IACSIT International Journal of Engineering and Technology, Vol.2, No.2, April 2010
ISSN: 1793-8236

Self _USS:A Self Umpiring System for Security in Mobile Ad hoc Network

Ayyaswamy Kathirvel, *Member, IACSIT* and Rengaramanujam Srinivasan

Abstract—**Protecting the network layer from malicious attacks is an important and challenging issue in both wired and wireless networks and the issue becomes even more challenging in the case of mobile ad hoc networks. In this paper we propose a solution of self-umpiring system that provides security for routing and data forwarding operations. In our system each node in the path from source to destination has dual roles to perform: packet forwarding and umpiring. In the umpiring role, each node in the path closely monitors the behavior of its succeeding node and if any misbehavior is noticed immediately flags off the guilty node. The umpiring system proposed is sufficiently general and can be applied to any networking protocol. For demonstration, we have implemented the self-umpiring system by modifying the popular AODV protocol. Simulation studies show vast improvement over the performance of the conventional AODV protocol, with only nominal overheads.**

Index Terms—**MANET, Security, AODV, and Self-Umpiring system.**

I. INTRODUCTION

A mobile ad hoc network (MANET) is a self-created self-organized and self-administering set of nodes connected via wireless links without the aid of any fixed infrastructure or centralized administrator. Each node moves and operates in a distributed peer-to-peer mode, generating independent data and acting as a router to provide multi-hop communication. MANET is ideally suited for potential applications in civil and military environments, such as responses to hurricane, earthquake, tsunami, terrorism and battlefield conditions. Security is an important aspect in such mission critical applications.

In this paper we tackle the problem of securing the network layer operations from malicious nodes. Malicious nodes may disrupt routing algorithms by transmitting a false hop count; they may drop packets, route the packets through unintended routes and so on. Our work rests on the foundations of two excellent systems already proposed: the twin systems of *watchdog* and *pathrater* [1] and SCAN [2]. A brief look at each one of them is in order.

Marti et al. [1] introduced two extensions to the Dynamic Source Routing Protocol DSR [3 - 5] to mitigate the effect of routing misbehaviors – *watchdog* and *pathrater*. The *watchdog* identifies misbehaving nodes while the *pathrater* avoids routing packets through these nodes. When a node forwards packets the node's *watchdog* verifies that the next node in the path also forwards the packet. The *watchdog* does this by listening promiscuously to the next hop transmissions. If the next node doesn't forward the packet then it is misbehaving. The *watchdog* detects the misbehavior and sends a message to the source, notifying it of the misbehaving node.

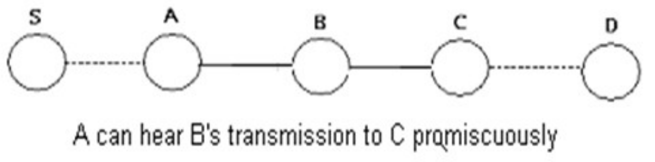


Fig. 1. watchdog mechanism

The *watchdog* method to detect misbehaving nodes is illustrated in Fig.1. Assume that, there exists an active path from source S to destination D through intermediate nodes A, B and C. Node A cannot transmit all the way to node C, but it can overhear what B is transmitting. Therefore A is in a position to tell whether B has correctly forwarded the packet sent by A to C. If encryption is not performed separately for each link – a costly proposition – then A can tell whether B has tampered with the payload or the headers.

The *pathrater*, run by each node in the network combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. For this each node maintains a rating for every other node it knows about; a node always rates itself with a rating of 1.0; neutral rating is 0.5; the *pathrater* increments the rating of nodes on all actively used paths by 0.01 at periodic intervals of 200ms; it decrements a node's rating by 0.05 when a link failure is detected during packet forwarding and the node becomes untraceable and all misbehaving nodes are assigned a special negative value of –100. The *pathrater* doesn't modify the ratings of nodes that are not currently in active use. If there are alternative paths available *pathrater* chooses the path having the highest metric.

Using *watchdog* and *pathrater* it has been shown that throughput is increased by 17% in the presence of 40% malicious nodes during moderate mobility, while increasing the ratio of overhead transmission to data transmission from the standard routing protocol's 9% to 17%.

In SCAN [2] two ideas are exploited to protect the mobile ad hoc network: (i) local collaboration where the neighboring nodes collectively monitor each other and (ii) information cross-validation by which each node monitors neighbors by cross-checking the overheard transmissions.

In SCAN, each node monitors the routing and packet-forwarding behavior of its neighbors and independently detects the existence of malicious nodes in its neighborhood. This is made possible because of wireless nature of the medium and all the involved nodes are within each other's transmission. In order to enable cross-checking they have modified AODV protocol and added a new field *next_ hop* in the routing messages so that

Manuscript received November 11, 2009.

each node can correlate the overheard packets accordingly.

While each node monitors it neighbors independently all the nodes in the neighborhood collaborate to convict a malicious node. An agreement between a minimum of k neighboring nodes is required for convicting a malicious node. Once its neighbors convict a malicious node the network reacts by depriving it of its right to access the network. In SCAN each node must possess a valid token in order to interact with other nodes. They have used asymmetric key cryptography to prevent forgeries of tokens. A group of nodes (minimum-k) can collaboratively sign a token, while no single node can do so. Further each node has to get its token renewed periodically by its neighbors. A node which behaves continuously in a good manner can get its token renewed at less frequent intervals as compared to a fresh entrant node.

Our self-umpiring system has been strongly influenced by the above two schemes. In our system all the active nodes have dual roles just as in *watchdog*; we also exploit promiscuous hearing functionality as done by both SCAN and *watchdog*. We have adopted the token concept from SCAN. However we have dropped partially the *pathrater* functionality. We believe link reliability assessment of *pathrater* may not be correct; a proper reliability metric for path assessment should consider the direction and velocity of movement of active nodes. Thus if a source is situated at south and its destination is situated vertically above at north and all the nodes are moving with a uniform velocity from south to north, good communication link will be maintained; on the other hand if alternate nodes in the path mentioned above, are moving from east to west there is a strong probability of link failure. Having dropped the link reliability factor from the *pathrater*, the only other functionality that remains is avoidance of malicious nodes. We achieve the avoidance of malicious nodes by a system of tokens, which is similar to the ones used in SCAN. Token is a pass or validity certificate enabling a node to participate in the network. It contains two fields: nodeID and status bit; nodeID is considered to be immutable. Initially the status bit of all participating nodes is set as 0 indicating "green flag" with freedom to participate in all network operations. It is assumed that a node cannot change its own status bit. When an umpiring node finds its succeeding node misbehaving it sends a M-Error message to the source and malicious node's status bit is changed using M-Flag message (set to 1 indicating "red flag"). With "red flag" on the culprit node is prevented from participating in the network.

Our objective is designing the security system is to keep the overhead as minimum as possible while optimizing the throughput. We do not use encryption or key algorithms as done by SCAN. We find that token issuing and token renewals and broadcasts to announce convictions create very large communication overheads and also degrade energy performance, which SCAN has completely over looked. There is no token renewal feature in our system. In our system all the nodes are pre issued with green tokens. They continue to enjoy the status until any immediate ancestor node, in its umpiring mode finds its next node misbehaving, sends the M-Error and M-Flag messages and red flag is set.

Just like SCAN in order to facilitate cogent promiscuous hearing we have used "*next_hop*" field with our AODV implementation. Our umpiring system can detect any false reporting of hop count during the route reply process RREP. In *watchdog* detection of malicious action is by a single node while in SCAN it is done by a set of neighbors. In our system the designated predecessor node in its umpiring role carries out both detection and conviction.

The rest of the paper is organized as follows: section II provides an overview of AODV routing protocol; section III formulates the network and USS models. Section IV presents simulation results; section V gives an analysis of simulation results; section VI discusses the related work and section VII gives the conclusions.

II. AODV Routing Protocol

Ad hoc on-demand distance vector (AODV) [6 - 10] routing protocol uses an on-demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. In an on-demand routing protocol, the source node floods the route request (RREQ) packets in the network when a route is not available for the desired destination. It may obtain multiple routes from a single route request. AODV uses destination sequence number (DestSeqNum) to determine an up-to date path to the destination. A node updates its path information if the DestSeqNum of the current packet received is greater that the last DestSeqNum stored in the node. If it possess a route towards the destination with a greater sequence number than the RREQ packet, it unicasts a route reply (RREP) back to the neighbor from which it received the RREQ packet, indicating it has a valid route to the destination. If RREQ is received multiple times, which is indicated by BcastId, SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid route to the destination, or the destination itself are allowed to send route reply (RREP) to the source. Every intermediate node, while forwarding a RREQ enters the previous node's address and its BcastID. A timer is used to delete this entry in case a RREP is not received before the stipulated time.

When a node receives RREP packet information about previous node from which packets was received, it is also stored in order to forward the data packet to the next node as the next hop towards the destination. When a link breaks, which is determined by observing the periodic beacons or through link level acknowledgements, the end nodes - source and destination nodes - are notified. When a source node learns about the path break, it reestablishes a path to the destination, if required by the higher layers. If an intermediate node detects the path break, the node informs the end nodes by sending an unsolicited RREP with hop count set as infinite value.

III. Models and Assumptions

Assumptions made in the design of Self_ Umpiring system are as follows:

1) A wireless ad hoc network where nodes are free to move about or remain at stand still, at their will is assumed.
2) Nodes may fail at any time.
3) There exists a bi-directional communication link between any pair of nodes, which is a requirement for most wireless MAC layer protocols including IEEE 802.11 for reliable transmission.
4) Wireless interfaces support promiscuous mode of operation.

Promiscuous hearing means, over hearing by a node say A, messages not addressed to it, transmitted by a second node B, situated in the communication range of A, to a third node C. Promiscuous hearing is possible because of wireless nature of communication network. As an example if a source node S sends a message "Good Morning" to the destination D, the nodes in the multi hop path hear directly the messages, when the message is forwarded to them, while all the neighbors hear the same message promiscuously. Most of the existing IEEE 802.11 based wireless cards support such promiscuous mode of operations, to improve routing protocol performance.

A. Self-Umpiring System Security Model: Self_USS

In the self-umpiring system each node is issued with a token at the inception. The token consists of two fields: NodeID and status. NodeID is assumed to be unique and deemed to be beyond manipulation; status is a single bit flag. Initially the status bit is preset to zero indicating a green flag. The token with green flag is a permit issued to each node, which confers it the freedom to participate in all network activities. Each node in order to participate in any network activity, say Route Request RREQ, has to announce its token. If status bit is "1" indicating "red flag" protocol does not allow the node to participate in any network activity. The working of the self-umpiring system is explained with reference to Fig. 2.

In the self-umpiring system all the nodes have dual roles – packet forwarding and umpiring. In the forward path during data forwarding, each node monitors the performance of immediate next node. That way, node A can tell correctly whether B is forwarding the packet sent by it, by promiscuously hearing B's transmissions. Similarly during reply process RREP, C can verify whether B is unicasting the route reply RREP and whether the hop count given by B is correct. Thus during forward path A is the umpire for B and C is the umpire for B during reverse path operations.

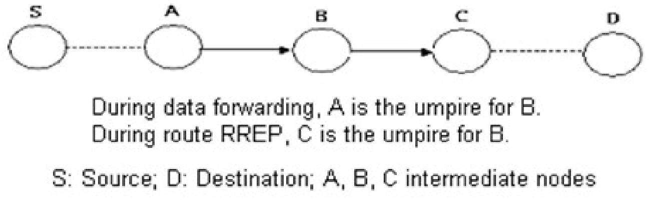

S
A
B
C
D
During data forwarding, A is the umpire for B.
During route RREP, C is the umpire for B.
S: Source; D: Destination; A, B, C intermediate nodes

Fig. 2. self umpiring system model

When a node is found to be misbehaving – say dropping packets, the corresponding umpire immediately sends a M-ERROR message to the source and the status bit of guilty node is set to "1" – red flag using M-Flag message. In order to correctly correlate the overheard messages an additional field next_hop has been introduced in all routing messages as done in SCAN [2]. Though there are several kinds of misbehavior that could be captured by promiscuous hearing we are focusing only on two types of malicious actions: dropping packets and transmitting false hop count.

The token system is similar to the one adopted by SCAN. In SCAN token is issued by a set of neighbors; minimum k neighbors are required to sign tokens; asymmetric cryptography has been adopted to prevent forgery of tokens. Further tokens are to be renewed at periodic intervals. In our system there is no change in the token – it can be used for the full lifetime of the node, if the node continuously behaves correctly. At the instance of the first offence the status of the guilty node is set to 1 preventing its further participation in the network.

We assume that no node can alter its own status bit. Only the designated umpire corresponding to the forward or reverse path under consideration can change the status bit. For example the status bit of B in Fig.2 can be changed only by A in the forward path and only by C in the reverse path. It is also assumed that a node cannot announce wrongly its token particulars – NodeID and status bit.

Our aim is designing the security system is to limit the overhead to as minimum as possible while getting a good improvement in throughput. SCAN system with minimum k neighbors signing, encryption, periodic renewal of tokens is definitely robust, but at a huge cost of control overhead and energy efficiency. There are some other connected issues, which are being discussed in later sections.

IV. Simulation and Results

TABLE I. Parameter Settings

Simulation Time	1500 seconds
Propagation model	Two-ray Ground Reflection
Transmission range	250 m
Bandwidth	2 Mbps
Movement model	Random way point
Maximum speed	0 – 20 m/s
Pause time	0 seconds
Traffic type	CBR (UDP)
Payload size	512 bytes
Number of flows	10 / 20

We use a simulation model based on QualNet 4.5 in our evaluation [11-12]. Our performance evaluations are based on the simulations of 100 wireless mobile nodes that form a wireless ad hoc network over a rectangular (1500 X 600 m) flat space. The MAC layer protocol used in the simulations was the Distributed Coordination Function (DCF) of IEEE 802.11 [13]. The performance setting parameters are given in Table 1.

Before the simulation we randomly selected a certain fraction, ranging from 0 % to 40 % of the network population as malicious nodes. We considered only two attacks – modifying the hop count and dropping packets. Each flow did not change its source and destination for the lifetime of a simulation run.

We have done two sets of studies. Set 1 corresponds to 10 flows with flows between 10 different source-destination pairs. We had kept the simulation time as 1500s, so as to enable us to compare our results with that of SCAN. We found that with set 1 all the network activities ceased by 900s and from 900s to 1500s the network was idle. Therefore in study 2, we introduced another 10 flows, starting at 900s between different sources and destinations (total 20 flows). We call this as set 2.

Our experiments are designed to throw light on four important questions:

1) What is the improvement in throughput with self-umpiring system as compared to plain AODV?
2) What is the probability that malicious nodes may not get caught and convicted? This is the failure to deduct probability, the so-called "False Negative" cases.
3) What is the probability that innocent nodes may get wrongly convicted? These are the "False Positive" cases.
4) What is the extra control overhead because of implementation of Self_USS?

Presently we go on to detail experimental results to elicit answers to each of these questions.

A. Throughput

In the world of MANET, packet delivery ratio has been accepted as a standard measure of throughput. Packet delivery ratio is nothing but a ratio between the numbers of packets received by the destinations to the number of packets sent by the sources. We present in Tables 2 and 3 the packet delivery ratios, for malicious node percentages of 0, 10, 20, 30 and 40, with node mobility varying between 0 to 20 m/s.

TABLE II. PACHET DELIVERY RATIOS FOR SET 1

Mobility (M/s)	Plain AODV					Self USS				
	Percentage of Malicious nodes					Percentage of Malicious nodes				
	0	10	20	30	40	0	10	20	30	40
0	98.28	82.56	76.89	70.44	64.34	98.28	89.48	84.81	76.22	69.28
5	96.28	70.59	57.85	45.18	37.24	96.28	83.32	77.99	73.04	59.56
10	95.11	65.67	51.28	37.89	31.89	95.11	81.41	74.79	70.25	57.69
15	94.12	63.45	47.51	32.55	26.17	94.12	80.77	73.87	69.58	55.02
20	93.73	61.22	45.57	32.07	26.04	93.73	78.89	73.23	68.46	53.18

TABLE III. PACHET DELIVERY RATIOS FOR SET 2

Mobility (M/s)	Plain AODV					Self USS				
	Percentage of Malicious nodes					Percentage of Malicious nodes				
	0	10	20	30	40	0	10	20	30	40
0	99.89	86.16	80.16	73.23	66.27	99.89	96.73	93.45	91.42	81.56
5	99.64	73.89	61.51	48.16	40.16	99.64	96.69	93.18	90.04	78.41
10	96.68	68.16	54.68	40.21	33.73	96.68	95.23	92.45	89.18	78.06
15	96.12	67.02	51.05	35.28	29.59	96.12	94.45	90.12	85.77	76.12
20	95.73	64.51	48.52	34.89	28.18	95.73	92.02	89.72	83.48	75.22

From Tables 2 and 3 the following conclusions can be drawn:

5) In general packet delivery ratio decreases as mobility and percentage of malicious nodes increase.
6) In the case of plain AODV, with 0% malicious nodes, packet delivery ratio drops from 98.28% (99.89% for set 2), when the nodes are stationary to 93.73% (95.73% for set 2), when the nodes are moving at 20 m/s.
7) We observe that the same results are obtained with Self_USS also. With zero percentage malicious nodes, self-umpiring system and plain AODV have almost identical performances.
8) With plain AODV, packet delivery ratio has a steep fall from 98.28 (0% malicious nodes, mobility = 0 m/s) to 26.04 (28.18%) (40% malicious nodes, mobility = 20 m/s). The corresponding values for self-umpiring system are 98.28, 53.18 for set 1 and 99.89, 75.22 for set 2. Thus throughput is increased by 104.22 % with set 1 and by 166.9% in set 2.
9) We find similar increase in throughput at all other combinations of malicious node percentages and mobility values, with self-umpiring system.

From the above results we conclude that self-umpiring system leads to a substantial improvement over plain AODV, from the point of view of throughput. The other question to be answered is how does Self_USS compare with SCAN? We present the details in Fig. 3, where a comparison corresponding to 30% malicious nodes with mobility varying from 0 to 20 m/s is given. The data for SCAN corresponds to Fig. 8 of the paper [2]. We find that set 1 and set 2 results are on either side of SCAN results. We make no claims and offer our comments in the analysis section.

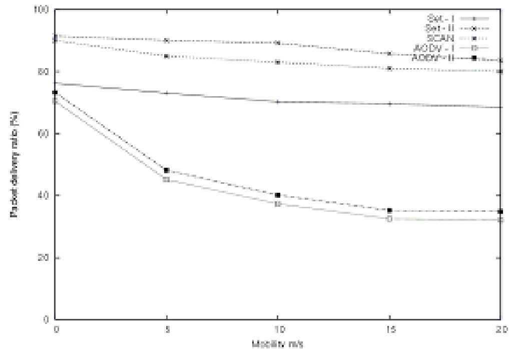


Fig. 3. Comparison of Packet Delivery Ratio (30 % Malicious nodes)

B. Failure to deduct (False Negatives) Probability

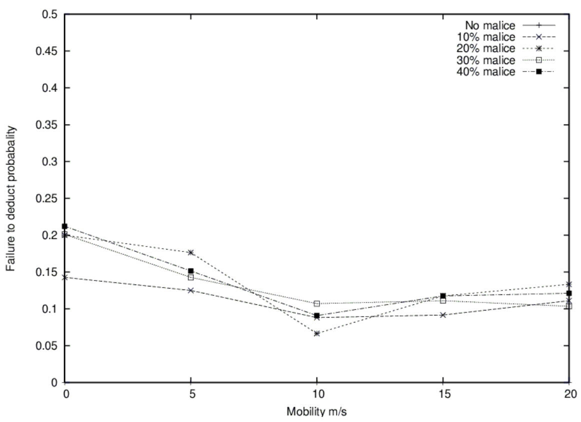


Fig. 4. False Negative Probability verses Mobility for Set – I

Fig. 4. Presents failure to deduct probability as a function of mobility and percentage malicious nodes.

False Negatives Probability can be defined as:

False Negatives Probability = number of malicious nodes left undetected/total number of malicious nodes

The above definition requires some elaboration. We can think of two groups of malicious nodes that are left undetected. In the first group are those nodes, which never played a part in the network operation; they were probably traveling along the boundaries and never had a chance to participate in the network activity.

The second groups of malicious nodes are those that played a role as a forwarding node, but went undetected. Clearly our umpiring system is responsible only for the second group. The first group of nodes is similar to reserve players in the sidelines and clearly any umpire cannot show red flag and march off players in the sidelines. Appropriately we have done the failure to detect probability calculation taking into consideration only those nodes, which took part in the network activity. Other researchers adopt the same approach also. The results are similar that of SCAN [2].

We further offer a comparison of set 1 and set 2 results with 30% malicious nodes in Fig.5. We find that false negative probability has decreased with set 2.

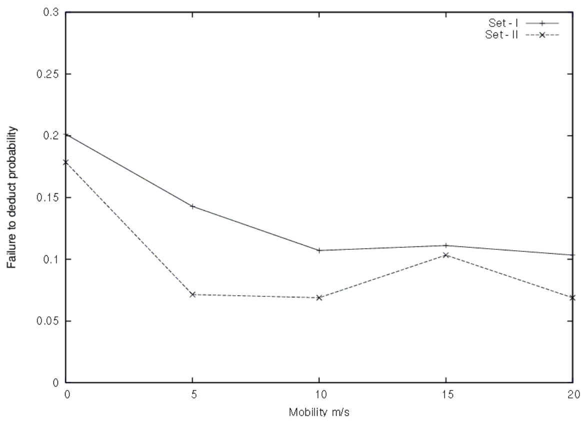


Fig. 5. Comparison of False Negative Probabilities between Set – I and Set – II (30 % Malicious Nodes)

C. False Accusation (False Positives) Probability

Fig. 6. Presents false accusation probability as a function of mobility and percentage malicious nodes for set 1. This is the probability of wrongly booking innocent nodes. We find false positive probability increases with increasing percentage of malicious nodes and increased mobility. The values vary between 0 to 10% and are similar to the patterns obtained for SCAN [2].

We present a comparison of False Positive Probability values between set 1 and set 2 in Fig. 7. It is seen that with set 2 False Positive Probabilities slightly increase.

D. Communication Overhead

Communication overhead can be evaluated based on the number of transmissions of control messages like RREQ, RREP, RERR in the case of plain AODV and in addition M_ERROR, M-Flag messages in the self umpiring system. RREQ are to be decimated to the entire network, where as RREP messages are unicasts.

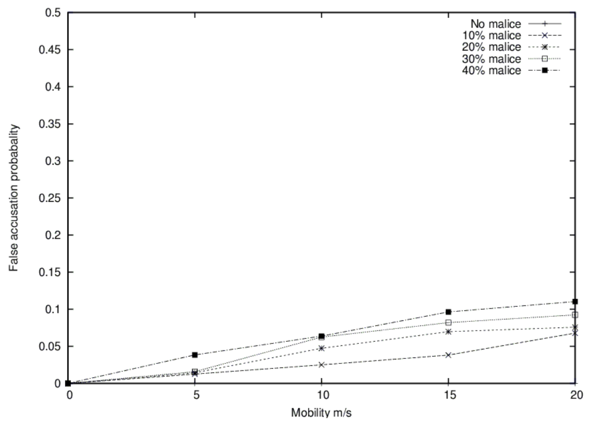


Fig. 6. False Positive Probability verses Mobility for Set – I

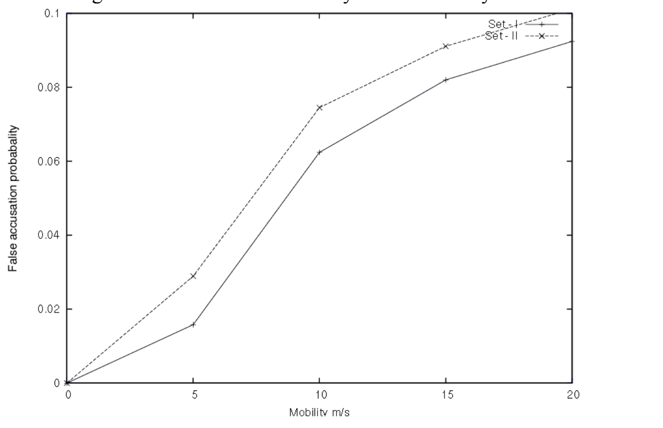


Fig. 7. Comparison of False Positive Probabilities between Set – I and Set – II (30 % Malicious Nodes)

We have taken appropriate weights for each message. For example the count of RREP message from destination to source will be k where k is the hop count. We present the communication overhead details in Table 4 for both plain AODV and for our Self_USS.

TABLE IV. COMMUNICATION OVERHEAD FOR SET 1

Mobility (m/s)	Plain AODV Percentage of Malicious nodes					Self-USS Percentage of Malicious nodes				
	0	10	20	30	40	0	10	20	30	40
0	100	122.4	144.9	156.3	167.7	100	125.5	150.9	167.4	183.1
5	106.3	128.2	150.2	161.4	173.2	106.3	133.5	162.3	177	192
10	110.8	133.9	155.5	166.7	178.9	110.8	142.1	171.1	185.9	202.2
15	116.9	139.4	161.1	172.2	184.9	116.9	152.3	180.4	195	213.6
20	132.6	144.7	166.7	177.8	191	132.6	159	188.3	203.1	222.5

From Table 4 following inferences can be drawn:

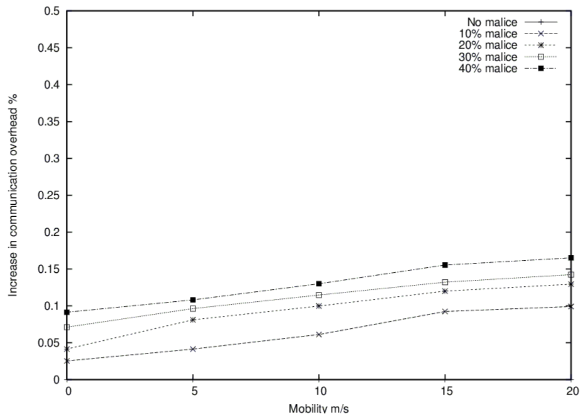


Fig. 8. Increase in Communication Overhead verses Mobility for Set – I

1) The communication overhead increases with increasing percentage of malicious nodes and mobility for both plain AODV and Self_USS.
2) For plain AODV, the increase's from 100% (0% malicious nodes; mobility = 0) to 191% (40% malicious nodes and mobility = 20 m/s). The corresponding variation for Self_USS is from 100 % to 222.5%.
3) Further we find that when there is no malicious nodes (0% malicious nodes) the nodes in their umpiring role have very few message packets to send and the communication overheads for plain AODV and Self_USS are nearly same.
4) In order to correctly appreciate the increase in communication overhead because of umpiring system we have generated Fig.8 from the data presented in Table 4. Fig.8 presents percentage increase in communication overhead with umpiring system corresponding to set 1 as compared to corresponding AODV situations. We find that maximum increase in communication overhead is 16.5 %

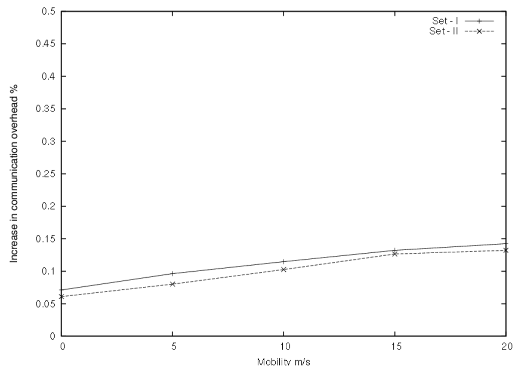


Fig. 9. Comparison of Communication Overheads between Set – I and Set – II (30 % Malicious Nodes)

1. Fig. 9. Provides a comparison of increase in communication overheads between sets 1 & 2 corresponding to 30% malicious nodes. We find that there is a reduction in communication overhead with set 2.

For set 1, we find that the largest increase in communication overhead is 16.5 % corresponding to 40% malicious nodes and mobility 20m/s. The corresponding figure for set 2 is 13.3%.We combine the results of throughput and communication overheads to state that our umpiring system yields 166.9 % increased throughput, with an increase in communication overhead of 13.3 % as compared to conventional AODV (set 2).

V. Analysis of Results

Let us analyze the detection and conviction performances of our umpiring system based up on the parameters – sensitivity, specificity, precision and accuracy [14]. Let TP, FN, FP and TN be defined as follows:

TP: Number of malicious nodes correctly flagged off by USS

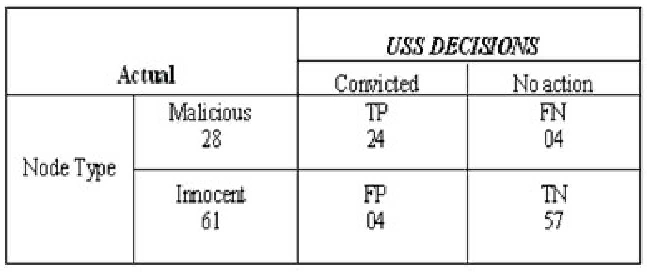
FN: Number of malicious nodes that could not be detected by USS

FP: Number of innocent nodes wrongly booked by USS

TN: Number of active good nodes left undisturbed by USS

The results from our simulation study with 30% malicious nodes and mobility is 5 m/s are as follows:

Total number of nodes pre chosen	= 100
Number of Active Nodes	= 89
Number of malicious nodes	= 30
Number of active malicious nodes	=28
Number of innocent nodes convicted=	04
Number of malicious nodes convicted=	24



Actual		USS DECISIONS	
		Convicted	No action
Node Type	Malicious 28	TP 24	FN 04
	Innocent 61	FP 04	TN 57

TABLE V. Contingency Table(Set 1)

The contingency table is presented as Table 5.

1. Sensitivity=True Positive Rate(TPR)= TP/(TP+FN) = 24/(24+4) = 0.8571

2. Specificity=True Negative Rate (TNR)= TN/(TN+FP) = 57/(57+4) = 0.9344

3. Precision=Recall=TP/(TP+FP)=24(24+4)= 0.8571

4. Accuracy = (TP + TN) / (TP + TN + FP + FN) = (24 + 57) / (24 + 57 + 4 + 4) = 0.9101

Our umpiring system not only detects but also quarantines malicious nodes from further participating in the network activity. Quarantining malicious nodes can be viewed as mine removal process and if we have started with say 40% malicious nodes, after a while (in our simulation study 1 after 900s) a large part of the malicious nodes are removed and any further transmission of messages between sources and destinations occur relatively at malicious nodes free atmosphere. Clearly if the simulation is continued for sufficiently long time, the throughput will incrementally build up until it reaches levels corresponding to almost 0%

malicious nodes.

Throughput: The above reasoning explains why throughput with our set 2 is considerably higher than set 1. SCAN's performance will also be similar and as simulation time is extended, throughput is likely inch towards performance levels at 0% malicious nodes. Therefore in the absence of exact details of simulation parameters, it is not possible to compare the performance of Self_USS with SCAN.

False Negatives Probability: With set 2, a few of the malicious nodes left undetected during the first 10 flows, are rounded up during the subsequent 10 flows, resulting in reduced false negatives probability.

False Positives Probability: Similarly, with in set 2 a few more innocent nodes are booked thus increasing the false positives probability.

Communication Overhead: Since with set 2 during latter portion of simulation time umpiring system operates with reduced percentage of malicious nodes, there is a decrease in communication overhead.

Thus for, we have offered justifications for the simulation results.

There are some other critical questions that we would like to answer before closing this section.

(i). What happens if umpires happen to be malicious?

For our simulation studies, we have defined two types of malicious umpires. Type1 (low level) malicious umpires are those, who when they detect a malicious node simply will ignore them. They are similar to sleeping umpires. Type 2 umpires (high level) are strongly malicious in that when a malicious node is detected they will ignore; further they will go about booking innocent nodes. We had experimented with all 3 situations – malicious umpires being exclusively of Type 1, Type 2 and a mixture of Type 1 and Type2. The results presented in this paper are those corresponding to Type 2 malicious umpires, i.e. strongly malicious umpires.

(ii). If that is so, how is that such a high output could be obtained?

We have made the assumption, that source and destination nodes are non-malicious. If source or destination is malicious in any flow, then the throughput corresponding to that flow will be zero. In the worst case if in all the n flows, either the source or destination happens to be malicious then the throughput of the entire simulation will be zero.

If source finds its next node malicious, it preempts it before malicious node can cause harm to the succeeding node. This works equally well in the reverse path too, with the destination being non malicious. We also deliberately made the source as malicious nodes and made our study. We found that, since we have implemented malicious umpires of Type 2, the malicious source node succeeds in killing 2 innocent nodes before regular AODV protocol packs it off.

(iii). Why are innocent nodes booked?

As discussed if source or destination happens to be malicious, innocent nodes will be shown red flag. Further if a genuine node is going out of communication range, its umpire will view the event as dropping packets. We propose to make modifications in our future designs so that, this type of wrong booking could be avoided.

(iv). Why are all malicious nodes not booked?

Some of the malicious nodes are always traveling in the periphery of the network. Clearly umpires can show red flag only to players who are within the playing area; mischievous sideliners as long as they continue to be in sidelines only, cannot be booked.

(v). What happens if malicious nodes behaves wrongly during RREQ stage?

The present design does not cater to this situation. We are working at alternative designs to overcome the above defect, without much increasing the overhead, which is our avowed design goal.

VI. Related Works

The Key Distribution Center (KDC) architecture is the main stream in wired network because KDC has so many merits: efficient key management, including key generation, storage, distribution and updating. The lack of Trusted Third Party (TTPs) key management scheme is a big problem in ad hoc network [15 - 32].

Kong et al. [16] describe a solution that supports ubiquitous services to mobile hosts. In their design they distribute the certification authority functions through a threshold secret sharing mechanism, in which each entity holds a secret share and multiple entities in a local neighborhood jointly provide complete services. Thus no single entity in the network knows or holds the complete system secret (e.g. - a certification authority's signing key). Instead, each entity holds a secret share of the certification authority's secret key. Multiple entities, say k in one hop network locality jointly provide complete security services, as if a single omni present certification authority provided them.

Yong et al. [18] propose a novel cryptography for ad hoc network security, where they present a new digital signature algorithm for identity authentication and key agreement scheme. Their scheme has no central administrator. They have shown that their scheme can withstand man-in-middle and Byzantine mode conspiracy attacks.

Hubaux et al. [22] make a survey of threats and possible solutions for one security of ad hoc network. They extend the idea of public key infrastructure. Their system is similar to Pretty Good Policy (PGP) in the sense public key certificates are issued by the users. However they do not rely on certificate directories for the distribution of certificates. They present two algorithms in this connection.

All the above schemes only try to protect the system from the attacker, but not bother about quarantining attackers. The twin systems of *watchdog* and *pathrater* [1] not only detect the mischievous nodes but also prevent their further participation in the network. SCAN [2] also has similar action, but is more comprehensive, in the sense not only packet dropping but also other misbehaviors like giving wrong hop count are covered. Our self-USS is an extension of the above two works.

Routeguard [32] is similar to *pathrater and* is run by each node. Routeguard introduces more detailed and natural classification system that rates each node into one of the five classes: *fresh, member, unstable, suspect* or *malicious.* Accordingly each node is treated differently.

VII. Conclusions

A self-umpiring system for security for mobile ad hoc network has been proposed. Simulation studies show that the proposed system increases throughput by 166.9 % with an increase in communication overhead of 13.3% as compared to plain AODV, when 40 % of the nodes are malicious and are roaming with a mobility of 20 m/s. We envisage that our system can profitably be used in civilian situations where invariably nodes are lean and energy starved. Research work is in progress for the development of an independent system of umpires.

VIII. Acknowledgements

We express our thanks to Dr. P. Kanniappan, the Vice Chancellor, Prof. V. M. Periasamy, the Register and Prof. K.M.Mehata, the Head, Department of CSE & Dean, School of Computer and Information Science B.S.A.Crescent Engineering College Chennai, Tamilnadu, India for the encouraging environment provided.

References

[1] Sergio Marti, T.J. Giuli, Kevin Lai and Mary Baker, “Mitigating routing misbehavior in mobile ad hoc networks”, in proc. ACM MobiCom, 2000, pp- 255-265.

[2] Hao Yang, James Shu, Xiaoqiao Meng and Songwu Lu, “SCAN: Self-Organized Network-Layer Security in Mobile ad hoc networks”, IEEE Journals on selected areas in communications, vol. 24, No. 2, February 2006.

[3] D. Johnson, D. Maltz, and J. Jetcheva, DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Network, Ad Hoc Networking. Reading, MA: Addison-Wesley, 2001, ch. 5.

[4] D.B. Johnson and D.A. Maltz, “Dynamic source routing in ad hoc wireless networks”, in Mobile Computing, volume 353, Kluwer Academic Publishers.

[5] D.B. Johnson, D.A. Maltz, and Y.C. Hu, “ The dynamic source routing protocol for mobile ad hoc networks (dsr). Internet Draft, draft-ietf-manet-dsr-09.txt, April, 2003.

[6] C. Perkins, and E. Royer, “Ad hoc on-demand distance vector routing”, in Proc. IEEE WMCSA, 1999, pp. 90-100.

[7] C. Perkins, E. Royer and S. Das, “Ad hoc on demand distance vector (AODV) routing”, Internet Draft, draft-ietf-manet-aodv-10.txt, 2002.

[8] S. Das, C. Perkins, and E. Royer, “Performance comparison of two on-demand routing protocols for ad hoc networks”, in Proc. IEE INFOCOM, 2003, pp. 3-12.

[9] C. Sivaram murthy and B.S. Manoj, “ Ad hoc wireless networks architectures and protocols”, Pearson Education, First edition, 2007.

[10] Jochen Schiller, “Mobile Communications”, Pearson Education, Second edition, 2007.

[11] Scalable Networks Technologies: QualNet simulator 4.5 http://www.scalable-networks.com/

[12] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla, “Glomosim : A scalable network simulation environment. Technical Report 990027, 1999.

[13] IEEE 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, August, 1999.

[14] Pang – Ning Tan, Michael Steinbach, Vipin Kumar, “Introduction to Data Mining”, Pearson Education, 2007.

[15] Marianne A. Azer, Sherif M. El-Kassas, and Magdy S. El-Soudani, “Certification and revocation schemes in ad hoc networks survey and challenges, in proc. IEEE ICSNC 2007.

[16] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, “Providing robust and ubiquitous security support for MANET”, in Proc. IEEE ICNP, 2001, pp. 251-260.

[17] Lei Feng-Yu, Cui Guo-Hua, and Liao Xiao-Ding, “Ad hoc Networks security mechanism based on CPK”, in proc. IEEE ICCISW, 2007, pp. 522 – 525.

[18] Pi Jian Yong, Liu Xin Song, Wu Ai, Liu Dan, “A Novel Cryptography for Ad Hoc Network Security”, in Proc. IEEE 2006, pp. 1448 -1451.

[19] Michael Hauspie, and Isabelle Simplot-Ryl, “Enhancing nodes cooperation in ad hoc networks”, in proc. IEEE 2007, pp. 130 – 137.

[20] S. Capkun, L. Buttyan and J. Hubaux, “Self-organized public-key management for mobile ad hoc networks”, IEEE Trans. Mobile Computing, vol. 2, No. 1, pp. 52-64, January, 2003.

[21] Pi Jian Yong, Liu Xin Song, Wu Ai, Liu Dan, “A Novel Cryptography for Ad Hoc Network Security”, in Proc. IEEE 2006, pp. 1448 -1451.

[22] J. Hubaux, L. Buttyan, and S. Capkun, “The quest for security in Mobile ad hoc networks”, in Proc. ACM MobiHoc, 2001, pp. 146-155.

[23] William Stallings, “Cryptography and network Security principles and Practices”, Pearson Education, First edition, 2007.

[24] Y. Zhang and W. Lee, “Intrusion detection in wireless ad hoc networks”, in Proc. ACM MobiCom, 2000, pp. 275-283.

[25] S. Capkun, J.Hubaux, and L. Buttyan, “Mobility helps security in ad hoc networks”, in Proc. ACM MobiCom, 2003, pp 46-56.

[26] Y. Hu, D. Johnson, and A. Perrig, “SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks”, in Proc. IEEE WMCSA, June 2002, pp. 3-13.

[27] Y. Hu, A. Perrig, and D. Johnson, “Ariadne: A Secure on-demand routing for ad hoc networks”, in Proc. ACM MobiCom, 2002, pp. 12-23.

[28] P. Papadimitratos and Z. Haas, “Secure routing for mobile ad hoc networks”, in Proc. CNDS, 2002, pp. 193-204.

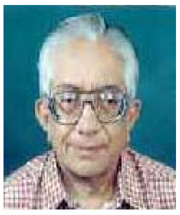
[29] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Royer, “A secure protocol for ad hoc networks,” in Proc. IEEEICNP, 2002, pp. 78-89.

[30] M. Zapata and N. Asokan, “Securing ad hoc routing protocols”, in Proc. ACM Wise, 2002, pp.1-10.

[31] Azeddine Attir, Farid Nait Abdesselem, Brahim Bensaou, and Jalel Ben-Othman, “Logical Wormhole Prevention in Optimized Link State Routing Protocol”, in proc IEEE GLOBECOM 2007, pp. 1011 – 1016.

[32] Nidal Nasser and Yunfeng Chen, “Enhanced Intrusion Detection System for discovering malicious nodes in mobile ad hoc networks”, in proc. IEEE ICC, 2007, pp. 1154- 1159.

A.Kathirvel born in 1976 in Erode, Tamilnadu, India, received his B.E. degree from the University of Madras, Chennai, in 1998 and M.E. degree from the same University in 2002. He is currently with B.S.A. Crescent Engineering College in the Department of computer science and Engineering and pursing Ph.D. degree with the Anna University, Chennai, India. He is a member of the ISTE. His research interests are protocol development for wireless ad hoc networks, security in ad hoc networks.



Rengaramanujam Srinivasan born in 1940 in Alwartirunagari, Tamilnadu, India, received B.E. degree from the University of Madras, Chennai, India in 1962, M.E. degree from the Indian Institute of Science, Bangalore, India in 1964 and Ph.D. degree from the Indian Institute of Technology, Kharagpur, India in 1971. He is a member of the ISTE and a Fellow of Institution of Engineers, India. He has over 40 years of experience in teaching and research. He is presently working as a Professor of Computer Science and Engineering at BSA Crescent Engineering College, Chennai, India and is supervising doctoral projects in the areas of data mining, wireless networks, Grid Computing, Information Retrieval and Software Engineering.