uses the format that

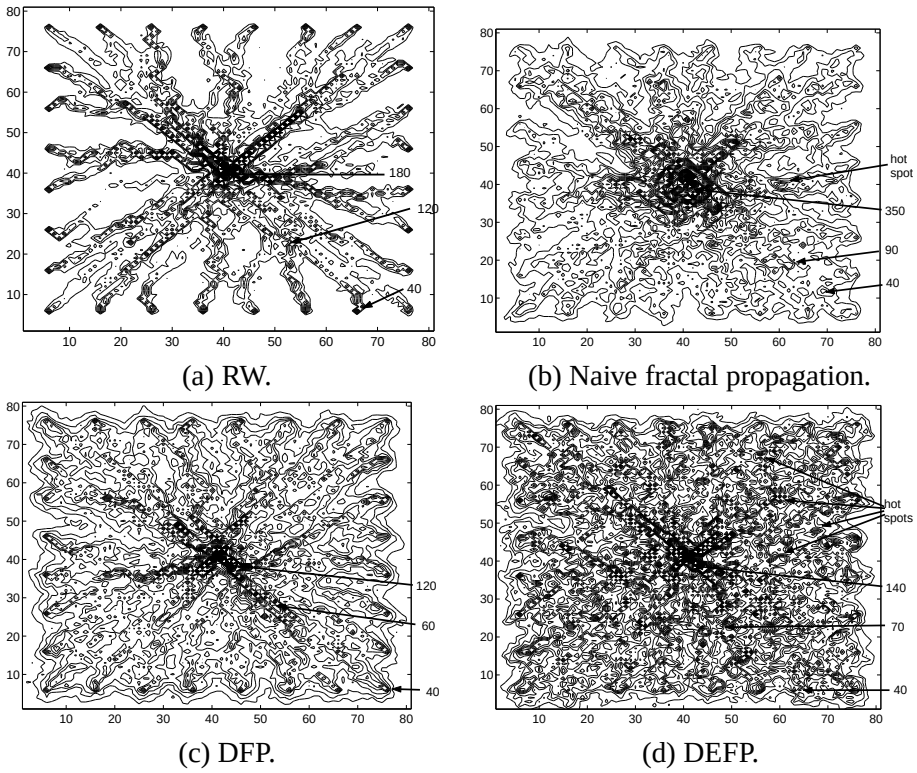
$$ID_{up_1}||E_{KC_u}(type||ID_{p_1u}||data)$$

If u is compromised, the adversary knows KC_u, ID_{up_1} and ID_{p_1u}, but she doesn't know p_1's IDs to p_1's parent nodes. She has to compromise u's neighboring nodes which are not u's children nodes, until she captures p_1. This mechanism increases the workload of node compromise. Table 1 shows the average number of nodes that need to be compromised in a random distributed network under a single parent routing scheme. We can see that if the first compromised node is 15 hops away from the base station, the adversary needs to compromise about 120 other nodes, instead of 15 nodes, to get to the base station. This shows that an adversary has to compromise a relatively large number of nodes along the path until reaching the base station.

In fractal propagation, if an adversary compromises a node, she can find out whether a packet is a fake or real. However, she cannot obtain any information other than what was discussed above. The adversary can attempt to launch a DoS attack by generating several fake packets and forwarding them to flood the network. However, the propagation area of a fake packet is limited by the value of the *length* parameter. A fake packet can propagate and generate new fake packets only within a small part of the network, so the damage due to such DoS attacks is limited to a small part of the network.

Cooperating adversaries can launch a collaborative attack. For example, two cooperating adversaries on different sides of a WSN can respectively determine the direction (a vector) where a base station is possibly located from their current location by analyzing packets over just a few hops. They can then form an estimate of the base station's location by intersecting the two vectors. However, such an attack requires that the direction in which a parent node is located is precisely the direction towards the base station. This is quite unlikely in a randomly distributed sensor network. In addition, MPR increases the difficulty in determining the precise geographic direction towards the base station, forcing the adversary to compromise a large number of nodes.

Finally, an adversary can also generate several forged data packets and forward them to the base station in an attempt to flood the base station. However, mechanisms currently exist that allow intermediate nodes to filter out forged data packets, e.g. see [24, 26]. In these mechanisms, intermediate nodes use randomly pre-distributed pair-wise keys to verify the authenticity of the data sent by the aggregator node. Forged packets are filtered out by each intermediate node with a certain probability and thus prevented from propagating over a long path.



(a) RW. (b) Naive fractal propagation.

(c) DFP. (d) DEFP.

Figure 8: Number of packets sent/forwarded by each node employing cumulatively more methods of randomization.

4.5 Simulation Results and Summary

We implemented our randomization techniques in our simulator, which is based on a standard discrete event generator. Figure 8 shows the cumulative routes taken by packets through a sensor network employing each of the techniques. The simulation results demonstrate that RW creates a more diffuse routing pattern than SP (compare to Figure 1). Similarly, fractal propagation techniques, DFP and DEFP introduce increasing degrees of randomness respectively that obscure the location of the base station. The network configuration for these simulations is a grid network described in Section 5.

A summary of the problems addressed by each of the randomization techniques presented in this section is given in Table 2. The set of techniques based on fractal propagation address both rate monitoring and time correlation attacks. Together, Table 2 and Figure 8 provide an intuitive sense of the scope and cost of these randomization techniques.

5 Evaluation

5.1 Evaluation Criteria

For the basic schemes described in Section 3, the cost of a rate control mechanism is assessed. For the randomized traffic based schemes described in Section 4, our evaluation measures how random the network

Randomization Technique	Rate Monitoring Attacks	Time Correlation Attacks
Multi-Parent Routing (MPR)	$\surd$	
Controlled Random Walk (RW)	$\surd$	
Fractal Propagation (FP)	$\surd$	$\surd$
Differential Fractal Propagation (DFP)	$\surd$	$\surd$
Differential Enforced Fractal Propagation (DEFP)	$\surd$	$\surd$

Table 2: Randomization techniques and the problems they address.

traffic is, and the cost of randomized traffic. In particular, we measure the randomness of network traffic and its effectiveness against *rate monitoring* attacks. We employ two metrics —*entropy* of the network traffic and the *GSAT* test. To estimate the cost of our techniques, the number of messages exchanged in our techniques is counted and compared with the number of messages exchanged in Shortest Path (SP) forwarding. Since our techniques incur very little memory cost on each sensor node, e.g. a few encryption keys and tickets of neighboring nodes, memory consumption overhead was not measured in our simulation. Also, the effectiveness of randomization in defending against *time-correlation* attacks was not evaluated. Intuitively, a higher forking probability (p_f) and a longer $length$ of fake path will make it more difficult to launch a *time correlation* attack. However, measuring the effect of this increased difficulty is more challenging, and is left as future work.

Entropy We use entropy to measure the randomness of network traffic. Entropy is a mathematical measure of information uncertainty, and it has been widely used as a metric to measure randomness in many applications, e.g. data communication, data compression, random number generators, and security of cryptographic algorithms. Entropy of a random variable X with a probability function $p(x)$ is defined as

$$H(X) = -\sum p(x)\log_2 p(x)$$

Suppose that during a time period T, a sensor node a sent/forwarded p_a packets, and a total of M packets were sent/forwarded in the WSN N. We define the network entropy with the following formula and use it to estimate the randomness of N during the time period

$$T : H(N) = -\sum_{a \in N} \frac{p_a}{M} \log_2 \frac{p_a}{M}$$

In general, a higher value of $H(N)$ implies that the communication traffic pattern of N is more random.

GSAT Test The GSAT test is intended to measure the ability of a randomization technique to guard against heuristic-based search algorithms that an adversary may use to locate a base station. The GSAT algorithm [19] was proposed for solving NP-hard satisfiability problems, such as the 3SAT problem [4]. In contrast to the traditional deterministic solutions, GSAT is a probabilistic algorithm that combines a hill-climbing search algorithm with a random restart mechanism. GSAT can solve most of the large 3SAT instances in a short time.

We use the idea of the GSAT algorithm to design a heuristic-based algorithm that an adversary employs to track the location of the base station. In this algorithm, an adversary starts at some location in the sensor network N. She monitors network traffic around her within her activity range. If she finds that a different node s within her activity range has the highest traffic, she moves to s, and continues to monitor traffic from s. Using this mechanism, she can move towards the locations that have higher and higher traffic volume. However, if she reaches a location that has the highest traffic within the neighborhood (local maxima), she

	Size	Average # Neighbors	Number of Aggregators	Sending Rate
Grid	81×81	8	28	4/minute
Random	4500	20	28	4/minute

Table 3: Network configuration Parameters

selects a direction at random, moves in that direction for some time, and then repeats the above algorithm. She continues to do this until she finds the base station.

The GSAT test measures the average number of hops an adversary takes to finally reach the base station using this heuristic algorithm. A large value of the GSAT test implies that the routing technique has better potential to guard against heuristic-based search algorithms that an adversary may employ to locate a base station.

In addition to the degree of randomness, the exact values of entropy and the GSAT test depend on several other network characteristics, e.g. network structure, network size, number and location of aggregator nodes. To evaluate our techniques, we have focused on differences in entropy and GSAT test values measured under the cases when one of the proposed traffic analysis countermeasures is used and the case when no traffic analysis defenses are used. We also experimented with different values of P_r in RW and P_f in DEFP, to understand the effects of these parameters. We simulated two network structures in our experiments: a grid topology and a random topology. Table 3 shows the parameters used in our simulation. Aggregators are located at the four edges of the network.

5.2 Message Overhead of Rate Control Mechanism

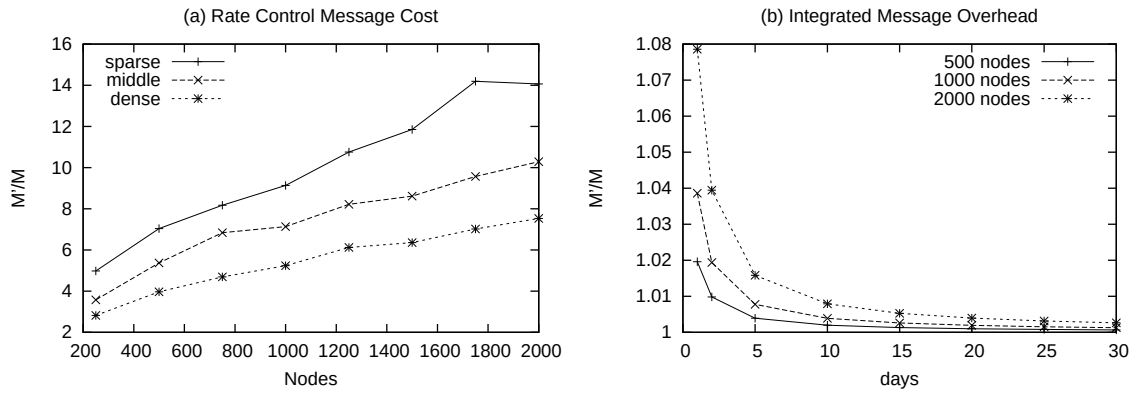


Figure 9: Overhead of uniform rate control countermeasures to traffic analysis.

We define $C = \frac{M'}{M}$ to measure the data transmission overhead of our basic suite of traffic analysis countermeasures, where C is the cost measurement, M is the number of messages without basic traffic analysis defenses, and M' is the number of messages with the our basic traffic analysis countermeasures. In this experiment, we simulated and measured the message overhead of the rate control scheme, since it introduces extra "dummy" packets. We ran three groups of tests. For each group of tests, we employed a different network topology. These networks differed from one another in the number of nodes, but had the same node density. The number of nodes varied from 250 to 2000. For each test, sensor nodes were randomly deployed in the network area.

	Entropy		Traffic		Center Traffic	
	(SP)	(BR)	(SP)	(BR)	(SP)	(BR)
Grid	9.64	11.40	39000	7×10^6	10080	4×10^5
Random	8.20	12.08	21000	5×10^6	2792	1.8×10^5

Table 4: Entropy and number of messages exchanged in SP and BR. (Traffic means the total messages exchanged in the network, and Center Traffic means the number of messages exchanged in the close vicinity of the base station.)

We set up routing, and measured M' and M. For the same number of nodes with the same network density, we repeated the test 50 times and calculated the average value. Figure 9 (a) shows the simulation result of $\frac{M'}{M}$ for three different network densities. We can see that the overhead of our rate control strategy increases as the size of the network increases. Our initial analysis and experiments show that $\frac{M'}{M} \propto \sqrt{N}$, where N is the size (number of nodes) of the network. That means the rate control overhead is not scaling directly with the size of the network.

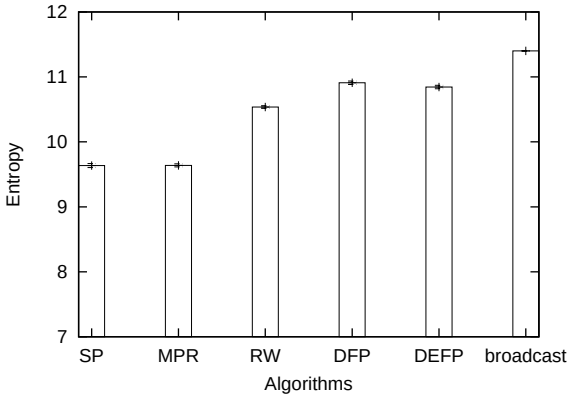
However, if topology information is not required frequently, the overhead of the rate control scheme only occupies a small part of the total cost. The network traffic is dominated by regular sensed data report, whose anti-traffic message overhead is 1. Figure 9 (b) shows the total message overhead combining sensor data packets and topology reports over an intermediate density network. We assume that every node reports its data once per minute, and the base station requires a topology report every one to thirty days. Figure 9 (b) shows that the total overhead reduces as the base station requires topology reports less frequently. For example, if the topology report is performed once a week, total overhead is less than 1.01. In this context, the overhead of sending "dummy" packets is much less noticeable.

5.3 Effectiveness and Cost of Randomized Traffic Techniques

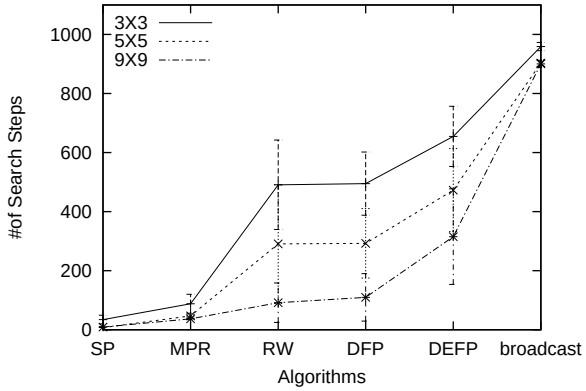
To evaluate the effectiveness of our randomized traffic countermeasures, we simulated them over a grid network (see Table 3) and measured the values of entropy, GSAT test, and energy cost (number of messages exchanged). We simulated the following techniques: MPR, MPR+RW, MPR+RW+DFP, and MPR+RW+DEFP. For simplicity, we use MPR, RW, DFP, and DEFP respectively to refer to these techniques in the rest of the paper. In these simulations, we set p_r to 0.6, p_f to 0.2, and K to 6.

To obtain an estimate of an upper bound of entropy and GSAT values, and a lower bound on the cost, we simulated two routing mechanisms. The first routing mechanism is SP, which selects the shortest path to the base station from each sensor node. SP provides a measure of lower bound on the cost of routing, but results in very pronounced communication patters as shown in Figure 1. The second routing mechanism is called the broadcast scheme (BR scheme). In this mechanism, every message sent by an aggregator node is flooded to the entire network. Since BR generates uniform network traffic, it provides a measure of an upper bound of entropy and GSAT values. Table 4 shows the entropy values and number of messages exchanged in SP and BR.

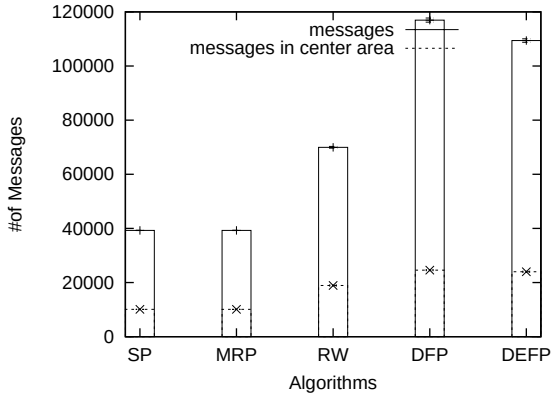
Figure 10 (a) shows the entropy measured for various routing techniques. All data reported here are an average over 20 runs. As expected, entropy is lowest for SP and highest for broadcast. Entropy for MPR and RW is higher than SP, but lower than DFP and DEFP, which confirms the progression of increasing randomness revealed in Figure 8's various subgraphs. Figure 10 (a) demonstrates that the idea of generating fake packets



(a) entropy



(b) GSAT test: number of search steps



(c) message cost

Figure 10: Effectiveness and cost of randomization countermeasures against traffic analysis.

in a controlled manner does aid in making the network traffic pattern more random. This is in addition to the original goal of defending against time-correlation analysis.

To determine resiliency against a GSAT search, we simulated the data traffic and recorded the number of packets sent/forwarded by each node in a log file. We initialized a starting point for the adversary in the network and used the GSAT algorithm to discover the *base station area*. We recorded the number of steps the adversary takes to get into the base station area. For each log file, we set 81 different initial locations. For each initial location, we ran GSAT to search for the base station area for 100 times, and recorded the number of hops the adversary takes to get into the base station area. Finally, we computed the average number of hops the adversary takes to get into the base station area for each technique. In addition, we experimented with three different activity ranges of the adversary: adversary could monitor data traffic over 3×3, 5×5, and 9×9 areas around her respectively.

Figure 10 (b) shows the results of the GSAT test. First, we notice that randomization countermeasures significantly increase the number of steps an adversary has to take to locate the base station. The addition of each technique increases the frustration of the adversary, with varying degrees of effectiveness. For example, she can discover the base station area in 34 steps in SP (activity range 3×3), and 653 steps in DEFP, which is about 19 times more. Notice that the number of search steps needed when all packets are broadcast is only about 1.5 times more than the number of steps needed in DEFP approach. In this sense, DEFP achieves close to the maximum decorrelation upper bound represented by broadcast.

The activity range of an adversary also impacts the GSAT value. If the activity range is larger, the corresponding GSAT value is smaller. This implies that the adversary can find the base station in fewer hops. Even when the activity range of the adversary is large (9×9), our traffic analysis defenses significantly increase the number of hops an adversary has to take to locate the base station area.

Also, we see that the GSAT values correlate with the entropy values shown in Figure 10 (a) (except DEFP). Higher entropy corresponds to a larger value of GSAT. This implies that both entropy and GSAT are useful metrics to measure the randomness in network traffic. The only exception is DEFP. Since DEFP converges some traffic together to form *hot spots*, it results in less entropy compared to DFP. However, those *hot spots* make it more difficult for an adversary to locate the base station using a GSAT search algorithm. This is evident from the higher values of GSAT in DEFP.

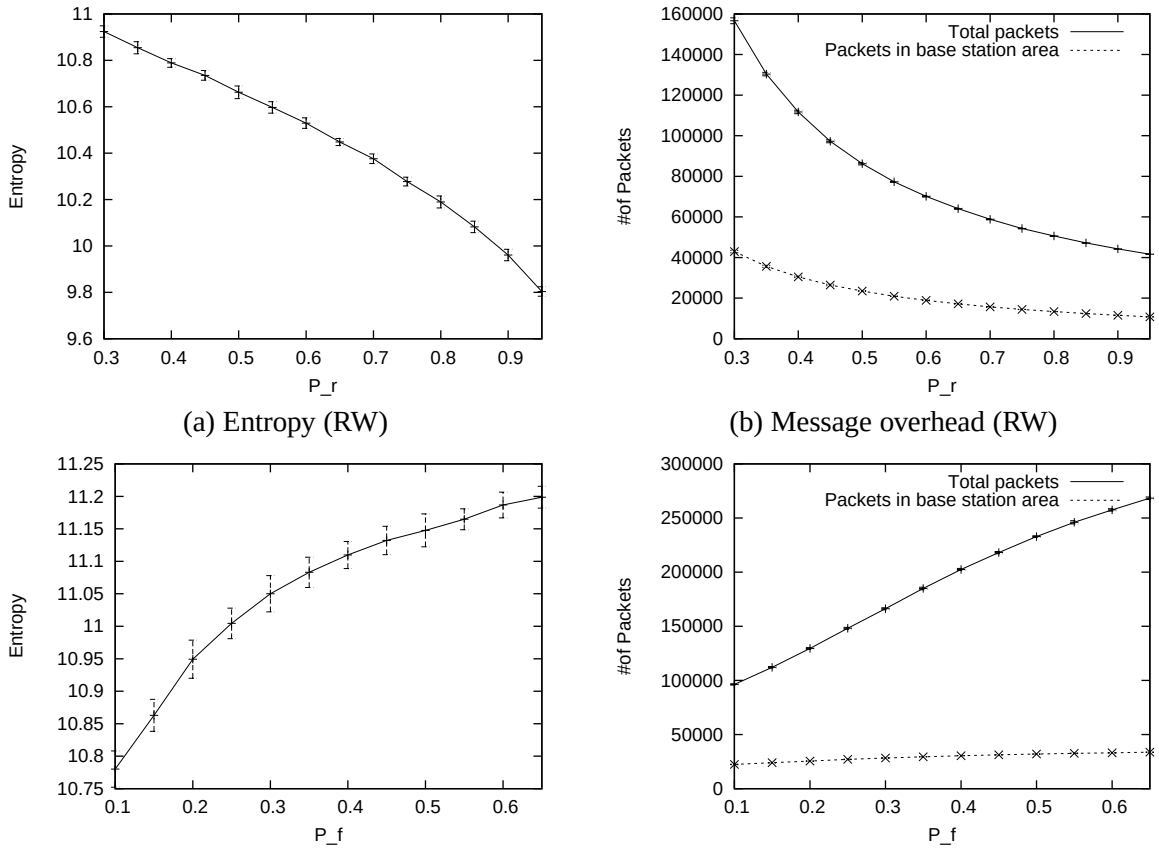
Figure 10 (c) shows the cost of randomization in terms of the total number of messages sent/forwarded by all nodes in the network, and the number of messages sent/forwarded by nodes near the base station (which is an area of 20×20 nodes with base station at center). The total traffic in RW is about 1.8 times larger than the total traffic in SP for the whole network and the area near the base station. The total message cost of DFP and DEFP is about 2.8 times the message cost of SP in the whole network, and 2.4 times near the base station. In our simulation, when aggregators send four packets per minute, the nodes directly connected to the base station forward about 14 packets per minutes in SP, and about 34 packets per minute in DEFP. This is easily feasible in the current sensor network technology. An important point to note is that the message cost of these algorithms is constant. It doesn't increase with increasing network size. The final observation is that broadcast costs about 70 times more messaging than DEFP, i.e. Table 4 give $7x10^6$ as the number of messages for broadcasting in a grid while Figure 10 (c) gives about 10^5 as the number of messages for DEFP. The overhead messaging cost provides an indication of the energy cost to the WSN, since packet transmission costs over a thousand times more energy per bit than computation, but is not a precise equivalence due to other factors such as duty cycling and degree of computation.

The big picture emerging from Figure 10's three graphs is that our most advanced randomization suite DEFP, equivalent to MPR+RW+DFP+DEFP, achieves nearly the best decorrelation capabilities afforded by broadcast at a fraction of the cost. DEFP achieves close to broadcast flooding's maximum decorrelation, within about 50% of the maximum number of GSAT search steps required of an adversary. Yet, DEFP's overhead of a hundred thousand messages is 70 times less than pure broadcast's overhead obtained from Table 4. This highlights the considerable advantage gained by employing our randomization algorithms.

5.4 Effectiveness of p_r and p_f

To understand the effect of different values of p_r and p_f, we varied parameters for RW and DEFP. We simulated them on both a grid network and a random network (Table 3). In RW, we varied p_r from 0.3 to 0.95. In DEFP, we fixed p_r at 0.6, and varied p_f from 0.1 to 0.65. The results are shown in Figures 11 and 12. We notice that the variation in the values of entropy and message cost based on p_r and P_f is similar in both grid and random networks. In RW, the entropy sub-linearly decreases and the number of messages decreases with increasing p_r. In DEFP, entropy sub-linearly increases and the number of messages dramatically increases with increasing p_f.

These results imply that we should chose p_f as small as possible, as long as it satisfies our requirements. In Section 4, we analyzed the relation between message cost, and p_r and p_f. The results from these experiments imply that there is a relation between the entropy of network traffic, and p_r and p_f, which is independent of the size of the network. Another observation is that although the total number of messages exchanged is quite



(a) Entropy (RW) (b) Message overhead (RW)

(c) Entropy (DEFP) (d) Message overhead (DEFP)

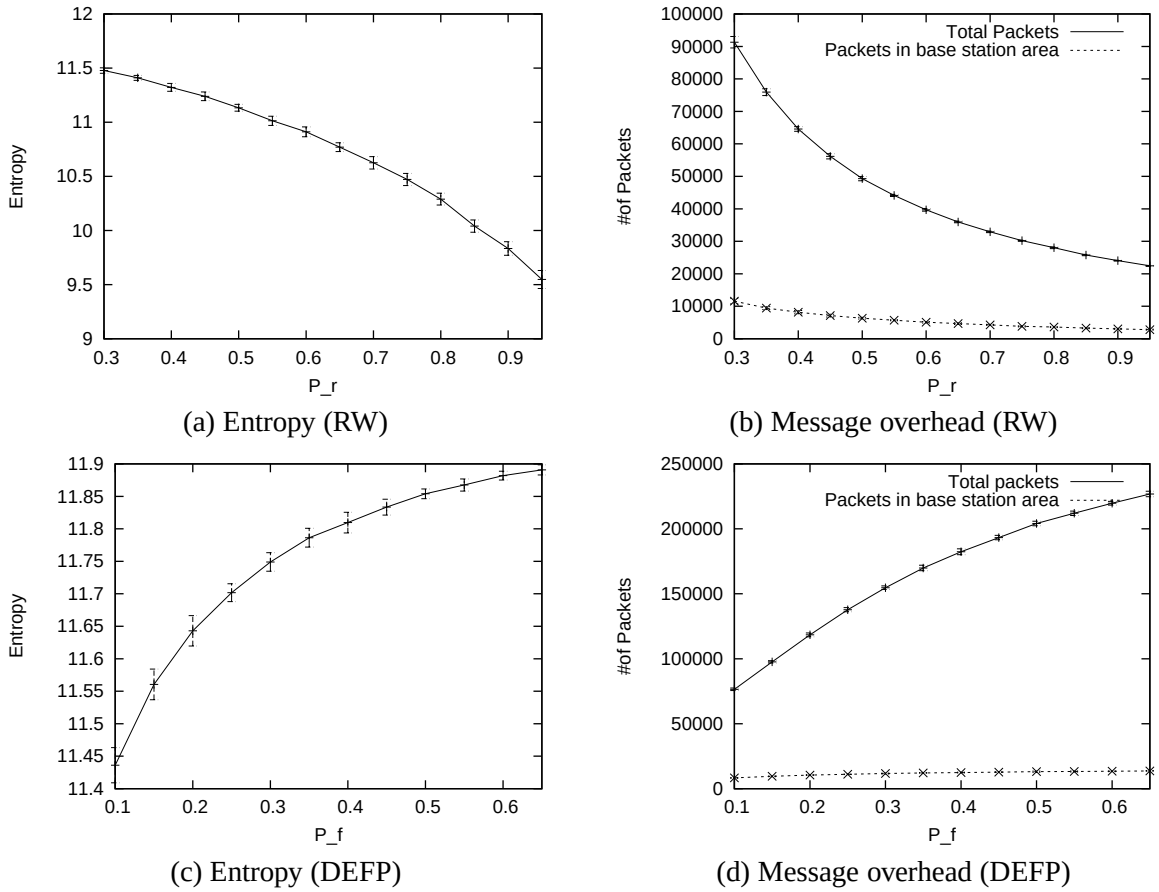
Figure 11: Effectiveness and cost as a function of p_r (a)-(b) and p_f (c)-(d) for a grid network.

large for very large values of p_f, the number of messages exchanged near the base station doesn't change a lot. This shows that the traffic control mechanism proposed in DFP and DEFP works quite well.

6 Related Work

Research in security issues in sensor networks has received much attention recently, e.g. secure data communication [16], secure routing [13, 11, 5], secure data aggregation [18], and pairwise key setup [8, 3, 7, 14, 25]. In the area of privacy in E-commerce, many techniques have been developed to protect the anonymity of message senders and receivers. Our anti-traffic analysis techniques are similar to the methods used in traditional privacy and anonymity research, but we have three unique properties: First, we focus on hiding the physical location of a base station, instead of hiding the identity of a message sender or receiver. Second, the communication pattern in sensor networks is highly asymmetric and converges on a base station. This make it more difficult to protect the base station against traffic analysis attacks. Third, traditional networks are resource-rich compared to a WSN, and so the techniques developed for traditional networks cannot be directly used in sensor networks.

In traditional privacy research, mist routing requires pre-deployed, hierarchical and trusted routers [2]. [10] requires that every node can talk to every other node. The Onion routing protocol [9] disguises who talks to



(a) Entropy (RW) (b) Message overhead (RW)

(c) Entropy (DEFP) (d) Message overhead (DEFP)

Figure 12: Effectiveness and cost as a function of p_r (a)-(b) and p_f (c)-(d) for a random network.

whom on the Internet by layered encryption and by forwarding received messages in a random order. In addition, a large number of messages are stored before forwarding them in a different order. A sensor node doesn't have enough memory to store many packets. The k-anonymous message transmission protocol proposed in [1] protects anonymity for both sender and receiver with low data transmission latency. Unfortunately, its high communication and computational requirements prevent it from being used in sensor networks. The techniques to disguise a receiver by routing each message to multiple receivers using a multicast mechanism are proposed in [17, 20]. Tor [6] is the second-generation onion router, which is a circuit-based low-latency anonymous communication service on the Internet. However, it needs to set up a large number of directory servers, which is difficult to envision in sensor networks.

Recently, techniques to randomize communications during the network setup phase to protect the anonymity of the sensor network infrastructure were proposed in [21]. In contrast, we focus on defending against traffic analysis during the data sending phase. In addition, we propose a more robust adversary model, and assume that an adversary can launch active attacks such as injecting traffic in the network, and compromising sensor nodes. Preserving source-location privacy in WSNs was proposed by C. Ozturk *et.al.* [15]. This work proposes randomization techniques such as fake packets, persistent fake sources, and a random walk to hide the location of the source of data packet from discovery. Unlike our approach, fake packets are always flooded, which incurs a high overhead cost. The key advantage of our approach is that it achieves much of the decorrelative effects of flooding at a fraction of the cost. Also, our focus is on the arguably more difficult task of hiding the *destination* of a data packet, i.e. base station, from discovery, since the patterns produced by the

tree-structured routing are quite pronounced and difficult to hide.

7 Future Work

There are a variety of research directions that could be more fully explored in the future. First, our work considers only one form of search algorithm, namely the GSAT search. More advanced search techniques could be evaluated. Second, the paper also has not developed a metric for evaluating the effectiveness of our proposed techniques against time correlation attacks. The fractal propagation approach makes it more difficult to trace a packet by inspecting transmission times of adjacent nodes, because the attacker may wind up following a fake path to a dead end. For example, one metric that captures the frustration level of an adversary mounting a time correlation attack is the number of dead ends reached until finding the base station. Third, this paper does not address a problem that the base station's forwarding behavior is somewhat different than a typical sensor node in that a packet just stops being transmitted after it has reached the base station. Since a large fraction of packets are destined for the base station, the sudden lack of forwarding is a strong indication that the base station area has been reached, even if we imposed a uniform sending rate on all nodes. We have considered a technique whereby a base station that has received a packet continues to forward a dummy version of that packet past the base station. These dummy packets will have a limited lifetime and can be treated by following nodes like the fake forked packets resulting from fractal propagation. However, we have not yet implemented or evaluated this idea. Fourth, the degree of aggregation has not been deeply explored. The tradeoff between the effectiveness and cost of randomization will be affected by more pervasive aggregation throughout the WSN. Fifth, one measurement that would have been useful to include was the cost in delay, or extra number of hops, due to random walking.

8 Conclusion

The tree-based routing structure of a wireless sensor network is rooted in a base station. The forwarding patterns of WSNs are highly pronounced, revealing the location of the base station through traffic volume and directionality of packet forwarding. An adversary can eavesdrop and employ *rate monitoring* and *time correlation* traffic analysis attacks to locate and destroy a base station, thus disabling the entire WSN. This paper proposed a suite of countermeasures aimed at decorrelating network traffic so that the location of a base station is disguised against traffic analysis techniques. First, three basic defenses were proposed that morph a packet's appearance at each hop via reencryption, impose a uniform sending rate throughout the network, and decorrelate packet sending times at each hop. Next, an improved suite of four more advanced solutions were proposed that overcome limitations of the basic defenses. We introduce controlled randomization into the multi-hop path a packet takes from a sensor node to a base station. We further introduce random fake paths to confuse an adversary from tracking a packet as it moves towards a base station. Finally, we create multiple, random hot spots of high communication activity to deceive an adversary as to the true location of the base station. The paper evaluated these techniques analytically and via simulation using three evaluation criteria: total randommness or entropy of the network, total energy consumed as represented by message overhead cost, and the ability to prolong a heuristic-based search technique called GSAT to locate a base station. The simulations showed that our combined suite of advanced randomization techniques, namely multi-parent routing plus controlled random walk plus differential enforced fractal propagation, together achieved decorrelation comparable to the best possible decorrelation represented by broadcast, at a fraction of broadcast's messaging cost.

9 Acknowledgements

We would like to thank the reviewers for their valuable comments.

References

[1] L. V. Ahn, A. Bortz, and N. J. Hopper. k-anonymous message transmission. In *10th ACM Conference on Computer and Communications Security*, pages 112–130, Washington D.C, USA, October 2003.

[2] J. Al-Muhtadi, R. Campbell, A. Kapadia, D. Mickunas, and S. Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *International Conference of Distributed Computing Systems (ICDCS 2002)*, Vienna, Austria, July 2002.

[3] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[4] S. A. Cook. The complexity of theorem-proving procedures. In *3rd Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–158, Shaker Heights, Ohio, USA, March 1971.

[5] J. Deng, R. Han, and S. Mishra. Inrusion tolerance and anti-traffic analysis strategies in wireless sensor networks. In *IEEE 2004 International Conference on Dependable Systems and Networks (DSN'04)*, Florence, Italy, June 2004.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *13th USENIX Security Symposium*, Dan Diego, CA, USA, August 2004.

[7] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington D.C, USA, October 2003.

[8] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *Conference on Computer and Communications Security, (CCS'02)*, Washington DC, USA, November 2002.

[9] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of ACM*, 42(2), February 1999.

[10] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao. Preventing traffic analysis for real-time communication networks. In *1999 IEEE MIlitary Communications Conference*, October 1999.

[11] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *11th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2004.

[12] C. Karlof, Y. Li, and J. Polastre. Arrive: Algorithm for robust routing in volatile environments. Technical Report Technical Report UCBCSD-02-1233, Computer Science Department, University of California at Berkeley, May 2002.

[13] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2-3), September 2003.

[14] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03*, Washingon D.C, USA, October 2003.

[15] C. Ozturk, Y. Zhang, and W. Trappe. Source-location privacy in energy-constrained sensor network routing. In *2004 ACM Workshop on Security of Ad Hoc and Sensor Networks*, October 2004.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security protocols for sensor networks. *Wireless Networks Journal(WINET)*, 8(5):521–534, September 2002.

[17] A. Pfitzmann and M. Waidner. Networks without user observability: Design options. In *Advances in Cryptology - EUROCRYPT'85*, Linz, Austria, April 1985.

[18] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *ACM SenSys'03*, Los Angeles, CA, USA, November 2003.

[19] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *10th National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446, San Jose, CA, USA, July 1992.

[20] C. Shields and B. Levine. A protocol for anonymous communications over the internet. In *7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.

[21] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones. On providing anonymity in wireless sensor networks. In *10th International Conference on Parallel and Distributed Systems*, Newport Beach, CA, USA, July 2004.

[22] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *1st Symposium on Operating Systems Design and Implementation(OSDI'94)*, Monterey, CA, USA, November 1994.

[23] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges fo reliable multihop routing in sensor networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Log Angeles, CA, USA, November 2003.

[24] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route detection and filtering of injected false data in sensor networks. In *23rd Conference of the IEEE Communications Society (INFOCOM'04*, Hong Kong, China, March 2004.

[25] S. Zhu, S. Setia, and S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM Conference on Computer and Communications Security*, Washington D.C, USA, October 2003.

[26] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *2004 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2004.