Detecting and Identifying of Packet Droppers and Modifiers in Wireless Sensor Networks

Eerla Raghu [#1], P V G K Jagannadha Raju [*2], Dr.S.Maruthu Perumal [*3]

[#1] IInd M.TECH (CSE) Student, [*2] Professor, [*3] Professor & HOD
Department of CSE
Godavari Institute of Engineering and Technology (GIET),
Rajahmundry, AP, India.

Abstract

In Wireless Sensor Network, sensors at different locations can generate streaming/ discrete data, which can be analyzed in Real-time/Non real-time to identify events of interest. A sensor node is often placed in an unfriendly environment to perform the monitoring and data collection tasks. When it is unfriendly environment, node may subject to compromise. After compromising one or multiple sensor nodes, an adversary may launch various attacks to disrupt the in-network communication. Packet dropping and modification are two common attacks that can disrupt communication in wireless multi-hop sensor networks. Many schemes have been proposed to reduce the attacks but none can effectively and efficiently identify the intruders. To address the problem, we propose a simple yet effective scheme, which can identify misbehaving forwarders that drop or modify packets. Extensive analysis and simulations using ns2 simulator have been conducted and verified the effectiveness and efficiency of the scheme.

Keywords:

Wireless networks, sensor networks, Topology, dynamic routing, heuristic ranking

1. Introduction

Wireless Sensor networks consist of large number of small sensor nodes having limited computation capacity, restricted memory space, limited power resource, and short-rage radio communication device. With a widespread deployment of these devices, one can precisely monitor the environment. Basically, sensor networks are application dependent and 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. 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 environment, it lacks physical protection and is subject to node compromise. After compromising one or multiple sensor nodes, an adversary may lunch various attacks to disrupt the in-network communication. This paper deals with two common attacks, dropping packets and modifying packets which can be launched by compromised nodes.

Sensor nodes in a wireless sensor network, monitor the environment, detect events of interest, produce data and co-operate in forwarding the data towards a sink, which could be a gateway, base station, storage node, or querying user. To perform the monitoring and data collection tasks a sensor network may often established in an unattended and hostile environment, which 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. Multi-path forwarding is a widely adopted counter-measure to deal with packet droppers. In multi-path forwarding each packet is forwarded along multiple redundant paths and hence packet dropping in only some path can be tolerated. This scheme introduces high extra communication overhead. Monitoring the behavior of forwarding nodes is another scheme of countermeasures. However, these schemes are subject to high energy cost incurred by the promiscuous operating mode of wireless interface. To deal with packet modifiers, most of existing countermeasures [1]–[4] are to filter modified messages within a certain number of hops.

However, without identifying packet droppers and modifiers, these countermeasures cannot fully solve the packet modification problems because the compromised nodes can continue attacking the network without being caught. To identify packet modifiers, Ye et al. [5] recently proposed a probabilistic nested marking (PNM) scheme to identify packet modifiers with a certain probability. However, the PNM scheme cannot be used together with the false packet filtering schemes [1]–[4], 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.

In this paper, we propose a simple yet effective scheme to catch both packet droppers and modifiers. According to the scheme, a dynamic routing tree rooted at the sink is first established. When sensor data is transmitted along the tree structure towards 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 rate associated with every sensor node, and then run our proposed **node categorization algorithm** to identify nodes that are droppers/ modifiers for sure or are suspicious droppers/modifiers. The tree structure dynamically changes in every time interval. One searching interval is called as a round. As the number of misbehaving node is increased, we have to find out the most suspicious node from among these nodes. So here we perform a *heuristic ranking algorithm* to find out the most suspicious bad node. This way, most of the bad nodes can be gradually identified with small false positive. Our system has the following unique characteristics compared with existing system: (1) being effective in identifying both packet droppers and modifiers, (2) low overhead in terms of both communication and energy consumption, and (3) being compatible with existing false packet filtering schemes [1]–[4], that is, it can be established together with the false packet filtering schemes, and therefore cannot only identify intruders but also filter modified packets immediately after the modification is detected.

2. System Model

The System Model covers the following topics:

2.1 Network Assumptions

We consider a typical deployment of sensor network, as shown in Fig. 1, where a large number of sensor nodes are established in a two dimensional area. Each sensor node generates sensing data periodically and all these nodes co-operate to forward packets that contain the data hop by hop towards a sink, located at some place within the network. We assume that all sensor nodes and the sink are time synchronized [6], which is required by many applications. The sink shares a unique key with all the nodes. The sink is aware of the network topology, which can be achieved by requiring nodes to report their neighboring nodes soon after deployment.

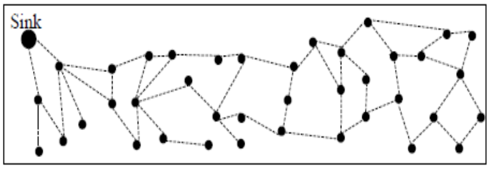

Fig. 1. Network System Model

2.2 Security Assumptions and Attack Model

We assume the network sink is trustworthy and free of compromise, but 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:

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

2) Packet Modification: A compromised node modifies all or some of the packets that it 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

In our scheme for identifying packet droppers and modifiers, a system initialization phase is followed by several equal duration rounds of intruder identification phases.

- ❖ In the initialization phase, sensor nodes form a dynamic routing tree rooted at the sink. The structure of the tree changes dynamically from round to round.
- ❖ In each round, data traffic is transmitted through the routing tree to the sink, and 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 the node categorization algorithm to identify nodes that must be droppers or modifiers and nodes that are suspiciously bad.
- ❖ 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.

3.1 Tree Establishment and Packet Transmission

Dynamic routing tree routed at the sink is first established. The sink knows the tree topology and shares a unique key with each node on the tree. When a node wants to send out a packet, it attaches a sequence number to the packet, encrypts the packet with the key shared with the sink, and then forwards the packet to its parent. 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 keeps tracking 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 rate for every node. Based on the dropping rate and the knowledge of tree 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.

1) System Initialization:

A dynamic routing tree routed at the sink is first established. The sink knows the tree topology structure, it setup a secret pair-wise keys between the sink and every regular sensor node and shares it with each node on the tree. The key facilitate packet forwarding from every sensor node to the sink.

Each sensor node *u* is preloaded the following information:

• ***Ku*** : a secret key exclusively shared between the node and the sink.
• ***Np***: the maximum number of candidate parent nodes that each sensor node records during the tree establishment procedure.
• ***Ns***: the maximum packet sequence number. For each sensor node, its first packet has sequence number 0, the Nth *s* packet is numbered *Ns* − 1, the (*Ns* + 1) *th* packet is numbered 0, and so on and so forth.

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 n has a data item D to report, it composes and sends the following packet to its parent node Pu:

$$\boxed{\langle P_u, \{R_u, u, C_p \text{ MOD } N_s, D, pad_{u,0}\}_{K_u}, pad_{u,1}\rangle,}$$

Where *Cp* MOD *Ns* is the sequence number of the packet. *Ru* is a random number between 0 and *Np* − 1 picked by node *u* during tree establishment. *{X}Y* represents the result of encrypting *X* using key *Y*.

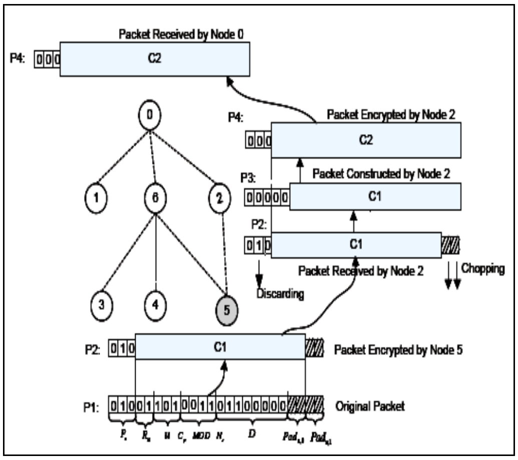
Packet Received by Node 0
P4:
C2
Packet Encrypted by Node 2
P4:
C2
Packet Constructed by Node 2
P3:
C1
P2:
C1
Packet Received by Node 2
Chopping
Discarding
P2:
C1
Packet Encrypted by Node 5
P1:
Original Packet
0
1
6
2
3
4
5
D

Fig. 2. Example of Packet Sending, Forwarding

3.2. Node Categorization Algorithm

In every round, for each sensor node *n*, the sink keeps track of the number of packets sent from *n*, 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 rate for each node *n*. Suppose *n: max* is the most recently seen sequence number, *n: flip* is the number of sequence number flips and *n: rcv* is the number of received packets. Based on the dropping rate 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 μ is first introduced. We assume that if a node's packets are not intentionally dropped by forwarding nodes, the dropping rate of this node should be lower than μ. Note that μ 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 μ, or with "-" otherwise. After all nodes have been marked with "+" or "-".

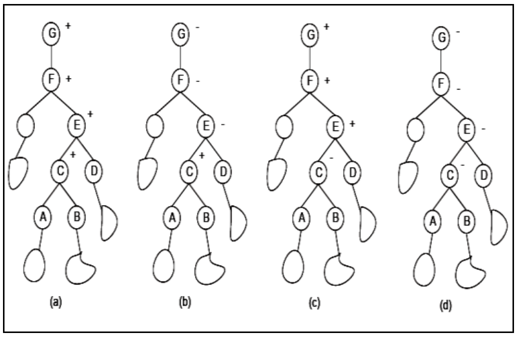

Fig. 3. Node Status Pattern

- + {+} +: The node and its one or more continuous immediate upstream nodes are marked as "+".
- + {-} +: The node is marked as "+", but it's one or more continuous immediate upstream nodes are marked as "-".
- - {+} +: The node is marked as "-", but its one or more continuous immediate upstream nodes are marked as "+".
- - {-} +: The node and its one or more continuous immediate upstream nodes are marked as "-".

For each of the above cases, we can infer whether a node (i) has dropped packets (called *bad for sure*), (ii) is suspected to have dropped packets (called *suspiciously bad*), (iii) has not been found to drop packets (called *temporarily good*), or (iv) must have not dropped packets (called *good for sure*):

Case 1: + {+} +. The node and its continuous immediate upstream nodes 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 temporally good.

Case 2: + {-} +. In the case, all nodes marked as “-” must be bad for sure. To show the correctness of this rule, we prove it by contradiction. Without loss of generality, where node C is marked as “+”, and node E, F, 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 and at least two hops above C. It is impossible for it to differentiate packets from E and C, so it 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 result. 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 “+” must be bad. But it cannot be further inferred whether (i) only the node with “-” is bad, (ii) only the node with “+” is bad, or (iii) both nodes are bad. Therefore, it is concluded that both nodes are *suspiciously bad*.

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 g is the highest-level node that is marked as “-”, and n is its parent.

4. Tree Reshaping

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. At the beginning of each round $i(i = 1, 2, \cdots)$, node u picks the $[h^i(K_u) \text{ MOD } n_{p,u}]th$ parent node as its parent node for this round, where h is a hash function and $h^i(K_u) = h(h^i - 1(K_u))$. 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 known by 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 specified as Algorithm 5.2 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

$$S_i = \{\langle u_j, v_j\rangle | \langle u_j, v_j\rangle \text{ is a suspicious pair and}$$

$$\langle u_j, v_j\rangle = \langle v_j, u_j\rangle\}.$$

5. Ranking Algorithms

The following are the three ranking algorithms discussed below:

5.1 Global Ranking-Based Approach

The GR method is based on the heuristic that, the more times a node are identified as suspiciously bad, the more likely it is a bad node. The node with the highest value is chosen as a most likely bad node and all the pairs that contain this node are removed.

Algorithm 1 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:     $u \leftarrow deque(Q)$
5:     $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
6:     remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$
```

5.2 Stepwise Ranking-Based Approach

It can be anticipated that the GR method will falsely accuse innocent nodes that have frequently been parents or children of bad nodes. 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.

Algorithm 2 The Stepwise Ranking-Based Approach

```
1: $\overline{S} \leftarrow \emptyset$
2: while $\bigcup_{i=1}^{n} S_i \neq \emptyset$ do
3:     $u \leftarrow$ the node has the maximum times of presence in
       $S_1, \cdots, S_n$
4:     $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
5:     remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$
```

5.3 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. 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.

Algorithm 3 The Hybrid 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:     $u \leftarrow deque(Q)$
5:     if there exists $\langle u, * \rangle \in \bigcup_{i=1}^{n} S_i$ then
6:         $\overline{S} \leftarrow \overline{S} \wedge \{u\}$
7:         remove all $\langle u, * \rangle$ from $\bigcup_{i=1}^{n} S_i$
```

5.4 Packet Modifiers

Modified packets can be detected with the afore-described scheme. Modified packets will be detected by sink and it will be dropped and hence packet modifier can be identified as packet dropper. To enable en-route detection of modifications, the afore-described procedures for packet sending and forwarding can be slightly modified as follows. when a node *u* has a data item *D* to report, it can obtain endorsement message authentication codes (MACs) from its neighbors, which are denoted as *MAC*(*D*), following existing en-route filtering schemes such as the statistical en-route filtering scheme (SEF) [7] and the interleaved hop-by-hop authentication scheme [8].

6. Performance Evaluation

Our packet dropper/modifier identification scheme is implemented in the ns-2 simulator (version 2.30) to evaluate the effectiveness and efficiency of the proposed scheme. We measure the performance of our scheme from two aspects: *the detection rate*, defined as the ratio of successfully identified bad nodes, and *the false positive probability*, defined as the ratio of mis accused innocent nodes over all innocent nodes. We run simulations on a 400 × 400*m*2 network with randomly generated network topology. Unless otherwise stated, we set the percentage of bad nodes to 10%, the network size to 100 sensor nodes, the per-node packet reporting interval to 3 seconds, and the length of each round to 300 seconds. Also, when a bad node decides to drop packet in a round, it drops 30% of the packets.

Attack Model

Compromised nodes might treat packets generated by themselves and those by other nodes differently. For their own packets, a compromised node may (1) drop the packets at each round, (2) drop the packets in some randomly rounds, or (3) do not drop all the time. For other nodes' packets that it is supposed to forward, a compromised node may (1) drop the packets in each round, or (2) drop the packets in some randomly rounds. Consider the combination of dropping behaviors in the above two categories, we obtain six attack models in total, namely, attack models 1-1, 1-2, 2-1, 2-2, 3-1 and 3-2, where the first index represents the dropping behavior towards the packets of the bad node itself and the second index represents the dropping behavior towards others' packets.

B. Simulation Results

We first report the simulation results when there is no node collusion and then the results when there is collusion. *1) Evaluation of Ranking Algorithms:* Fig 4 shows the detection rate and false positive probability of our scheme under different attack models. From the figure, we can see that the stepwise ranking (SR) algorithm provides a bit lower detection rate than the other two ranking algorithms in the first several rounds, but after 8 rounds, the three ranking algorithms achieve almost the same detection rate. In terms of false positive probability, the global ranking (GR) algorithm introduces much higher false positive probability than the other two, while the other two algorithms result in almost the same number of false positives. This is because the global ranking (GR) algorithm identifies bad nodes only based on the number that a node is suspected. Therefore, if an innocent node does not have many choices to select its parents in different rounds, or many of its possible parent nodes are actually compromised, the times that this innocent node is suspected will be large. On the contrary, the hybrid ranking (HR) and the stepwise ranking (SR) algorithms do not select a node which is suspected many times when that node has always been suspected together with some already-identified bad nodes, which results in less number of mis-accusations. Consider both the metrics, it is determined that the hybrid ranking is the best ranking algorithm among the three for its high detection rate and low false positive.

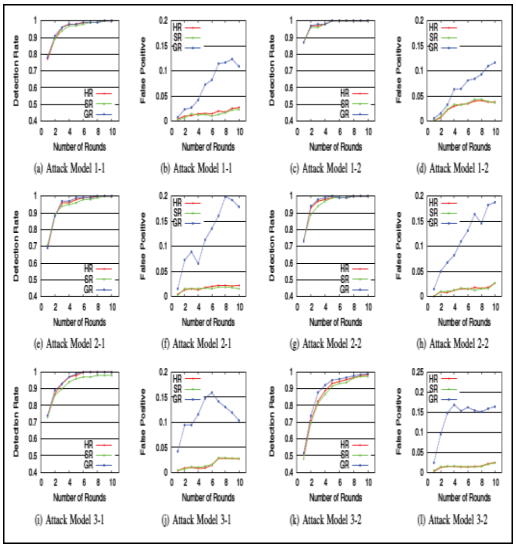

Fig. 4. Comparing Ranking Strategy under Various Attack Models

2) Impact of the Number of Rounds

We study the number of rounds needed to collect information such that a stable and high detection rate as well as a low false positive probability is achieved. We use the hybrid ranking (HR) algorithm here and first plot the detection rate under the six attack models in each round in Fig. 5. From the figure, we can see almost all bad nodes can be identified after 8 rounds regardless of the attack model. Among them, under attack model 1-2, the bad nodes will be detected quickly after 5 rounds. This is because a bad node does not drop packets from its downstream nodes at some intervals, which results in the +*{−}*+ case and the bad nodes can be identified immediately according to our proposed rule. On the contrary, under attack model 3-2, more rounds are needed to achieve a higher detection rate. In this case, bad nodes are sly and do not drop their self generated packets. Consequently, they are only categorized as suspiciously bad nodes. More rounds are needed before they are eventually identified via a ranking algorithm.

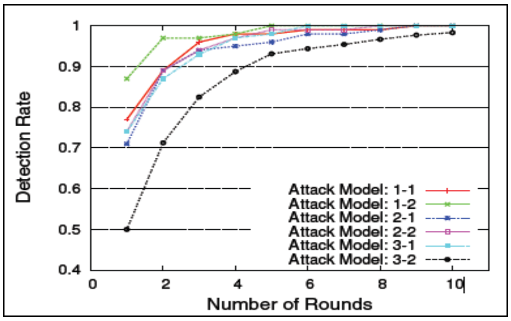

Fig. 5. Number of Rounds vs. Detection Rate

7. Conclusion

In this paper, to address the problem of packet dropping and modification, we proposed a simple yet effective scheme to identify misbehaving forwarders that drop or modify packets. The node categorization and heuristic ranking algorithms are used for this purpose .Extensive analysis and simulations have been conducted and verified the effectiveness of the proposed scheme in various scenarios.

8. References

[1] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-route Filtering of Injected False Data in Sensor Networks," IEEE INFOCOM, March 2004.

[2] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," IEEE Symposium on Security and Privacy, 2004.

[3] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), May 2005.

[4] Z. Yu and Y. Guan, "A Dynamic En-route Scheme for Filtering False Data in Wireless Sensor Networks," IEEE Infocom 2006, April 2006.

[5] F. Ye, H. Yang, and Z. Liu, "Catching Moles in Sensor Networks," IEEE International Conference on Distributed Computing Systems (ICDCS), June 2007.

[6] Q. Li and D. Rus, "*Global clock synchronization in sensor networks," IEEE INFOCOM,* 2004.

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

[8] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical Enroute Filtering of Injected False Data in Sensor Networks," in *IEEE INFOCOM*, 2004.

9. About the Authors

Eerla Raghu is currently pursuing his M.Tech (CSE) in Computer Science & Engineering Department, GIET, Rajahmundry. His area of interests includes Networks

P V G K Jagannadha Raju is currently working as a Professor in Computer Science & Engineering Department, GIET College, Rajahmundry. His research interests include Image Processing, Networks.

Dr. S. Maruthu Perumal is currently working as a Head of the Department for Computer Science & Engineering Department, GIET, Rajahmundry. He is awarded with PhD in related field. His research interests include Image Processing, Data Mining & Warehousing, Networks and Security, Software Engineering.