Detecting and Evading Wormholes in Mobile Ad-hoc Wireless Networks

Asad Amir Pirzada and Chris McDonald
(Corresponding author: Asad Amir Pirzada)

School of Computer Science & Software Engineering, The University of Western Australia
135 Stirling Highway, Crawley, W.A. 6009, Australia. (Email: {pirzada,chris}@csse.uwa.edu.au)

(Received Sept. 16, 2005; revised and accepted Nov. 6, 2005)

Abstract

Mobile ad-hoc wireless networks are established in improvised environments through the mutual cooperation of its participating nodes. These nodes often operate in a physically insecure environment and, as a result, are vulnerable to capture and compromise. In addition, the nature of the wireless communication medium restricts enforcement of rigourous node memberships and so a number of malicious nodes may also participate in the network. These nodes, in order to snoop or sabotage, can undertake a variety of attacks against the network. Among these, wormhole attacks have unusual significance primarily due to their modus operandi and peculiar attack pattern. In such attacks, two or more malicious colluding nodes create a higher level virtual tunnel in the network, which is employed to transport packets between the tunnel endpoints. These tunnels emulate shorter links in the network and so act as bait to unsuspecting network nodes which, by default, seek shorter routes. The benefit gained by the malicious nodes is that they are able to conduct a variety of attacks against the tunnelled traffic. In this paper, we present a novel trust-based scheme for identifying and isolating nodes that create a wormhole in the network, without engaging any cryptographic means. With the help of extensive simulations, we demonstrate that our scheme functions effectively in the presence of malicious colluding nodes and does not impose any unnecessary conditions upon the network establishment and operation phase.

Keywords: Ad-hoc, attacks, network, routing protocol, security, trust

1 Introduction

An ad-hoc network is built, operated, and maintained by its constituent wireless nodes. These nodes generally have a limited transmission range and so each node seeks the assistance of its neighbouring nodes in forwarding packets. In order, to establish routes between nodes, which are farther than a single hop, specially configured routing protocol are engaged. The unique feature of these protocols is their ability to trace routes in spite of a dynamic topology. These protocols can be categorised into two main types: Reactive and Proactive [19]. The nodes in an ad-hoc network generally have limited battery power and so active routing protocols endeavour to save upon this, by discovering routes only when they are essentially required. In contrast, proactive routing protocols continuously establish and maintain routes, so as to avoid the latency that occurs during new route discoveries. Both types of routing protocols require persistent cooperative behaviour, with intermediate nodes primarily contributing to the route development. Similarly each node, which acts like a mobile router, has absolute control over the data that passes through it. In essence, the membership of any ad-hoc network indisputably calls for sustained benevolent behaviour by all participating nodes.

In real life, such an altruistic attitude is more than often extremely difficult to realise and so we often find malicious nodes also present in the same network. Some of these are alien nodes, which enter the network during its establishment or operation phase, while others may originate indigenously by compromising an existing benevolent node [2]. These malicious nodes can carry out both Passive and Active attacks against the network [5]. In passive attacks a malicious node only eavesdrop upon packet contents, while in active attacks it may imitate, drop or modify legitimate packets [3]. The severity of such attacks increases multifold especially when these are performed in collusion. A typical example of such a cooperative attack is a wormhole [7] in which a malicious node tunnels the packets from one end of the network to another. The tunnel essentially emulates a shorter route through the network and so naive nodes prefer to use it rather than alternate longer routes. The advantage gained by the colluding nodes is obvious as they are now, for all intents and purposes, in charge of a high usage route through the network. The consequences of such a wormhole on the network can be catastrophic, and in worst case scenarios, may lead to a vertex cut in the network [14].

To counter known attacks, a number of secure routing protocols have been developed that essentially make use of cryptographic techniques to protect the network traffic. However, a comparison [15] indicates that for effective execution of these protocols a number of mandatory requirements must be met before and after the network establishment phase. These requirements include configuration of the nodes prior to joining the network and/or creation of a centralised or distributed trusted third party. Ad-hoc networks, which are inherently impulsive, lack the instinctive attribute of planning and therefore oppose extraneous requirements. To facilitate deployment of secure routing protocols, ad-hoc networks have been grouped into two types: Managed and Pure [16]. Managed ad-hoc networks are those which are established in a scenario where some information regarding the composition and condition of the network is available beforehand. This permits configuration of the nodes with encryption keys and certificates, along with the creation of a key repository [17] for the implementation of various security services [20]. In contrast, pure ad-hoc networks have no a-priori knowledge of their future setup and so are not based upon any assumptions. They tend to adopt the human pattern of reaction in unrehearsed situations and hence believe in improvisation. These networks are thus operated by nodes that put trust [10] in each other's capabilities to get their individual chores accomplished. This trust is affected by past experiences with other nodes over a period of time and is reinforced through reference.

In this paper, we apply a similar trust based scheme to the Dynamic Source Routing (DSR) protocol to detect and evade wormhole attacks in a pure ad-hoc network. Each node in the network autonomously executes the trust model and maintains its own evaluation regarding other nodes in the network. Each node, based upon its own experience, rewards collaborating nodes for their benign behaviour and penalises malicious nodes for their malignant conduct. Source nodes use this trust information to circumvent malicious nodes by computing the most trustworthy path to a particular destination. The routes computed in this manner are neither protected in terms of security and possibly not minimal in terms of hops. However, these routes traverse nodes that have been found to be more trustworthy compared to others and are hence more dependable in an unfamiliar situation.

In Section 2 of this paper we present some relevant previous work. In Section 3 we describe our proposed scheme in detail. Simulation results are presented in Section 4. An analysis of the proposed scheme is presented in Section 5 with concluding remarks in Section 6.

2 Previous Work

2.1 Packet Leashes

Packet Leash [7] is a mechanism to detect and defend against wormhole attacks. The mechanism proposes two types of leashes for this purpose: Geographic and Temporal. In Geographic Leashes, each node knows its precise position and all nodes have a loosely synchronised clock. Each node, before sending a packet, appends its current position and transmission time to it. The receiving node, on receipt of the packet, computes the distance to the sender and the time it took the packet to traverse the path. The receiver can use this distance and time information to deduce whether the received packet passed through a wormhole or not. In Temporal Leashes, all nodes are required to maintain a tightly synchronised clock. All sending nodes append the time of transmission to each sent packet. The receiver compares the time to its locally maintained time and assuming that the transmission propagation speed is equal to the speed of light, computes the distance to the sender. The receiver is thus able to detect, whether the packet has travelled on additional number of hops before reaching the receiver. Both types of leashes require that all nodes can obtain an authenticated symmetric key of every other node in the network. These keys enable a receiver to authenticate the location and time information in a received packet.

2.2 SECTOR

The Secure Tracking of Node Encounters in Multi-hop Wireless Networks (SECTOR) [1] is a set of mechanisms, which can be used to prevent wormhole attacks without requiring any clock synchronisation or location information. SECTOR uses a distance-bounding protocol called MAD (Mutual Authentication with Distance-bounding) to determine the distance between any two communicating parties. MAD assumes that each node is equipped with a special hardware transceiver module that can receive a single bit, perform a two bit XOR operation on it and transmit a single bit without involving the main CPU. This module is used to perform a series of bit exchanges among the two nodes, so as to compute the precise interval between sending and receiving the bit patterns. This interval helps to compute the distance between the communicating parties. To ensure the integrity of the bit patterns, MAD makes use of one-way hash chains and hash trees. To prove the authenticity of the messages, MAD uses message authentication codes (MAC), which are secured using pairwise secret keys. The security of the MAD protocol depends upon the number of bits exchanged between the sender and the receiver. The accurate execution of the MAD protocol provides the receiver with the exact distance to a sender. The receiving node can thus confirm whether the packets being received are actually traversing a legitimate path or are being tunnelled through colluding nodes.

2.3 Directional Antennas

Wormhole attacks can also be countered using directional antennae as suggested by Hu et al. [8]. All nodes in the network share their directional information with other nodes in the network in order to prevent wormhole attacks. In doing so, each node maintains accurate information about the bearing of its immediate neighbours. Any direct messages from a non-neighbour are immediately discarded. The scheme presents two types of neighbour discovery protocols that can be used to prevent wormholes. The first protocol, called the Verified Neighbour Discovery Protocol, protects from adversaries that have a single endpoint in the neighbourhood. During the execution of the protocol, the announcing node exchanges six messages with its neighbouring node, before the announcer or its neighbour accepts each other as neighbours. The protocol also requires the services of a third node called the verifier, which is in a different direction than that of the neighbour and the announcer, so as to protect the discovery protocol itself from wormhole attacks. The messages shared between the announcer, neighbour and the verifier include `HELLO` packets, nonce and identity tags. The neighbour shares two secret keys one each with the announcer as well as the verifier. All critical inter-node messages are encrypted before transmission using these keys. Another reinforced variant of the first protocol, Strict Neighbour Discovery Protocol, protects against wormhole attacks even when the adversary has two endpoints in the neighbourhood. However, there may be situations when no verifying node is available in the vicinity leading to the failure of both protocols.

2.4 MDS-VOW

The Multi-Dimensional Scaling Visualisation Of Wormhole (MDS - VOW) [21] is a mechanism to detect wormholes in sensor networks. The mechanism does not require any special hardware such as positioning devices, synchronised clocks or directional antennas. MDS-VOW uses multi-dimensional scaling to reconstruct the network and detects the attack by visualising the anomaly introduced by the wormhole. Each node in the network estimates the distance to its immediate neighbours based upon the received signal strength. These distances are then sent to the centralised controller, which uses the Dijkstra algorithm [4] to calculate the distance between all nodes. The controller uses MDS to find the virtual position of each sensor and uses a smoothing function to cater for measurement errors. If the smoothed surface is flat, it indicates the absence of a wormhole. However, if the surface between two nodes is bend towards each other, it suggests the likely existence of a wormhole. The controller computes the wormhole indicator for each node and distributes them to the sensors. The sensors use these indicators to evade subsequent communication through the wormholes. In order to ensure the integrity of packets containing distance information and wormhole indicators, MDS-VOW uses Message Authentication Codes that are formed using a group key shared between the sensors and the controller.

The aforementioned schemes implement various mechanisms to detect and protect against wormholes in an ad-hoc network. The use of cryptography, clock synchronisation and special devices indeed helps in protecting against wormholes but is considered an extraneous requirement, which contradicts the spontaneous nature of ad-hoc network. All schemes have certain pre and post establishment conditions, which restrict their application only to managed ad-hoc wireless networks.

3 Trust-based Wormhole Detection and Evasion in DSR

3.1 Dynamic Source Routing (DSR) Protocol

The DSR protocol [9] is a reactive routing protocol. As the name suggests it uses IP source routing. All data packets that are sent using the DSR protocol contain the complete list of nodes that the packet has to traverse. During route discovery, the source node broadcasts a `ROUTE REQUEST` packet with a unique identification number. The `ROUTE REQUEST` packet contains the address of the target node to which a route is desired. All nodes that have no information regarding the target node or have not seen the same `ROUTE REQUEST` packet append their IP addresses to the `ROUTE REQUEST` packet and re-broadcast it. In order to control the spread of the `ROUTE REQUEST` packets, the broadcast is done in a non-propagating manner with the IP TTL field being incremented in each route discovery. The `ROUTE REQUEST` packets keep on spreading until they reach the target node or any other node that has a route to the target node. The recipient node creates a `ROUTE REPLY` packet, which contains the complete list of nodes that the `ROUTE REQUEST` packet had traversed. Based upon implementation, the target node may respond to one or more incoming `ROUTE REQUEST` packets. Similarly, the source node may accept one or more `ROUTE REPLY` packets for a single target node. The selection of the `ROUTE REPLY` can be made both on minimal hop count or latency.

For optimisation reasons, nodes maintain a `PATH CACHE` or a `LINK CACHE` scheme [6]. All nodes either forwarding or overhearing data and control packets, add all useful information to their respective route cache. This information is used to limit the spread of control packets for subsequent route discoveries. The DSR protocol also provides the facility for 'route shortening' to avoid unnecessary intermediate nodes. For example, if a node overhears a data packet that is supposed to traverse a number of nodes before passing through it, this node creates a shorter route known as Gratuitous `ROUTE REPLY` and sends it to the original sender.

3.2 Wormhole Creation

In any ad-hoc network, a wormhole can be created through the following three ways:

- Tunnelling of packets above the network layer
- Long range tunnel using high power transmitters
- Tunnel creation via external wired infrastructure

In the first type of wormhole, all packets which are received by a malicious node are duly modified, encapsulated in a higher layer protocol and dispatched to the colluding node using the services of the network nodes. These encapsulated packets traverse the network in the regular manner until they reach the collaborating node. The recipient malicious node, extracts the original packet, makes the requisite modifications and sends them to the intended destination. In the second and third type of wormholes, the packets are modified and encapsulated in a similar manner. However, instead of being dispatched through the network nodes, they are sent using a point-to-point specialised link between the colluding nodes. In this paper, we only discuss solutions to the first type of wormhole which in our opinion has greater applicability to pure ad-hoc networks.

In an ad-hoc network executing the DSR protocol, each packet contains the complete list of nodes that it has to traverse in order to reach the destination. This feature, although excludes intermediate nodes form making any routing decisions, can still be exploited to create a wormhole. Such wormholes can be created in a number of topological scenarios. However, all such settings are primarily derived from scenarios where the colluding nodes (M1,M2) are not the immediate neighbours of the source (S) and destination (D) nodes, as shown in Figure 1.

Wormhole creation in such a scenario is generally accomplished using the following steps:

Sustained Routes between Colluding Nodes. M1 and M2 periodically establish and maintain routes to each other in the network at all times. This route serves as a higher layer tunnel for all other nodes whose traffic is routed through M1 and M2.

Fallacious Response to Source Node Route Requests. Whenever a `ROUTE REQUEST` packet from S is received by M1, it immediately sends a `ROUTE REPLY` packet so as to portray minimal delay. M1 also makes the `ROUTE REPLY` packet (S-1-M1-M2-D) as short as possible, indicating D as an immediate neighbour of M2. Such `ROUTE REPLY` packets, have a high probability of being selected by S as they have minimal hop-count and latency.

Route Development till the Destination Node. M1 informs M2 to initiate a route discovery to D through a pre agreed upon higher layer protocol and also performs the same. In the mean time, all data packets from S to D are buffered for a certain interval at M1. While waiting for a route to D, if M1 receives a `ROUTE REPLY` packet from D to S, it verifies whether it can reach D through M2. If yes, it creates a new working source route option from M2 to D (S-M1-M2-5-D) for the buffered packets, encapsulates and sends them to M2, else it waits for the `ROUTE REPLY` packet to be received in response to the `ROUTE REQUEST` packet that was initiated by itself and M2 . Upon receipt of these `ROUTE REPLY` packets, M1 traces an optimal route to D through M2. However, if during this waiting period, the buffer interval expires or an overflow occurs, M1 sends a `ROUTE ERROR` packet to S for the last received data packet.

Deception through Gratuitous Route Replies. As an alternate mechanism, if M1 overhears any ongoing communication between S and D (S-1-2-3-4-5-D). It may initiate a new route discovery to D and also request the same through M2. Upon receipt of a route from M1 to D via M2, it can create a new Gratuitous `ROUTE REPLY` packet (S-1-M1-M2-D) and send it to S. Based upon the same criterion for route selection, S may classify the newly received route as optimal and discard the one that was already in use.

Translation of IP Addresses. IP Address translation is done both at M1 and M2 to successfully route all data through the created tunnel.

3.3 Trust Model

We detect and evade wormholes in the network using an effort-return based trust model. The trust model uses the inherent features of the Dynamic Source Routing (DSR) protocol to derive and compute respective trust levels in other nodes. For correct execution of the model, the following conditions must be met by all participating nodes:

- All nodes support promiscuous mode operation.
- Node transceivers are omnidirectional and that they can receive and transmit in all directions.
- The transmission and reception ranges of the transceivers are comparable.

Each node executing the trust model, measures the accuracy and sincerity of the immediate neighbouring nodes by monitoring their participation in the packet forwarding mechanism. The sending node verifies the different fields in the forwarded IP packet for requisite modifications through a sequence of integrity checks. If the integrity checks succeed, it confirms that the node has acted in a benevolent manner and so its direct trust counter is incremented. Similarly, if the integrity checks fail or the forwarding node does not transmit the packet at all, its corresponding direct trust measure is decremented. We

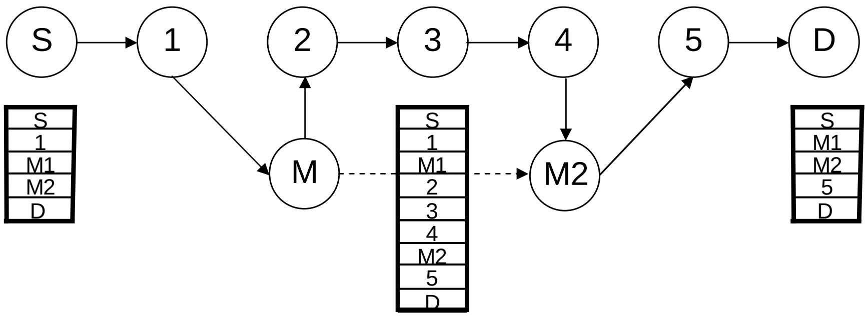

Figure 1: Wormhole creation in DSR

represent the direct trust in a node y by node x as T_{xy} and is given by the following equation:

$$\mathrm{T}_{xy} = \mathrm{P_P} \, . \, \mathrm{P_A}$$

where $\mathrm{P_P} \in [0,1]$, represents the situational trust category Packet Precision, which essentially indicates the existence or absence of a wormhole through node y. $\mathrm{P_A}$ represents the situational trust category Packet Acknowledgements that preserves a count of the number of packets that have been forwarded by a node. The category $\mathrm{P_P}$ and $\mathrm{P_A}$ are employed in combination to protect the DSR protocol against wormhole attacks and for identifying selfish node behaviour respectively. Any benevolent node not able to forward a data packet, due to radio interference, hardware faults, software bugs or environmental conditions, is classified as selfish. However, in case no other alternate trusted nodes are available, these selfish nodes will be engaged into the routing process. However, any node incorrectly forwarding a data packet, by not ensuring its integrity, will be classified as malicious and not included in any subsequent data connections.

3.4 Wormhole Detection

During wormhole detection, each node in the network measures the accuracy and sincerity of its immediate neighbouring nodes. The relevant C++ code, for DSR based networks, is shown in Figure 2. The detection process works in the following manner:

- Each node, before transmission of a data packet, buffers the DSR Source Route header. After transmitting the packet, the node places its wireless interface into the promiscuous mode for the Trust Update Interval (TUI). The TUI fundamentally represents the time a sending node must wait after transmitting a packet until the time it overhears the retransmission by its neighbour. This interval is critically related to the mobility and traffic of the network and needs to be set accordingly. If this interval is made too small it may result in ignoring of the retransmissions, similarly a large value may induce errors due to nodes moving out of range.
- If during the TUI, the node is able to overhear its immediate node retransmit the same packet, the sending node increases the situational trust category $\mathrm{P_A}$ for that neighbour. It then verifies whether the retransmitted packet's DSR Source Route header is the same as the one that was buffered earlier. If the Salvage field[1] of the DSR Source Route option is zero, then these list of addresses should be exactly the same. If this integrity check passes, the situational trust category $\mathrm{P_P}$ is not set, indicating an absence of a wormhole. However, if the retransmitting node, modifies the DSR Source Route header, the detecting node sets $\mathrm{P_P}$ to true.
- In case no retransmission is heard and a timeout occurs when the TUI has exceeded, the situational trust category $\mathrm{P_A}$ for that neighbour is reduced and the DSR Source Route buffer is cleared.
- With the passage of time, the number of inter-node interactions also increase, increasing each node's knowledge of the behaviour of other nodes. Any forwarding node, which had earlier detected wormhole creation by any of its immediate neighbour, drops all packets that were destined to go through that neighbour and generates a corresponding `ROUTE ERROR` packet. This packet informs the source and all intermediate nodes regarding the unavailability of the route through the wormhole. Consequently, the wormhole is circumvented in subsequent data connections.

3.5 Wormhole Evasion

The relevant C++ code for wormhole evasion is shown in Figure 3. In DSR, before initiating a new route discovery, the cache is first scanned for a working route to the destination. In the event of unavailability of a route from the cache, the `ROUTE REQUEST` packet is propagated. When the search is made for a route in the cache, the Dijkstra algorithm [4] is executed, which returns the shortest path in terms of number of hops. In the `LINK CACHE` scheme the default cost of each link is one, which signifies uniform

[1]In case the Salvage field is nonzero in the DSR Source Route option, it implies a non working route through the forwarding node and hence calls for a new route discovery by the source node. Such an activity may be ignored or taken into consideration by the preceding node, depending upon the frequency of such Salvage operations.

```
bool P_P = FALSE;                             //Node wormhole status
int P_A = NEUTRAL;                            //Initial Node selfishness level
int TUI=5;                                    //Trust Update Interval (5 seconds)

DSRAgent::handleForwarding(SRPacket &p)       //DSR forwarding mechanism
{
   if (p.route = TRUE )                       //Find a route for the packet
   {
      P_P = getstatus(nexthop);               //Check the next hop status
      if (!P_P)                               //If next hop not a wormhole
      {
         buffer_packet(p)                     //Buffer packet SR header
         schedule forwarding(p);              //Schedule packet dispatch
         Trust_Scheduler(TRUE);               //Trust_Scheduler started
      }
      else
      xmitFailed(p.pkt, DROP_PACKET);         //Drop packet and Send ROUTE ERROR
   }
}

DSRAgent::buffer_packet(Packet *p)            //Packet buffering before forwarding
{
   hdr_sr *buff_srh=hdr_sr::access(p);        //Buffer packet SR Header
}

DSRAgent::tap(const Packet *packet)           //DSR promiscuous mode tap
{
   P_A++;                                     //Packet forwarded
   P_P = verify_packet_integrity(p)           //Detect Wormhole
   Trust_Scheduler(FALSE);                    //Trust_Scheduler disabled
}

DSRAgent::verify_packet_integrity(packet *p)//Verify P_P
{
   hdr_sr *srh = hdr_sr::access(p);
   return (srh != buff_srh);                  //Return TRUE if wormhole detected
}

Trust_Scheduler::handle(bool status)          //Called every TUI seconds
{
   buff_srh=NULL;                             //Clear SR buffer
   P_A--;                                     //Packet not forwarded
   Trust_Scheduler(FALSE);                    //Trust_Scheduler disabled
}
```

Figure 2: Code for wormhole detection

spread of the inter-node trust levels. We replace this cost with the actual trust level of a node to which this particular link is directed. Now, each time a new route is required, a modified variant of the search algorithm is executed, which finds routes with the maximum trust level. However, before cost assignment to any link, each node first checks the wormhole status of the link end node. If it has been classified as a wormhole, the cost of that link is set to infinity. This method ensures that wormholes nodes are avoided in all future data connections.

4 Simulation

4.1 Set-up

To evaluate the effectiveness of the proposed scheme, we simulated the scheme in NS-2 [13]. The simulation parameters are listed in Table 1. We implement the random waypoint movement model for the simulation, in which a node starts at a random position, waits for the pause time, and then moves to another random position with a velocity chosen between 0 m/s and the maximum simulation speed. All benign nodes execute the trust model for the duration of the simulation. The TUI value is set to 5 seconds, which has been found optimal in prior experiments [18], for networks where the nodes have a maximum speed of up to 20 m/s with a transmission range of 250 metres. The performance metrics are obtained through ensemble averaging [23] over 100 simulations, each with a different mobility and connection pattern.

4.2 Metrics

To evaluate the performance of the proposed scheme, we have used the following metrics:

4.2.1 Throughput

It is the ratio between the number of packets received by the application layer of destination nodes to the number of packets sent by the application layer of source nodes.

4.2.2 Packet Overhead

This is the ratio between the total number of control packets generated to the total number of data packets received during the simulation time.

4.2.3 Average Latency

Gives the mean time (in seconds) taken by the packets to reach their respective destinations.

4.2.4 Path Optimality

It is the ratio between the number of hops in the optimal path to the number of hops in the path taken by the

```
t(n)  The trust value assigned to node n by the source node s
p     Set of predecessors for each node on the most trustworthy path from the
      source
S     Set of nodes whose most trustworthy path from the source has been found
Q     Set of nodes whose most trustworthy path from the source has yet to be
      determined


void DSRAgent::handlePktWithoutSR(SRPacket& p, bool retry)
{
   if (route_cache->findRoute(p.dest, p.route))//Find route for destination
      sendOutPacketWithRoute(p, true);         //Send out packet with route
   else
      getRouteForPacket(p, retry);             //Initiate new route discovery
}

LinkCache::findRoute(ID dest, Path& route)
{
    dijkstra();                                //Execute modified Dijkstra
}

dijkstra ()
{
   Link *v;
   for (all v links from s)
   {
      P_P = getstatus(v->ln_dst);              //Check link end node status
      if (!P_P)                                //If node not a wormhole
      v->ln_cost = 1/P_A;                      //Set link cost to P_A^-1
      else
      v->ln_cost = INFINITY;                   //Else set cost to infinity
   }
   initialize(s)
   S = Q = {}
   add s to Q
   while Q is not empty
   {
      u = extract-most-trustworthy(Q);
      add u to S;
      for each node v which is an immediate neighbor of u
      {
         relax-neighbors(u, v, v->ln_cost );
      }
   }
}

initialize(Node s)
for all v nodes in the link-cache
{
   t(v) = 0;                                   //Set initial trust to distrust
   p[v] = NULL;
   t(s) = INFINITY;                            //Source node own trust level
}

extract-most-trustworthy(Q)
{
   find the most trustworthy node in Q
   remove it from Q
   return the node
}

relax-neighbors(Node u, Node v, double cost)
{
   if t(v) <= t(u) + cost                      //Lower trust nodes are bypassed
   {
      t(v) = t(u) + cost;
      p(v) = u;                                //Set predecessor of v to be u
   }
}
```

Figure 3: Code for wormhole evasion

Table 1: Simulation parameters

Examined Protocol	DSR
Simulation time	900 seconds
Simulation area	1000 x 1000 m
Number of nodes	50
Transmission range	250 m
Movement model	Random way point
Propagation Model	Two-ray Ground Reflection
Maximum speed	20 m/s
Pause time	10 seconds
Traffic type	CBR (UDP)
Maximum Connections	30
Payload size	512 bytes
Packet rate	4 pkt/sec
Malicious nodes	2
Number of wormholes	1

packets.

4.2.5 Detection Probability

It is the ratio between the number of nodes whose behaviour (malicious or benevolent) is identified correctly to the actual number of such nodes present in the network.

4.3 Results and Discussion

The results shown in Figure 4, indicate that the throughput of the Trusted DSR protocol is lower than that of the Standard DSR in the presence of a single wormhole in the network. This is due to the fact that the benevolent nodes detecting the wormhole are dropping legitimate data packets before they enter the tunnel. While, the standard DSR protocol, permits the packets to traverse the tunnel and so depicts a higher throughput. However, it can be observed that at zero mobility, the wormhole position remains constant in the network and so a higher throughput is achieved using the trust based DSR protocol. This is due to the fact that the trust level of any node not capable of sustaining the required traffic flow is automatically downgraded when it dumps the packets and some other node having a higher trust level is selected for the routing process. This feature helps to reduce traffic congestion onto trustworthy nodes by transferring the traffic load onto other available nodes in the neighbourhood ensuring a best-effort delivery for the generated traffic.

In case of detection of a wormhole by an intermediate node, all data packets leading towards the tunnel are dropped and a corresponding `ROUTE ERROR` packet is generated. The generation of these packets augments with the speed of the network as the colluding nodes are constantly varying their positions in the network. This primarily leads to an increase in the packet overhead when the trust based DSR protocol is used. The average latency of the network also increases with speed, as the trusted paths are not always the shortest in terms of number of hops. The latency is also influenced by additional delays associated with new route discoveries in response to received `ROUTE ERROR` packets. The path optimality of the trust based DSR protocol also degrades, reflecting the selection of trustworthy routes in favour of the shortest routes.

The probability of detection of wormholes significantly increases with speed. The results shown in Figure 5, indicate that the mobility of the network supplements the trust derivation mechanism. At higher speeds the number of interactions with the nodes creating the wormhole increase considerably. This helps to spread trust information in the network at a appreciably higher rate. Up to 60% of the nodes executing the trust based DSR protocol were able to correctly identify at least one end of the wormhole. However, with increased mobility, the probability of detection of at least one colluding node by all network nodes becomes almost 100%. Similarly, the detection probability for benevolent behaviour also follows a similar trend under increasing speeds. A number of nodes, whose behaviour pattern could not be analysed, were primarily those who were not part of any data connection during the simulation.

The standard DSR protocol, does not take into account the trust levels of the nodes and so we see that a number of packets were tunnelled through the wormhole. In contrast, each node using the trust based routing scheme takes into account the behaviour of the next node before forwarding a packet and so the total number of tunnelled packets drops appreciably. It can also be observed that at varying speeds, there are still some packets which are routed through the wormhole. The justification for such an occurrence is that the wormhole detection mechanism is based upon a minimal threshold (presently set to consecutive modification of two DSR source route headers) before it stops the data communication through the wormhole. This permits a small number of data packets to permeate the wormhole.

5 Analysis

The quantitative analysis of the trust model was carried out under variable speed networks. The results indicate that the throughput and path optimality of the trust based DSR protocol is lower than that of the standard DSR protocol in the presence of a wormhole in the network. Similarly, the packet overhead and latency of the trust based scheme is higher than that of the standard DSR protocol. This degradation in performance can be attributed to the fact that the standard DSR protocol does not has any knowledge regarding the presence of the wormhole in the network and so nodes continue to route data connections through the wormhole. However, the trust based DSR protocol, computes real-time trust in neighbouring nodes and so can make instantaneous decisions related to the routing process. Any source node,

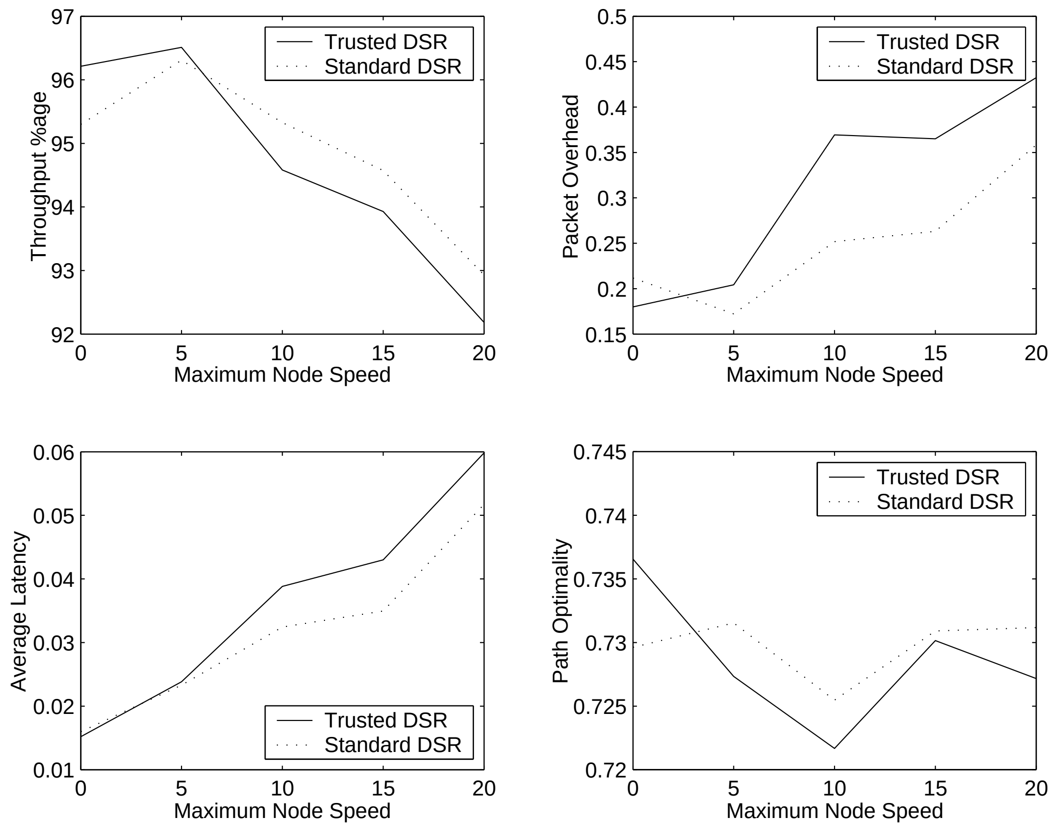

Figure 4: Simulation results of trust-based wormhole detection and evasion

having prior knowledge regarding the wormhole, creates a route that would evade the wormhole. Even though some operational routes exist in the LINK CACHE, they are discarded because they contain malicious nodes. The increase in the packet latency and deviation from the optimal paths can be attributed to the fact that the routes obtained from the LINK CACHE are not optimal in terms of hops but instead consist of nodes that have been found to be more trustworthy than the others. This indirectly leads to a decrease in the path optimality, as the packets now traverse longer routes, which in turn increases the latency of the network. Similarly, any intermediate node, which finds a wormhole, drops all packets being routed towards it and generates a corresponding ROUTE ERROR packet. This in turn reduces the throughput of the network, while increasing the packet overhead. The nodes that execute the trust model, can successfully detect the wormhole behaviour and this detection probability increases with the mobility of the network. The detection of such wormholes facilitates nodes to bypass the wormhole and to direct data through alternate routes.

Any intermediate node that drops data packets in order to stop them from being tunnelled, may be graded as selfish by its preceding node. This is due to the fact that the preceding node was unable to overhear its packets being forwarded during the trust update interval. However, when the next node sends the related ROUTE ERROR packet, the preceding node rectifies the corresponding trust levels. Similarly, a malicious node may send fallacious ROUTE ERROR packets to degrade working routes. Such packets are not considered as malevolent by the trust model, but regarded as normal events related to link breakage. Any node receiving these packets, first scans the LINK CACHE for any alternate routes and if unsuccessful initiates a new discovery for the destination node.

In its present state, the trust model detects nodes that portray selfish behaviour and may not include them in the routing process. Such scenarios may help a selfish node to conserve its battery power. However, as the trust level of such nodes is known to other nodes in the network, a punishing scheme may be implemented in which such nodes are not provided the requisite network services. Such a mechanism may help to invigorate cooperation rather than selfishness in the participating nodes.

The trust model maintains the level of trust for nodes based upon a node's IP address. This technique is, however, susceptible to a number of impersonation attacks where a node may frequently change its IP address after launching an attack in order to attain a higher trust level. In both wired and wireless networks, node identification is accomplished using various cryptographic mechanisms, which we have deliberately tried to avoid primarily due to their associated requirements.

The method for promiscuous mode trust assignment also has certain drawbacks, which have been highlighted by Marti et al. [12]. The foremost is the ambiguous col-

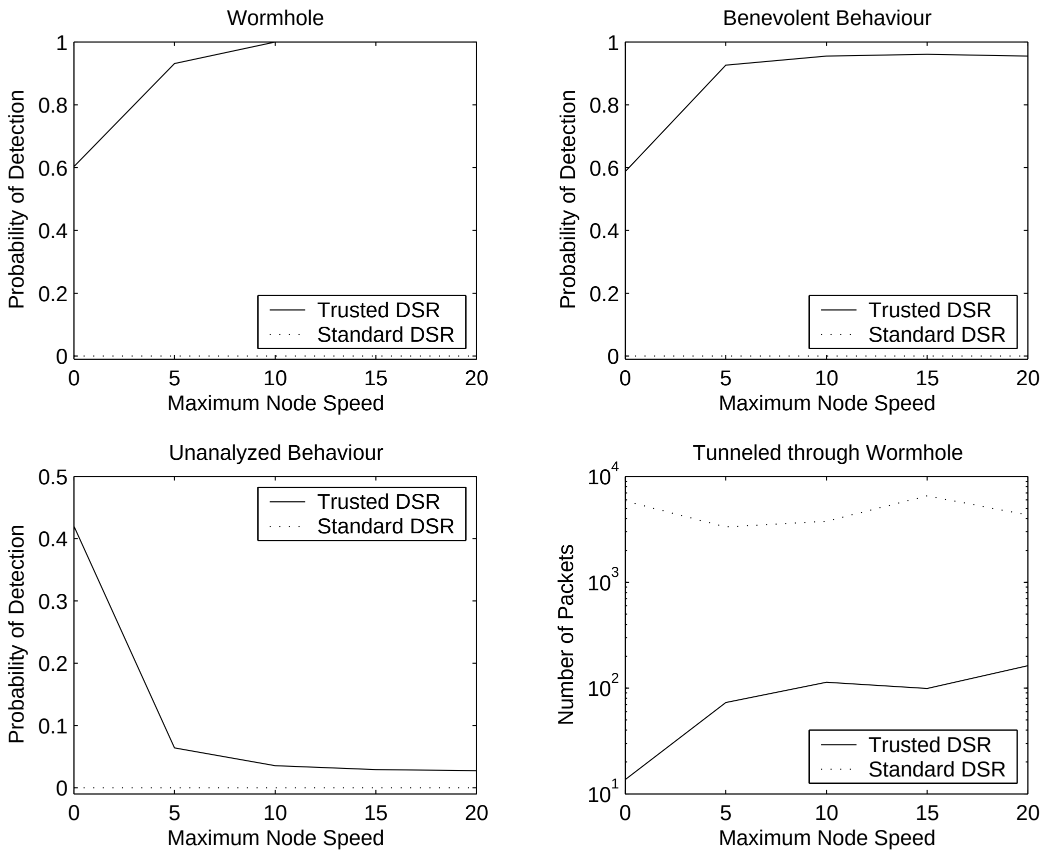

Figure 5: Wormhole detection probabilities and packets tunnelled

lision problem in which a node A cannot hear the broadcast from neighbouring node B to node C, due to a local collision at A. In the receiver collision problem node A overhears node B broadcasting a packet to C but cannot hear the collision which occurs at node C. Similarly, if nodes have varying transmission power ranges, the mechanism of passive trust assignment might not work properly. Such collisions may be carefully planned so as to deceive the trust derivation mechanism. Similarly, communication jamming [22] at the physical layer can create a similar set of problems.

Multiple colluding nodes may also be able to help each other in order to gain higher trust levels, provided they are successively located along the path of the wormhole and data connections. These nodes are capable of portraying shorter routes to nodes using a particular data connection without being detected. These address translations, as being done in series, are not perceivable by the benevolent nodes employing the proposed scheme. The isolation of such malicious nodes, which depict complex behaviour like attacks against multiple layers of the protocol suite, launching attacks in series or by operating at the verge of benevolent and malevolent behaviour, is quite intricate. We recommend using Intrusion Detection systems such as those proposed by Zhang et al. [24] and Kachirski et al. [11] for isolating such nodes in ad-hoc networks.

6 Conclusions

Ad-hoc wireless networks are generally established on the fly in an impromptu manner. Due to the physical vulnerability of both the environment and the nodes, a number of attacks may be undertaken against these networks. A wormhole is one such prominent attack, that is formed by malicious colluding nodes. The detection and evasion of such wormholes in an ad-hoc network is still considered a challenging task. In order to protect from wormholes, current security-based solutions propose the establishment of ad-hoc networks in a controlled manner, often requiring specialised node hardware to facilitate deployment of cryptographic mechanisms. Such solutions, although successful in achieving self organisation during the operation, essentially violate the self organised nature of an ad-hoc network. In this paper, we have deviated from the customary approach of using cryptography and instead employ a trust-based scheme to detect and evade wormholes. In our scheme, we derive trust levels in neighbouring nodes based upon their sincerity in execution of the routing protocol. This derived trust is then used to influence the routing decisions, which in turn guide a node to avoid communication through the wormholes. Through extensive testing, we have established that the trust model can effectively locate dependable routes through the network in the presence of a wormhole in the network. The routes established in this manner may not be the shortest in terms of number of hops, but they definitely contain nodes which have been found more trustworthy than the others. We accentuate that our trust-based scheme will be most suitable for pure ad-hoc networks, which can be rapidly established without entailing any constraining assumptions.

Acknowledgements

This work was supported by the University of Western Australia International Postgraduate Research Scholarship and the University Postgraduate Award. We would also like to thank Diana Senn from the ETH Information Security Group (Swiss Federal Institute of Technology, Zurich) for contributing NS-2 code in support of the simulations.

References

[1] S. Capkun, L. Buttyan, and J. Hubaux, "SECTOR: Secure tracking of node encounters in multi-hop wireless networks," in *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 21-32, 2003.

[2] S. Carter and A. Yasinsac, "Secure position aided ad hoc routing protocol," in *Proceedings of the IASTED Conference on Communications and Computer Networks (CCN)*, pp. 329-334, 2002.

[3] B. Dahill, B. N. Levine, E. Royer, and C. Shields, "A secure routing protocol for ad hoc networks," in *Proceedings of the International Conference on Network Protocols (ICNP)*, pp. 78-87, 2002.

[4] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[5] Y. C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 12-23, 2002.

[6] Y. C. Hu and D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 231-242, 2000.

[7] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1976-1986, 2003.

[8] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 131-141, 2004.

[9] D. B. Johnson, D. A. Maltz, and Y. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," *IETF MANET, Internet Draft (work in progress)*, 2003.

[10] A. Josang, "The right type of trust for distributed systems," in *Proceedings of the ACM New Security Paradigms Workshop*, pp. 119–131, 1996.

[11] O. Kachirski and R. Guha, "Intrusion detection using mobile agents in wireless ad hoc networks," in *Proceedings of the IEEE Workshop on Knowledge Media Networking (KMN)*, pp. 153-158, 2002.

[12] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 255-265, 2000.

[13] NS, *The Network Simulator*, http://www.isi.edu/nsnam/ns/, 1989.

[14] A. Perrig, Y. C. Hu, and D. B. Johnson, *Wormhole Protection in Wireless Ad Hoc Networks*, Technical Report TR01-384, Department of Computer Science, Rice University, 2001.

[15] A. A. Pirzada and C. McDonald, "Secure routing protocols for mobile ad-hoc wireless networks," in *Advanced Wired and Wireless Networks*, T. A. Wysocki, A. Dadej, and B. J. Wysocki, editors, pp. 57–80, Springer, 2004.

[16] A. A. Pirzada and C. McDonald, "Establishing trust in pure ad-hoc networks," in *Proceedings of the 27th Australasian Computer Science Conference (ACSC)*, vol. 26, pp. 47-54, 2004.

[17] A. A. Pirzada and C. McDonald, "Kerberos assisted authentication in mobile ad-hoc networks, in *Proceedings of the 27th Australasian Computer Science Conference (ACSC)*, vol. 26, pp. 41-46, 2004.

[18] A. A. Pirzada, C. McDonald, and A. Datta, "Performance comparison of trust-based reactive routing protocols," *(to appear in) IEEE Transactions on Mobile Computing*, 2006.

[19] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications Magazine*, vol. 6, no. 2, pp. 46–55, 1999.

[20] W. Stallings, *Network Security Essentials*, Prentice Hall, 2000.

[21] W. Wang and B. Bhargava, "Visualization of wormholes in sensor networks," in *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, pp. 51–60, 2004.

[22] C. Ware, T. Wysocki, and J. Chicharo, "Hidden terminal jamming problems in IEEE 802.11 mobile ad hoc networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 1, pp. 261-265, 2001.

[23] W. H. Yuen and R. D. Yates, "Inter-relationships of performance metrics and system parameters in mobile ad hoc networks," in *Proceedings of the IEEE MILCOM*, vol. 1, pp. 519–524, 2002.

[24] Y. Zhang and W. Lee, "Intrusion detection in wireless ad hoc networks," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 275-283, 2000.

Asad Amir Pirzada is presently doing his PhD on trust and security issues in ad-hoc wireless networks at The University of Western Australia. His current research interests include wireless communications, networking, cryptography, real-time programming and data acquisition systems. He holds a BE Avionics from NED University Pakistan, a MSc Computer Science from Preston University USA and a MS Information Security from the National University of Sciences and Technology Pakistan.

Chris McDonald holds a BSc(Hons) and PhD in Computer Science from The University of Western Australia, and currently holds the appointments of senior lecturer in the School of Computer Science & Software Engineering at UWA and adjunct professor in the Department of Computer Science at Dartmouth College, New Hampshire. Chris has recently taught in the areas of computer networking, operating systems, computer & network security, computer architecture, distributed systems programming and, together with these areas, his research interests include network simulation, ad-hoc & mobile networking, programming language implementation, open-source software.