©2010 International Journal of Computer Applications (0975 – 8887)
Volume 1 – No. 7

Advanced Algorithm for Detection and Prevention of Cooperative Black and Gray Hole Attacks in Mobile Ad Hoc Networks

Shalini Jain
Maharaja Surajmal Institute Of Technology
(Affiliated to Guru Gobind Singh Indraprastha University)
Department Of Information Technology
New Delhi, India

Mohit Jain
Maharaja Surajmal Institute Of Technology
(Affiliated to Guru Gobind Singh Indraprastha University)
Department Of Information Technology
New Delhi, India

Himanshu Kandwal
Maharaja Surajmal Institute Of Technology
(Affiliated to Guru Gobind Singh Indraprastha University)
Department Of Computer Science
New Delhi, India

ABSTRACT
In this paper, we propose an algorithm to detect a chain of cooperative malicious node in ad-hoc network that disrupts transmission of data by feeding wrong routing information along with the detection algorithm. We also propose a mechanism to detect and remove the black and gray hole attacks. Our technique is based on sending data in terms of equal but small sized blocks instead of sending whole of data in one continuous stream. The flow of message is monitored independently at the neighborhood of both source and destination. The result of monitoring is gathered by a backbone network of trusted nodes. Our algorithm takes O(n) time on average to find the chain of malicious nodes which is better than earlier O(n^2) time bound for detecting a single black hole network.

Categories and Subject Descriptors
C.2.1 [**Networks**]: Mobile Ad Hoc Networks – *Attacks, Security*

RES-290

General Terms
Algorithms, Performance, Security.

Keywords
Packet forwarding misbehavior, Mobile ad-hoc network, Gray hole attack, Black hole attack.

1. INTRODUCTION
Mobile ad hoc networks are highly susceptible to routing attacks because of their dynamic topology and lack of any infrastructure. Two of the major routing attacks are black hole and gray hole attacks. In a black hole attack, the malicious node (referred to as black hole) replies to every routing request saying that it has a route to the given destination. . So, unsuspecting nodes start sending data to the destination through the black hole. This way a black hole diverts most of the traffic in the network to itself, and later dumps it. A gray hole attack is a variation of the black hole attack, where the malicious node is not initially malicious, it turns malicious sometime later. This anomalous behavior of malicious nodes prevents a trust based security solution from detecting them.

In this paper we tackled two types of routing attacks namely Gray hole attack and Black hole attack which exhibits packet forwarding misbehavior. In a black hole attack malicious node (called black hole) replies to every route request by falsely claiming that it has a fresh enough route to the destination. In this way all the traffic of the network are redirected to that malicious node which then dumps them all. A gray hole attack is a variation of black hole attack, where an adversary first behave as an honest node during the route discovery process, and then silently drops some or all of the data packets sent to it for further forwarding even when no congestion occurs. Detection of gray hole attack is harder because nodes can drop packets partially not only due to its malicious nature but also due to overload, congestion or selfish nature. A selfish node is unwilling to spend its battery life, CPU cycles or available network bandwidth to forward packets not of direct interest to it, even though it expects others to forward packets on its behalf.

In this paper we present a mechanism capable of detecting and removing the malicious nodes launching these two types of attacks. Our approach consists of an algorithm which works as follows. Instead of sending the total data traffic at a time we divide the total traffic into some small sized blocks. So that malicious nodes can be detected and removed in between the transmission of two such blocks by ensuring an end-to-end checking. Source node sends a prelude message to the destination node before start of the sending any block to alert it about the incoming data block. Flow of the traffic is monitored by the neighbors of the each node in the route. After the end of the transmission destination node sends an acknowledgement via a postlude message containing the no of data packets received by destination node.

Source node uses this information to check whether the data loss during transmission is within the tolerable range, if not then the source node initiate the process of detecting and removing malicious node by aggregating the response from the monitoring nodes and the network.

The rest of this paper is organized as follows. In section 2, we discuss the related work. Assumption proposed in our algorithm is discussed in section 3. We present the methodology and relevant algorithms in section 4. Finally, we discussed the conclusion in section 5.

2. RELATED WORK

Marti et al [3] proposed to trace malicious nodes by using watchdog/pathrater. In watchdog when a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet by promiscuously listening to the next node's transmissions. If the watchdog finds the next node does not forward the packet during a predefined threshold time, the watchdog will accuse the next node as a malicious node to the source node; The proposal has two shortcomings: 1) to monitor the behavior of nodes two or more hops away, one node has to trust the information from other nodes, which introduces the vulnerability that good nodes may be bypassed by malicious accusation; 2) The watchdog cannot differentiate the misbehavior from the ambiguous collisions, receiver collisions, controlled transmission power, collusion, false misbehavior and partial dropping. In pathrater algorithm each node uses the watchdog's monitored results to rate its one-hop neighbors. Further the nodes exchange their ratings, so that the pathrater can rate the paths and choose a path with highest rating for routing. Shortcoming of this algorithm is that the idea of exchanging ratings genuinely opens door for blackmail attack.

SCAN [4] exploits two ideas to protect the mobile ad hoc networks: 1) local collaboration: the neighboring nodes collectively monitor each other and sustain each other; and 2) information cross-validation: each node monitors its neighbors by cross-checking the overheard transmissions, and the monitoring results from different nodes are further cross validated. As a result, the security solution is self-organized, distributed, and fully localized. In SCAN once a malicious node is convicted by its neighbors, the network reacts by depriving its right to access the network by revoking its token. A powerful collusion among the attackers will break SCAN as it violates the assumption of the polynomial secret sharing scheme.

S. Ramaswamy et al presented an algorithm in [5] which claims to prevent the cooperative black hole attacks in ad hoc network. In this algorithm each node maintains an additional Data Routing Information (DRI) table. Whenever a node (say IN) responded to a RREQ it send the id of its next hop neighbor (NHN) and DRI entry for NHN to the source. If IN is not a trustable node for source then source sends a further route request (FRq) to NHN. NHN in turn responds with FRp message including DRI entry for IN, the next hop node of current NHN, and the DRI entry for the current NHN's next hop. If NHN is trusted node then source checks whether IN is a black hole or not using the DRI entry for IN replied by NHN. If NHN is not trustable node then the same cross checking will be continued with the next hop node of NHN. This cross checking loop will be continued until a trusted node is found. Moreover, in the case when the network in not under the attack, the algorithm takes more time to complete. This algorithm is based on a trust relationship between the nodes, and hence it cannot tackle gray hole attacks.

Gonzalez et al [6] presents a methodology, for detecting packet forwarding misbehavior, which is based on the principle of flow conservation in a network. That states that if all neighbors of a node *vj* are queried for i) the amount of packets sent to *vj* to forward and ii) the amount of packets forwarded by *vj* to them, the total amount of packets sent to and received from *vj* must be equal. They assume a threshold value for non malicious packet drop. A node *vi* maintains a table with two metrics *Tij* and *Rij*, which contains an entry for each node *vj* to which vi has respectively transmitted packets to or received packets from. Node vi increments *Tij* on successful transmission of a packet to *vj* for *vj* to forward to another node, and increments *Rij* on successful receipt of a packet forwarded by *vj* that did not originate at *vj*. All nodes in the network continuously monitor their neighbors and update the list of those they have heard recently. This algorithm does not require many nodes to overhear each others' received and transmitted packets, but instead it uses statistics accumulated by each node as it transmits to and receives data from its neighbors. Since there is no collaborative consensus mechanism, such an algorithm may lead to false accusations against correctly behaving nodes.

3. ASSUMPTIONS

The goal of our algorithm is to detect malicious dropping of data packets by an intruder node. In our approach each node in the route is monitored by its neighbors. Neighbors counts the no of data packets forwarded by the node (say dataCount) and on receiving query message from the source which contains no of packets actually sent by the source (say n_i) neighbors of each node check if (dataCount $\neq n_i$) then it replies to source via a result message. Now the problem is that mobile ad hoc networks are resource limited. So nodes may drop packets due to overloaded, lack of CPU cycles, buffer space or bandwidth to forward packets. For these the above straight forward comparison cannot be applied in a rigorous manner. Therefore we assume a threshold probability of packets dropped by a node through no fault of its own.

Let α be the threshold probability of non malicious packet drop by each node then each monitor node check if ($n_i(1-\alpha) \leq$ dataCount) then it is not a suspected node. In our algorithm source node will issue a query message to detect malicious node only when it found that no of packets received by destination (say d_count) is significantly less than the no packets actually sent. If the threshold probability of non malicious packet drop at source node is $\bar{\alpha}$. Then source will start gray/black hole removal process only if (d_count < $n_i(1-\bar{\alpha})$) can be estimated from α as follows. If the non malicious data loss at first node in the route is α then the volume of data actually forwarded by the node to the next node is $n_i(1-\alpha)$. Similarly if at the next node data loss is α then the next node actually forwards $n_i(1-\alpha)(1-\alpha)$ volume of data. So at the destination total data loss due to non malicious packet drop is $(n_i - n_i(1-\alpha)^N$, where N is the total number of nodes in the route. Therefore,

$$\bar{\alpha} = 1-(1-\alpha)^N$$

4. METHODOLOGY

The main idea behind this method is to formulate a list of malicious nodes locally at each node whenever they act as source node. The behavior of each node in the route is monitored by all the neighbors of that node. We employ the idea of dividing the total traffic volume into a set of small data blocks so that the malicious nodes can be captured in between the transmission of two such blocks. We choose a window size w which is used to determine the total no. of such data blocks say k. Before starting the transmission of the data packets from the first block source node (say S) sends a prelude message to the destination node (say D). On receiving prelude message destination will be alert of the incoming data packets. So destination node sets a timer for the end of the incoming transmission and start counting the no. of the data packets received. After the timer expired it sends a postlude message to the source containing the no. of data packets received by it. On the other hand after sending prelude message source node broadcasts a monitor message to all its neighbors instructing them to monitor the action of the next node in the route and start transmitting data. After finishing the transmission source node sets a time out for the receiving of the postlude message. If source node received a postlude message before the timeout expire and the no. of the data packets received by destination is same as the no. of data packets sent by source or the data loss is within tolerable range then source starts the transmission of the next data block. Else it starts detection and removal of the malicious nodes in the route. Here we have assumed a threshold data loss rate α at each node and total data loss rate threshold $\overline{\alpha}$ which can be estimated from α as shown in equation (1) of the previous section. Selection of the value of α plays an important role in the detection power of our proposed algorithm, i.e. the capability of the algorithm to detect misbehaving nodes. The lower the α is the more likely it is that our algorithm detects any malicious behavior. However, it also means that the probability of a false detection will increase with the lower value of α . Also it should be taken into account the total data loss rate should not be higher otherwise source node will not invoke the process of malicious node detection at all. We suggest to assume the maximum value of $\overline{\alpha}$ first, depending on the path length (which is the hop count for the route in AODV routing), then from $\overline{\alpha}$ to estimate the value of α .

On receiving the monitor message neighbors of the source node checks whether it is the neighbor of the next hop node in route or not. If it is neighbor of the hop node in route then it starts monitoring the action of the node. It first initializes a counter to count the no. of the data packets forwarded by the node also infer the id of the next node to which it is forwarding the data. To do so monitor nodes can maintains a copy of the neighbor's routing table and determines the next-hop node to which the neighbor should forward the packet; if the packet is not overheard as being forwarded, it is considered to have been dropped. Also the monitor nodes again broadcast a monitor message to all its neighbors containing the id of the next node to which this node is forwarding the data, instructing them to monitor the action of the next node. This process will continue until the next node is the destination node. If the receiving node of the monitor message is not the neighbor of the next hop node in route it simply forward the message to all its neighbors.

Whenever a source node wish to initiate the gray/black hole detection and removal process it broadcasts a query message to all its neighbors and sets a time out for the receipt of the result message from the monitoring nodes. When the timeout not expired each time a result message or the node is malicious message is received for any node source node will append that node in its findMalicious Table and initialize the voteCount as 1 if it is not already there, otherwise increments its voteCount by 1 and check if voteCount is greater than a predefined thresholdCount or not. If greater, then source node will remove that node from the findMalicious table and enter it into the Black/Gray Hole table. Broadcasts that the node is malicious to the network and modify the malicious status of that route by setting the findHoleStatus as true for that route in its routing table. When the timeout expired source node will start voting for the nodes left in the findMalicious table. It broadcasts vote request message to the network containing the id of each node in the findMalicious table one by one. Sets a timeout for the receipt of the vote reply and on receiving a reply voteCount is incremented by 1. Check if the voteCount is greater than a predefined thresholdCount remove that node from the findMalicious table and enter it into the Black/Gray Hole table. Also broadcasts that the node is malicious to the network and modify the malicious status of that route by setting the findHoleStatus as true for that route in its routing table. Finally the source node checks the findHoleStatus of the route and if it is true then it terminates sending data until it finds a new route to the destination. If it is not true then it retries sending data of the same block.

In the above process source node actually elect the malicious node from the result messages sent by the neighbors based on the reference thresholdCount for both result if the node is voted as malicious by the neighbors or suspected as malicious by neighbors. By doing so we are avoiding the chance of accusing a legitimate node as malicious node by colluding neighbors. Also the vote method from the network enhances the possibility of detecting a really malicious node which is voted as legitimate by the colluding neighbors by not replying to the query message. Our methodology is based on the assumption that a neighborhood of any node in the ad hoc network has more trusted nodes than malicious nodes.

On receiving a query message monitoring nodes checks if the no. of data packets forwarded by the node under monitor is same as the no. of data sent to it or the data loss rate is within the tolerable range (determined by α). If so then it simply broadcast the query message to all its neighbors by replacing the node id to be queried as the next node id to which the monitored node is forwarding the data packets and no. of data packets sent to next node by the data count of the monitoring node. Else monitoring nodes checks if the next id to which the monitored node is forwarding the data packets is NULL then it infers that the monitored node is a black hole node and replies to source as monitored node is malicious. If the next node id is not NULL monitoring nodes replied to the source that monitored node is suspected as malicious node by sending result message to the source. Also it again generate a further query message by replacing the node id to be queried as the next node id to which the monitored node is forwarding the data packets and no. of data packets sent to next node by the data count of the monitoring node and broadcast them to all its neighbors to check if there is any other cooperative malicious nodes exists or not. All the replies to the source are traversed through a reverse path of the query message; therefore, the need for broadcast messages will be minimized.

On receiving a vote request for any node a regular node in the network check their Black/Gray Hole table. If an entry for that node is found it replies to the source node (i.e. the generator of the vote request) via a vote reply message. Here we assume that if the node is not a newly joined node then there is a possibility that node has traversed from the different region of the network. So any other node in the network may have used this node for forwarding traffic and found it as malicious.

On receiving a node is malicious message all regular nodes in the network first check if they already have an entry for the node in their Black/Gray Hole table. If not then they make and entry for that node in their findMalicious table and initialize voteCount as 1. If the node already exists in any of the above tables ignore the message. We are doing so because if we black list the node or increment its voteCount then there is a chance of completely banning a legitimate node from the network by false probing.

Here in our method we propose to modify AODV protocol by introducing three more tables maintained at each node.

First one is DRI (Data Routing Information) table maintained at each node for the purpose of monitoring each of its neighbors. Another table is the findMalicious table which keeps the track of the nodes suspected as malicious with their voteCount. And the Black/Gray hole table which keeps the track of the black listed nodes. We also modified the routing table of the AODV by adding a new field called findHoleStatus which is set as true if a malicious node s found in the route. With the help of the following Figure 1 which shows a current network topology each of the above tables are depicted below.

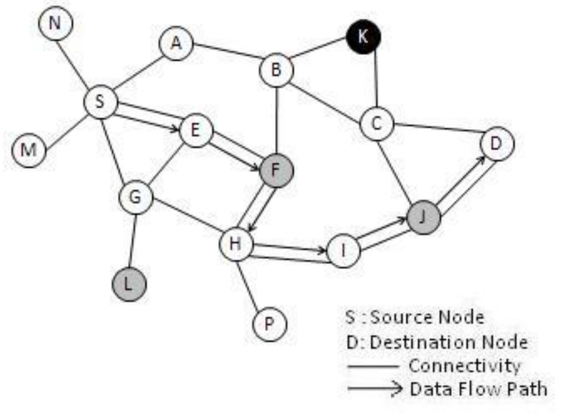

Figure 1: Current Network Topology

Table 1: Data routing table at S

DESTINATION NODE ID	ROUTE	findHoleStatus
D	E,F,H,I,J	False
P	A,B,F,H	False
J	G,H	False

Table 2: List of Neighbors maintained at S

NEIGHBOR NODE ID
E
G
A
M
N

Table 3: Data routing information table maintained at node B for monitoring neighbors

MONITOR D NODE ID	NEXT NODE ID	DATA COUNT
F	H	5
K	NULL	0

Table 4: FindMalicious table maintained at S

NODE ID	VOTE COUNT
F	2
J	1

Table 5: Black/Gray Hole Table Maintained at S

NODE ID
L
K

Pseudo code of our algorithm is as follows.

4.1 Algorithm for Detecting Gray/Black Hole

Step 1: Divide the data packets to be sent in **k** equal parts.
DATA [1,….,K];
Initialize **i = 1;**
Comment: Chose window size **w**, If total no of data packets **n** then **k = ceiling (n/w)**

Step 2: Send ***prelude(S, D, n_i)*** message to the destination node **D**. Where **n_i** is the no of data packets to be sent in current block.

Step 3: Broadcast ***monitor (S, D, NNR)*** message to all its neighbors. Instructing neighbors to monitor next node in the route **(NNR).**

Step 4: Starts transmitting data packets from the block **Data[i]** to **D**.

Step 5: Sets timeout TS for the receipt of the ***postlude(D, S, d_count)*** message containing d_count, no of data packets received by D.

Step 6: If **T_S** not expired and **postlude** message received, if ($n_i (1 - \bar{\alpha}) \le d\_count$)
Increment i by 1 and go to Step 8.
else Start Gray/Black hole removal process.
Comment: Where $\bar{\alpha}$ is a threshold value ranges between 0 and 1 indicates the fraction of total packets gets lost due to error prone wireless channel. If we assume that α is the permissible packet loss in each node in the route then $\bar{\alpha} = 1 - (1 - \alpha)^N$, where N is the total no of nodes in the route (hop count).

Step 7: If **T_S** expired and *postlude* message not received then start Gray/Black hole removal process.

Step 8: Continues from **Step 2** when **i** less than equal to **k**.

Step 9: Terminates S's action.

4.1.1 Action by Destination Node D

Step 1: On receiving ***prelude(S, D, n_i)*** message from S extracts **n_i**
Initialize **d_count = 0.**

Step 2: Sets timeout **T_D** for the receipt of the current data sample and waits for the data packets.

Step 3: When **T_D** not expired and a data packet received Update **d_count += 1**

Step 4: When **T_D** expired send ***postlude(D, S, d_count)*** message to S.

Step 5: Terminates D's action.

4.1.2 Action by neighbors On receiving monitor (S, D, NNR) message

Step 1: On receiving ***monitor*** **(S, D, NNR)** message nodes extracts the id of the next node in the route **NNR**, source node id S and destination node id **D**.

Step 2: If the receiving node is neighbor of NNR then,
  Step 2.1: Turn on Promiscus mode.
  Step 2.2: Initialize **$dataCount_{NNR}$** = 0.
  Step 2.3: Find next node id **N_{next}** to which **NNR** is forwarding the data packets.
  Step 2.4: start counting data packets by incrementing **$dataCount_{NNR}$ += 1.**
  Step 2.5.: If Nnext is not destination node D then
    Step 2.5.1: Broadcast ***monitor (S, D, NNR)*** message to all its neighbors replacing NNR by **N_{next}.**

Step 3: Else Rebroadcast ***monitor (S, D, NNR)*** message to all its neighbors.

Step 4: Terminates its action.

4.2 Gray/Black Hole Removal process

4.2.1 Action by Source Node S

Step 1: Broadcast ***query(S, D, N_{RREP}, n_i)*** message to all its neighbors. Where **N_{RREP}** is the id of the node sending route reply message to **S**.

Step 2: Sets timeout **T_{RES}** for the receipt of the ***result (MN, S, N_{RREP})*** message from the monitoring node **MN**.

Step 3: When TRES not expired and result message received or **"N_{RREP} Malicious"** received then extracts **N_{RREP}**.
  Step 3.1 If NRREP already exists in **FindMalicious** table
    Step 3.1.1: Then increment **voteCount** for **N_{RREP}** by 1.
    Step 3.1.2: If **votecount >= thresholdCount**
      Step 3.1.2.1: Remove **N_{RREP}** from **FindMalicious** table and append **N_{RREP}** in **Gray/BlackHole table**.
      Step 3.1.2.2: Broadcast **"N_{RREP} Malicious"** to the Network.
      Step 3.1.2.3: Set **findHoleStatus = true** in the routing table of S for the route to D.
  Step 3.2: Else
    Step3.2.1: Append **N_{RREP}** in **FindMalicious**.
    Step 3.2.2: Initialize **voteCount = 1.**

Step 4: Initialize **j = 1**.

Step 5: When j <= length of **FindMalicious** table
  Step 5.1: Broadcast ***VREQ(S, N_j)*** to the network requesting other nodes in the network to vote for Nj if it is malicious.
  Step 5.2: Sets timeout TVREP for reply from the network ***VREP(RN, S, N_j)*** where RN is id of any regular node in the network.
  Step 5.3: When **T_{VREP}** not expired and **VREP** message received then
    Step 5.3.1: increment **voteCount** for **N_j** by **1**.
  Step 5.4: If **voteCount >= thresholdCount**
    Step 5.4.1: Remove **N_{RREP}** from **FindMalicious** table and append **N_{RREP}** in **Gray/BlackHole table**.
    Step 5.4.2: Broadcast **"N_{RREP} Malicious"** to the Network.
    Step 5.4.3: Set **findHoleStatus = true** in the routing table of S for the route to **D**.
  Step 5.5: Increment **j** by **1**.

Step 6: If **findHoleStatus** is **True**
  Step 6.1: Terminate sending data. Find new route to D.

Step 7: Resume its normal action.

4.2.2 Action by Neighbors on receiving on receiving query(S, D, N_{RREP}, n_i) message

Step 1: On receiving ***query(S, D, N_{RREP}, n_i)*** message nodes extracts **N_{RREP}** (id of the node sending route reply message to **D**), **S, D** and **n_i**(no of data packets sent to **D**).
Step 2: If the receiving node is neighbor of NRREP then,
  Step 2.1: If $n_i\,(1-\alpha) \leq$ dataCount
    Step 2.1.1: when **N_{next}** is not **D**
      Step 2.1.1.1: Broadcast ***query(S, D, N_{RREP}, n_i)*** message to all its neighbors replacing **N_{RREP}** by **N_{next}.**
  Step 2.2: Else
    Step 2.2.1: If **N_{next}** equals to **NULL** then **N_{next}** itself dropping all the packets

```
        Step 2.2.1.1: Reply "N_RREP Malicious" to S.
    Step 2.2.2: Else
        Step 2.2.2.1: Reply result (MN, S, N_RREP) to S,
            which means N_RREP may be malicious.
        Step 2.2.2.2: Broadcast query(S, D, NRREP, ni)
          message to all its neighbors replacing NRREP by N_next
and n_i by dataCount for N_RREP.

Step 3: If the receiving node is not neighbor of N_RREP then
broadcast query(S, D, N_RREP, n_i) message to all its neighbors.

Step 4: Terminates its action.
```

4.2.3 Action by any regular nodes (RN) on receiving on receiving VREQ(S, N_j) message

```
Step 1: On receiving VREQ(S, N_j) message nodes extracts N_j
Step 2: If N_j exists in Gray/BlackHole table
  Step2.1: Reply VREP(RN, S, N_j) to S.
Step 3: Terminates its action.
```

4.2.4 Action by any regular nodes (RN) on receiving on receiving "N_{RREP} Malicious"

```
Step 1: On receiving "N_RREP Malicious" all regular nodes in the
network check Gray/BlackHole table.

Step 2: If N_RREP not exists in Gray/BlackHole table, then
  Step 2.1: If N_RREP not exists in FindMalicious table.
    Step 2.1.1: Append N_RREP in FindMalicious table.
    Step 2.2.2: Initialize voteCount = 1.

Step 3: Terminates its action.
```

5. CONCLUSION

Black hole attacks are significant attacks, probably that need to be addressed in mobile ad hoc networks. Although, substantial research had been done to combat black hole attacks, here we successfully attempted to detect and prevent the cooperative black and gray hole attacks. . The theoretical results indicate that node working under this algorithm have the potential of the over half of the actual nodes comprised by attack. Finally we also proposed a feasible solution for detection and removal of chain of cooperative black and gray hole attack in AODV protocol with improved complexity O(n) which is half of the previous complexity $O(n^2)$ in previous work. In our solution each node can locally maintain its own table of black listed nodes whenever it tries to send data to any destination node and it can also aware the network about the black listed nodes. This list of malicious nodes can be applied to discover secure paths from source to destination by avoiding multiple black/ gray hole nodes acting in cooperation.

6. REFERENCES

[1] "Security Issues in Mobile Ad Hoc Networks- A Survey" Wenjia Li and Anupam Joshi, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County.

[2] Wireless/Mobile Network Security, Y. Xiao, X. Shen, and D.-Z. Du (Eds.) pp. 170 – 196,2006 Springer, "A Survey on Intrusion Detection in Mobile Ad Hoc Networks" Tiranuch Anantvalee.

[3] S. Marti, T. J. Giuli, K. Lai, and M. Baker, Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. Proceedings of the 6th annual international conference on Mobile Computing and Networking (MOBICOM), Boston, Massachusetts, United States, 2000, 255-265.

[4] H. Yang, J. Shu, X. Meng, and S. Lu, "SCAN: Self-organized network-layer security in mobile ad hoc networks," IEEE Journal on Selected Areas in Communications, vol. 24, issue 2, pp. 261-273, February 2006.

[5] S. Ramaswamy, H. Fu, M. Sreekantaradhya, J. Dixon, and K. Nygard. Prevention of cooperative black hole attack in wireless ad hoc networks. In Proceedings of 2003 International Conference on Wireless Networks (ICWN'03), pages 570–575. Las Vegas, Nevada, USA, 2003.

[6] Oscar F. Gonzalez, Michael Howarth, and George Pavlou, Detection of Packet Forwarding Misbehavior in Mobile Ad-Hoc Networks Center for Communications Systems Research, University of Surrey, Guildford, UK. Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on May 21, 2007.