A Novel Packet Loss Control in Wireless Sensor Networks

M.Aparna and K.Ravi chand

Abstract: In wireless multi-hop sensor networks, an intruder may launch some attacks due to packet dropping in order to disrupt the communication. Packet dropping and modification are common attacks that can be launched by an adversary to disrupt communication in wireless multi-hop 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 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 counter-measure is multipath forwarding 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 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.

M.Aparna is a M.tech (CSE) Student Department of CSE, EVMCET, Narasaraopet, Guntur(Dist), Ap, India.
And K.Ravi chand is working as Professor, Head of the Department of CSE, EVMCET, Narasaraopet, Guntur(Dist), Ap, India.
mudiyalaaparna.89@rediffmail.com #1, k_ravichand@yahoo.com#2

To locate and identify packet droppers and modifiers, it has been proposed that nodes continuously monitor the forwarding behaviours of their neighbours 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 non-neighbour node is trustable. 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. 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 and effective system to catch both packet droppers and modifiers. In this system, 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 modifiers. Our proposed system 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.

1. SYSTEM MODEL

A) 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 which is required by many applications. Attack-resilient time synchronization schemes, which have been widely investigated in wireless sensor networks, 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.

B) Security 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:

i) 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.

ii) 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.

2. PROPOSED SYSTEM

We propose a simple effective scheme to catch 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 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.

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.

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

i) 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.

ii) 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).

iii) The routing tree is reshaped every round. As a certain number of rounds have passed, the sink will have collected information about node behaviours in different routing topologies. The information includes which nodes are bad for sure, which nodes are suspiciously bad, and the nodes' topological relationship.

2.1 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.

System Initialization

The purpose of system initialization is to set up secret pair wise 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:

- K_u: A secret key exclusively shared between the node and the sink.
- L_r: The duration of a round.
- N_p: The maximum number of parent nodes that each node records during the DAG establishment procedure.
- N_s: the maximum packet sequence number. For each sensor node, its first packet has sequence number 0, the N_sth packet is numbered N_s - 1, the (N_s + 1)th packet is numbered 0, and so on and so forth.

Topology establishment.

After deployment, the sink broadcasts to its one-hop neighbours a 2-tuple (0, 0). 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, d_v), node u sets its own distance to the sink as $d_u = d_v + 1$.
2. Node u records each node w (including node v) as its parent on the DAG if it has received (w, d_w) where $d_w = d_v$. 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 N_p, only N_p parents are recorded while others are discarded. The actual number of parents it has recorded is denoted by $n_{p,u}$.
3. After a certain time interval,[1] node u broadcasts 2-tuple (u, d_u) 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 node u randomly picks one (whose ID is denoted as P_u) as its parent on the routing tree. Node u also picks a random number (which is denoted as R_u) between 0 and N_p - 1. As to be elaborated later, random number R_u 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 P_u, R_u 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 L_r. 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.

i) Packet Sending and Forwarding

Each node maintains a counter C_p 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 P_u:

$$[P_u; \{R_u, u, C_p \text{ MOD } N_s, D, pad_{u,0}\}_{Ku}, pad_{u,1}],$$

where C_p MOD N_s is the sequence number of the packet. R_u ($0 \leq R_u \leq N_p \leq 1$) is a random number picked by node u during the system initialization phase, and R_u 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 $pad_{u,0}$ and $pad_{u,1}$ 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 puddings are constructed as follows:

For a packet sent by a node which is h hops away from the sink, the length of $pad_{u,1}$ 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 L_{id} bits and data D be L_D bits. $pad_{u,0}$ should be Lp - $L_{id} * 2$ - log(Np) – h- log(N_s) - L_D bits, where $L_{id} * 2$ bits are for P_u and u fields in the packet, field R_u is log(Np) bits long, field $pad_{u,1}$ is log(Np) $*$ (h -1) bits long, and Cp MOD N_s is log(N_s) bits long. Setting $pad_{u,0}$ 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

P_v: $[P_v, \{ R_v, m^l \}_{Kv}]$,

Where m^l is obtained by trimming the rightmost log(Np) bits off m. Meanwhile, R_v, which has log N_p bits, is added to the front of m^l. Hence, the size of the packet remains unchanged. Suppose on a routing tree, node u is 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.

ii) Packet Receiving at Sink

We use node 0 to denote the sink. When the sink receives a packet [0, m^l], it conducts the following steps:

1. Initialization. Two temporary variables u and m are introduced. Let u = 0 and m = m^l initially.
2. The sink attempts to find out a child of node u, denoted as v, such that dec(K_v, m) results in a string starting with R_v, where dec(K_v, m) means the result of decrypting m with key K_v.
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(K_v, m) starts with (R_v, 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 R_v from the leftmost. Then, steps 2-4 are repeated.

Algorithm 1. Packet Receipt at the Sink

```
 1: Input: packet (0; m).
 2: u = 0, $m^l$ = m;
 3: hasSuccAttemp = false;
 4: **for** each child node v of node u **do**
 5:     P = dec($K_v$, $m^l$ );
 6:     **if** decryption fails **then**
 7:     continue;
 8:     **else**
 9:         hasSuccAttemp = true;
10:         if P starts with ($R_v$, v)i then
11:             record the sequence number; / * v is the
                sender */
12:             **break**;
13:         **else**
14:             trim $R_v$ from P and get $m^l$; / * v is
                a forwarder */
15:             u $\leftarrow$ v, hasSuccAttemp = false; go to line 4;
16: **if** hasSuccAttemp = false **then**
17:     drop this packet;
```

2.2 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 N_s - 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 $n_{u,max}$ is the most recently seen sequence number, $n_{u,flip}$ is the number of sequence number flips, and $n_{u,rcv}$ is the number of received packets.

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 Θ 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 Θ Note that Θ 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 Θ, or with "-" otherwise. After then, for each path from a leaf node to the

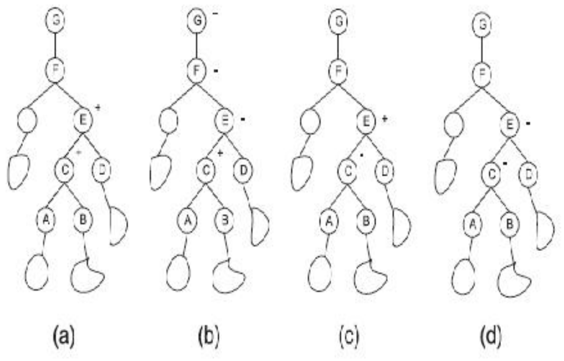

Fig 2.2.1. .Node Status Pattern

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 "+."
- –{-}: 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):

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.

Algorithm 2

Tree-Based Node Categorization Algorithm

```
 1: Input: Tree T, with each node u marked by “+” or “-,”
    and its dropping ratio $d_u$.
 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 $d_v - d_u > \Theta$ then
21:                Set v as bad for sure;
22:             else if $d_u - d_v > \Theta$ then
23:                Set u and v as suspiciously bad;
24:       u ← v, v ← v's parents node.
```

3.2.1 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.

2.3 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 down-stream 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.

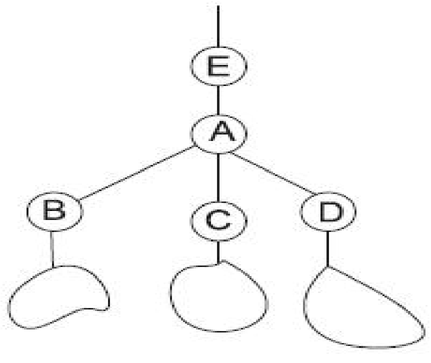

Fig. 2.3.1 Collusion Scenarios

2.4 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 differ-entiate 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.

3. 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.

REFERENCES

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

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.