Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2353–2364
computer communications
www.elsevier.com/locate/comcom

An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks

Edith C.H. Ngai [a], Jiangchuan Liu [b,*], Michael R. Lyu [a]

[a] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
[b] School of Computing Science, Simon Fraser University, Burnaby, BC, Canada V5A 1S6

Available online 6 May 2007

Abstract

In a wireless sensor network, multiple nodes would send sensor readings to a base station for further processing. It is known that such a many-to-one communication is highly vulnerable to a *sinkhole attack*, where an intruder attracts surrounding nodes with unfaithful routing information, and then performs selective forwarding or alters the data passing through it. A sinkhole attack forms a serious threat to sensor networks, particularly considering that the sensor nodes are often deployed in open areas and of weak computation and battery power.

In this paper, we present a novel algorithm for detecting the intruder in a sinkhole attack. The algorithm first finds a list of suspected nodes through checking data consistency, and then effectively identifies the intruder in the list through analyzing the network flow information. The algorithm is also robust to deal with multiple malicious nodes that cooperatively hide the real intruder. We have evaluated the performance of the proposed algorithm through both numerical analysis and simulations, which confirmed the effectiveness and accuracy of the algorithm. Our results also suggest that its communication and computation overheads are reasonably low for wireless sensor networks.

Keywords: Wireless sensor network; Sinkhole attack; Intruder detection; Intruder identification

1. Introduction

A wireless sensor network (WSN) consists of a set of geographically distributed sensor nodes, which continuously monitor their surroundings and forward the sensing data to a base station (referred to as a *sink*) through multi-hop routing. It has become increasingly popular in solving such challenging real-world problem as geographical sensing and environmental monitoring.

Given the importance of the sensed data, various attacks have targeted sensor networks [1,2]. While some of the attacks are common in different types of networks, the many-to-one communication pattern between the sensors and the sink poses unique challenges [3–5]. A typical example is the sinkhole attack, where an intruder attracts surrounding nodes with unfaithful routing information, and then alters the data passing through it or performs selective forwarding [1,6,7]. The sinkhole attack prevents the base station from obtaining complete and correct sensing data, and thus leads to a serious threat. It is particularly severe for wireless sensor networks given that the wireless links are vulnerable and the sensors are often deployed in open areas with weak computation and battery power. The existing routing protocols in sensor networks are generally susceptible to the sinkhole attack [1,8–10]. Although some secure mechanisms are proposed and make use of cryptographic techniques to protect network traffic [11–14], they are often localized, or suffer from high computation overheads and require time synchronization among the nodes.

In this paper, we propose a novel lightweight algorithm for detecting the sinkhole attack and identifying the intruder involved. We deviate from the traditional strategy of

* Corresponding author. Tel.: +1 604 291 4336; fax: +1 604 291 3045.
E-mail addresses: chngai@cse.cuhk.edu.hk (E.C.H. Ngai), jclu@cs.sfu.ca (J. Liu), lyu@cse.cuhk.edu.hk (M.R. Lyu).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.025

defending against an attack using cryptography; instead, we first detect the attack by observing the network flow information, and then identify the intruder and malicious nodes, which can later be isolated to protect the network.

We focus on a general many-to-one communication model, where the routes are established based on the reception of route advertisements. Our solution explores the asymmetricity between the sensor nodes and the base station, and makes effective use of the relatively-high computation and communication power of the base station [1,15,16]. It consists of two major parts: First, a secure and low-overhead algorithm for the base station to collect the network flow information from the attacked area; Second, an efficient identification algorithm that analyzes the routing pattern and locates the intruder. We also consider the complex scenario with colluding nodes that cooperatively cheat the base station about the intruder location. Specifically, we examine multiple suspicious nodes and identify the intruder with a voting method. We show that the solution is correct as long as the normal nodes are dominant.

The performance of the proposed algorithm is evaluated through simulations, which confirm the effectiveness and accuracy of the algorithm. Our results also suggest that its communication and computation overheads are reasonably low for wireless sensor networks.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3, we formally describe the sinkhole attack in wireless sensor networks, and state the problem to be solved. In Section 4, we present our 2-step detection algorithm, which collects network flow information and identifies the intruder. In Section 5, we enhance the algorithm to handle multiple malicious nodes and prove its correctness. The performance of the proposed algorithm is further evaluated in Sections 6 through simulations. Finally, Section 7 concludes this paper and offers some future research directions.

2. Related work

Intrusion detection has long been an active research topic in the Internet evolution [17]. Recently, many detection algorithms have been proposed for wireless ad hoc networks as well. Most of them assume uniform nodes and symmetric data communication patterns between the nodes [7,18,19]. The one-to-many communication pattern in wireless sensor networks however poses different challenges, in particular, the sinkhole attack. The weaker computation and battery power of the sensor nodes further aggravates the problem. Pirzada et al. [20] applied a trust scheme to the routing protocol to detect sinkhole and wormhole attacks in a sensor network, but it requires the nodes to operate in a promiscuous mode. Hu et al. [21] introduced *packet leash*, which confides the maximum transmission time and distance of each packet. It assumes that a node can obtain a key for any other node and authentication is applied to each data packet. On the contrary, our work does not require the nodes to support promiscuous mode nor authenticate every data packet.

Our work is also motivated by the existing studies on filtering false reports or avoiding jammed/failed nodes in sensor networks. Specifically, Doumit and Agrawal [22] showed a self-organized criticality (SOC) and Hidden Markov models based algorithm, which can effectively detect data inconsistencies in sensor networks. Wood et al. [23] proposed a mechanism for detecting and mapping jammed regions. They described a protocol for identifying the surrounding nodes of a jammer, thus avoiding the broken links and congested nodes in the region. Staddon et al. [24] demonstrated that the topology of the network could be efficiently conveyed to the base station, allowing for quick tracing of failed nodes with moderate communication overhead. Ding et al. [25] further proposed a localized algorithm for identifying faulty sensors. Ye et al. [26] presented a Statistical En-route Filtering (SEF) mechanism for detecting and dropping false reports. It integrates multiple authentication codes, probabilistic verification, and data filtering to determine the truthfulness of each report. Our work differs from them in that we focus on sinkhole attacks, where the intruder and multiple malicious nodes actively disturb the one-to-many data communications or even interfere the identification algorithm itself.

3. Network model and problem statement

We consider a wireless sensor network that consists of a *base station* (BS) and a collection of geographically distributed sensor nodes, each denoted by a unique identifier ID_v. The sensor nodes continuously collect and forward the sensed environmental data to the base station in a multi-hop fashion.

As mentioned earlier, this commonly used many-to-one communication pattern is vulnerable to sinkhole attacks. In this type of attack, an intruder usually attracts network traffic by advertising itself as having the shortest path to the base station. For example, as shown in Fig. 1a, an intruder, which is equipped with much higher computation and communication power than a normal sensor node, creates a high-quality single-hop link to the BS. It can then advertise imitated routing messages about the high quality route, spoofing the surrounding nodes to create a sinkhole (SH). A sinkhole can also be performed using a wormhole [27], which creates a metaphorical sinkhole with the intruder being the center; the intruder then relays the messages received in one part of the network toward the sink using a tunnel (see Fig. 1b).

We assume the sensor nodes are either *normal* or *malicious*. The center of a sinkhole attack is a malicious node compromised by the intruder. Note that, even if there is only one compromised node, it can affect many surrounding normal nodes by creating a high quality route to the base station. Furthermore, this intruder may also collude

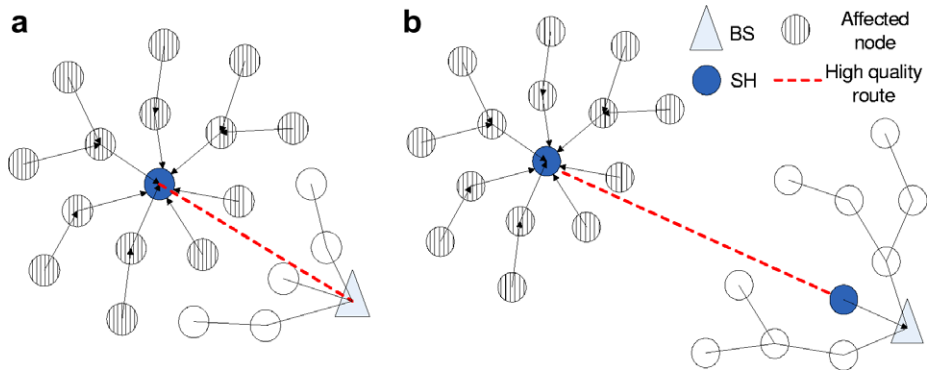

Fig. 1. Two examples of sinkhole attack in wireless sensor networks. (a) Using an artificial high quality route; (b) using a wormhole.

with some other malicious nodes. They could even collaboratively cheat the detection algorithm by suggesting a normal node as the intruder (the victim, denoted as SH′).

The focus of our work is to effectively identify the real intruder (SH) in the sinkhole attack. Once it is identified, a routing protocol or a higher-layer application can easily isolate the intruder from the network to avoid further loss. We assume that the base station is physically protected or has tamper-robust hardware [15]; hence, it acts as a central trusted authority in our algorithm design. The base station also has a rough understanding on the location of nodes, which could be available after the node deployment stage or obtained by various localization mechanisms [28]. For ease of exposition, in Table 1, we list the major notations used throughout this paper.

4. Intruder detection for sinkhole attack

We now describe our algorithm for detecting a sinkhole attack, and then efficiently identifying the intruder. We first focus on the case of a single malicious node only, i.e., the intruder (SH). We assume that, in this simple scenario, the *SH* will affect sensor data collection, but will not interfere the detection algorithm through dropping or altering the control messages. In the next section, we will present enhancements dealing with multiple malicious nodes that collude and even interfere the detection algorithm.

Table 1
List of notation

BS	Base station
SH	Real intruder in the sinkhole attack
SH′	False intruder (victim) in the sinkhole attack
ID_v	Identity of sensor node v
p	Probability of a node being malicious
d	Packet drop rate
k	No. of neighbors to which a message will be forwarded
$h_{\max}$	Hops from the farthest node to the *BS*
h_{rc}	Hops from the *BS* where root correction takes place
l	Levels from the *BS* in the tree of network flow
t^l	Number of nodes at level l
N	Total number of nodes in the attacked area
F_r	Number of correct network flow information collected
F_m	Number of incorrect network flow information collected
F_s	Number of missing network flow information
F_n	Total number of network flow information collected

4.1. Estimating the attacked area

In a sinkhole attack, the intruder can drop, alter, or selectively forward the sensing data. The *BS* can suspect the existence of an attack through various statistical or application-specific data analysis. The sensors, which are closely located, are expected to have similar readings from the environment. We divide the network into a number of sub-areas and compare the data within each of them. The *BS* can detect the attack by finding the inconsistent data between the normal sensors and attacked sensors in the sub-areas. It can also detect the missing data from the attacked sensors and identify the attacked area. Since the area affected by the attack is limited in size, it is impossible for an intruder to alter the data from all the sensors in the network. Thus, the attack must be detected in some of the sub-areas.

For illustration, consider a monitoring application in which sensor nodes submit data to the *BS* periodically. Let $X_1, \ldots, X_n$ be the sensing data collected in a sliding window, and $\overline{X}$ be their mean. Define $f(X_j)$ as,

$$f(X_j) = \sqrt{\frac{(X_j - \overline{X})^2}{\overline{X}}}.$$

A simple measure for identifying a suspected node is whether $f(X_j)$ is greater than a certain threshold, for the data from this node is noticeably inconsistent with others in the same area. More advanced statistical methods can be found in [29,30].

After identifying a list of suspected nodes, the *BS* can estimate where the sinkhole locates. Specifically, it can circle a potential *attacked area*, which contains all the suspected nodes. An example is shown in Fig. 2, where the shaded nodes are found to contain missing or inconsistent data. Note that all the nodes in the circle could be attracted by the sinkhole sooner or later, and we thus refer to them as *affected nodes*.

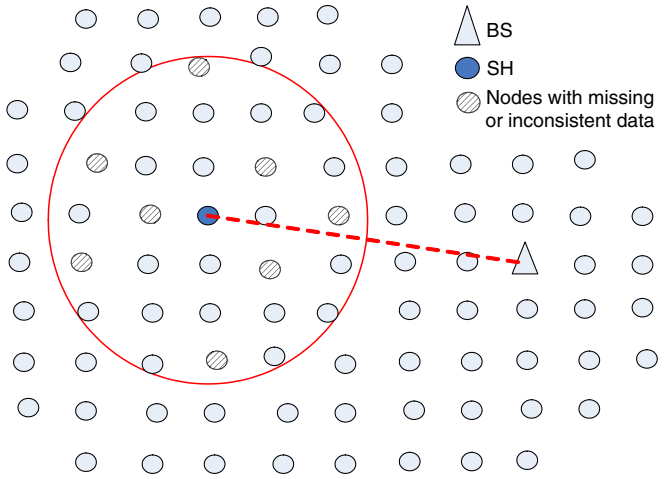

Fig. 2. Estimate the attacked area.

4.2. Identifying the intruder

Since the attacked area may contain many nodes, and the sinkhole is not necessarily the center of the area in a multi-hop sensor network, it is necessary to further locate the exact intruder and isolate it from the network. This can be achieved through analyzing the routing pattern in the affected area.

We now demonstrate a method for collecting the network flow information, which facilitates the routing pattern analysis. First, the *BS* sends a request message to the network. The message contains the *ID*s of the affected nodes, and is flooded hop by hop. For each node receiving the request, if its *ID* is there, it should respond to the *BS* with a message, which includes its own *ID*, the *ID* of the next-hop node, and the cost for routing, e.g, hop-count to the *BS*. Note that the next-hop and the cost could already be affected by the attack; hence, the response message should be transmitted along the reverse path in the flooding, which corresponds to the original route with no intruder.

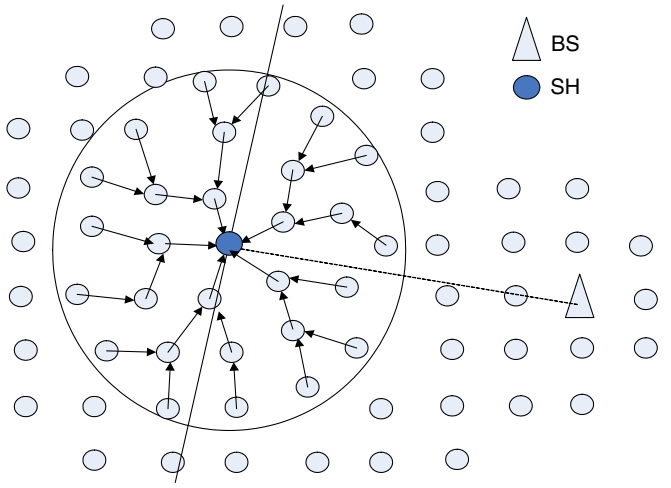

Fig. 3. Network flow in the attacked area.

At the *BS*, each piece of network flow information can be represented by a directed edge, $a \xrightarrow{ct} b$, where a denotes an affected node, b denotes the next hop of a, and ct is the cost from a to the *BS*. The *BS* can then visualize the routing pattern by constructing a tree using the collected next hop information. Note that the area invaded by a sinkhole attack has a special routing pattern, where all network traffic flows toward the same destination, that is, the intruder *SH*. As shown in Fig. 3, once the tree is constructed, the *BS* can easily identify the *SH*, which is exactly the root of the tree in this single malicious node case.

5. Enhancements against multiple malicious nodes

As mentioned before, there could be multiple malicious nodes that prevent the *BS* from obtaining correct and complete flow information for intruder detection. Specifically, they may cooperate with the intruder to perform the following misbehaviors:

1. Forward the response messages selectively or even drop all (denial of service);
2. Modify the response messages passing through; and
3. Respond with false network flow information of itself.

In this section, we present effective enhancements that address these problems.

5.1. Dealing with dropped flow information

Malicious nodes may drop the response messages of network flow information, as shown in Fig. 4a. To mitigate the problem, the sensors can forward the information to the *BS* through multiple redundant paths. Specifically, a node can forward reply messages to k neighbors, $k \geqslant 1$. Let p be the probability that a node is malicious, h_{max} be the number of hops from the farthest node to the *BS*, and the number of nodes in level l (i.e., hop count of l to the *BS*) be t^l. For uniformly distributed sensors, t^l can be estimated as

$$t^l = [(lR)^2\pi - (l-1)^2R^2\pi] * D = (2l-1)R^2\pi * D,$$

where R is the transmission range, and D is the sensor distribution density.

Consider the extreme case in which a malicious node does not generate a response and drops any responses passing by, the probability that the response message from a node at level l reaches the *BS* is $(1-p)(1-p^k)^{l-1}$. Thus, the expected number of responses reaching the *BS* is

$$n = \sum_{l=1}^{h_{\max}} (1-p)(1-p^k)^{l-1} t^l.$$

As an example, for $p = 0.1$, $k = 2$, $h_{\max} = 5$, $R = 10$ m, $D = 0.01$ node/m^2, we have $n = 67.73$. That is, on average, 67.73 responses will reach the *BS*. Such a result is reasonably good, given that the total number of normal nodes is around 78 in this setting.

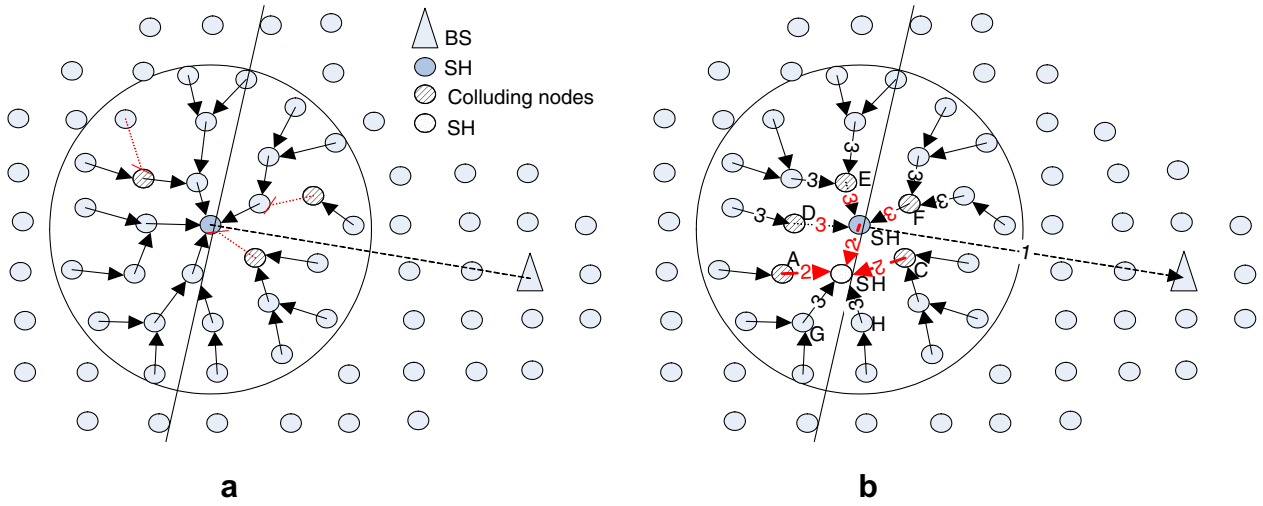

Fig. 4. Attacked area with colluding nodes. (a) Dropping responses; (b) providing false responses.

Since there are still losses of the responses, the tree to be constructed based on the network flow information might be broken into several subtrees. Algorithm 1 shows a procedure to construct these trees, and the intruder is clearly in the tree with direct connection to the *BS*.

In some extreme cases, the malicious nodes may perform denial of service attacks. They may refuse to reply and drop all the messages passing through. However, this kind of attacks can be easily detected, as the information from the same area is completely lost.

Algorithm 1. Identify multiple subtrees

```
R = φ            /*R: set of subtrees*/
for each v ∈ S   /*S: set of nodes in the attacked area*/
  if v has no incoming edge
    R = R ∪ FindSubtree(v)
end for
subroutine FindSubtree(node u)
    R' = φ
    if u is not yet visited
      mark u is visited
    else
      return φ
    if u has no outgoing edge
      return {u}
    for each e(u,v)
      R' = R' ∪ FindSubtree(v)
    end for
    return R'
end FindSubtree
```

5.2. Dealing with tampered or false flow information

In the process for collecting the network flow information, the malicious nodes could even tamper the responses passing by or generate false responses. The receiver thus has to protect the reply message, so as to prevent an attacker from forging the network flow information. To this end, we assume every node v shares a secret key K_v with the *BS*, which they use in conjunction with a message authentication code (MAC) function (for example HMAC [31]) to authenticate control messages. This key can be loaded to the node through a pre-distribution protocol, e.g., that in [32]. To send a reply message R, v actually sends $\langle R, MAC_{Kv}(R)\rangle$ to *BS*, where the notation $MAC_{Kv}(R)$ is the MAC message authentication code computed over message R with key K_v. MAC message authentication code is a short piece of information used to authenticate a message. An MAC algorithm accepts a secret key and an arbitrary-length message to be authenticated as input, and outputs an MAC (sometimes also known as a tag). Although the encryption process for generating the MAC imposes additional calculations to the sensors and the base station, the overhead is affordable with the existing lightweight symmetric encryption algorithms. Recent studies have shown that symmetric encryption and hashing function schemes can be efficiently implemented in various small sensing devices [33,34]. The code sizes in some of these algorithms, like RC4 and RC5, are very small. They are suitable for the low-cost processors in sensors which lack large amounts of program memory. When *BS* receives this message, it can verify the authenticity of the message by comparing the received MAC value to the MAC that it computes for itself over the received message with K_v. More importantly, the encryption applies to the responses, whose volume is much smaller than that of the normal data traffic.

The encryption, however, cannot solve the problem that the malicious nodes themselves provide false responses to hide the real *SH*. As shown in Fig. 4b, two colluding nodes A and C, together with the real *SH*, suggest outgoing edges to a victim node *SH'*. To deal with this problem, the *BS* can detect the inconsistency among the hop count information. For instance, we can see that the incoming edges of the *SH'* have different number of hop counts, which is suspicious because the nodes sending messages via the same next hop should have the same hop counts to the *BS*. Also note that the malicious nodes D, E, and F have identical

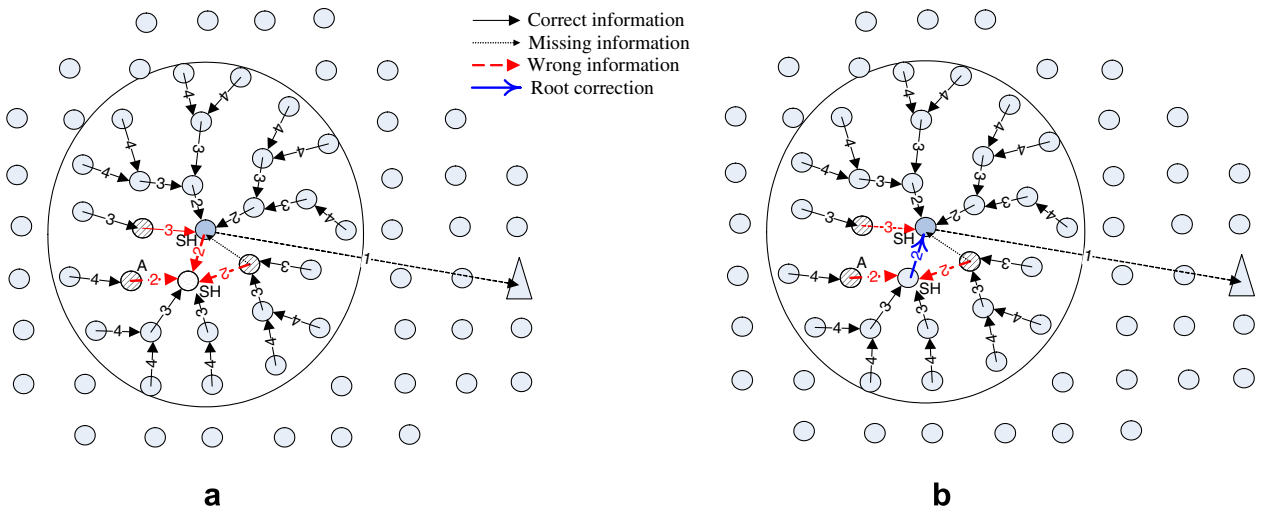

Fig. 5. Example of intruder identification with multiple malicious nodes.

hop counts in their incoming and outgoing edges, which is again abnormal. In our algorithm, we calculate the difference between the hop counts provided by a node and the number of edges from the node to the current root. We then identify the *SH* and other suspicious nodes by spotting the inconsistency of the hop counts.

To this end, we maintain an array *Count*, where the *i*-th entry stores the total number of nodes having hop count difference *i*. Note that index *i* can be negative, which indicates that the hop count provided by a node is smaller than its actual distance from the current root. Intuitively, if *Count*[0] is not the dominated one in the array, it means the current root is unlikely the real intruder. Through analyzing the array *Count*, we can estimate the hop counts from the *SH′* to the *SH*. For example, if most non-zero entries of *Count* fall in the index range $[-2,2]$, we suspect that the *SH* is two hops away from the *SH′*, which is the current root. Given this estimation, the *BS* can make root correction and re-calculate the entries of *Count* for those nodes within two hops from the *SH′*. After several iterations, it can conclude the intruder if a majority of the nodes agree with a consistent result. A formal description of the above procedure can be found in Algorithm 2.

As an example, consider the attacked area in Fig. 5a, where the node *SH′* is the original root of the network flow tree, and its array *Count* is as follow,

i	−2	**−1**	**0**	1	2
Count[*i*]	0	**14**	**8**	6	0

Algorithm 2. Find the real intruder with root corrections

```
for each root r
    initialize a new Array count[]
    initialize a new Path correctPath
    checkRootByCount(r, count, 1)
    S = {x > 0|forall y > 0,count[x] + count[−x] > count[y] + count[−y]}
    x = min(S)
    correctRoot(r, r, x, 0, correctPath, count[0])
    apply correctPath on Network G
end for
subroutine checkRootByCount (Node r, Array count[], int depth)
    depth = depth +1
    for each precedent Node c of r
      increase count[hop_count(c) – depth] by 1
      checkRootByCount (c, count, depth)
    end for
end checkRootByCount
subroutine correctRoot (Node r, Path p, int totalLevel, int currentLevel,
  Path correctPath, int bestCount)
    if (currentLevel >= totalLevel)
      return
    end if
    currentLevel = currentLevel+1
    for each precedent node c of r
      initialize a new array count[]
      reverse edge (c,r)
      checkRootByCount (c, count, 1)
        if (count[0] > bestCount)
          correctPath = p − >c
          bestCount = count[0]
        end if
      correctRoot (c, p − >c, totalLevel, currentLevel, correctPath,
  bestCount)
      reverse edge (c,r)
    end for
end correctRoot
```

It shows that only 8 nodes agree that the *SH′* is the intruder. However, 14 nodes do not agree with it. Instead, they suggest that the *SH* should be one hop closer to the *BS* than the node *SH′*. Since they are the majority, our correction algorithm runs again to look for a new root. After that, the node *SH* becomes the new root (Fig. 5b), and the corresponding *Count*[] becomes,

i	−2	−1	**0**	1	2
Count[*i*]	0	1	**21**	6	0

We can see that 21 nodes provide consistent information about the current root *SH*. Since the value of *Count*[0] is the majority, the *SH* is concluded as the intruder.

The time complexity for calculating array *count* is $O(N)$, and that for correcting the roots is $\sum_{l=1}^{h_{rc}} t^l \cdot N = O(t^{h_{rc}} \cdot N)$. Here, h_{rc} is the average number of hops where a root correction will take place, which can be estimated from *count*[] and is in general quite small. The total time is thus $O(N) + O(N) + O(t^{h_{rc}} \cdot N) = O(t^{h_{rc}} \cdot N)$, which is relatively low.

5.3. Proof of correctness

We now give a simple analysis on the correctness of the above algorithm. We assume that N is the number of nodes in the attacked area, namely, $N = F_n + F_s = (F_m + F_r) + F_s$.

Property: For any F_m, our sinkhole detection algorithm works if there are more than $2F_m$ sensors reporting their network flow information successfully to the *BS* and at most F_m are malicious among them.

Proof. Since there are more than $2F_m$ sensors successfully reporting their network flow information, we have $F_n > 2F_m$, and consequently $F_r > F_m$. In the worst case, all the F_m malicious sensors are colluding and suggesting victim SH' as the intruder. Yet, F_r normal sensors will suggest *SH*, the real intruder, and the number of these nodes (F_r) is greater than the malicious nodes (F_m). Hence, our algorithm can correctly identify the real intruder through the majority vote.

Our algorithm might not work if $F_n \leqslant 2F_m$ and the malicious nodes are all colluding. Nevertheless, it unlikely happens in most sensor networks, where a majority of sensors should be in normal condition.

6. Performance evaluation

We further evaluate the performance of our sinkhole detection algorithm through simulations. We simulate a wireless sensor network with a 200 meter by 200 meter field in which 400 nodes are placed with uniform random distribution. The sensors adopt IEEE 802.11 MAC protocol with radio range of 10 meter. A base station is placed at the center of the network to collect data from the sensors. Moreover, a sinkhole is added to the network at x- and y-coordinates $(50, 50)$ for emulating a sinkhole attack. We are interested in evaluating the accuracy on intruder identification, communication overhead, and energy consumption of our intruder detection algorithm. Table 2 shows the default environment settings of our implementation, mostly adapted from [23,26].

Table 2
Parameter settings of the experiments

Number of nodes of network	400
Size of network	200 m × 200 m
Transmission range	10 m
Location of *BS*	(100, 100)
Location of sinkhole	(50, 50)
Percentage of malicious nodes	m
Percentage of colluding nodes	m_c
Message drop rate (d)	0–80%
No. of neighbors which a message is forwarded to (k)	1–2
Packet size	30 bytes
Max. number of reply messages per packet	5

6.1. Accuracy of intruder identification

In the first set of experiments, we investigate the accuracy of our intruder detection algorithm for sinkhole attacks. We focus on the following three measures: (1) *success rate*, which is the percentage that our algorithm can correctly identify the *SH*; (2) *false-positive rate*, which is the percentage that our algorithm incorrectly identifies the *SH*; and (3) *false-negative rate*, which is the percentage that our algorithm is not able to identify any intruder but it does exist.

We first consider a mildly hostile environment in which 20% nodes are malicious. For those networks with less malicious nodes or even one (the intruder itself) only, the results are even better. We vary the ratio of colluding nodes from 0% to 20% of the total nodes in the network. When the ratio is 20%, all the malicious nodes are colluding with the intruder; for the ratios lower than 20%, a malicious node might randomly drop response messages only but would not collaborate with the intruder to provide misleading routing information, if it is not a colluding node. Fig. 6 shows that the success rates for dropping rates of 0, 0.2, 0.4, 0.6 and 0.8, respectively. For the zero dropping rate at the malicious nodes, we can see that the success rates are 100%. Also the corresponding false-positive and false-negative rates are zero, as shown in Figs. 7 and 8, respectively. This is because the number of correct routing information pieces is much more than that of the incorrect ones in this scenario, and the intruder can always be identified through routing pattern analysis and majority vote. When the drop rate increases, however, the success rate

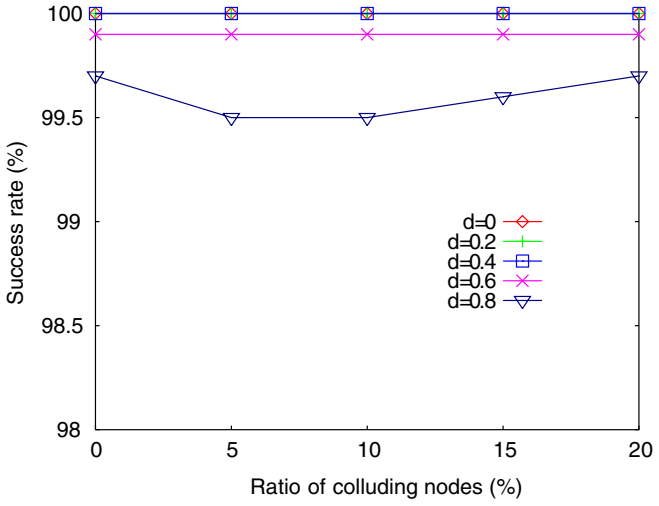

Fig. 6. Success rate in intruder identification ($m = 20\%$).

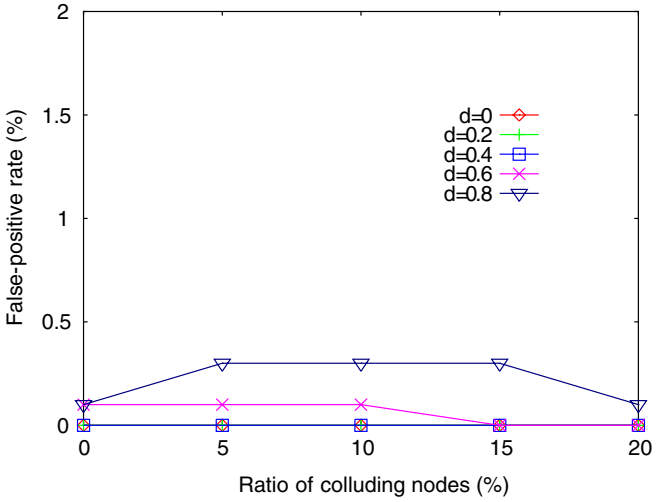

Fig. 7. False-positive rate in intruder identification ($m = 20\%$).

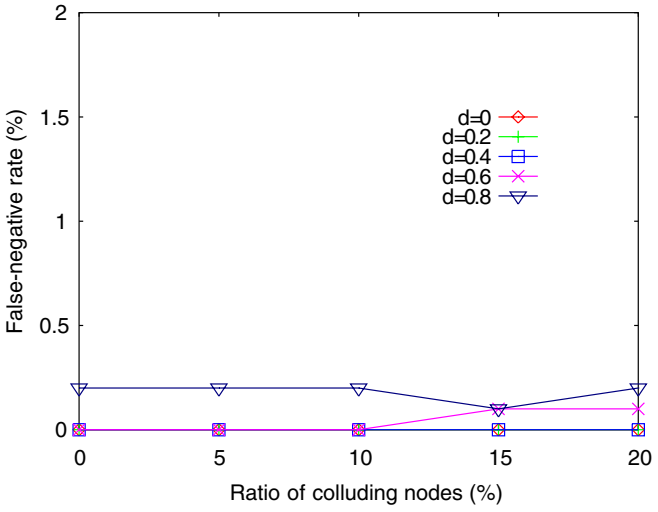

Fig. 8. False-negative rate in intruder identification ($m = 20\%$).

slightly decreases. Intuitively, when the malicious nodes randomly drop a lot of responses, it is possible that the correct information pieces become less than the incorrect ones, which leads to lower success rate, and, not surprisingly, higher false-positive and false-negative rates. Nevertheless, the change is quite minor, even with a dropping rate of 80%. Such results suggest that our algorithm works well under a normally hostile environment.

Given the above results, it is clear that for less than 20% malicious nodes, our intrusion detection algorithm can be even more effective. We are thus more interested in its performance with other extreme hostile environments. To this end, we repeat the above experiments for $m = 50\%$ and 80%, that is, more than half of the nodes are malicious. We again vary the ratio of the colluding nodes (with upper bounds of 50% and 80%, respectively). The corresponding results are shown in Figs. 9–14. For $m = 50\%$, the success, false-positive, and false-negative rates have similar trends

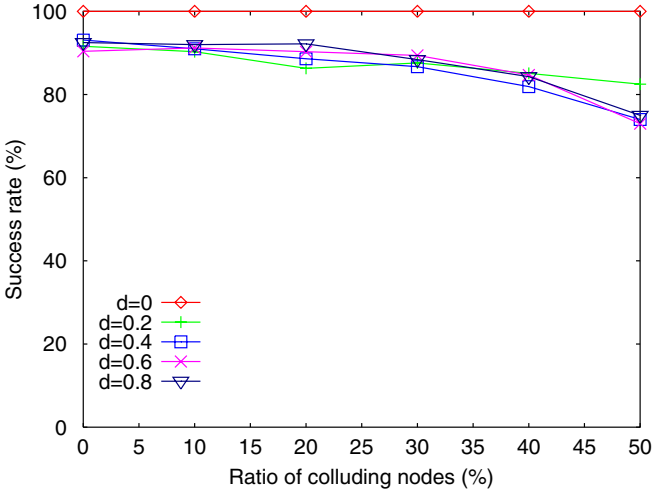

Fig. 9. Success rate in intruder identification ($m = 50\%$).

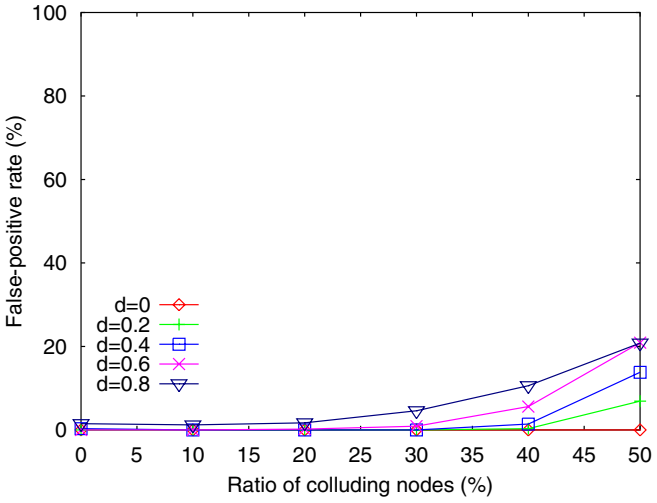

Fig. 10. False-positive rate in intruder identification ($m = 50\%$).

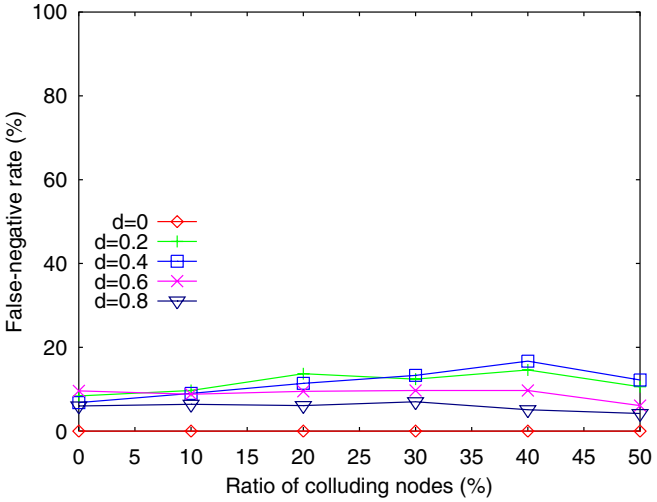

Fig. 11. False-negative rate in intruder identification ($m = 50\%$).

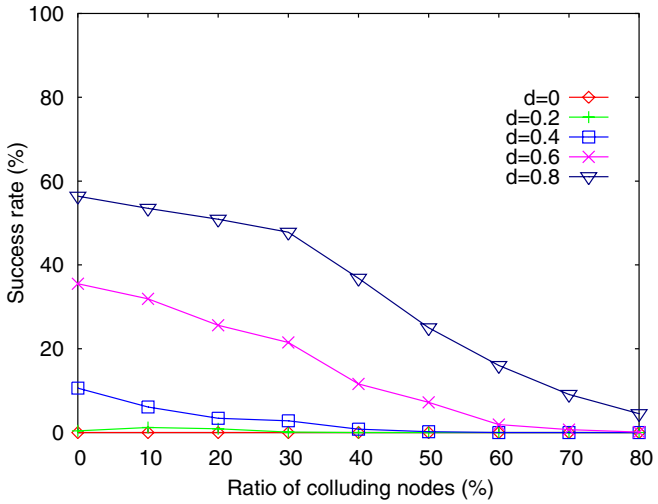

Fig. 12. Success rate in intruder identification ($m = 80\%$).

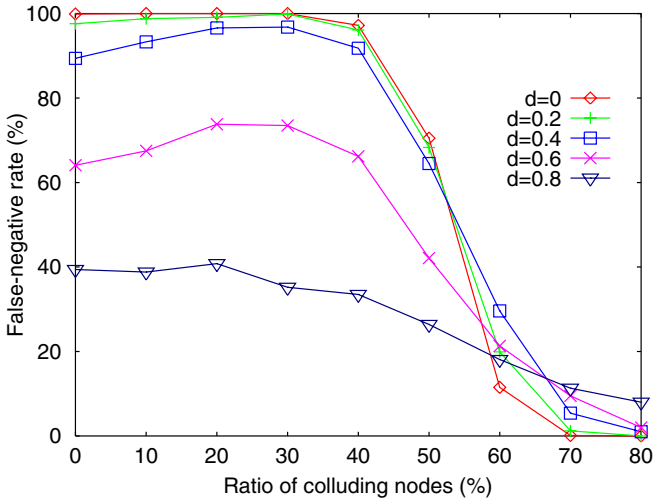

Fig. 14. False-negative rate in intruder identification ($m = 80\%$).

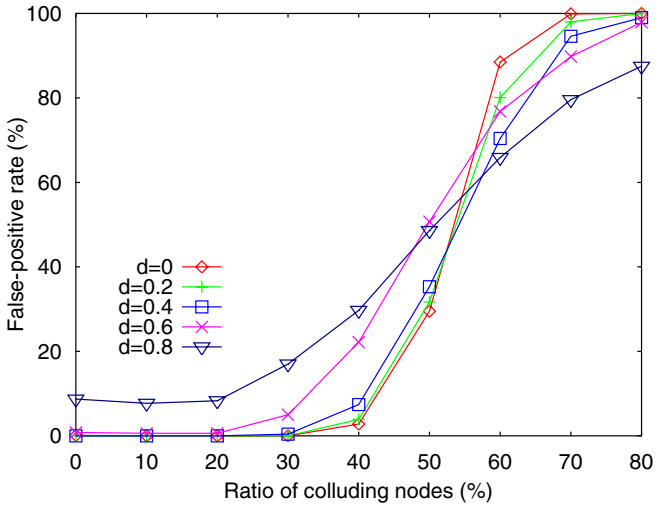

Fig. 13. False-positive rate in intruder identification ($m = 80\%$).

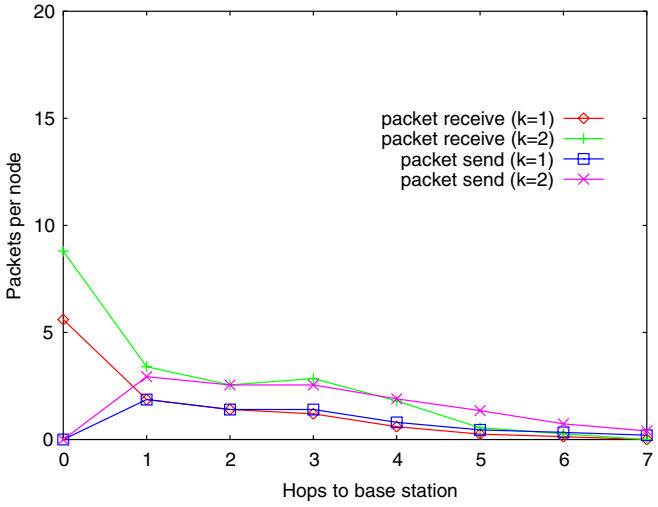

Fig. 15. Communication cost for intrusion detection.

as those with $m = 20\%$. Though the results are generally worse, they are still acceptable. It is worth noting that some of the curves are not monotonic, e.g., Fig. 11 (false-negative rates), when the ratio of colluding nodes is around 40%. This is mainly due to the random drops by the malicious nodes.

For $m = 80\%$, the performance of our detection is generally unacceptable, because the malicious nodes become dominating in the network. Also note that the trends of the curves in this case are often the inverse of that in the previous two cases. For example, the success rate is the highest with a dropping rate of 0.8, and the false-negative rate decreases with increasing the ratio of colluding nodes. The reason again is because the misbehaved nodes dominate the network. Nevertheless, we do not expect any detection algorithms would work well in such an extreme environment.

6.2. Communication cost

In this experiment, we evaluated the communication overhead of our algorithm. Fig. 15 shows the number of packets sent or received by nodes of different hops to the *BS*. We can see that the nodes closer to the *BS* have higher overheads. This is because most of the communication overhead is incurred in collecting the network flow information, and, in this process, a node closer to the *BS* has to relay more messages. Note that, however, the *BS* (0 hop) sends one request messages only, and does not have to relay response messages. The figure also shows the overhead when path redundancy is introduced for responses, where k represents the number of neighbors to which a message will be forwarded. Clearly, the larger the value of k is, the less the

Table 3
Parameters of energy consumption

Communication circuit power	5×10^{-8} J/bit
Communication antenna power	1×10^{-10} J/bit/m^2
Encryption and MAC computation	3×10^{-9} J/bit

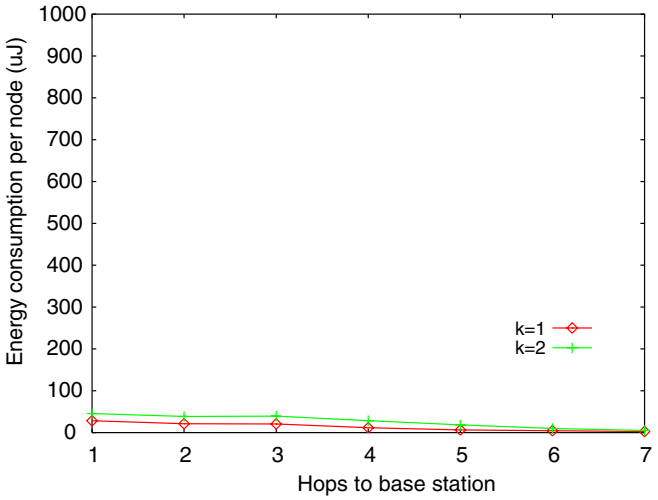

Fig. 16. Energy consumption for intruder identification.

network flow information will be dropped by malicious nodes. Yet, our experiment shows that, when $k = 2$, the overhead is reasonably low while a good delivering ratio can be expected. In addition, for the nodes that are far away from the attacked area (in Fig. 15, those of six or more hop counts), their communication overhead is almost independent of k because they do not have to respond to the request from the *BS*.

6.3. Energy consumption

Finally, we study the energy consumption for our intruder identification algorithm. As mentioned, we assume that the *BS* has high-enough computation and battery power; hence, we mainly focus on the energy consumption in the individual sensors. It is related to both the transmission cost for request and response messages and the computation cost for encrypting messages. Table 3 shows a typical energy consumption for a sensor node, as adapted from [16,35].

Fig. 16 shows the average energy consumption for our algorithm at each single node as a function of its hop counts to the *BS*. It can be seen that the consumption monotonically decreases with increasing the hop count. This is consistent with the data in Table 3 and the result from the previous section; basically, the communication overhead is the dominated one, and the computation part is less than 5% in the total consumption. In addition, similar to the previous experiments, applying redundant paths consumes more energy, as seen in the curve for $k = 2$. Nonetheless, the energy consumption for our intruder detection algorithm is indeed lightweighted. For example, consider a sensor node equipped with two 3 V 1.2 Amp-Hour batteries [36,37]. The total energy available at this node is $E_t = 7.2$ V·A·Hour, which translates into 2000 μJ. Hence, the node needs to spend only a minor portion of the available energy for intruder identification throughout its lifetime.

7. Conclusion and future work

In this paper, we presented an effective method for identifying sinkhole attacks in a wireless sensor network. The algorithm consists of two steps: It first locates a list of suspected nodes by checking data consistency, and then identifies the intruder in the list through analyzing the network flow information. We also presented a series of enhancements to deal with cooperative malicious nodes that interfere the detection algorithm and attempt to hide the real intruder.

The performance of the proposed algorithm was examined through simulations. The results demonstrated the effectiveness and accuracy of the algorithm. They also suggested that its communication and computation overheads are reasonably low for wireless sensor networks. There could be many future directions toward enhancing this work; in particular, we are working on more effective statistical methods for identifying data inconsistency, which will facilitate our algorithm to precisely locate the suspected nodes in sinkhole attacks.

Acknowledgements

The work described in this paper was substantially supported by grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4205/04E). J. Liu's work was supported in part by a Canadian NSERC Discovery Grant 288325, an NSERC Research Tools and Instruments Grant, a Canada Foundation for Innovation (CFI) New Opportunities Grant, and an SFU President's Research Grant.

References

[1] C. Karlof, D. Wagner, Secure routing in sensor networks: attacks and countermeasures, in: Proceedings of the 1st IEEE Workshop on Sensor Network Protocols and Applications, May 2003, pp. 1–15.
[2] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, IEEE Computer 35 (2002) 54–62.
[3] C. Karlof, N. Sastry, D. Wagner, Tinysec: a link layer security architecture for wireless sensor networks, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 162–175.
[4] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, D.E. Culler, Spins: security protocols for sensor networks, Wireless Network Journal 8 (2002) 521–534.
[5] A. Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, H. Wong, Decentralized intrusion detection in wireless sensor networks, in: Proceedings of the 1st ACM International Workshop on Quality

of Service & Security in Wireless and Mobile Networks, 2005, pp. 16–23.
[6] B.J. Culpepper, H.C. Tseng, Sinkhole intrusion indicators in DSR MANETs, in: Proceedings of BroadNets '04, October 2004, pp. 681–688.
[7] H. Deng, W. Li, D.P. Agrawal, Routing security in wireless ad hoc networks, IEEE Communications Magazine 40 (2002) 70–75.
[8] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: Proceedings of MobiCom '00, Aug 2000, pp. 56–67.
[9] F. Ye, A. Chen, S. Lu, L. Zhang, A scalable solution to minimum cost forwarding in large sensor networks, in: Proceedings of ICCCN '01, Oct 2001, pp. 304–309.
[10] D. Braginsky, D. Estrin, Rumour routing algorithm for sensor networks, in: Proceedings of the WSNA '02, September 2002, pp. 22–31.
[11] J. Hubaux, L. Buttyan, S. Capkun, The quest for security in mobile ad hoc networks, in: Proceedings of ACM MobiHoc '01, October 2001, pp. 146–155.
[12] A.A. Pirzada, C. McDonald, Secure routing protocols for mobile ad-hoc wireless networks, in: T.A. Wysocki, A. Dadej, B.J. Wysocki (Eds.), Advanced Wired and Wireless Networks, Springer, 2004.
[13] F. Delgosha, F. Fekri, Key pre-distribution on wireless sensor networks using multivariate polynomials, in: Proceedings of SECON '05, September 2005, pp. 118–129.
[14] J. Deng, R. Han, S. Mishra, INSENS: intrusion-tolerant routing for wireless sensor networks, Elsevier Computer Communications 29 (2006) 216–230.
[15] E. Shi, A. Perrig, Designing secure sensor networks, IEEE Wireless Communications 11 (2004) 38–43.
[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, in: Proceedings of MobiCom '01, July 2001, pp. 189–199.
[17] D.E. Denning, An intrusion detection model, in: Proceedings of the IEEE Symposium on Security and Privacy, 1986, pp. 118–131.
[18] Y. Zhang, W. Lee, Intrusion detection in wireless ad-hoc networks, in: Proceedings of MobiCom '00, August 2000, pp. 275–283.
[19] Y. Huang, W. Lee, A cooperative intrusion detection system for ad hoc networks, in: Proceedings of SASN '03, October 2003, pp. 135–147.
[20] A.A. Pirzada, C.S. Mcdonald, Circumventing sinkholes and wormholes in ad-hoc wireless networks, Proceedings of International Workshop on Wireless Ad-hoc Networks, London, England, Kings College, London, 2005.
[21] Y.-C. Hu, A. Perrig, D.B. Johnson, Packet leashes: a defense against wormhole attacks, in: Proceedings of INFOCOM '05, March 2005, pp. 1976–1986.
[22] S.S. Doumit, D.P. Agrawal, Self-organized critically & stochastic learning based intrusion detection system for wireless sensor networks, in: Proceedings of MILCOM '03, October 2003, pp. 609–614.
[23] A.D. Wood, J.A. Standovic, S.H. Son, JAM: A jammed-area mapping service for sensor networks, in: Proceedings of RTSS '03, December 2003, pp. 286–297.
[24] J. Staddon, D. Balfanz, G. Durfee, Efficient tracing of failed nodes in sensor networks, in: Proceedings of WSNA '02, September 2002, pp. 122–130.
[25] M. Ding, D. Chen, K. Xing, X. Cheng, Localized fault-tolerant event boundary detection in sensor networks, in: Proceedings of INFOCOM '05, March 2005, pp. 902–913.
[26] F. Ye, H. Luo, S. Lu, L. Zhang, Statistical en-route filtering of injected false data in sensor networks, in: Proceedings of INFOCOM '04, March 2004, pp. 2446–2457.
[27] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, L.W. Chang, Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach, in: Proceedings of WCNC '05, March 2005, pp.1193–1199.
[28] L. Hu, D. Evans, Localization for mobile sensor networks, in: Proceedings of MobiCom '04, September 2004, pp. 45–57.
[29] D. Wagner, Resilient aggregation in sensor networks, in: Proceedings of SASN '04, October 2004, pp. 78–87.
[30] N. Ye, Q. Chen, An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems, Quality and Reliability Engineering International 17 (2001) 105–112.
[31] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, in: Neal Koblitz (Ed.), Advances in Cryptology CRYPTO '96. vol. 1109, of Lecture Notes in Computer Science, Springer-Verlag, Berlin Germany, 1996, pp. 1–15.
[32] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, in: Proceedings of CCS '03, October 2003, pp. 52–61.
[33] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu, Analyzing and modeling encryption overhead for sensor network nodes, in: Proceedings of WSNA '03, September 2003, pp. 151–159.
[34] S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in: Proceedings of CCS '03, October 2003, pp. 62–72.
[35] W.B. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Communications 1 (2002) 660–670.
[36] MPR/MID User's Manual, Document 7430-0021-06, Rev. B, Crossbow Technology, Inc., April 2005.
[37] R. Jurdak, C.V. Lopes, P. Baldi, Battery lifetime estimation and optimization for underwater sensor networks, IEEE Sensor Network Operations (2004).

Edith C. H. Ngai received her B.Eng. degree in Computer Engineering and M.Phil. degree in Computer Science and Engineering from The Chinese University of Hong Kong in 2002 and 2004, respectively. She is currently a Ph.D. candidate in the Computer Science and Engineering Department of The Chinese University of Hong Kong. Her research interests include mobile ad hoc networks, wireless sensor networks, network security, network protocol and algorithm design, and video information processing. She is a student member of IEEE.

Jiangchuan Liu received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology in 2003, both in computer science. He is currently an assistant professor in the School of Computing Science, Simon Fraser University, BC, Canada, and was an assistant professor at The Chinese University of Hong Kong from 2003 to 2004. His research interests include Internet architecture and protocols, media streaming, wireless ad hoc networks, and service overlay networks. He serves as TPC member for various international conferences, including IEEE INFOCOM and IWQoS. He was Information System Co-Chair for IEEE INFOCOM'04, and a guest-editor for ACM/Kluwer Journal of Mobile Networks and Applications (MONET), Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks. He is an editor of IEEE Communications Surveys and Tutorials. He is a member of IEEE and ACM, and an elected member of Sigma Xi.

Michael R. Lyu received the B.S. (1981) in electrical engineering from National Taiwan University, the M.S. (1985) in computer engineering from University of California, Santa Barbara, and the Ph.D. (1988) in computer science from University of California, Los Angeles. He is a Professor in the Computer Science and Engineering Department of the Chinese University of Hong Kong. He worked at the Jet Propulsion Laboratory, Bellcore, and Bell Labs; and taught at the University of Iowa. His research interests include software reliability engineering, software fault tolerance, distributed systems, image and video processing, multimedia technologies, and mobile networks. He has published over 200 papers in these areas. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. Professor Lyu was frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was Program Chair for ISSRE'1996, Program Co-Chair for WWW10 and SRDS'2005, and General Chair for ISSRE'2001 and PRDC'2005. He also received Best Paper Awards in ISSRE'98 and in ISSRE'2003. He is the editor-in-chief for two book volumes: Software Fault Tolerance (Wiley, 1995), and the Handbook of Software Reliability Engineering (IEEE and McGraw-Hill, 1996). He has been an Associate Editor of IEEE Transactions on Reliability, IEEE Transactions on Knowledge and Data Engineering, and Journal of Information Science and Engineering. Professor Lyu was elected to IEEE Fellow (2004) and AAAS Fellow (2007) for his contributions to software reliability engineering and software fault tolerance.