DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks

Hon Sun Chiu and King-Shan Lui
Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, PRC.
Tel: (852) 2859-2692; Email: {hschiu, kslui}@eee.hku.hk

***Abstract* – In mobile ad hoc networks, data transmission is performed within an untrusted wireless environment. Various kinds of attack have been identified and corresponding solutions have been proposed. Wormhole attack is one of the serious attacks which forms a serious threat in the networks, especially against many ad hoc wireless routing protocols and location-based wireless security system. We identify two types of wormhole attacks. In the first type, malicious nodes do not take part in finding routes, meaning that, legitimate nodes do not know their existence. In the second type, malicious nodes do create route advertisements and legitimate nodes are aware of the existence of malicious nodes, just do not know they are malicious. Some researchers have proposed detection mechanisms for the first type. In this paper, we propose an efficient detection method called Delay Per Hop Indication (DelPHI). By observing the delays of different paths to the receiver, the sender is able to detect both kinds of wormhole attacks. This method requires neither synchronized clocks nor special hardware equipped mobile nodes. The performance of DelPHI is justified by simulations.**

***Keywords* – Security, Wormhole, Tunnel, Ad Hoc, Wireless**

I. Introduction

A mobile ad hoc network (MANET) is formed by a group of mobile wireless devices, such as mobile laptop computers, PDAs, and wireless phones, that cooperatively communicate with each other without a fixed network infrastructure [1]. It generally uses a wireless radio communication channel. The advantages of MANET are rapid deployment and low cost of operation. On the other hand, MANET utilizes an untrusted environment for data transmission, and therefore it is subjected to various kinds of security attacks [2, 3].

For example, the blackhole attack refers to an attacker which drops all the traffic passing through it, while white hole attack refers to an attacker floods the network with a large amount of traffic. An attacker can also easily eavesdrop on communication, record packets, and replay the packets in wireless networks. Most of these attacks have been extensively investigated, and the proposed solutions, such as the watchdog and the pathrater [5], provide encouraging results [2-7].

All the attacks mentioned above are preformed by a single attacker. In this paper, we focus on an attack which is launched by a pair of collaborating attackers: wormhole attack [8, 9, 10]. In a wormhole attack, an attacker records packets (or bits from a packet) at one location in the network, tunnels them to a second attacker in another location, and the packets are replayed by the second attacker there. Since the contents of the packets are not modified, wormholes cannot be detected by cryptographic techniques. However, as these two malicious nodes are acting as neighbors to other nodes, hiding the fact they are in fact several hops away by tunneling, this attack imposes severe threats to ad hoc network routing protocols. For example, in AODV, the path with smallest hop count is selected. Since the malicious nodes are acting as neighbors, the AODV routing protocol would get wrong hop count information and select an inappropriate path. Malicious nodes can also lure other nodes to send traffic through them by advertising apparently short paths so as to launch other attacks to the data packets.

Some mechanisms have been developed to detect wormhole attacks. Hu et al. proposed in [8] to put information in a packet to restrict the transmission distance of the packet so as to avoid tunneling. The authors called the information packet leash and they proposed two types of packet leashes: geographical leash and temporal leash. In the geographical leashes, the location information and loosely synchronized clocks together verify the neighbor relation. In the temporal leashes, the packet transmission distance is calculated as the product of signal propagation time and the speed of light.

There are also some other methods proposed to defense against wormhole attacks. In [6], distributed Network Monitors were developed to monitor the control messages of the AODV routing protocol, and observe whether the behavior violates the "correct behavior" captured by the specifications. This "correct behavior" is pre-defined in the monitors and is manually configured.

[10] studied how to enhance the security of routing protocol in ad hoc networks. The defense mechanism simply uses the fastest path, instead of using the path with smallest hop count. It can prevent wormhole attack with actual path length longer than the false hop count produced by the malicious pair. However, it is not a detection mechanism.

The mechanism developed in [11], called SECTOR, assumes each node is equipped with a special hardware that can respond to a one-bit challenge without any delay. The challenger measures the round-trip-time (RTT) of the signal with an accurate clock to calculate the distance between the nodes. The probability that an attacker can guess all bits correctly decreases exponentially as the number of challenges increases.

In [12], similar to SECTOR, per-hop RTT is used for the detection of a wormhole attack. Whenever a node receives a route request message, before forwarding, it will send a verification message to the pervious node and wait for the reply. The request is forwarded only if the RTT is approved.

In this paper, we present a more efficient method in detecting wormhole attacks – Delay Per Hop Indication (DelPHI) Wormhole Detection. DelPHI allows the sender to

0-7803-9410-0/06/$20.00 ©2006 IEEE.

check whether there are any malicious nodes sitting along its paths to the receiver and trying to launch wormhole attacks. We obtain the delay and the hop count information of some disjoint paths between the sender and the receiver and use this information to indicate whether a certain path among these disjoint paths is subjected to wormhole attacks. The advantages of DelPHI are that it does not require clock synchronization and position information, and it does not require the mobile nodes to be equipped with some special hardware, which in turns provides higher power efficiency.

The remainder of the paper is organized as follows. We first present and compare two kinds of wormhole attacks in the next section. In Section III, we present our DelPHI detection mechanism. The performance of DelPHI is evaluated by simulations in Section IV. In Section V, we address the message overhead issue and finally, we conclude the paper in Section VI.

II. Two Kinds Of Wormhole Attacks

In a wormhole attack, two attackers work together. One receives the packets, tunnels the packets to its partner, and then the partner replays them into the network. There are two kinds of wormhole attacks. In the first type, malicious nodes hide the fact that they forward a packet, meaning that, legitimate nodes do not know their participation in packet forwarding. In the second type, legitimate nodes are aware of the fact that the malicious nodes are forwarding packets, just do not know they are malicious. For the ease of discussion, we refer the first type as *hidden attack* while the second type as *exposed attack*.

Hidden Attack – The attackers do not modify the content of the packet and the packet header, even the packet is an AODV advertisement packet. Instead, they simply tunnel the packet from one point and replay it at another point. This kind of wormhole attacks makes the sender treat the receiver as its immediate neighbor. Suppose that S wants to establish a route to R using AODV. S would broadcast a RREQ message. Any node that receives the RREQ should check whether it knows how to get to R. If not, it should continue to broadcast RREQ if it receives RREQ for the first time. It should also update the hop count information and put its identity in the packet header. However, in the hidden attack, malicious nodes do not update the packet header as it should. As shown in Fig. 1(a), the packet from S is received by M1, then M1 tunnels the packets to M2 and replays them to R, without modifying the packet header. Since M1 and M2 do not include themselves in the header, what R will find is that the pervious hop is S. The same observation can be obtained in the reverse path, such that S finds R as its immediate neighbor, and the path found is {S, R}. This is obviously not correct since S and R are separated by M1, M2, and other nodes that are in the tunnel.

Exposed Attack – In this kind of attacks, the attackers do not modify the content of the packet, but include themselves in the packet header following the route setup procedure. Other nodes are aware that the malicious nodes lie on the path but they would think that the malicious nodes are direct

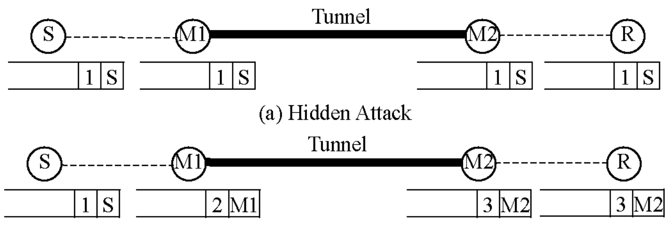


Fig. 1. Two types of wormhole attacks

neighbors. Let's consider the situation where S wants to establish a route to R. As illustrated in Fig. 1(b), when M1 receives the packet, it modifies the pervious hop field to M1 and increases the hop count by 1. Then the RREQ packet is tunneled to M2 and M2 performs the same setup procedure and broadcasts the RREQ packet to R. R finds its previous hop is M2 with hop count equals to 3. The same thing happens in the reverse path. When S receives the RREP packet, it finds its pervious hop is M1 with hop count equals to 3. And the route is setup as {S, M1, M2, R}.

In both kinds of attacks, there is at least one pair of "neighbors" that are actually not direct neighbors. In Fig. 1(a), S and R perceive themselves as neighbors but they are not. We call this kind of neighbors "false neighbors." In Fig. 1(b), M1 and M2 are false neighbors. Since neighbors should be within transmission range with each other, if we are able to know the distance between neighbors and find that the distance between two neighbors are out of range, we can tell whether a wormhole attack occurs.

Packet leashes [8] provide solution to the hidden attack based on this observation. The main idea behind packet leashes is to limit the transmission distance to one hop. In the first case in Fig. 1(a), S and R treat themselves as neighbor, but in reality, they are a few hops away, therefore wormhole can be detected by packet leashes in R. However, it is not the case in the exposed attack in Fig. 1(b). S knows M1 is its neighbor, and R knows M2 is its neighbor, both transmission distances are found to be within one hop, i.e. {S, M1} and {M2, R}. Hence wormhole is not detected.

Similar to packet leashes, the main idea of SECTOR [11] and the mechanism in [12] are to calculate the upper bound on the distance within one hop. Thus they also cannot provide solution to the exposed wormhole attack problem.

The mechanisms proposed in [6, 10] are able to tackle both the hidden and exposed wormhole attacks. More specifically, [6] monitors the network behavior with reference to the pre-defined specifications. Since both attacks produce message flows that are different from the specifications, the monitors can detect them. [10] always chooses the fastest path with the reason that the paths under wormhole attacks must not be the fastest.

Although the mechanisms [6, 10] can deal with both kinds of wormhole attacks, there are still some drawbacks regarding to these mechanisms. For example, [6] requires a set of monitors to have pre-defined specification. These monitors must be placed carefully to cover the whole network. On the other hand, these monitors require manual configuration. This is not suitable for dynamic networks. It also suffers from the single point of failure problem. The mechanism in [10] also has some disadvantages. The mechanism tries to avoid the use of the wormhole-attacked paths, but it is not a wormhole detection mechanism. Consider the case that some paths are congested such that the wormhole is the fastest path, wormhole is still selected.

To avoid using synchronized clocks, positioning device and special hardwares, but having the ability of detecting both the hidden and exposed attacks, we devise DelPHI, which will be described in detail in the following section.

III. Delay Per Hop Indication (DELPHI) Detection Mechanism

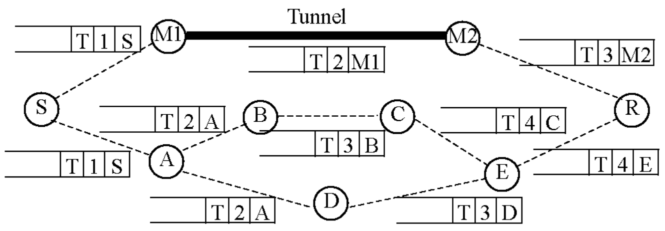
In our DelPHI wormhole detection presented in this paper, we collect both hop count and delay information of disjoint paths and calculate the delay/hop value to serve as the indicator of detecting wormhole attacks, which provides a general solution for both kinds of wormhole attacks. The reason behind is that under normal situation, the delay a packet experiences in propagating one hop should be similar along each hop along the path. However, under a wormhole attack, the delay for propagating across false neighbors should be unreasonably high since there are in fact many hops between them. Therefore, if we compare the delay per hop of a legitimate path with the delay per hop of a path that is under wormhole attack, we should find that the delay/hop of the legitimate path is smaller. Therefore, if a path has a distinguishable high delay/hop value, it is likely to be subjected to a wormhole attack.

To avoid the need of synchronized clocks, positioning device and other special hardwares, DelPHI collects information and performs detection at the sender. DelPHI obtains delay and hop count information in a way similar to the AODV route setup mechanism [13]. When the detection is initiated, the sender broadcasts a request message to the receiver, and the receiver replies all the request messages received. In this way, the sender can obtain the information of some disjoint paths to the receiver. By comparing the delay/hop values among these disjoint paths, a wormhole can be identified.

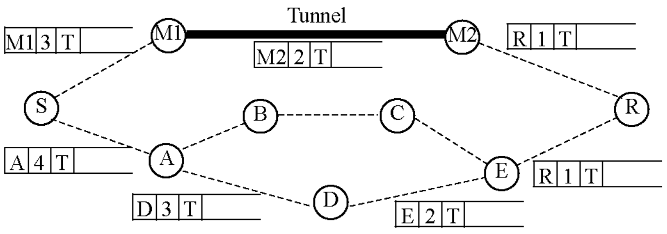
There are two phases in our mechanism. In the first phase, delay and hop count information is collected. In the second phase, the sender analyzes the information obtained in the first phase to detect whether there is any wormhole attack.

A. First Phase: Data Collection

In this phase, the sender initiates the detection and collects information. There are two kinds of messages: DelPHI Request (DREQ) and DelPHI Reply (DREP). Similar to the AODV RREQ and RREP packets, DREQ is used for the sender to find disjoint paths to the receiver, while DREP is sent from the receiver back to the sender to identify paths. Both DREQ and DREP packets include a pervious hop field, hop count field, and a timestamp field. We use *pervious hop*



(a) DREQ roadmap



(b) DREP roadmap

Fig. 2. Two possible disjoint paths

but not the *whole path information* simply because of saving the network resources. If the path is long, then the packet will be large.

When the sender initiates DelPHI wormhole detection, it broadcasts a DREQ packet to the receiver, which is illustrated in Fig. 2(a). The previous hop field is filled with the sender's node ID, the hop count field is set to 1, and the timestamp field is set with the time when the packet is sent. The previous hop field and the hop count field will be modified by intermediate nodes while the timestamp field is never changed by other nodes, even the receiver. Therefore, the sender should protect the integrity of the timestamp. This can be achieved by signing the message authentication code of the timestamp.

When an intermediate node receives a DREQ packet, it records the previous hop field and establishes a reverse path to the sender. Then it puts its node ID into the pervious hop field and increases the hop count field by 1. The resulted DREQ packet is then broadcasted.

The forwarding of DREQ is somewhat similar to the AODV RREQ forwarding. Any node in the network broadcasts DREQ received and sets up a reverse path when it receives the packet in the first time. When the same packet is received at the second time, it can be simply dropped. Unlike the AODV route setup, a node must forward the DREQ no matter there is a record in its routing table or not, until the packet reaches the receiver. To secure this procedure, the sender and receiver can sign the packet such that no intermediate node can impersonate the receiver to reply a DREQ message.

When the receiver gets a DREQ, it unicasts a DREP packet to the sender through the reverse path, and is illustrated in Fig. 2(b). It puts its node ID in the pervious hop field, sets the hop count field to 1, and copies the timestamp field of the DREQ packet to the DREP packet. Similar to the request procedure, an intermediate node puts its node ID into the pervious hop field and increases the hop count field by 1 upon receiving the DREP packet. Every intermediate node only forwards the DREP packet once for each corresponding DREQ.

Noted that the receiver replies to every DREQ packet received (compare with AODV, receiver only replies to the first RREQ received), and each node only broadcasts the DREQ packet once. Hence the sender can receive a number of DREP packets where each of them follows a path which is disjoint from the paths of other DREP packets. In other words, the DREP packets collect information of a set of disjoint paths from the sender to the receiver. As shown in Fig. 2(b), S can receive 2 DREP packets, one from M1 and one from A.

Each DREP carries the hop count information of the path that it is associated with. It also carries the timestamp of the time that the sender sent the corresponding DREQ. Therefore, the round trip time of the path is the time difference between the time at which the sender receives DREP and the timestamp carried in DREP. Then, the sender is able to calculate the delay/hop value of the corresponding path.

DREQ and DREP packets could be lost. To enhance reliability of the information collected, the data collection procedure is repeated 3 times. It is possible that the hop counts of the three DREPs received from the same neighbor are different. In this case, we select the delay/hop of the shortest path for analysis. It is because a path that is under wormhole attack tends to be shorter. Among all the shortest path DREPs, we take the average of the delays for wormhole detection. For example, again refer to Fig. 2, that during the second broadcast, E receives a DREQ from C prior than D, then the path formed becomes {S, A, B, C, E, R}, and S receives a DREP from A with hop count set to 5. Knowing that there is a path to R through A is 4 hops away, the 5-hop information is ignored. Similarly, if the first broadcast obtains the 5-hop information while the 4-hop information is obtained in a later broadcast, then the 5-hop record is deleted and updated by the 4-hop data. If there are two trials of 4-hop and one trial of 5-hop, we take the average of the two 4-hop trials for phase 2.

To distinguish the DREQs of the three different trials, we have to put some identifiers in the packet headers. As there are many standard mechanisms to do that and is not the focus of this paper, we leave the details to the readers and describe the detection in details in the following.

B. Second Phase: Data Analysis and Detection

Suppose that the sender initiates the detection, i.e. broadcasts the DREQ packet, at time t_s, and receives a DREP packet from a neighbor node i at time t_i, then the round trip time (*RTT*) of the path through node i is given by $RTT_i = t_i - t_s$. If the hop count field in the DREP from node i is h_i, then the delay/hop value (*DPH*) of the path to the receiver through node i is given by

$$DPH_i = \frac{RTT_i}{2h_i} = \frac{t_i - t_s}{2h_i} \qquad (1)$$

In normal situation, a smaller h provides a smaller value of

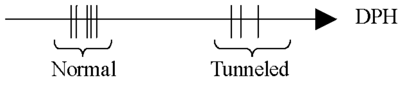


Fig. 3. Relationship of normal and tunneled paths

RTT. It can be explained by the fact that a shorter path should have a smaller round trip time. Hence the *DPHs* of normal paths should have similar values independent to h.

However, it is not the case in the paths which suffer from wormhole attacks. Recall that a tunnel is formed by two malicious nodes. No matter how long the tunnel is, the malicious pair M1 and M2 advertise to others that they are 1 hop away. Therefore, the longer the tunnel, the larger the *RTT*, but the hop count remains the same. The resulted *DPH* value will be larger than normal path.

We performed some simulations and observed that the *DPH* values of normal paths usually appear as small values when compared with those of tunneled paths. It can be easily observed that the *DPH* values of normal and tunneled paths form two separate groups as shown in Fig. 3. The difference between "*the smallest DPH in the tunneled group*" and "*the largest DPH in the normal group*" is always larger than the gap between any 2 *DPH* values within the same group.

Therefore, to identify a wormhole attack, we arrange the *DPH* values in descending order, and find whether there is a large difference between 2 adjacent values. If DPH_i is larger than the next *DPH* value by a *Threshold* (*T*), then the path through node i and all other paths with *DPH* values larger than DPH_i are treated as under wormhole attack.

As our detection mechanism is based on the distinguishable difference of *DPH* values between normal paths and tunneled paths, DelPHI does not work well when all the paths are tunneled. How to enhance DelPHI to work also in the situation when all the paths are attacked is left for future work.

IV. Performance Evaluations

In this section, the performance of DelPHI is evaluated by simulation using the LBNL network simulator *ns* [14]. Random topologies with N nodes and size LxL are randomly generated by a random generator provided by *ns*. Sender (S), receiver (R) and malicious pair (M1 and M2) are put in the corresponding places as shown in Fig. 4, e.g. S is randomly put in square A with size 100x100 in the lower left hand corner. In the simulation, the malicious nodes are not necessarily to be the neighbor of the sender and receiver.

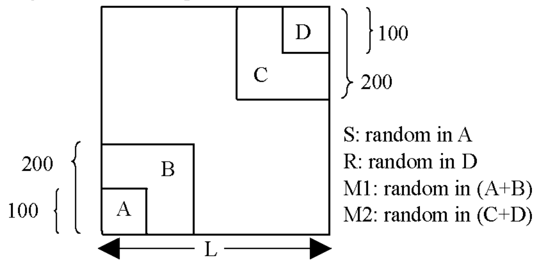


Fig. 4. Simulation topology

Table 1. Detection rate with different threshold (L=1000)

Threshold (T) /ms	Normal path /%	Tunneled path /%
5	98	89
3	95.6	93.7
2	91.5	95.8
1	85.8	97.6

Table 2 Detection rate with different threshold (L=750)

Threshold (T) /ms	Normal path /%	Tunneled path /%
5	96.9	57
3	93.7	78.6
2	90.2	86.2
1	83.8	90.5

Table 3 Detection rate with different threshold (L=500)

Threshold (T) /ms	Normal path /%	Tunneled path /%
5	99	10.4
3	96.3	30.7
2	88.3	47.96
1	81.9	65

We have performed a number of simulations, and due to space limitation, we only present some representative results in this paper. The settings of these simulations are as follows: (a) L = 1000, N = 50; (b) L = 750, N = 30; and (c) L = 500, N = 15. Please note that a smaller network has a shorter tunnel.

For each setting, we generated 1000 different networks with random node placement. In these 1000 networks, we ignored those where all the paths are tunneled. We then measured the detection rate over these remaining networks, which are more than 900 topologies for each setup.

We first conducted simulations with different settings against different threshold values (T), the results are shown in Tables 1-3. Column "Normal path" refers to the percentage of normal paths (paths that are not attacked) that are detected correctly. Column "Tunneled path" is defined in a similar way. As expected, the smaller the threshold (T) is, the easier to detect wormhole attack. However, it also leads to a higher rate of treating a normal path as a tunneled path if no path is under wormhole attack. Since smaller L has a larger chance that the tunnel is short, the DPH values of tunneled paths and normal paths become comparably close, which in turns lower the detection rate (will be explained by further simulations later). In order to maintain the detection rate of normal paths above 90% and that of wormhole paths should be remained as high as possible, we set T = 3ms in the following simulations, which evaluate the performance of DelPHI in the absence of background traffic, and study the effect of background traffic on DelPHI.

Fig. 5 shows the simulation results when there is no background traffic. We started our simulation with tunnel length set to 2 hops. This is simply because 1 hop is not regarded as a tunnel. Here, the tunnel length stands for the hop count from M1 to M2. It is found that the longer the tunnel length, the higher the detection rate of wormhole attack. And the detection rate of normal path is independent to the tunnel length. This phenomenon can be explained by equation 1 in Section III(B). When the path is under wormhole attack, no matter how long the tunnel is, it always treats it as 1 hop, hence h remains small. On the other hand, a longer tunnel gives a larger RTT. Therefore, the longer the tunnel, the larger the DPH, and the gap between normal path and tunneled path will be more obvious. If the tunnel length is small, the DPH will be similar to that of a normal path, and it is the reason why short tunnel length leads to a low detection rate.

In the case of normal path, as explained in Section III(B), RTT is directly proportional to h. Therefore the DPH value is independent to the tunnel length, and it gives the reason why the detection rate stays at the same level throughout the simulation.

Another set of simulations were conducted with background traffic. The background traffic was set in the following way: Connections are setup in randomly chosen non-overlapping node pairs, i.e. there is only one-to-one transmission, no node is a sender of more than 1 connection and no node is a receiver of more than 1 connection.

Fig. 6 presents the result when there is heavy background traffic. We define *heavy* as all nodes are having connections, i.e. the number of connections equals to 50% of the number of nodes. It is found that the detection rate of tunneled path is much higher than that with no background traffic. And the detection rate of normal path is only slightly decreased.

It can be explained as follows: Due to the claimed small hop count through the wormhole, the traffic will choose this path as their shortest path, which leads to heavy congestion in the tunnel. Therefore, the DPH value through the tunneled path is dramatically increased. The difference between the DPH of tunneled path and that of normal path becomes more obvious and larger than T. Therefore the detection rate of

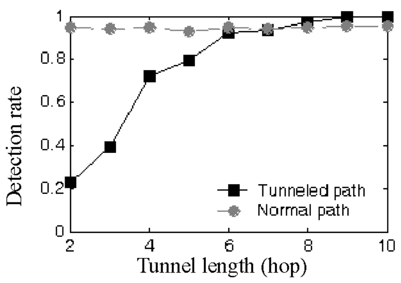


Fig. 5. No background traffic

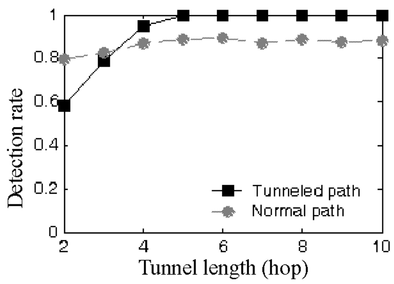


Fig. 6. Heavy background traffic

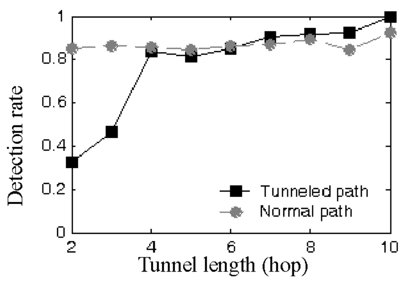


Fig. 7. Light background traffic

wormhole attack is higher. For the normal paths, since there is background traffic, the *DPH* value is also increased. When no path is under wormhole attack, a gap may appears because of the delay due to background traffic. Hence, the detection rate is slightly decreased, as shown in Fig. 6.

Finally, we conducted simulations with light background traffic and the result is shown in Fig. 7. In this simulation, we randomly choose connection pairs with the number of connections equals to 10% of the number of nodes. Again, the detection rate of wormhole is higher than that with no background traffic, and the detection rate of normal path is similar to that with heavy background traffic. It can be explained by the same reason in the last simulation with heavy background traffic.

V. Discussion

In this section, the overhead of DelPHI is addressed. Let the number of nodes be N, the number of disjoint paths be P, and the number of hops in path i be h_i.

Consider in each request, every node (except the receiver) broadcasts a DREQ packet once, there are totally N-1 request packets transmitted in the network. The number of DREQ packets that the receiver can receive is equals to the number of disjoint paths P. In DelPHI, P is bounded by the number of neighbors of the sender and the number of neighbors of the receiver. Therefore, $P << N$. Since the receiver replies to all DREQ packets, the number of DREP packet is given by

$$\sum_{i=1}^{P} h_i \tag{2}$$

Noted that the detection procedure consists of 3 requests, therefore the total message overhead is give by

$$3 \cdot \left(N - 1 + \sum_{i=1}^{P} h_i \right) \tag{3}$$

AODV is chosen to provide comparison because the route setup procedure is similar to DelPHI. For AODV route setup, since the sender only broadcast the RREQ once, and the receiver only replies to one RREP, therefore the message overhead is given by $N - 1 + h$.

If we set $N = 50$, by simulation, we find that E[P]=3.19470 and E[h]=4.29055. Hence, the message overhead of DelPHI is 188.12106, while that of AODV route setup is 53.29055. The major factor is the triple request procedures in providing reliability. There is a tradeoff between providing reliability and minimizing the message overhead.

VI. Conclusion

In this paper, we have described an efficient algorithm for detecting wormhole attack in mobile ad hoc networks. We call it Delay Per Hop Indication (DelPHI). The advantages of DelPHI are that it does not require clock synchronization and position information, and it does not require the mobile nodes to be equipped with some special hardwares, thus it provides higher power efficiency. The performance of DelPHI has been evaluated by conducting various simulations using the *ns* simulator. It has been shown that DelPHI can achieve higher than 95% in detecting normal path and 90% in detecting wormhole attack, in the absence of background traffic. Simulations has also shown that DelPHI can maintain above 85% detection rate for both normal and tunneled paths given that there is background traffic.

The message overhead of DelPHI has also been addressed in this paper. We compared it with AODV route setup procedures and found that the major factor is the triple request procedures in providing reliability. There is a tradeoff between providing reliability of DelPHI and minimizing the message overhead, and may need further investigation.

References

[1] S. Corson and J. Macker, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," *RFC2501*, January 1999.

[2] L. Buttyan and J. P. Hubaux, "Report on a Working Session on Security in Wireless Ad Hoc Networks," *Mobile Computing and Communications Review*," vol. 7, no. 1, Jan. 2003, pp. 74 – 94.

[3] Y. Hu and A. Perrig, "A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security & Privacy*, May/June 2004, pp. 28 – 39.

[4] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad-Hoc Networks," *MobiCom'2000*, Boston, Massachusetts, Aug. 6-11, 2000, pp. 275 – 283.

[5] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *MobiCom'2000*, Boston, Massachusetts, Aug. 6-11, 2000, pp. 255 – 265.

[6] C. Y. Tseng, P. Balasubramanyam, C. Ko, R. Limiprasittiporn, J. Rowe, and K. Levitt, "A Specification-based Intrusion Detection System for AODV," *Proc. of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks*, Fairfax, Virginia, 2003, pp. 125 – 134.

[7] Y. Huang and W. Lee, "A Cooperative Intrusion Detection System for Ad Hoc Networks," *Proc. of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks*, Fairfax, Virginia, 2003, pp. 135 – 147.

[8] Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," *Proc. of INFOCOM'2003*, April 2003, pp. 1976 – 1986.

[9] P. Papadimitrators and Z. J. Haas, "Secure Routing for Mobile Ad Hoc Networks," *Proc. of CNDS*, San Antonio, TX, Jan. 27-31 2002.

[10] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A Secure Routing Protocol for Ad hoc Networks," *Proc. ICNP*, 2002.

[11] S. Capkun, L. Buttyan, and J. Hubaux, "SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks," *Proc. of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003, pp. 21 – 32.

[12] J. Zhen and S. Srinivas, "Preventing Replay Attacks for Secure Routing in Ad Hoc Networks," *ADHOC-NOW 2003*, Montreal, Canada, Oct 8-10, 2003, pp. 140-150.

[13] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, Feb. 1999, pp. 90 – 100.

[14] ns: UCB/LBNL/VINT Network Simulator – ns (version 2), http://www-mash.cs.berkeley.edu/ns/