FINDING PACKET MODIFIERS IN WIRELESS NETWORK USING PACKET DROPPING RATIO

M.ANANDHARAJ[1], DR.K.RAVIKUMAR[2]

[1]*Part time Ph.D Department of CS,/Bharathiar University, Tamil Nadu, (India)*

[2]*Asst. prof. Department of CS,/Tamil University, Tamil Nadu, (India)*

ABSTRACT

In the recent days, wireless communication usage's are huge and modern improvement's as stipulate for wireless system goes on escalating to mount. Today's, the majority traditional and emerging system in wireless Network is Mobile Ad hoc Network (MANET) as number of mobile consumers are goes on increasing day by day. MANET is the wireless network and it has no infrastructure (no centralized server to maintain the entire networks). So that it is appropriate in various and numerous pasture for communication networks such as used in many applications like military communications system, adroit planning operations, and environmental assault marks networks. In the MANET, geo-routing of mobile nodes necessitate to preserve and continue to up-to-date their respective and their immediate neighbors nodes positions in the networks for making effective forwarding of data packets. For the regular updating of the bonfire packets to the neighboring nodes we proposed the adaptive position update for mobile nodes. 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.

I 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 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.

In this paper, we propose a simple yet 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 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, behaviors of sensor nodes can be observed in a large variety of scenarios. As the information of node behaviors 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 can not 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.

II 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 nodesmayormaynot collude with each other. A compromised node can launch the following two attacks:

2.1 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.

2.2 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.

III 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 Packet Transmission. 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 System Initialization

The purpose of system initialization is to set up secret pair wise keys between the sink and every regular sensor node, 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_s th packet is numbered N_s- 1, the (N_s+1) th packet is numbered 0, and so on and so forth.

3.2 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 P_u

$(P_u, \{R_{u,u,} C_p \text{ MOD } N_{S,} D, pad_{u,0}\} K_u, pad_{u,1})$

Where C_p MOD $N_{S,}$ is the sequence number of the packet. R_u ($0 <= R_u <= N_s - 1$) 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 paddings 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 $\log(N_p)*(h-1)$ bits. As to be described later, when a packet is forwarded for one hop, $\log(N_p)$ bits information will be added and meanwhile, $\log(N_p)$ bits will be chopped off.
- Let the maximum size of a packet be L_p bits, a node ID be L_{id} bits and data D be L_D bits. $pad_{u,0}$ should be $L_p - L_{id}*2 - \log(N_p) - h - \log(N_s) - L_{id}$ bits, where L_{id}-2 bits are for P_u and u fields in the packet, field R_u is $\log(N_p)$ bits long, field $pad_{u,1}$ is $\log(N_p) * (h-1)$ bits long, and C_p MOD N_S is $\log(N_p)$ bits long. Setting $pad_{u,0}$ to this value ensures that all packets in the network have the same length L_p.

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'} K_v), where m' is obtained by trimming the rightmost $\log(N_p)$ bits off m. Meanwhile, R_v, which has $\log(N_p)$ bits, is added to the front of m. 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. We take special consideration for the collusion scenarios, which are to be elaborated later.

3.3 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. Let u=0 and m=m' 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:

Algorithm 1. Packet Receipt at the Sink

step 1: Input: packet (0, m).

```
step 2: u=0, m’=m;
step 3: hasSuccAttemp =false;
step 4: for each child node v of node u do
step 5: P= dec($K_v$,m’);
step 6: if decryption fails then
step 7: continue;
step 8: else
step 9: hasSuccAttemp = true;
step 10: if P starts with ($R_v$,v) then
step 11: record the sequence number;
step 12: break;
step 13: else
step 14: trim $R_v$ from P and get m’;
step 15: u < v, hasSuccAttemp = false; go to  step 4;
step 16: if hasSuccAttemp = false then
step 17: drop this packet;
```

IV 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 behavior 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.

4.1 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 K_u which is exclusively shared with the sink. At the beginning of each round i(i=1,2,…), node u picks the [$h^i(K_u)$MOD $n_{p,u}$] th parent node as its parent node for this round, where $h^i(K_u)=h(\ h^{i-1}(K_u))$ and $n_{p,u}$ is the number of candidate parent nodes that node u recorded during the tree establishment phase. Recall that node u’ candidate parent nodes are those which are one hop closer to the sink and within node u’ 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 K_u 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.

4.2 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 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

$$S_i=\{(u_{j,}v_j)| (u_{j,}v_j) \text{ is a suspicious pair and } u_{j,}v_j = v_{j,}u_j \}.$$

Algorithm 2. The Global Ranking-Based Approach

```
1. Sort all suspicious nodes into queue Q according to the
 descending order of their accused account values
2. $\overline{S} \leftarrow \emptyset$
3. While $\bigcup_{i=1}^{n} S_i \neq \emptyset$ do
4. $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
5. Remove all (u,*) from $\bigcup_{i=1}^{n} S_i$
```

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' 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 the GR method, after a node u is identified as bad, all suspicious pairs with format (u,*) are removed from S1,…..Sn.

4.3 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.

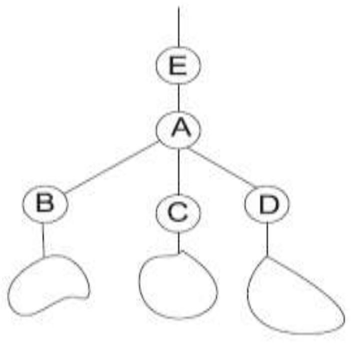
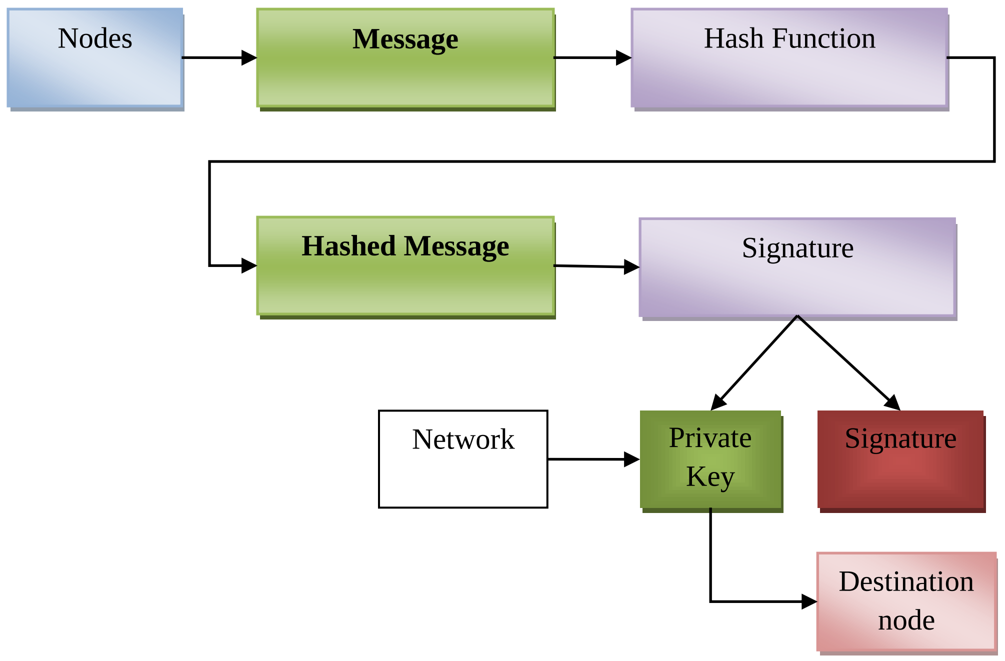


Fig.1: Collusion Scenarios.

BLOCK DIAGRAM



- 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.

V CONCLUSION

We propose a simple 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 behaviors 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 MobiCom, 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. 842 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23, NO. 5, MAY 2012

[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.