ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (765-776), Month: April - June 2015, Available at: www.researchpublish.com

Detecting Packet Droppers and Modifiers in Wireless Sensor Networks

[1]Ms. Shruthi N M, [2]Mrs. Veena M

[2]Assistant Professor, [1, 2] Department of Computer Science, PES College of Engineering, Mandya, India

Abstract: **Packet dropping and modification are common attacks that can be launched by an adversary to disrupt communication in wireless multihop sensor networks. Many schemes have been proposed to mitigate or tolerate such attacks, but very few can effectively and efficiently identify the intruders. To address this problem, we propose a simple yet effective scheme, which can identify misbehaving forwarders that drop or modify packets. Extensive analysis and simulations have been conducted to verify the effectiveness and efficiency of the scheme.**

Keywords: **Packet dropping, packet modification, intrusion detection, wireless sensor networks.**

1. INTRODUCTION

In a wireless sensor network, sensor nodes monitor the environment, detect events of interest, produce data, and collaborate in forwarding the data toward a sink, which could be a gateway, base station, storage node, or querying user. Because of the ease of deployment, the low cost of sensor nodes and the capability of self-organization, a sensor network is often deployed in an unattended and hostile environment to perform the monitoring and data collection tasks. When it is deployed in such an environment, it lacks physical protection and is subject to node compromise. After compromising one or multiple sensor nodes, an adversary may launch various attacks [1] to disrupt the in-network communication. Among these nodes, an adversary may launch various attacks [1] to disrupt the in-network communication. Among these attacks, two common ones are dropping packets and modifying packets, i.e., compromised nodes drop or modify the packets that they are supposed to forward.

To deal with packet droppers, a widely adopted countermeasure is multipath forwarding [2], [3], [4], [5], in which each packet is forwarded along multiple redundant paths and hence packet dropping in some but not all of these paths can be tolerated. To deal with packet modifiers, most of existing countermeasures [6], [7], [8], [9] aim to filter modified messages en-route within a certain number of hops. These countermeasures can tolerate or mitigate the packet dropping and modification attacks, but the intruders are still there and can continue attacking the network without being caught.

To locate and identify packet droppers and modifiers, it has been proposed that nodes continuously monitor the forwarding behaviours of their neighbours [10], [11], [12], [13], [14], [15] to determine if their neighbours are misbehaving, and the approach can be extended by using the reputation based mechanisms to allow nodes to infer whether a nonneighbor node is trustable [16], [17], [18], [19]. This methodology may be subject to high-energy cost incurred by the promiscuous operating mode of wireless interface; moreover, the reputation mechanisms have to be exercised with cautions to avoid or mitigate bad mouth attacks and others. Recently, Ye et al. proposed a probabilistic nested marking (PNM) scheme [20]. But with the PNM scheme, modified packets should not be filtered out en route because they should be used as evidence to infer packet modifiers; hence, it cannot be used together with existing packet filtering schemes.

In this paper, we propose a simple yet effective scheme to detected both packet droppers and modifiers. In this scheme, a routing tree rooted at the sink is first established. When sensor data are transmitted along the tree structure toward the sink, each packet sender or forwarder adds a small number of extra bits, which is called packet marks, to the packet. The format of the small packet marks is deliberately designed such that the sink can obtain very useful information from the

marks. Specifically, based on the packet marks, the sink can figure out the dropping ratio associated with every sensor node, and then runs our proposed node categorization algorithm to identify nodes that are droppers/modifiers for sure or are suspicious droppers/ modifiers. As the tree structure dynamically changes every time interval, behaviours of sensor nodes can be observed in a large variety of scenarios. As the information of node behaviours has been accumulated, the sink periodically runs our proposed heuristic ranking algorithms to identify most likely bad nodes from suspiciously bad nodes. This way, most of the bad nodes can be gradually identified with small false positive.

Our proposed scheme has the following features: 1) being effective in identifying both packet droppers and modifiers, 2) low communication and energy overheads, and 3) being compatible with existing false packet filtering schemes; that is, it can be deployed together with the false packet filtering schemes, and therefore it cannot only identify intruders but also filter modified packets immediately after the modification is detected. Extensive simulation on ns-2 simulator has been conducted to verify the effectiveness and efficiency of the proposed scheme in various scenarios.

In the rest of the paper, Section 2 defines the system model. Section 3 describes the proposed scheme and Section 4 reports the evaluation results. Section 5 discusses the related work, and Section 6 concludes the paper.

2. SYSTEM MODEL

2.1 Network Assumptions:

We consider a typical deployment of sensor networks, where a large number of sensor nodes are randomly deployed in a two dimensional area. Each sensor node generates sensory data periodically and all these nodes collaborate to forward packets containing the data toward a sink. The sink is located within the network. We assume all sensor nodes and the sink are loosely time synchronized [21], which is required by many applications. Attack resilient time synchronization schemes, which have been widely investigated in wireless sensor networks [22], [23], can be employed. The sink is aware of the network topology, which can be achieved by requiring nodes to report their neighbouring nodes right after deployment.

2.2 Security Assumptions and Attack Model:

We assume the network sink is trustworthy and free of compromise, and the adversary cannot successfully compromise regular sensor nodes during the short topology establishment phase after the network is deployed. This assumption has been widely made in existing work [8], [24]. After then, the regular sensor nodes can be compromised. Compromised nodes may or may not collude with each other. A compromised node can launch the following two attacks:

Packet dropping: A compromised node drops all or some of the packets that is supposed to forward. It may also drop the data generated by itself for some malicious purpose such as framing innocent nodes.

Packet modification: A compromised node modifies all or some of the packets that is supposed to forward. It may also modify the data it generates to protect itself from being identified or to accuse other nodes.

3. THE PROPOSED SCHEME

Our proposed scheme consists of a system initialization phase and several equal-duration rounds of intruder identification phases.

- In the initialization phase, sensor nodes form a topology which is a directed acyclic graph (DAG). A routing tree is extracted from the DAG. Data reports follow the routing tree structure.

- In each round, data are transferred through the routing tree to the sink. Each packet sender/ forwarder adds a small number of extra bits to the packet and also encrypts the packet. When one round finishes, based on the extra bits carried in the received packets, the sink runs a node categorization algorithm to identify nodes that must be bad (i.e., packet droppers or modifiers) and nodes that are suspiciously bad (i.e., suspected to be packet droppers and modifiers).

- The routing tree is reshaped every round. As a certain number of rounds have passed, the sink will have collected information about node behaviors in different routing topologies. The information includes which nodes are bad for sure, which nodes are suspiciously bad, and the nodes' topological relationship. To further identify bad nodes from the potentially large number of suspiciously bad nodes, the sink runs heuristic ranking algorithms.

In the following sections, we first present the algorithm for DAG establishment and packet transmission, which is followed by our proposed categorization algorithm, tree structure reshaping algorithm, and heuristic ranking algorithms. To ease the presentation, we first concentrate on packet droppers and assume no node collusion. After that, we present how to extend the presented scheme to handle node collusion and detect packet modifiers, respectively.

3.1 DAG Establishment and Packet Transmission:

All sensor nodes form a DAG and extract a routing tree from the DAG. The sink knows the DAG and the routing tree, and shares a unique key with each node. When a node wants to send out a packet, it attaches to the packet a sequence number, encrypts the packet only with the key shared with the sink, and then forwards the packet to its parent on the routing tree. When an innocent intermediate node receives a packet, it attaches a few bits to the packet to mark the forwarding path of the packet, encrypts the packet, and then forwards the packet to its parent. On the contrary, a misbehaving intermediate node may drop a packet it receives. On receiving a packet, the sink decrypts it, and thus finds out the original sender and the packet sequence number. The sink tracks the sequence numbers of received packets for every node, and for every certain time interval, which we call a round, it calculates the packet dropping ratio for every node. Based on the dropping ratio and the knowledge of the topology, the sink identifies packet droppers based on rules we derive. In detail, the scheme includes the following components, which are elaborated in the following.

3.1.1 System Initialization:

The purpose of system initialization is to set up secret pairwise keys between the sink and every regular sensor node, and to establish the DAG and the routing tree to facilitate packet forwarding from every sensor node to the sink.

Preloading keys and other system parameters: Each sensor node u is preloaded the following information:

. Ku: a secret key exclusively shared between the node and the sink.

. Lr: the duration of a round.

. Np: the maximum number of parent nodes that each node records during the DAG establishment procedure.

. Ns: the maximum packet sequence number. For each sensor node, its first packet has sequence number 0, the Nsth packet is numbered Ns -1, the (Ns+1)th packet is numbered 0, and so on and so forth.

Topology establishment: After deployment, the sink broadcasts to its one-hop neighbors a 2-tuple h0; 0i. In the 2-tuple, the first field is the ID of the sender (we assume the ID of sink is 0) and the second field is its distance in hop from the sender to the sink. Each of the remaining nodes, assuming its ID is u, acts as follows:

1. On receiving the first 2-tuple <v,dv>, node v sets its own distance to the sink as du =dv+1.

2. Node u records each node w (including node v) as its parent on the DAG if it has received <w,dw> where dw =dv. That is, node u records as its parents on the DAG the nodes whose distance (in hops) to the sink is the same and the distance is one hop shorter than its own. If the number of such parents is greater than Np, only Np parents are recorded while others are discarded. The actual number of parents it has recorded is denoted by np,u.

3. After a certain time interval,1node u broadcasts 2- tuple <u,du> to let its downstream one-hop neighbors to continue the process of DAG establishment. Then, among the recorded parents on the DAG, node u randomly picks one (whose ID is denoted as Pu) as its parent on the routing tree. Node u also picks a random number (which is denoted as Ru) between 0 and Np _ 1. As to be elaborated later, random number Ru is used as a short ID of node u to be attached to each packet node u forwards, so that the sink can trace out the forwarding path. Finally, node u sends Pu, Ru and all recorded parents on the DAG to the sink.

After the above procedure completes, a DAG and a routing tree rooted at the sink is established. The routing tree is used by the nodes to forward sensory data until the tree changes later; when the tree needs to be changed, the new structure is still extracted from the DAG.

The lifetime of the network is divided into rounds, and each round has a time length of Lr. After the sink has received the parent lists from all sensor nodes, it sends out a message to announce the start of the first round, and the message is forwarded hop by hop to all nodes in the network. Note that, each sensor node sends and forwards data via a routing tree

which is implicitly agreed with the sink in each round, and the routing tree changes in each round via our tree reshaping algorithm presented in Section 3.3.

3.1.2 Packet Sending and Forwarding:

Each node maintains a counter Cp which keeps track of the number of packets that it has sent so far. When a sensor node u has a data item D to report, it composes and sends the following packet to its parent node Pu:

<Pu , {Ru, u, Cp MOD Ns, D,padu,0}Ku, padu,1>,

Where Cp MOD Ns is the sequence number of the packet. Ru (0 _ Ru _ Np _ 1) is a random number picked by node u during the system initialization phase, and Ru is attached to the packet to enable the sink to find out the path along which the packet is forwarded. {X}Y represents the result of encrypting X using key Y .

Paddings padu,0 and padu,1 are added to make all packets equal in length, such that forwarding nodes cannot tell packet sources based on packet length. Meanwhile, the sink can still decrypt the packet to find out the actual content. To satisfy these two objectives simultaneously, the paddings are constructed as follows:

. For a packet sent by a node which is h hops away from the sink, the length of padu,1 is log(Np)*(h-1) bits. As to be described later, when a packet is forwarded for one hop, log(Np) bits information will be added and meanwhile, log(Np) bits will be chopped off.

. Let the maximum size of a packet be Lp bits, a node ID be Lid bits and data D be LD bits. padu,0 should

be Lp _ Lid * 2 _ log(Np) * h _ log(Ns) _ LD b i t s , where Lid * 2 bits are for Pu and u fields in the packet, field Ru is log(Np) bits long, field padu,1 is log(Np)*(h-1) bits long, and Cp MOD Ns is log(Ns) is log(Np)*(h-1) bits long, and Cp MOD Ns is log(Ns) bits long. Setting padu,0 to this value ensures that all packets in the network have the same length Lp.

When a sensor node v receives packet <v,m>, it composes and forwards the following packet to its parent node Pv:

<Pv,{Rv,m'}Kv>,

Where m' is obtained by trimming the rightmost log Np bits off m'. Meanwhile, Rv, which has logNp bits, is added to the front of m'. Hence, the size of the packet remains unchanged. Suppose on a routing tree, node u s the parent of node v and v is a parent of node w. When u receives a packet from v, it cannot differentiate whether the packet is originally sent by v or w unless nodes u and v collude. Hence, the above packet sending and forwarding scheme results in the difficulty to launch selective dropping, which is leveraged in locating packet droppers. We take special consideration for the collusion scenarios, which are to be elaborated later.

3.1.3 Packet Receiving at the Sink:

We use node 0 to denote the sink. When the sink receives a packet <0,m'>, it conducts the following steps

1. Initialization. Two temporary variables u and m are introduced.

2. The sink attempts to find out a child of node u, denoted as v, such that dec(Kv,m) results in a string starting with Rv, where dec(Kv,m) means the result of decrypting m with key Kv.

3. If the attempt fails for all children nodes of node u, the packet is identified as have been modified and thus should be dropped.

4. If the attempt succeeds, it indicates that the packet was forwarded from node v to node u. Now, there are two cases:

a. If dec(Kv,m) starts with <Rv, v>, it indicates that node v is the original sender of the packet. The sequence number of the packet is recorded for further calculation and the receipt procedure completes.

b. Otherwise, it indicates that node v is an intermediate forwarder of the packet. Then, u is updated to be v, m is updated to be the string obtained by trimming Rv from the leftmost.

Then, steps 2-4 are repeated.

Algorithm 1. Packet Receipt at the Sink

1: Input: packet <0,m>.

2: u =0, m' = m;

3: has Succ Attemp = false;

4: for each child node v of node u do

5: P = dec<Kv,m');

6: if decryption fails then

7: continue;

8: else

9: has SuccAttemp =true;

10: if P starts with <Rv, v> then

11: record the sequence number;

12: break;

13: else

14: trim Rv from P and get

15: u <-v, hasSuccAttemp =false; go to line 4

16: if hasSuccAttemp= false;

17: drop this packet;

3.2 Node Categorization Algorithm:

In every round, for each sensor node u, the sink keeps track of the number of packets sent from u, the sequence numbers of these packets, and the number of flips in the sequence numbers of these packets, (i.e., the sequence number changes from a large number such as Ns _ 1 to a small number such as 0). In the end of each round, the sink calculates the dropping ratio for each node u. Suppose nu;max is the most recently seen sequence number, nu,flip is the number of sequence number flips, and nu,rcv is the number of received packets. The dropping ratio in this round is calculated as follows:

du =nu,flip *Ns + nu,max + 1 - nu,rcv

nu;flip _ Ns þ nu;max þ 1

Based on the dropping ratio of every sensor node and the tree topology, the sink identifies the nodes that are droppers for sure and that are possibly droppers. For this purpose, a threshold _ is first introduced. We assume that if a node's packets are not intentionally dropped by forwarding nodes, the dropping ratio of this node should be lower than _. Note that _ should be greater than 0, taking into account droppings caused by incidental reasons such as collisions. The first step of the identification is to mark each node with "+" if its dropping ratio is lower than _, or with "_" otherwise. After then, for each path from a leaf node to the sink, the nodes' mark pattern in this path can be decomposed into any combination of the following basic patterns, which are also illustrated by Fig. 1:

. +{+}: a node and its parent node are marked as "+"

. + - {-}*: a node is marked as "+," but its one or more continuous immediate upstream nodes are marked as "_."

.-{+} : a node is marked as "-," but its parent node is marked as "þ+"

. -{-}: a node and its parent node are marked as "_."

For each of the above cases, we can infer whether a node

1. Has dropped packets (called bad for sure),

2. Is suspected to have dropped packets (called suspiciously bad),

3. Has not been found to drop packets (called temporarily good), or

4. Must have not dropped packets (called good for sure):

Case 1: +{+}. The node and its parent node do not drop packets along the involved path, but it is unknown whether they drop packets on other forwarding paths. Therefore, the sink infers that these nodes are temporarily good. For example, in Fig. 1a, node C and E are marked "+" and are regarded as temporarily good. A special case is, if a leaf node is marked as "+," it is safe to infer it as good since it cannot drop other's packets.

Case 2: + -{-}*. In the case, all nodes marked as "_"must be bad for to show the correctness of this rule, we prove it by contradiction. Without loss of generality, we examine the scenario illustrated in Fig. 1b, where node C is marked as "+," and nodes E, F, and G are marked as "_." If our conclusion is incorrect and node E is good, E must not drop its own packets. Since E is marked as "_," there must be some upstream nodes of E dropping E's packets. Note that the bad upstream nodes are at least one hop above E, i.e., at least two hops above C. It is impossible for them to differentiate packets from E and C, so they cannot selectively drop the packets from E while forwarding the packets from C. Even if C and the bad upstream node collude, they cannot achieve this. This is because every packet from C must go through and be encrypted by E, and therefore the bad upstream node cannot tell the source of the packet to perform selective dropping. Note that, if a packet is forwarded to the bad upstream node without going through E, the packet cannot be correctly decrypted by the sink and thus will be dropped. Therefore, E must be bad. Similarly, we can also conclude that F and G are also bad.

Case 3:-{+}. In this case, either the node marked as "-" or its parent marked as "þ" must be bad. But it cannot be further inferred whether 1) only the node with "_" is bad, 2) only the node with "þ" is bad, or 3) both nodes are bad. Therefore, it is concluded that both nodes are suspiciously bad. The correctness of this rule can also be proved by contradiction. Without loss of generality, let us consider the scenario shown in Fig. 1c, where node C is marked as "_," and node E is marked as "þ." Now suppose both C and E are good, and hence there must exist at least one upstream node of E which is a bad node that drops the packets sent by C. However, it is impossible to find such an upstream node since nodes F and G, and other upstream nodes cannot selectively drop packets from node C while forwarding packets from node E. Hence, either node C is bad or node E is bad in this case.

Case 4: -{-}. In this case, every node marked with "-" could be bad or good. Conservatively, they have to be considered as suspiciously bad. Specifically, suppose v is the highest-level node that is marked as "_," and u is its parent node. If u is the sink, v must be bad for sure; otherwise, both u and v are suspiciously bad. On the other hand, suppose v is a child of u and they are both marked as "_." If the dropping ratio of u is larger than that of v by at least _ (i.e., dv < du and du _ dv > _, recalling that _ is a threshold used to tolerate incidental droppings), node u is bad for sure. Otherwise, both u and v are suspiciously bad.

Based on the above rules, we develop a node categorization algorithm to find nodes that are bad for sure or suspiciously bad. The formal algorithm is presented in

Algorithm 2. Tree-Based Node Categorization Algorithm

1: Input: Tree T, with each node u marked by "+" or "-," and its dropping ratio du.

2: for each leaf node u in T do

3: v <- u's parent;

4: while u is not the Sink do

5: if u,mark = ''+'' then

6: if v,mark = ''-'' then

7: b <- v;

8: repeat

9: e <- v;

10: v <- v's parents node;

11: until v.mark = ‘‘+’’ or v is Sink

12: Set nodes from b to e as bad for sure;

13: else

14: if v is Sink then

15: Set u as bad for sure;

16: if v.mark = ‘‘+’’ then

17: if v is not bad for sure then

18: Set u and v as suspiciously bad;

19: else

20: if dv - du > _ then

21: Set v as bad for sure;

22: else if du - dv > _ then

23: Set u and v as suspiciously bad;

24: u <- v, v <- v’s parents node

3.3 Tree Reshaping and Ranking Algorithms:

The tree used to forward data is dynamically changed from round to round, which enables the sink to observe the behaviour of every sensor node in a large variety of routing topologies. For each of these scenarios, node categorization algorithm is applied to identify sensor nodes that are bad for sure or suspiciously bad. After multiple rounds, sink further identifies bad nodes from those that are suspiciously bad by applying several proposed heuristic methods.

3.3.1 Tree Reshaping:

The tree used for forwarding data from sensor nodes to the sink is dynamically changed from round to round. In other words, each sensor node may have a different parent node from round to round. To let the sink and the nodes have a consistent view of their parent nodes, the tree is reshaped as follows. Suppose each sensor node u is preloaded with a hash function h(.) and a secret number Ku which is exclusively shared with the sink. At the beginning of each round i (i =1; 2; . . .), node u picks the [hi(Ku) MOD np,u]th parent node as its parent node for this round, where hi(Ku) = h(hi-1(Ku)) and np,u is the number of candidate parent nodes that node u recorded during the tree establishment phase. Recall that node u’s candidate parent nodes are those which are one hop closer to the sink and within node u’s communication range. Therefore, if node u choose node w as its parent in a round, node w will not select node u as its parent, and the routing loop will not occur. Note that, how the parents are selected is predetermined by both the preloaded secret Ku and the list of parents recorded in the tree establishment phase. The selection is implicitly agreed between each node and the sink. Therefore, a misbehaving node cannot arbitrarily select its parent in favor of its attacks.

3.3.2 Identifying Most Likely Bad Nodes from Suspiciously Bad Nodes:

We rank the suspiciously bad nodes based on their probabilities of being bad, and identify part of them as most likely bad nodes. Specifically, after a round ends, the sink calculates the dropping ratio of each node, and runs the node categorization algorithm specified as Algorithm 2 to identify nodes that are bad for sure or suspiciously bad.

Since the number of suspiciously bad nodes is potentially large, we propose how to identify most likely bad nodes from the suspiciously bad nodes as follows. By examining the rules in Cases 3 and 4 for identifying suspiciously bad nodes, we can observe that in each of these cases, there are two nodes having the same probability to be bad and at least one of them must be bad. We call these two nodes as a suspicious pair. For each round i, all identified suspicious pairs are recorded in a suspicious set denoted as

Si ={<uj.vj>|< uj,vj> is a suspicious pair and <uj,vj>=<vj,uj>}

Therefore, after n rounds of detection, we can obtain a series of suspicious sets: S1; S2; . . . ; Sn.

We define S as the set of most likely bad nodes identified from S1,S2, . . . , Sn, if S has the following properties:

. Coverage. 8hu; vi 2 Si (i ¼ 1; . . . ; n), it must hold that either u 2 S or v 2 S. That is, for any identified suspicious pair, at least one of the nodes in the pair must be in the set of most likely bad nodes.

. Most-likeliness. 8hu; vi 2 Si (i ¼ 1; . . . ; n), if u 2 S but v 62 S, then u must have higher probability to be bad than v based on n rounds of observation.

. Minimality. The size of S should be as small as possible in order to minimize the probability of misaccusing innocent nodes. Among the above three conditions, the first one and the third one can be relatively easily implemented and verified. For the second condition, we propose several heuristics to find nodes with most-likeliness.

Global ranking-based (GR) method. The GR method is based on the heuristic that, the more times a node is identified as suspiciously bad, the more likely it is a bad node. With this method, each suspicious node u is associated with an accused account which keeps track of the times that the node has been identified as suspiciously bad nodes. To find out the most likely set of suspicious nodes after n rounds of detection, as described in Algorithm 3, all suspicious nodes are ranked based on the descending order of the values of their accused accounts. The node with the highest value is chosen as a most likely bad node and all the pairs that contain this node are removed from S1; . . . ; Sn, resulting in new sets. The process continues on the new sets until all suspicious pairs have been removed. The GR method is formally presented in Algorithm 3.

Algorithm 3. The Global Ranking-Based Approach1: Sort all suspicious nodes into queue Q according to the descending order of their accused account values

2: S ;

3: while Sn i¼1 Si 6¼ ; do

4: u dequeðQÞ

5: S S ^ fug

6: remove all hu; _i from Sn i¼1 Si

Step wise ranking-based (SR) method. It can be anticipated that the GR method will falsely accuse innocent nodes that have frequently been parents or children of bad nodes: as parents or children of bad nodes, according to previously described rules in Cases 3 and 4, the innocents can often be classified as suspiciously bad nodes. To reduce false accusation, we propose the SR method. With the SR method, the node with the highest accused account value is still identified as a most likely bad node. However, once a bad node u is identified, for any other node v that has been suspected together with node u, the value of node v's accused account is reduced by the times that u and v have been suspected together. This adjustment is motivated by the possibility that v has been framed by node u. After the adjustment, the node that has the highest value of accused account among the rest nodes is identified as the next mostly like bad node, which is followed by the adjustment of the accused account values for the nodes that have been suspected together with the node. Note that, similar to then GR method, after a node u is identified as bad, all suspicious pairs with format hu; _i are removed from S1; . . . ; Sn. The above process continues until all suspicious pairs have been removed. The SR method is formally presented in Algorithm 4.

Algorithm 4. The Stepwise Ranking-Based Approach

1: S ;

2: while Sn i¼1 Si 6¼ ; do

3: u the node has the maximum times of presence in S1; . . . ; Sn

4: S S ^ fug

5: remove all hu; _i from Sn i¼1 Si

Hybrid ranking-based (HR) method. The GR method can detect most bad nodes with some false accusations while the SR method has fewer false accusations but may not detect as many bad nodes as the GR method. To strike a balance, we further propose the HR method, which is formally presented in Algorithm 5. According to HR, the node with the highest

accused account value is still first chosen as most likely bad node. After a most likely bad node has been chosen, the one with the highest accused account value among the rest is chosen only if the node has not always been accused together with the bad nodes that have been identified already. Thus, the accusation account value is considered as an important criterion in identification, as in the GR method; meanwhile, the possibility that an innocent node being framed by bad nodes is also considered by not choosing the nodes which are always being suspected together with already identified bad nodes, as in the SR method. The HR method is formally presented in Algorithm 5.

Algorithm 5. The Hybrid Ranking-Based Approach

1: Sort all suspicious nodes into queue Q according to the descending order of their accused account values

2: S ;

3: while Sn i¼1 Si 6¼ ; do

4: u dequeðQÞ

5: if there exists hu; _i 2 Sn i¼1 Si then

6: S S ^ fug

7: remove all <u,*> from Sn i=1 Si

3.4 Handling Collusion:

Because of the deliberate hop by hop packet padding and encryption, the packets are not distinguishable to the upstream compromised nodes as long as they have been forwarded by an innocent node. The capability of launching collusion attacks is thus limited by the scheme. However, compromised nodes that are located close with each other may collude to render the sink to accuse some innocent nodes. We discuss the possible collusion scenarios in this section and propose strategies to mitigate the effects of collusion.

As the four cases described in Section 3.2, the attackers do not gain any benefit if the collusion triggers the scenarios of Cases 1 and 2. However, the attackers may accuse honest nodes if the collusion triggers the scenarios of Cases 3 and 4. By exploiting the rules used by the node categorization algorithm and rank algorithm, there are two possible collusion strategies to make the sink accuse innocent nodes. We use Fig. 2 as a general example to discuss the collusion scenarios.

Horizontal collusion. If nodes B, C, and D are compromised and collude, they will drop all or some of the packets of their own and their downstream nodes. Consequently, according to the rules in Case 3, <A,B>, <A,C>, and <A,D> are all identified as pairs of suspiciously bad nodes. Since A has been suspected for more times than B, C, and D, it is likely that A is falsely identified as bad node.

Vertical collusion. If nodes B and E are compromised and collude, B may drop some packets of itself and its downstream nodes, and then E further drops packets from its downstream nodes including B and B's downstream nodes. Note that, E cannot differentiate the packets forwarding/generating by B since they are encrypted by node A. Consequently, the dropping rates for B and its downstream nodes are higher than that for node A. According to Case 4, <E,A> and <A,B> are both identified as pairs of suspiciously bad nodes. Since A has been suspected for more times than B and E, it is likely to be identified as a bad node.

To defeat collusion that may lead to false accusation, our scheme is extended as follows:

The concept of suspicious pair is extended to suspicious tuple which is a nonordered sequence of suspicious nodes. Note that, a suspicious pair is a special case of suspicious tuple, i.e., suspicious 2-tuple.

A new rule is introduced: for each round i, if there exists multiple suspicious tuples of which each contains a certain node u, <u, v1,1, . . . , v1,m1> <u, vn,1, . . . , vn,m>, all these tuples should be combined into a single tuple without duplication. For example, if the original tuples are <u, v1>, <u, v2, v3>, and <u, v3>, these tuples will be replaced with <u, v1, v2, v3>, where each of the four nodes is suspected for only once.

As to be shown in our simulation results, the above enhancement can deal with collusion at the cost of slightly degraded detection rate.

3.5 An Extension for Identifying Packet Modifiers:

The proposed scheme can be extended for identifying packet modifiers. Particularly, it can be slightly modified so that the statistical en route filtering scheme (SEF) [6] and the interleaved hop-by-hop authentication scheme [7] can be deployed to filter the modified packets. Details are presented in Section 3.5 in the supplementary file, available in the online supplemental material.

4. PERFORMANCE EVALUATION

The effectiveness and efficiency of the proposed scheme are evaluated in the ns-2 simulator (version 2.30). The detailed performance metric, methodology as well as the attack models are discussed in Section 4.1 in the supplementary file, available in the online supplemental material.

The simulation results are presented in the supplementary file, available in the online supplemental material. We first study the impact of various system parameters on the detection performance from Sections 4.2.1 to 4.2.7 when there is no collusion. We then evaluate our proposed scheme under node collusion attacks in Section 4.2.8.

To identify packet modifiers and droppers, it has been proposed to add nested MACs to address this problem in [20] and [25]. We compare our proposed scheme with the PNM scheme [20] regarding detection performance and communication overhead. Details are presented in Section 4.3 in the supplementary file, available in the online supplemental material.

As the proposed scheme outperforms the PNM scheme in terms of detection performance and communication overhead, we further measure the computational overhead of the packet sending and forwarding scheme on TelosB motes, which are widely used resource-constrained sensor motes [26]. Details are shown in Section 4.4 in the supplementary file, available in the online supplemental material.

5. RELATED WORK

The approaches for detecting packet dropping attacks can be categorized as three classes: multipath forwarding approach, neighbour monitoring approach, and acknowledgment approach. Multipath forwarding [4], [5] is a widely adopted countermeasure to mitigate packet droppers, which is based on delivering redundant packets along multiple paths. Another approach is to exploit the monitoring mechanism [10], [13], [14], [16], [17], [18], [19], [27]. The watchdog method was originally proposed to mitigate routing misbehaviour in mobile ad hoc networks [10]. It is then adopted to identify packet droppers in wireless sensor network [13], [27], [28]. When the watchdog mechanism is deployed, each node monitors its neighborhood promiscuously to collect the firsthand information on its neighbuor nodes. A variety of reputation systems have been designed by exchanging each node's firsthand observations, which are further used to quantify node's reputation [16], [17], [18], [19]. Based on the monitoring mechanism, the intrusion detection systems are proposed in [15] and [29]. However, the watchdog method requires nodes to buffer the packets and operate in the promiscuous mode, the storage overhead and energy consumption may not be affordable for sensor nodes. In addition, this mechanism relies on the bidirectional communication links, it may not be effective when directional antennas are used [30].Particularly, this approach cannot be applied when a node does not know the expected output of its next hop since the node has no way to find a match for buffered packets and overheard packets. Note that, this scenario is not rare, for example, the packets may be processed, and then encrypted by the next hop node in many applications that security is required. Since the watchdog is a critical component of reputation systems, the limitations of the watchdog mechanism can also limit the reputation system. Besides, a reputation system itself may become the attacking target. It may either be vulnerable to bad mouthing attack or false praise attack [30]. The third approach to deal with packet dropping attack is the multihop acknowledgment technique [31], [32], [33]. By obtaining responses from intermediate nodes, alarms, and detection of selective forwarding attacks can be conducted. To deal with packet modifiers, most of existing countermeasures [6], [7], [8], [9] are to filter modified messages within a certain number of hops so that energy will not be wasted to transmit modified messages. The effectiveness to detect malicious packet droppers and modifiers is limited without identifying them and excluding them from the network. Researchers hence have proposed schemes to localize and identify packet droppers; one approach is the acknowledgment-based scheme [24], [25], [34] for identifying the problematic communication links. It can deterministically localize links of malicious nodes if every node reports ACK using onion report.

However, this incurs large communication and storage overhead for sensor networks. The probabilistic ACK approaches are then proposed in [24] and [25], which seek trade-offs among detection rate, communication overhead, and storage overhead. However, these approaches assume the packet sources are trustable, which may not be valid in sensor networks. As in sensor networks, base station typically is the only one we can trust. Furthermore, these schemes require to set up pair wise keys among regular sensor nodes so as to verify the authenticity of ACK packets, which may cause considerable overhead for key management in sensor networks. Ye et al. [20] proposed a scheme called PNM for identifying packet modifiers probabilistically. However, the PNM scheme cannot be used together with the false packet filtering schemes [6], [7], [8], [9], because the filtering schemes will drop the modified packets which should be used by the PNM scheme as evidences to infer packet modifiers. This degrades the efficiency of deploying the PNM scheme.

6. CONCLUSION

We propose a simple yet effective scheme to identify misbehaving forwarders that drop or modify packets. Each packet is encrypted and padded so as to hide the source of the packet. The packet mark, a small number of extra bits, is added in each packet such that the sink can recover the source of the packet and then figure out the dropping ratio associated with every sensor node. The routing tree structure dynamically changes in each round so that behaviours of sensor nodes can be observed in a large variety of scenarios. Finally, most of the bad nodes can be identified by our heuristic ranking algorithms with small false positive. Extensive analysis, simulations, and implementation have been conducted and verified the effectiveness of the proposed scheme.

REFERENCES

[1] H. Chan and A. Perrig, “Security and Privacy in Sensor Networks,” Computer, vol. 36, no. 10, pp. 103-105, Oct. 2003.

[2] C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,” Proc. IEEE First Int’l Workshop Sensor Network Protocols and Applications, 2003.

[3] V. Bhuse, A. Gupta, and L. Lilien, “DPDSN: Detection of Packet- Dropping Attacks for Wireless Sensor Networks,” Proc. Fourth Trusted Internet Workshop, 2005.

[4] M. Kefayati, H.R. Rabiee, S.G. Miremadi, and A. Khonsari, “Misbehavior Resilient Multi-Path Data Transmission in Mobile Ad-Hoc Networks,” Proc. Fourth ACM Workshop Security of Ad Hoc and Sensor Networks (SASN ’06), 2006.

[5] R. Mavropodi, P. Kotzanikolaou, and C. Douligeris, “Secmr—A Secure Multipath Routing Protocol for Ad Hoc Networks,” Ad Hoc Networks, vol. 5, no. 1, pp. 87-99, 2007.

[6] F. Ye, H. Luo, S. Lu, and L. Zhang, “Statistical En-Route Filtering of Injected False Data in Sensor Networks,” Proc. IEEE INFOCOM, 2004.

[7] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An Interleaved Hop-by- Hop Authentication Scheme for Filtering False Data in Sensor Networks,” Proc. IEEE Symp. Security and Privacy, 2004.

[8] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, “Toward Resilient Security in Wireless Sensor Networks,” Proc. Sixth ACM Int’l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc ’05), 2005.

[9] Z. Yu and Y. Guan, “A Dynamic En-route Scheme for Filtering False Data in Wireless Sensor Networks,” Proc. IEEE INFOCOM, 2006.

[10] S. Marti, T. Giuli, K. Lai, and M. Baker, “Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” Proc. ACM Mobi Com,2000.

[11] M. Just, E. Kranakis, and T. Wan, “Resisting Malicious Packet Dropping in Wireless Ad Hoc Networks,” Proc. Int’l Conf. Ad-Hoc Networks and Wireless (ADHOCNOW ’03), 2003.

[12] R. Roman, J. Zhou, and J. Lopez, “Applying Intrusion Detection Systems to Wireless Sensor Networks,” Proc. IEEE Third Consumer Comm. Networking Conf. (CCNC), 2006.

[13] S. Lee and Y. Choi, "A Resilient Packet-Forwarding Scheme Against Maliciously Packet-Dropping Nodes in Sensor Networks," Proc. Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06), 2006.

[14] I. Khalil and S. Bagchi, "MISPAR: Mitigating Stealthy Packet Dropping in Locally-Monitored Multi-Hop Wireless Ad Hoc Networks," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm '08), 2008.

[15] I. Krontiris, T. Giannetsos, and T. Dimitriou, "LIDeA: A Distributed Lightweight Intrusion Detection Architecture for Sensor Networks," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm '08), 2008.

[16] S. Ganeriwal, L.K. Balzano, and M.B. Srivastava, "Reputation-Based Framework for High Integrity Sensor Networks," ACM Trans. Sensor Networks, vol. 4, no. 3, pp. 1-37, 2008.

[17] W. Li, A. Joshi, and T. Finin, "Coping with Node Misbehaviors in Ad Hoc Networks: A Multi-Dimensional Trust Management Approach," Proc. 11th Int'l Conf. Mobile Data Management (MDM '10), 2010.

[18] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," Proc. IFIP TC6/TC11 Sixth Joint Working Conf. Comm. and Multimedia Security: Advanced Comm. and Multimedia Security, 2002.

[19] S. Buchegger and J. Le Boudec, "Performance Analysis of the Confidant Protocol," Proc. ACM MobiHoc, 2002.

[20] F. Ye, H. Yang, and Z. Liu, "Catching Moles in Sensor Networks," Proc. 27th Int'l Conf. Distributed Computing Systems (ICDCS '07), 2007.

[21] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," Proc. IEEE INFOCOM, 2004.

[22] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "Tinysersync:Secure and Resilient Time Synchronization in Wireless Sensor Networks," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), 2006.

[23] H. Song, S. Zhu, and G. Cao, "Attack-Resilient Time Synchronization for Wireless Sensor Networks," Ad Hoc Networks, vol. 5, no. 1, pp. 112-125, 2007.

[24] B. Xiao, B. Yu, and C. Gao, "Chemas: Identify Suspect Nodes in Selective Forwarding Attacks," J. Parallel and Distributed Computing, vol. 67, no. 11, pp. 1218-1230, 2007.

[25] X. Zhang, A. Jain, and A. Perrig, "Packet-Dropping Adversary Identification for Data Plane Security," Proc. ACM CONEXT Conf. (CoNEXT '08), 2008

[26] Crossbow, "Wireless Sensor Networks," http://www.xbow.com/ Products/Wireless_Sensor_Networks.htm, 2011.

[27] T.H. Hai and E.N. Huh, "Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-Hops Neighbor Knowledge," Proc. IEEE Seventh Int'l Symp. Network Computing and Applications (NCA '08), 2008.

[28] F. Liu, X. Cheng, and D. Chen, "Insider Attacker Detection in Wireless Sensor Networks," Proc. IEEE INFOCOM, 2007.

[29] K. Ioannis, T. Dimitriou, and F.C. Freiling, "Towards Intrusion Detection in Wireless Sensor Networks," Proc. 13th European Wireless Conf., 2007.

[30] A. Srinivasan, J. Teitelbaum, H. Liang, J. Wu, and M. Cardei, "Reputation and Trust-Based Systems for Ad Hoc and Sensor Networks," Proc. Algorithms and Protocols for Wireless Ad Hoc and Sensor Networks, 2008.

[31] J.M. Mccune, E. Shi, A. Perrig, and M.K. Reiter, "Detection of Denial-of-Message Attacks on Sensor Network Broadcasts," Proc. IEEE Symp. Security and Policy, 2005.

[32] B. Yu and B. Xiao, "Detecting Selective Forwarding Attacks in Wireless Sensor Networks," Proc. 20th Int'l Symp. Parallel and Distributed Processing (IPDPS), 2006.

[33] K. Liu, J. Deng, P.K. Varshney, and K. Balakrishnan, "An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in Manets," IEEE Trans. Mobile Computing, vol. 6, no. 5, pp. 536-550, May 2007.

[34] B. Barak, S. Goldberg, and D. Xiao, "Protocols and Lower Bounds for Failure Localization in the Internet," Proc. Eurocryp.