INTRODUCTION

What is AD HOC Network?

In computer networking, an ad hoc network refers to a network connections established for a single session and does not require a router or a wireless base station. It is defined as the category of wireless network that utilize multi-hop radio relaying and capable of operating without the support of any fixed infrastructure. Hence, it is also called infrastructure-less network.Wireless mobile *ad hoc* networks are self-configuring, dynamic networks in which nodes are free to move. Wireless networks lack the complexities of infrastructure setup and administration, enabling devices to create and join networks.

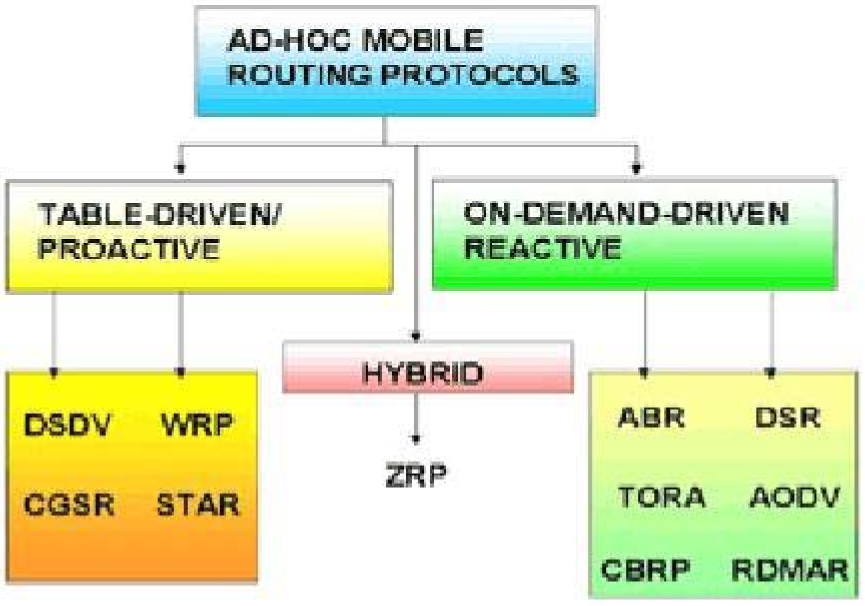
Routing in Ad Hoc Networks:-

An ad-hoc network is a self-configuring network of wireless links connecting mobile nodes. These nodes may be routers and/or hosts. The mobile nodes communicate directly with each other and without the aid of access points, and therefore have no fixed infrastructure. They form an arbitrary topology, where the routers are free to move randomly and arrange themselves as required.
Each node or mobile device is equipped with a transmitter and receiver. They are said to be purpose-specific, autonomous and dynamic. This compares greatly with fixed wireless networks, as there is no master slave relationship that exists in a mobile ad-hoc network. Nodes rely on each other to established communication, thus each node acts as a router. Therefore, in a mobile ad-hoc network, a packet can travel from a source to a destination either directly, or through some set of intermediate packet forwarding nodes.
In a wireless world, dominated by Wi-Fi, architectures which mix mesh networking and ad-hoc connections are the beginning of a technology revolution based on their simplicity.

Routing protocols in Ad Hoc network:-

Routing protocols between any pair of nodes within an ad hoc network can be difficult because the nodes can move randomly and can also join or leave the network. This means that an optimal route at a certain time may not work seconds later. Discussed below are three categories that existing ad-hoc network routing protocols fall into:

1. Table Driven Protocols
2. On Demand Protocols
3. Hybrid Protocols



Description of the Ad-hoc Routing Protocols:-

Ad Hoc on-Demand Distance Vector Routing (AODV)

In AODV, the network is silent until a connection is needed. At that point the network node that needs a connection broadcasts a request for connection. Other AODV nodes forward this message, and record the node that they heard it from, creating an explosion of temporary routes back to the needy node. When a node receives such a message and already has a route to the desired node, it sends a message backwards through a temporary route to the requesting node. The needy node then begins using the route that has the least number of hops through other nodes. Unused entries in the routine tables are recycled after a time.

AODV adopts a very different mechanism to maintain routing information. It uses traditional routing tables, one entry per destination. This is in contrast to DSR, which can maintain multiple route cache entries for each destination. Without source routing, AODV relies on routing table entries to propagate an RREP back to the source and, subsequently, to route data packets to the destination. AODV uses sequence numbers maintained at each destination to determine freshness of routing information and to prevent routing loops. All routing packets carry these sequence numbers. An important feature of AODV is the maintenance of timer-based states in each node, regarding utilization of individual routing table entries.

Advantages & Disadvantages:-

The main advantage of this protocol is having routes established on demand and that destination sequence numbers are applied to find the latest route to the destination. The connection setup delay is lower. One disadvantage of this protocol that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries. Also, multiple RouteReply packets in response to a single RouteRequest packet can lead to heavy control overhead.

Why we use AODV Routing Protocol:-

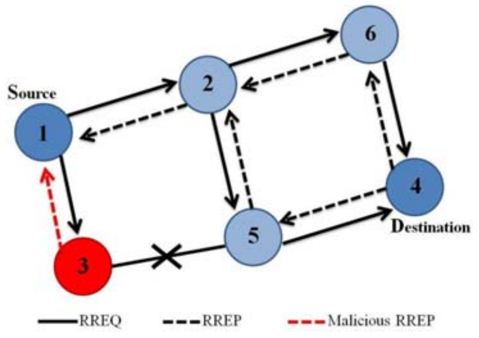
AODV (Ad hoc on Demand distance vector routing) algorithm will be used primarily for developing detection as well as prevention algorithms for Wormholes encountered in the MANET. The AODV (Ad-Hoc On-Demand Distance Vector) routing protocol is a reactive routing protocol that uses some characteristics of proactive routing protocols. Routes are established on-demand, as they are needed. However, once established a route is maintained as long as it is needed. Reactive (or on-demand) routing protocols find a path between the source and the destination only when the path is needed (i.e., if there are data to be exchanged between the source and the destination). An advantage of this approach is that the routing overhead is greatly reduced.

Attacks on Ad Hoc Networks:-

In addition to often being wireless the structure of an Ad Hoc network, or lack thereof, leads to some special kinds of attacks. Such as –

Black Hole

A black hole problem means that one malicious node utilizes the routing protocol to claim itself of being the shortest path to the destination node, but drops the routing packets but does not forward packets to its neighbors. A single black hole attack is easily happened in the mobile ad hoc networks. An example is shown that node 1 stands for the source node and node 4 represents the destination node. Node 3 is a misbehavior node who replies the RREQ packet sent from source node, and makes a false response that it has the quickest route to the destination node. Therefore node 1 erroneously judges the route discovery process with completion, and starts to send data packets to node 3. As what mentioned above, a malicious node probably drops or consumes the packets. This suspicious node can be regarded as a black hole problem in MANETs. As a result, node 3 is able to misroute the packets easily, and the network operation is suffered from this problem. The most critical influence is that the PDR diminished severely.



Grey Hole

A special case of the black hole attack is an grey hole attack .In this attack the adversary selectively drops some kinds of packets but not other. For example the attacker might forward routing packets but not data packets.

Wormhole

Wormhole nodes fake a route that is shorter than the original one within the network; this can confuse routing mechanisms which rely on the knowledge about distance between node. It has one or more malicious nodes and a tunnel between them. The attacking node captures the packets from one location and transmits them to other distant located node which distributes them locally. A wormhole attack can easily be launched by the attacker without knowledge of the network or compromising any legitimate having nodes or cryptographic mechanisms.

Rushing Attack

Many reactive routing protocols keep a sequence number for duplication suppression at every node. An attacker can distribute a large number of route requests with increasing sequence numbers forged to appear to be from other nodes.
This way when the actual route request is sent out many nodes suppress it as a duplicate and thereby disrupt the actual route discovery.

Sybil Attack

Sybil attack is an attack which uses several identities at a time and increases lot of misjudgments among the nodes of a network or it may use identity of other legitimate nodes present in the network and creates false expression of that node in the network. Like this, it disturbs the communication among the nodes of the network.

OBJECTIVE OF THE PROJECT

Our objective is to detect the wormhole nodes in the networks and prevent the wormhole effect in the network by using ids and compare the performances with and without wormhole attack using X graph.

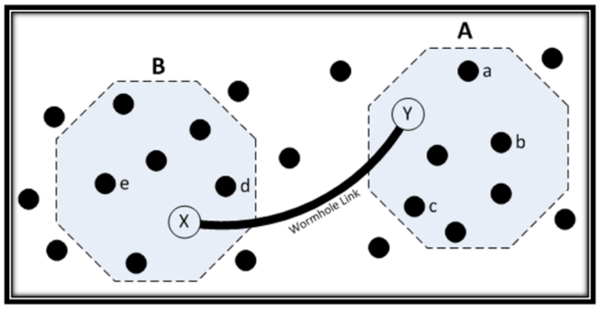
MODULES OF PROJECT:

- Creation of nodes and transmission of packets through AODV protocol.
- Cloning of AODV protocol to inject the wormhole behavior.
- Cloning of AODV protocol to prevent wormhole attack.
- Calculation of performance metrics for AODV, wormhole AODV and ids AODV.
- Comparing the performance metrics for AODV, wormhole AODV and ids AODV.

DETAILS ABOUT ATTACK & IT'S ALGORITHM:- WORM HOLE ATTACK

In the wormhole attack, there are two colluding nodes that are far apart are connected by a tunnel giving an illusion that they are neighbors. Each of these nodes receive route request and topology control messages from the network and send it to the other colluding node via tunnel which will then replay it into the network from there. By using this additional tunnel, these nodes are able to advertise that they have the shortest path through them. Once this link is established, the attackers may choose each other as multipoint relays (MPRs), which then lead to an exchange of some topology control (TC) messages and data packets through the wormhole tunnel. Since these MPRs forward flawed topology information, it results in spreading of incorrect topology information throughout the network. On receiving this false information, other nodes may send their messages through them for fast delivery. Thus, it prevents honest intermediate nodes from establishing links between the source and the destination. Sometimes, due to this, even a wormhole attacker may fall victim to its own success. In, a particular type of wormhole attack known as "in-band wormhole attack" is identified. A game theoretic approach has been followed to detect intrusion in the network. Presence of a central authority is assumed for monitoring the network. This is a limitation in wireless scenario such as military or emergency rescue. No experimental result is reported in. In the wormhole attacks are classified as

1) In-band wormhole attack, which require a covert overlay over the existing wireless medium and
2) Out-of-band wormhole attack, which require a hardware channel to connect two colluding nodes.



Algorithm For WormHole Attack:-

New Fresh Algorithm

Step 1: Whenever a source node needs a route to destination the protocol starts route discovery. During route discovery, source node broadcast RREQ packets through neighboring nodes. RREQ packet contains destination address and sequence number along with source address. Sequence number provides the freshness of route.

Step 2: Once an RREQ packet is received by an intermediate node and verifies destination address. If the destination address not matches with the RREQ packet then forwards it to its next hop. This process is repeated until it reaches the final destination.

Step 3: Route path nodes are saved in routing table.

Step 4: When source node starts sending packets, it sends to next node and that node sends to next until it reaches destination. The traversed path nodes are checked with the path nodes in routing table.

Step 5: If the traversed path nodes are not in the routing table, wormhole is detected and it is out band wormhole.

Step 6: While sending packets to next neighbor node, PDR is calculated for each node. The ratio of sent packets to received packets is calculated for each node.

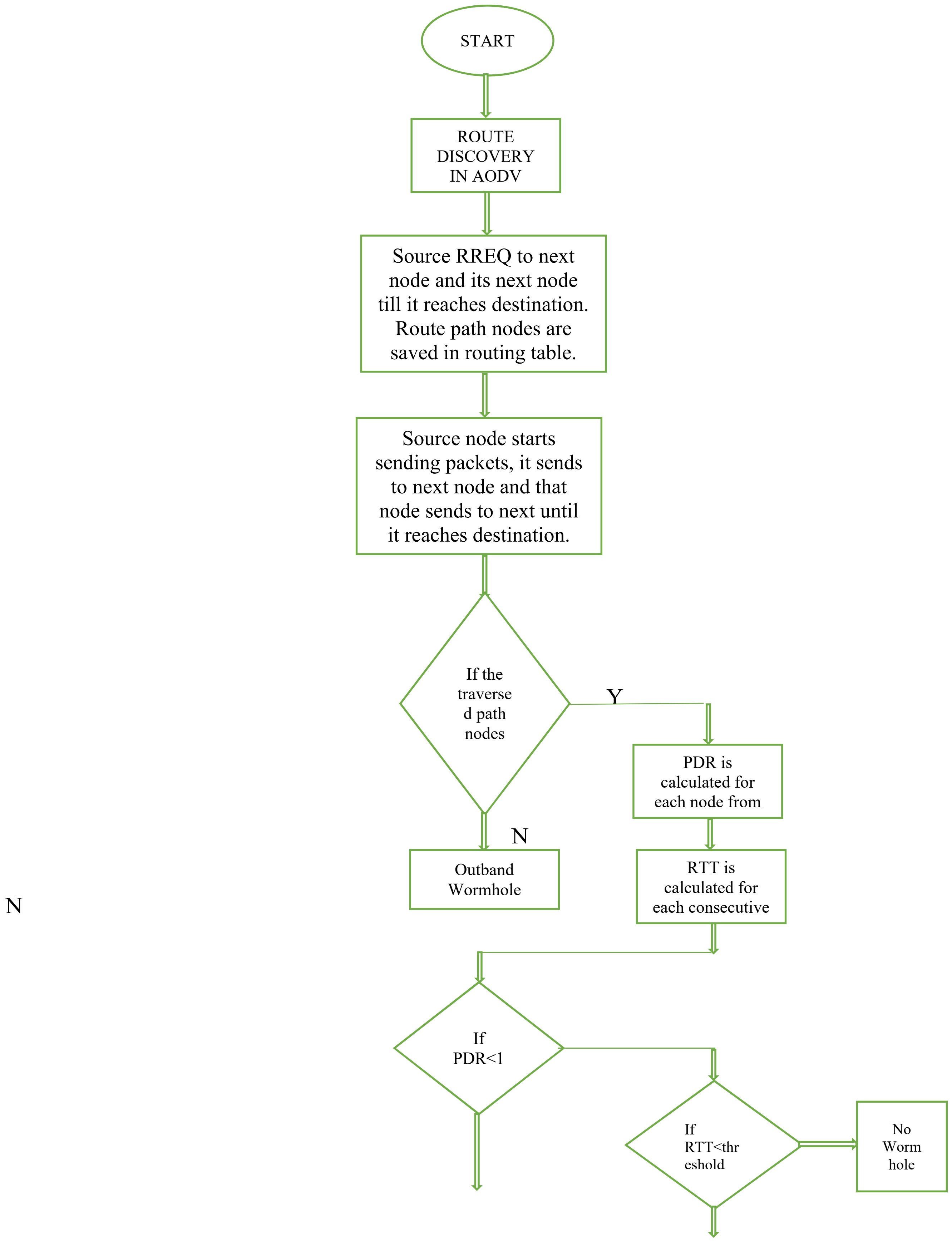
Step 7: Hello packets are also sending to each node along with packets until it reaches destination. Roundtrip time is calculated for each consecutive node. If the roundtrip time is less than threshold, that link is high speed link and the two nodes are malicious and detected as wormhole. And also if the PDR is less than 1, that node is wormhole node. The wormhole detected is active wormhole as it affects the packets.

Step 8: If PDR less than 1 and RTT is not less than threshold means the loss may be due to traffic.

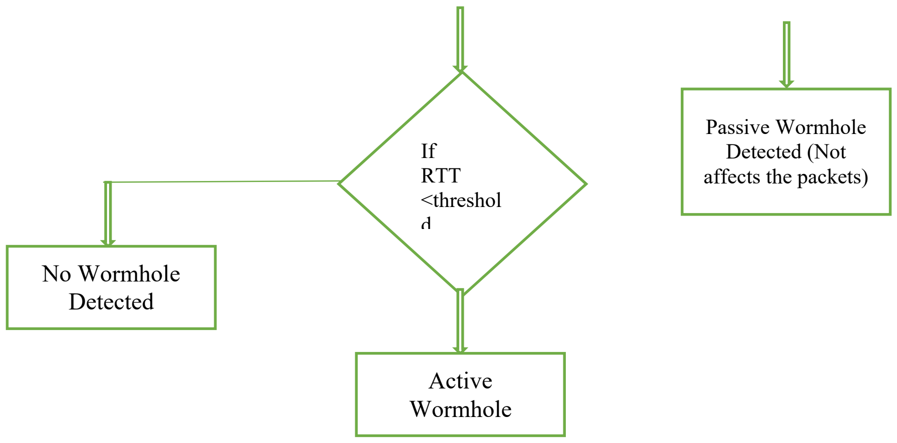
Step 9: If PDR not less than 1, check for RTT less than threshold or not. If it is less passive wormhole is detected as the packets are not affected. If it is not less than threshold, there is no wormhole.

Step 10: Wormhole nodes are announced to all other nodes. All nodes remove wormhole node id from its neighbor table and Routing Table. If any forwarding node receives the wormhole announcement node, it will send RERR message to source. It will reinitiate route discovery process, and find the new path to the destination without wormhole node.

Flow chart of new fresh algorithm:



START
ROUTE DISCOVERY IN AODV
Source RREQ to next node and its next node till it reaches destination. Route path nodes are saved in routing table.
Source node starts sending packets, it sends to next node and that node sends to next until it reaches destination.
If the traverse d path nodes
Y
N
PDR is calculated for each node from
Outband Wormhole
RTT is calculated for each consecutive
N
If PDR<1
If RTT<thr eshold
No Worm hole



Code for Worm Hole Attack:

```
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     25                         ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      1186                       ;# X dimension of topography
set val(y)      584                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end
set val(t1)     0.0                        ;
set val(t2)     0.0                        ;
```

Initialization:

```
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
```

```
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
```

Mobile node parameter setup:

```
$ns node-config -adhocRouting  $val(rp) \
            -llType        $val(ll) \
            -macType       $val(mac) \
            -ifqType       $val(ifq) \
            -ifqLen        $val(ifqlen) \
            -antType       $val(ant) \
            -propType      $val(prop) \
            -phyType       $val(netif) \
            -channel       $chan \
            -topoInstance  $topo \
            -agentTrace    ON \
            -routerTrace   ON \
            -macTrace      ON \
            -movementTrace ON
```

Nodes Definition:

```
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$n0 color Red
$ns initial_node_pos $n0 30
```

```
$n0 start
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 30
set n2 [$ns node]
$n2 set X_ 791
$n2 set Y_ 406
$n2 set Z_ 0.0
$ns initial_node_pos $n2 30
set n3 [$ns node]
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 30
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 30
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 30
set n6 [$ns node]
$n6 set X_ 650
$n6 set Y_ 40.0
$n6 set Z_ 0.0
$ns initial_node_pos $n6 30
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 30
set n8 [$ns node]
$n8 set X_ 761
```

```
$n8 set Y_ 234
$n8 set Z_ 0.0
$ns initial_node_pos $n8 30
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 30
set n10 [$ns node]
$n10 set X_ 714
$n10 set Y_ 121
$n10 set Z_ 0.0
$ns initial_node_pos $n10 30
set n11 [$ns node]
$n11 set X_ 825
$n11 set Y_ 140
$n11 set Z_ 0.0
$ns initial_node_pos $n11 30
set n12 [$ns node]
$n12 set X_ 509
$n12 set Y_ 34
$n12 set Z_ 0.0
$ns initial_node_pos $n12 30
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 30
set n14 [$ns node]
$n14 set X_ 822
$n14 set Y_ 51
$n14 set Z_ 0.0
$ns initial_node_pos $n14 30
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 30
set n16 [$ns node]
```

```
$n16 set X_ 903
$n16 set Y_ 255
$n16 set Z_ 0.0
$ns initial_node_pos $n16 30
set n17 [$ns node]
$n17 set X_ 908
$n17 set Y_ 344
$n17 set Z_ 0.0
$ns initial_node_pos $n17 30
set n18 [$ns node]
$n18 set X_ 600
$n18 set Y_ 180
$n18 set Z_ 0.0
$ns initial_node_pos $n18 30
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 30
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
$ns initial_node_pos $n20 30
set n21 [$ns node]
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 30
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 30
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 30
```

```
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 30
$n1 color red
$ns at 1.8 "$n1 color red"
$n7 color red
$ns at 0.0 "$n7 color red"

$n13 color red
$ns at 0.0 "$n13 color red"

$n20 color green
$ns at 0.0 "$n20 color green"

$n21 color green
$ns at 0.0 "$n21 color green"

$n17 color blue
$ns at 0.0 "$n17 color blue"

$n9 color blue
$ns at 0.0 "$n9 color blue"
```

Multiple wormhole nodes:

```
$ns at 1.8 "[$n1 set ragent_] wormhole "
$ns at 0.0 "[$n7 set ragent_] wormhole "
$ns at 0.0 "[$n13 set ragent_] wormhole "
```

Generate movement:

```
$ns at 0 " $n21 setdest 150 150 40 "
$ns at 0 " $n20 setdest 150 150 40 "



#$ns at 5 " $n6 setdest 650 490 40 "
#Setup a UDP connection
```

```
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n17 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n9 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n11 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500
```

```
#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
$ns attach-agent $n17 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 100.0 "$cbr4 stop"
```

Termination:

```
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

IMPLEMENTATION DETAILS

Network Simulation:-

Network Simulation is a technique whereby a software program models the behavior of a network either by calculating the interaction between the different network entities (routers, switches, nodes, access points, links etc.). Most simulators use discrete event simulation - the modeling of systems in which state variables change at discrete points in time. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network / protocols would behave under different conditions.

List Of Network Simulators:-

There are both free/open-source and proprietary network simulators available. Examples of notable network simulators / emulators include:

- ns (open source)
- OPNET (proprietary software)
- NetSim (proprietary software)

What Is NS-2:-

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. It is a discrete event simulator for networking research. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, HTTP and DSR. It simulates wired and wireless network.

Why Using NS2:-

The goal of the ns-2 project is to create an open simulation environment for networking research that will be preferred inside the research community. Ns-2 was initiated based on a refactoring. Presently ns-2 consists over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution.

Installing NS2:-

STEP 1:

Install all necessary dependencies using below commands one after another.

```
sudo apt-get install tcl8.5-dev tk8.5-dev
sudo apt-get install build-essential autoconf automake
sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev
```

STEP 2:

1. Download the NS2 Package from this link.
2. Copy the downloaded file to your /Home folder in ubuntu 14.04.
3. Right click on the file and select "Extract here" option. (You can also do this using command line).

STEP 3:

Now go to ns-allinone-2.35/ns-2.35/linkstate sub folder.
double click on "ls.h" file to open.
go to line number 137 and change the below line

```
from
```

```
void eraseAll() { erase(baseMap::begin(), baseMap::end()); }

to
void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
```

STEP 4:
Open the Terminal by pressing "ALT+CNTL+T" keys combination. And move to ns-allinne-2.35 folder from home through terminal

```
 maggi@maggi-PC:~$ cd ns-allinone-2.35/
 maggi@maggi-PC:~/ns-allinone-2.35$
```

Now type ./install on terminal

```
maggi@maggi-PC:~/ns-allinone-2.35$ ./install
```

hit enter and wait for some time till it shows path information. That's done now and you are installed NS2.
STEP 5:
Now it's time to set the path information. In the terminal use sudo gedit .bashrc and hit enter. It will ask for password to enter (Its not visible).

```
maggi@maggi-PC:~$ sudo gedit .bashrc
[sudo] password for maggi:
```

Go to the last line of the newly opened file (bashrc), copy and paste these 3 lines. Make sure that you changed maggi with your username on ubuntu.

```
PATH=$PATH:/home/maggi/ns-allinone-2.35/bin:/home/maggi/ns-allinone-2.35/tcl8.5.10/unix:/home/maggi/n
s-allinone-2.35/tk8.5.10/unix

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/maggi/ns-allinone-2.35/otcl-1.14:/home/maggi/ns-allin
one-2.35/lib

TCL_LIBRARY=$TCL_LIBRARY:/home/maggi/ns-allinone-2.35/tcl8.5.10/library
```

Save the document and close. Reload the .bashrc using the following command.

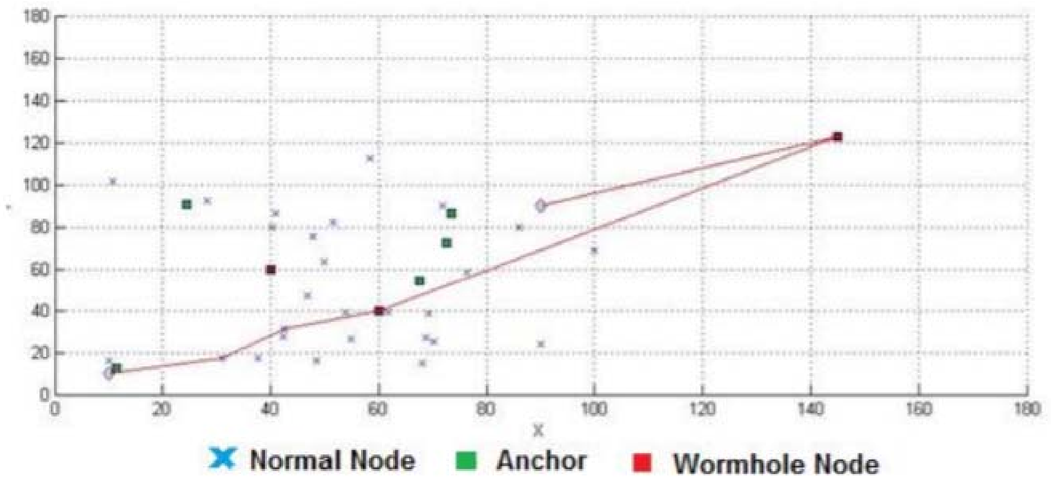
```
source ~/.bashrc
```

STEP 6:
Its done! open the terminal and type "ns" hit enter. You will get a % sign, it indicates the successful installation.

Simulation Parameter:

In our simulations and as in [53][51], we assumes that physical layer has a fixed communication range pattern, i.e. two nodes can directly communicate with each other successfully only if they are in each other communication range. We randomly deployed 50 nodes within an area of 100 x 100 meters. A fraction of these nodes was randomly selected to wormhole misbehave. The Trust Factor value of each node is initialized to TFactor = zero. Simulations are implemented with one source node and one destination node. The source node is located at the most left-bottom region of the simulation area, while the destination node is placed at the most right-upper area of simulation environment. This assumption ensures that our results are representative of a long multi-hop path from source to destination; also, it permits potential failures at various distances from the source.

Each experiment was repeated for 100 random network topologies. A brief summary of the basic simulation parameters are listed.



Parameter	Value
Simulation Area	1000 x 1000 (m)
Number of nodes	50
Number of wormhole nodes	1, 2, 4, 8, 16
Communication Range	250 m
Routing Protocol	Modified AODV
Node Speed	10 m/s

Performance Evaluation Metrics :

The evaluation of the proposed model is measured in accordance to the following three metrics:

• Average Hop-Count: Average hop count per route refers to the Total Hop Count of demands over Number of demands as in Average Hop Count = Number Of Demand / Total Hop Count Of Demand.

• Detection rate:

which is the ratio of the number of nodes that are possibly attacked by a wormhole to the number of how many of them are successfully detected as in. it is used to determine the wormhole detection rate:

Wormholes Detection Rate = Total Wormholes / Total Detected

• Detection Accuracy:

It is the ratio of the number of links declared as attacked by a wormhole to the number of how many of them are actually affected as in [43]. The following formula is used to determine the detection accuracy:

Detection Accuracy = Total Actual Wormholes / Total Detected Wormholes

Simulation Scenarios

To support different research methods, different scenarios chosen to let the wormhole attack work in more than one mode. Every mode has its own advantages for certain scenarios.

First Scenario

The simulation parameters that used in first scenario are a MANET with different sizes. Here, we assume the network size are 20, 30, 40 and 50 nodes and are randomly distributed in 1000m×1000m area. No wormhole nodes are considered in these experiments. The scenario is simulated for 100 times. Experiment results listed in the results of average hop-count according to different network size.

No. of Nodes	Average hop-count
20	5.6
30	6.3
40	6.65
50	7.9

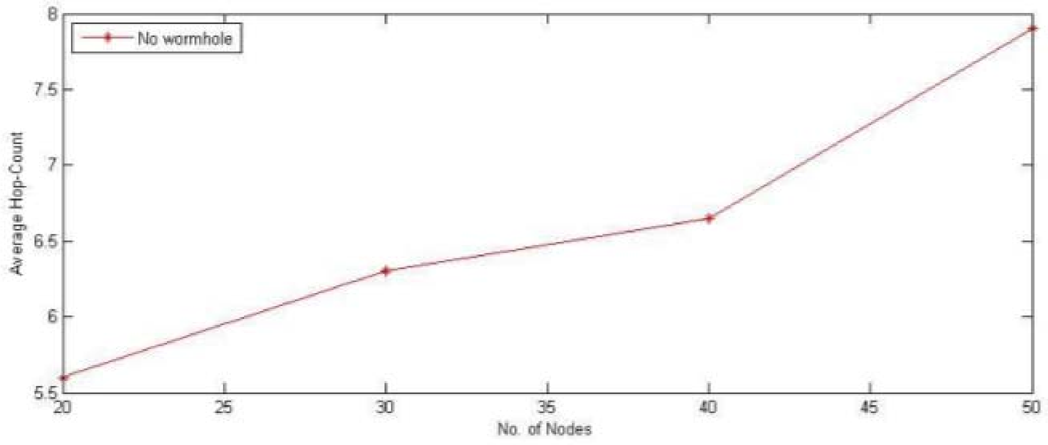


Figure 5- 2: No-Wormhole Scenario

Second Scenario

A simulation conducted with same simulation parameters that used in above scenario except that two wormhole nodes are considered. Results listed in depicts the results of average hop count according to assumed parameters.

No. of Nodes	Average Hop-Count
20	4.37
30	5.4
40	6
50	7.63

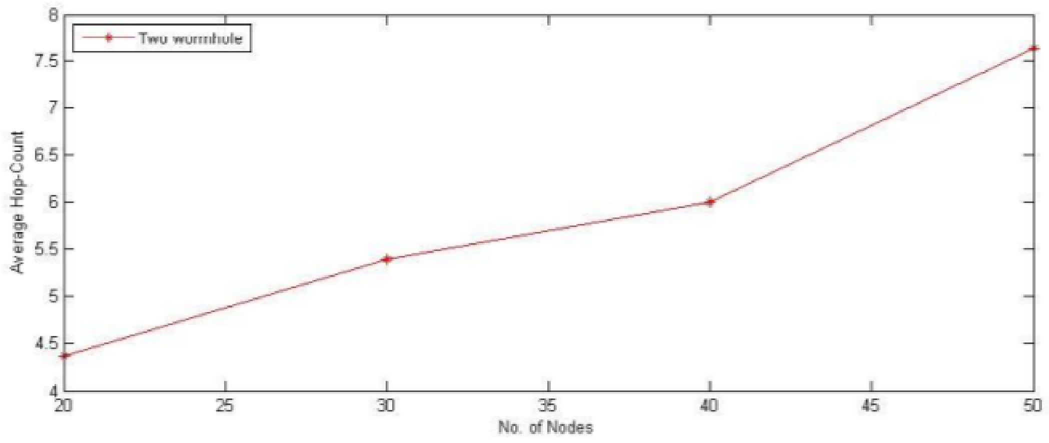


Figure 5- 3: Two Wormhole Nodes Scenario

Third Scenario

Another simulation results listed in table 5-4 and figure 5-4 depicts these results for an eight wormhole nodes. A significant change in average hop-count depicted compared to first and second experiments and this lead us to a conclusion that hop-count play an important role in detecting wormhole attack.

No. of Nodes	Average Hop-Count
20	3.2
30	4.18
40	5.75
50	7.01

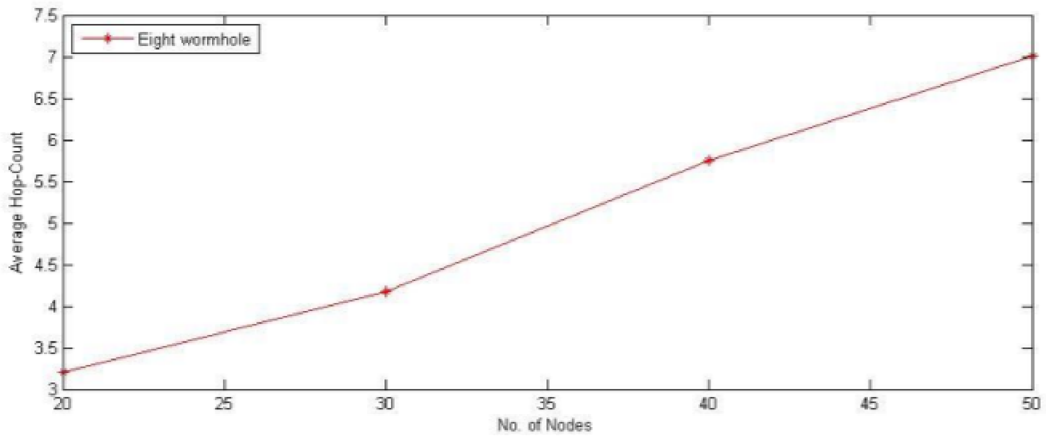


Figure 5- 4: Eight Wormholes Nodes Scenario

Experiment Results and Performance Evaluation:

All scenarios with different network sizes are obtained. Founded results are listed. In the following graph, x-axis represents number of nodes and y-axis represents the average Hop-Count. A comparison between number of nodes and the average hop-count obtained for every different scenario presented. We change the number of nodes from 20 to 50. We can find that as the number of wormhole increases, the average hop-count decreases rapidly. Thus, Hop-count metric gives us a good pointer for an existence of wormhole.

No. Nodes	No Wormhole	One Wormhole	Two Wormhole	Four Wormhole	Eight Wormhole	16 Wormhole
20	5.6	5.12	4.37	3.66	3.2	3.08
30	6.3	5.64	5.4	5.3	4.18	3.98
40	6.65	6.61	6	5.9	5.75	5.39
50	7.9	7.85	7.63	7.31	7.01	6.16

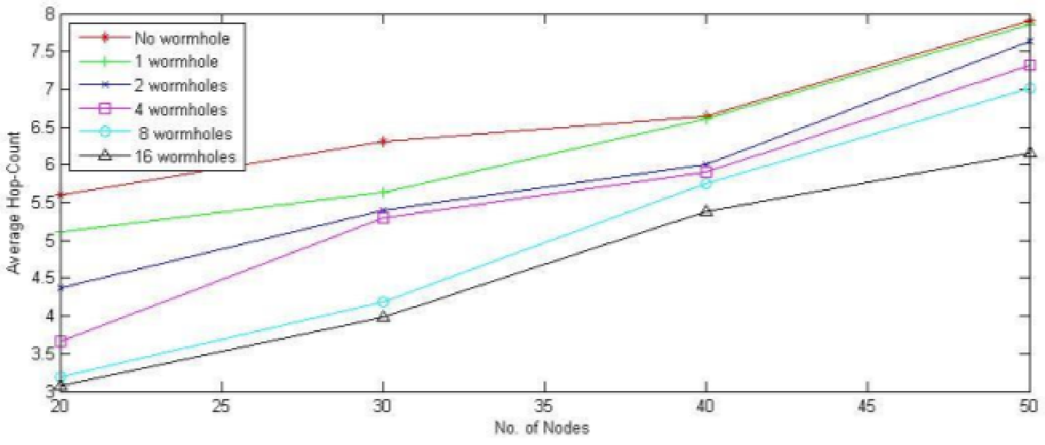
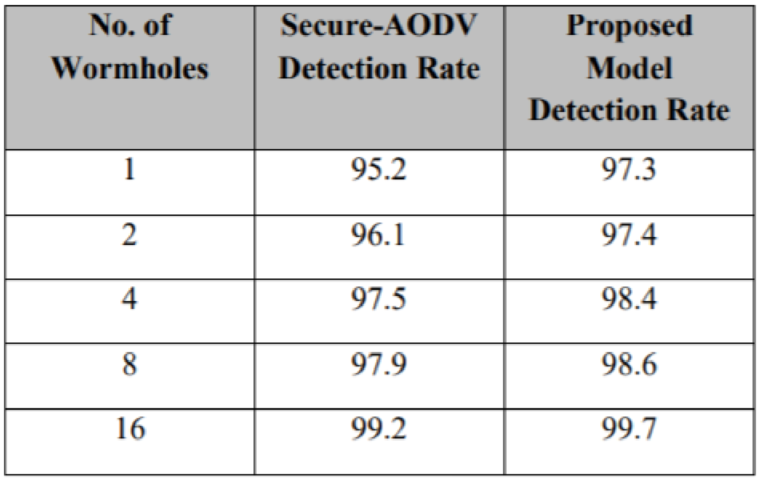


Figure 5- 5: Relation between number of nodes and number of Hop-Count

<u>Calculating Wormhole Detection Rate Metric:</u>

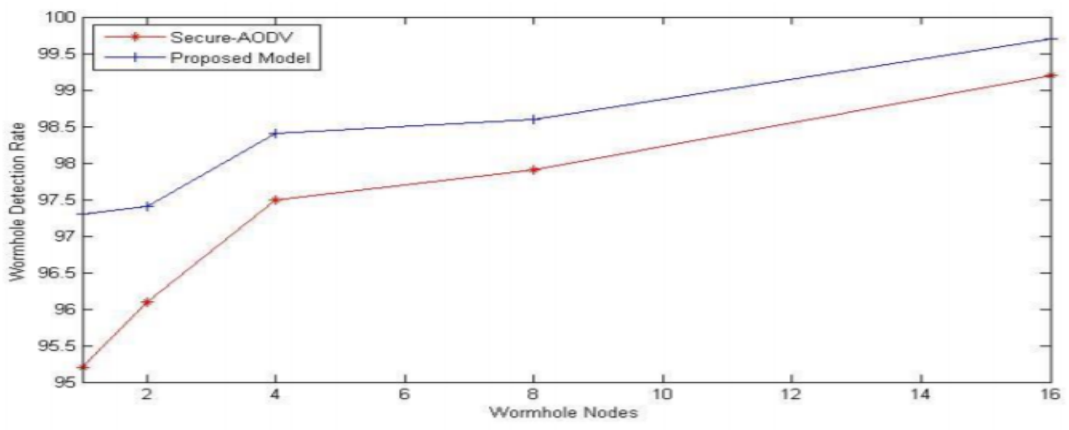
Wormhole detection rate metric calculated, we obtain the total detected wormholes for different number of wormholes in each routing model Secure-AODV and our proposed model. In Secure-AODV, the total detected wormholes were 1809, 1826, 1853, 1860 and 1885 and total of wormholes was 19. So, the detection rates calculated according to eq. 5.2 and listed. In our proposed model, the total detected wormholes were 1849, 44 1851, 1670, 1873 and 1894 and the total wormholes was 19. So, detection rates calculated. We list the experiments results obtained for different wormhole nodes to measure the wormhole detection rate. The wormhole detection rate versus the number of wormholes for AODV routing protocols compared to proposed model. It can be seen that the wormhole detection rate shows an increasing trend as the number of the wormholes is increased. This is because that with larger wormhole sizes, the probability of the actually attacked neighbors being included in the suspected part of the source's Neighbor-List is almost certain due to the hop-count between them. The detection rate curves are almost bend slightly for larger wormhole sizes because the probability of suspected nodes is much higher than the rate of change in number of one hop neighbors. The proposed model, with blue line, shows better detection rate compared to AODV routing protocol under same network configuration.

Wormhole detection rate:



No. of Wormholes	Secure-AODV Detection Rate	Proposed Model Detection Rate
1	95.2	97.3
2	96.1	97.4
4	97.5	98.4
8	97.9	98.6
16	99.2	99.7

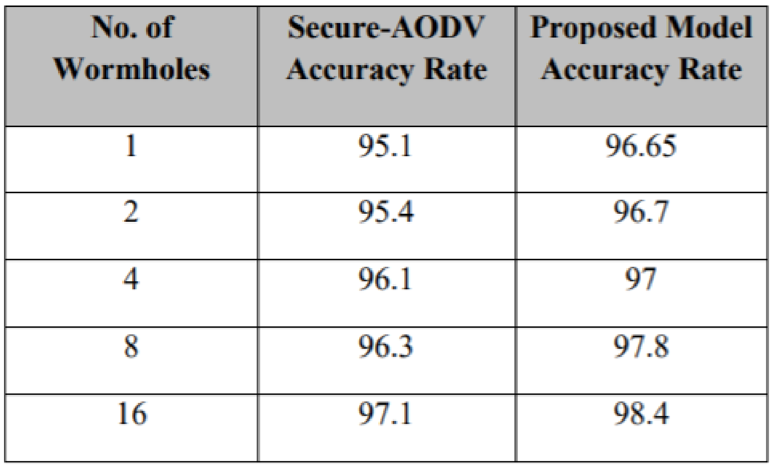
No of wormhole vs worm hole detection rate:



Calculating Wormhole Detection Accuracy Rate Metric:

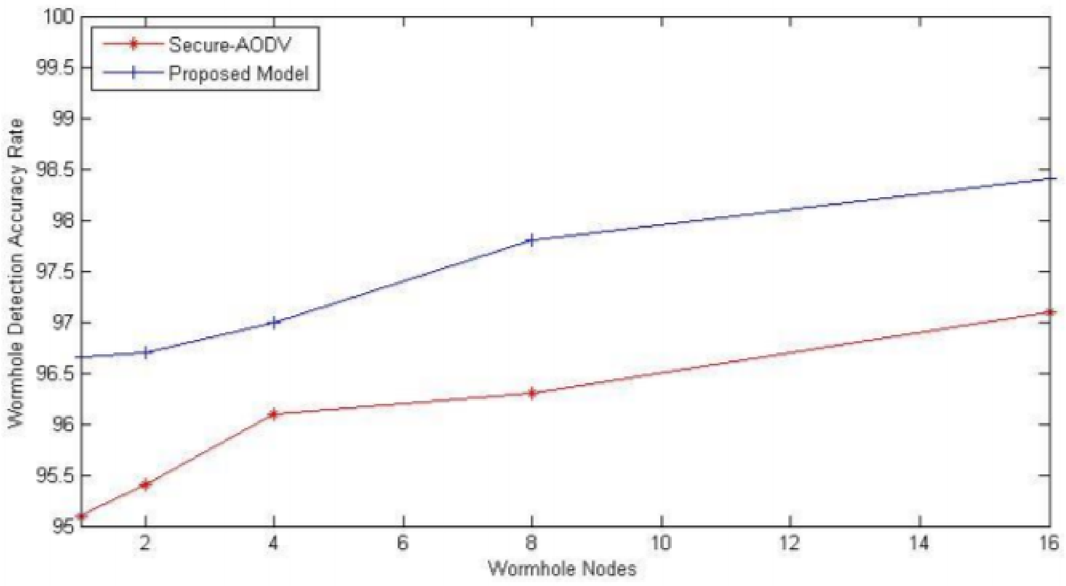
Wormhole detection accuracy rate metric calculated by the e, we obtain the total detected wormholes for different number of wormholes in each routing model Secure-AODV and our proposed model. In Secure-AODV, the total detected wormholes were 1902, 1908, 1922, 1926 and 1942 and total of wormholes was 20. So, the detection rates calculated. In our proposed model, the total detected wormholes were 1933, 1934, 1940, 1956 and 1968 and the total wormholes was 20. So, detection rates calculated, we list the experiments results obtained for different wormhole nodes to measure the wormhole accuracy rate. A comparison between AODV routing protocol and proposed model presented to show the accuracy of wormhole detection. From the results, it can be seen that our model, with blue line, achieves much higher accuracy of alarms because the number of neighbors that can be selected to form wormhole tunnels by malicious nodes. When the number of wormhole nodes in the network is equal to 1, the number of any node's neighbors is more likely to be small; as the number of wormhole increases, it becomes rarely obvious to find another route similar to that of the detected wormhole tunnel.

Worm hole detection accuracy rate:



No. of Wormholes	Secure-AODV Accuracy Rate	Proposed Model Accuracy Rate
1	95.1	96.65
2	95.4	96.7
4	96.1	97
8	96.3	97.8
16	97.1	98.4

No of wormhole vs detection accuracy rate:



FUTURE SCOPE :-

In our thesis work we will improve the security of AODV protocol using Cryptography algorithm. After improving security we will compare the modified protocol with simple AODV protocol. Further comparisons can be obtained with other protocols such as DSDV, DSR.

REFFEENCES

[1]Ravinder Ahuja, Alisha Banga Ahuja, and Pawan Ahuja. Performance evaluation and comparison of aodv and dsr routing protocols in manets under wormhole attack. In Image Information Processing (ICIIP), pages 699 – 702, 2013.

[2] Parmar Amish and V.B. Vaghela. Detection and prevention of wormhole attack in wireless sensor network using aomdv protocol. Procedia Computer Science, 79:700 – 707, 2016.

[3] Swati Bhagat and Trishna Panse. A detection and prevention of wormhole attack in homogeneous wireless sensor network. In International Conference on ICT in Business Industry Government (ICTBIG), pages 1 – 6, 2016.

[4] J. Biswas, A. Gupta, and D. Singh. Wadp: A wormhole attack detection and prevention technique in manet using modified aodv routing protocol. In 9th International Conference on Industrial and Information Systems (ICIIS), pages 1 – 6, 2014