DelPHIX: A Simple and Efficient Mechanism for Wormhole Detection in Ad Hoc Networks

Hon Sun Chiu, King-Shan Lui and Kwan L. Yeung

Abstract – **Data transmission in a mobile ad hoc network is performed within an untrusted wireless environment. It is subjected to many kinds of security attacks. In a wormhole attack, two malicious nodes work together to tunnel packets from one to the other, making other nodes perceive a path to have a smaller hop count. We identify two types of wormhole attacks. In the first type, malicious nodes do not expose themselves in route finding process and 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. Existing solutions usually can identify only one kind of attacks. In this paper, we propose a simple and efficient detection method called DelPHIX. 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 to be equipped in mobile nodes. The performance of DelPHIX 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. MANET is especially attractive for scenarios where it is infeasible or expensive to deploy significant networking infrastructure. Most previous research has focused on the realization and practical implementation of MANET, in terms of data aggregation protocols and routing protocols, with the assumption that MANET works in a trusted environment.

However, 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 dropping all the traffic passing through it, while white hole attack refers to an attacker flooding 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 [4], provide encouraging results [2-7].

All the attacks mentioned above have a common feature – they are preformed by a single attacker. In this paper, we focus on a different attack that is launched by a pair of collaborating attackers: wormhole attack [8-10]. 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 second attacker replays the packets 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, and 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 the Ad hoc On Demand Distance Vector (AODV) routing mechanism [17], 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.

We identify two types of wormhole attacks: *Hidden Attack* and *Exposed Attack*. In a hidden attack, malicious nodes do not take part in finding routes, that is, legitimate nodes do not know their existence. On the other hand, in an exposed attack, malicious nodes do create route advertisements and legitimate nodes are aware of the existence of malicious nodes, just do not know they are malicious. In the literature, most of the proposed protocols, with additional hardware or synchronized clocks, can only identify the hidden attack, but not the exposed attack.

In this paper, we present an efficient method in detecting both kinds of wormhole attacks – DelPHIX, an extension to our earlier work Delay Per Hop Indication (DelPHI) Wormhole Detection [16]. DelPHIX allows the sender to check whether there are any malicious nodes sitting along its paths to the receiver that are 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 collect the average round trip delay of immediate neighbors, then use this information to indicate whether a certain path among these disjoint paths is subjected to wormhole attacks. The

Manuscript received January 25, 2006 and revised May 04, 2006. This research was supported in part by the Seed Funding established by the University of Hong Kong. This paper was presented in part at the International Symposium on Wireless Pervasive Computing (ISWPC) 2006.

Authors are with the Department of Electrical and Electronic Engineering, the University of Hong Kong, Pokfulam Rd, Hong Kong, (e-mail: hschiu@eee.hku.hk; kslui@eee.hku.hk; kyeung@eee.hku.hk).

1845-6421/05/5180

advantages of DelPHIX are that it neither requires clock synchronization nor position information. Moreover, 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. Then some related works are presented in Section III. In Section IV, we present our DelPHIX detection mechanism. The performance of DelPHIX is evaluated by simulations in Section V. In Section VI, we address the message overhead issue and finally, we conclude the paper with the future work direction in Section VII.

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. In a hidden attack, malicious nodes hide the fact that they forward a packet and legitimate nodes do not know their participation in packet forwarding. In an exposed attack, legitimate nodes are aware of the fact that the malicious nodes are forwarding packets, just do not know they are malicious.

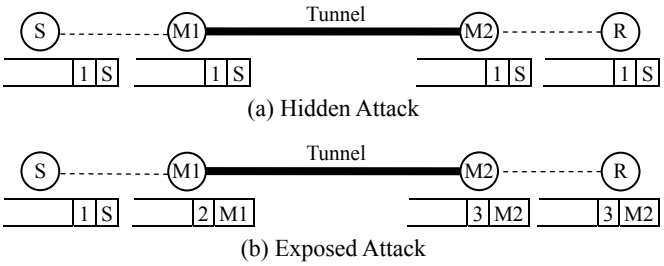

Fig. 1. Two types of wormhole attacks

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 (route request) 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. The intermediate nodes 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), when S initiates a route setup procedure, it broadcast a RREQ packet to the network, with *previous_ node_id* = S and *hop_count* = 1 in the header. When M1 receives the RREQ, it tunnels the packets to M2 and M2 replays them to R, without modifying the packet header. Since M1 and M2 do not include themselves in the header, R will think the received RREQ was sent from S that is 1 hop away. Then, R mistakenly treats S as its immediate neighbor. 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 also do not modify the content of the packet, but they expose themselves in the route finding process. Other nodes are aware that the malicious nodes lie on the path but they would think that the malicious nodes are direct neighbors. Let's consider the situation where S wants to establish a route to R. As illustrated in Fig. 1(b), when S initiates a route setup procedure, it broadcasts a RREQ packet to the network, with *previous_node_id* = S and *hop_count* = 1 in the header. When M1 receives the packet, it modifies the *previous_node_id* field to M1 and increases the *hop_count* by 1, just like the normal AODV operations. Then the RREQ packet is tunneled to M2 and M2 performs the same route setup procedure and broadcasts the RREQ packet to R. R finds that it is 3 hops away from S through neighbor M2. The same thing happens in the reverse path. When S receives the RREP packet, it finds that through M1, there are 3 hops to R. And the route is formed 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. Please note that a hidden attack can usually generate more false neighbor pairs than an exposed attack. It is because any node in the neighborhood of M1 can be a false neighbor of any node in the neighborhood of M2 in a hidden attack. On the other hand, there is only a pair of false neighbor, M1 and M2, in an exposed attack. 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.

III. Related Works

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 leash, the location information and loosely synchronized clocks together verify the neighbor relation. In the temporal leash, the packet transmission distance is calculated as the product of signal propagation time and the speed of light. The main idea behind packet leashes is to limit the single hop transmission distance. However, this approach can only identify the hidden attack but not the exposed attack. As shown in Fig. 1(a), S and R treat themselves as neighbors, but in reality, they are a few hops away, and thus wormhole can be detected by packet leashes. 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 the transmission distances of {S, M1} and {M2, R} are found to be within one hop. Definitely, M1 and M2 will not tell others that they are in fact out of range. Hence the wormhole is not detected.

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 neighboring nodes. The probability that an attacker can guess

all bits correctly decreases exponentially as the number of challenges increases. Instead of limiting the packet transmission distance of a single hop, the idea of SECTOR is to use accurate RTT for calculating the distance with a certain neighbor, and compare the distance with the transmission range. Thus it also cannot provide solution to the exposed wormhole attack problem.

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. With similar reason, this method can only solve the hidden attack but not the exposed attack. Moreover, this method adds extra delay in the route setup procedure.

Another method called NVP is proposed in [13] for replay-based attacks. NVP considers both RTT-based approach and power-based approach for calculating the transmission range. By combining the two approaches, the authors claim that the transmitting power to receiving power ratio is a function of round trip propagation delay. If the transmission is passing through an intruder, then the power values are altered. Intruder cannot find a suitable transmission power to suit the function, as it does not know the required receiving power strength and the distance between the sender and the receiver. NVP works well in hidden attack as it is able to find false neighbors in the network. However, in an exposed attack, the malicious nodes would not tell others the fact that they are false neighbors. Thus NVP fails in the detection.

SAM [14, 15] is a detection mechanism with the ability to detect exposed attacks. However, it fails in detecting hidden attacks. SAM employs a multi-path routing mechanism that multiple non-disjoint paths between sender and receiver are found. The main idea is that the tunneled link between the two attackers will be selected with higher probability than other links due to the smaller hop count than other routes. The receiver gathers the multi-path information from the route request messages and calculates the frequency of the selected links. The frequency of tunneled link should be much higher than that of normal links. However, this mechanism may fail to detect hidden attacks since there may be many false neighbors, making the detection inaccurate. No simulation results about detection rate were reported in the papers.

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. Although Network Monitors can handle both the hidden and exposed attacks, there are still some drawbacks with this mechanism. The monitors must be placed carefully to cover the whole network. In addition, these monitors require manual configuration. This is not suitable for dynamic networks. It also suffers from the single point of failure problem.

[10] studies 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 and cannot tell whether there is a wormhole attack in the network or not.

The mechanisms [6, 8, 11-13] either use special hardware, location information and/or synchronized clocks, increasing the cost, complexity and power consumption of the wireless nodes. Still, [8, 11-13] cannot provide solution in detecting the exposed wormhole attacks. Although [6] provides solutions to both kinds of wormhole attacks, it is not suitable for dynamic networks.

Our approach DelPHIX is different from [6, 8, 11-13] in the way that it does not require synchronized clocks, positioning device and special hardware. In other words, DelPHIX is a simpler approach, with lower cost and power consumption. More importantly, DelPHIX is able to detect both the hidden and exposed wormhole attacks.

DelPHIX is similar to SAM [14, 15] as it also gets rid of using synchronized clocks, positioning device and special hardware. However, there are many differences. First, although both DelPHIX and SAM collect information of multiple paths between sender and receiver, DelPHIX only requires disjoint path information while SAM requires non-disjoint paths as well. Moreover, SAM requires the hop-by-hop information of whole path is known but DelPHIX does not. In other words, the overhead in collecting path information of DelPHIX is smaller. On the other hand, SAM is executed by the receiver, while DelPHIX is executed by the sender. Therefore, DelPHIX is less vulnerable to denial-of-service attacks. Another main difference is that, SAM counts "links" from the multi-path information, and uses the frequency of the appeared link as the detection criterion. This is a fatal weakness in detecting hidden wormhole attack, as the "tunneled link" is hidden by the malicious nodes, and the "high frequency" is spread out by their neighbors. Instead, DelPHIX utilizes the whole path information for detection, and therefore it can detect both kinds of wormhole attacks.

We developed a wormhole detection mechanism called DelPHI earlier [16]. However, DelPHI works only when there is at least one normal path and under some network traffic assumptions. We fix some of the problems of DelPHI in DelPHIX and detailed detection mechanism of DelPHIX will be described in the next section. In the rest of our discussion, we refer a path that is under wormhole attack as a "wormhole path".

IV. DelPHIX Detection Mechanism

In our DelPHIX wormhole detection, 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. Hence the delay/hop values should also be similar for all the normal paths. 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. It also leads to an increase in the delay/hop value of the path. Therefore, if we compare the delay/hop value of a

legitimate path with the delay/hop value of a wormhole path, we should find that the delay/hop value of the legitimate path is smaller. In addition, if a path has a distinguishable high delay/hop value, it is likely to be subjected to a wormhole attack.

However, a congested legitimate path may have a high delay/hop value. In order to further reduce false detection that mistakenly treats a congested normal path as a wormhole path, we also use the average round trip time between the sender and its neighbors as a reference. It is expected that the delay/hop value of a legitimate path is also similar to the time needed to traverse from the sender to its neighbors. Therefore, if the delay/hop of a path is larger than the average round trip delay of immediate neighbors (multiplied by some factors), it is very possible that the path is under a wormhole attack.

To avoid the need of synchronized clocks, positioning device and other special hardware, DelPHIX collects information and performs detection at the sender. DelPHIX obtains delay and hop count information in a way similar to the AODV route setup mechanism [17]. When the detection is initiated, the sender broadcasts a request message to the receiver. Every intermediate node re-broadcasts the request message if it is the first time to the message; otherwise, simply drops it. In AODV, intermediate node replies the request message if it has a valid route to the receiver in its routing table. DelPHIX is different and requires all intermediate nodes to broadcast the request message without checking the routing table. Another different from AODV is that DelPHIX allows the receiver to reply all the request messages received, while receiver using AODV only replies to the first received request message. In this way, a DelPHIX sender can obtain the information of some disjoint paths to the receiver. By comparing the delay/hop values among these disjoint paths, paths under wormhole attack can be identified.

DelPHIX has two phases. 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 for detecting whether there is any wormhole attack.

A. First Phase: Data Collection

In this phase, the sender initiates the detection and collects information. Two kinds of messages are introduced: DelPHIX Request (DREQ) and DelPHIX 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_node_id* field, a *hop_count* field, and a *timestamp* field. We use *pervious_node_id* but not the *whole path information* simply because of saving the network resources. If the path is long, then the packet with whole path information will be large.

When the sender initiates DelPHIX wormhole detection, it broadcasts a DREQ packet to the receiver, which is illustrated in Fig. 2(a). The *previous_node_id* 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_node_id* 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* using some standard security mechanisms.

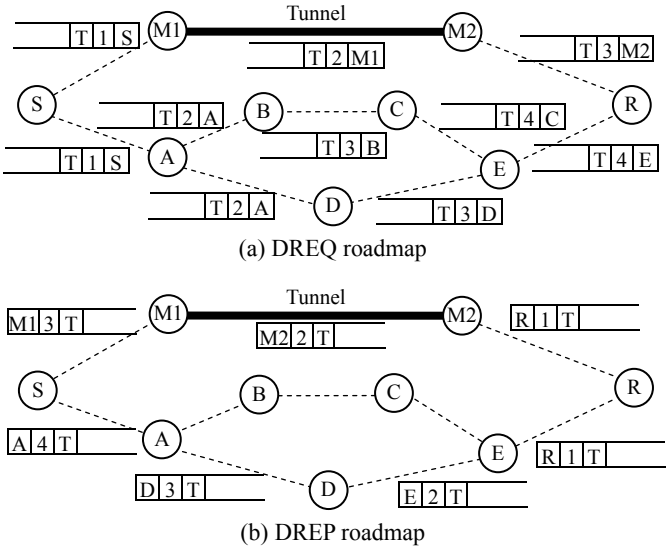

Fig. 2. Two possible disjoint paths

When an intermediate node receives a DREQ packet, it records the *previous_node_id* field and establishes a reverse path to the sender. Then it puts its node ID into the *pervious_node_id* 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_node_id* 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_node_id* 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 that 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. It should be noted that the number of disjoint paths is bounded by the number of neighbors of the sender. As shown in Fig. 2(b), S can receive 2 DREP packets, one from M1 (for path {S, M1, M2, R}) and one from A (for path {S, A, D, E, R}).

Each DREP carries the hop count information of the path that it is associated with. It also carries the *timestamp* indicating 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 the DREP. Then, the sender is able to calculate the delay/hop value of the corresponding path.

Since MANET works in an unreliable wireless environment, DREQ and DREP packets could be lost. To enhance reliability of the information collected, the data collection procedure is repeated several (*num_repeats*) times. It is possible that the hop counts of the 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, 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 in length, 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 several trials of 4-hop and 5-hop, we take the average of the 4-hop trials for phase 2.

To distinguish the DREQs of the *num_repeats* different trials, we have to put some identifiers in the packet headers. As there are many standard mechanisms for this purpose 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

In this phase, the collected data are analyzed. There are two analytical tests for detecting a wormhole: First, we identify normal paths, and put the remaining uncertain paths into a *suspected set*. Then, the suspected set is further tested for detecting a wormhole attack.

Suppose 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 per hop (*DPH*) value of the path to the receiver through node i is given by

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

In normal situation, a smaller h provides a smaller value of *RTT*. It can be explained by the fact that a shorter path should have a smaller round trip time. Hence the *DPH* of normal paths should have similar values independent to h. However, it is not the case in the wormhole paths. 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 small. The resulted *DPH* value of a wormhole path will be larger than that of a normal path.

To prove its correctness, suppose there is a normal path found through node A, while a wormhole path is found through node M, and the round trip times are about the same, then

$$h_M < h_A$$
$$\frac{RTT}{h_M} > \frac{RTT}{h_A}$$
$$DPH_M > DPH_A$$

We performed some simulations, based on the topology in Fig. 4 presented in the next section, in order to get the insight of the relationship between the *DPH* values and the *hop_count*. The relationship is shown in Fig. 6, in which the "path length" indicates the actual *hop_count* of the path including the nodes within the tunnel. We observed that the *DPH* values of normal paths usually appear as smaller values when compared with those of wormhole paths. The *DPH* values of normal and wormhole paths in the same network setting form two separate groups as shown in Fig. 3. The difference between "*the smallest DPH in the wormhole group*" and "*the largest DPH in the normal group*" is always larger than the difference between any 2 *DPH* values within the same group.

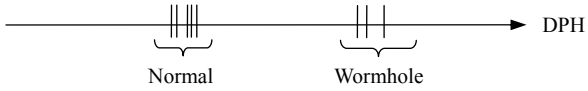

Fig. 3. Relationship of normal and wormhole paths

Based on this observation, we first arrange the *DPH* values in ascending order in the first test. Then, we find whether there is a large difference between 2 adjacent values. If DPH_i is less than the next *DPH* value by a *Threshold* (T_1), or

$$DPH_j - DPH_i > T_1,$$

then all paths with *DPH* values larger than DPH_i are put in the suspected set for another testing, while the remaining paths with smaller *DPH* are treated as normal paths.

It is worth noting that we put the paths with large *DPH* into a suspected set instead of immediately treat them as wormhole paths. The reason is that some paths may have larger delay than the others due to congestion, interference, etc. If we simply treated all paths in the suspected set as wormhole paths, DelPHIX would falsely identify some legitimate paths as wormhole paths.

If we cannot separate the paths into two groups, we also put all the paths in the suspected set. It is because there are three cases that we are unable to locate the large difference (> T_1) between the two groups:

- *All are normal paths* – In this case, the *DPH* values are of similar small values and within the normal group.
- *All are wormhole paths* – In this case, the *DPH* values are of similar large values and within the wormhole group.
- *Congested normal paths* – In this case, suppose

congestion is outside the tunneled link, the *DPH* values of the congested normal paths are as large as that of wormhole paths, all *DPH*s are within the wormhole group.

To identify the wormhole paths in the above cases, we rely on the second test. In the second test, we use the average RTT of all the immediate neighbors (*NRTT*), times a weighting factor, as a threshold (T_2). Immediate neighbors are nodes within transmission range. *NRTT* can be obtained through the AODV *hello message* [17]. When a node receives a hello message, it replies with a short message, e.g. simply its *node_id*, and therefore the overhead is minimal. *NRTT* can reflect the network situation. If the network is congested, *NRTT* is large; otherwise it appears as a small value. We observe from extensive simulations that

$$DPH \geq \frac{NRTT}{2}.$$

When the connection is only one hop, both sides of the inequality have more or less the same value. In multi-hop transmission, the propagation delays of different paths may vary. Some paths may have larger delays than the others due to congestion, interference, etc. In other words, *DPHs* of both normal and tunnel paths may be larger than *NRTT*/2. An interesting observation is found with the value $(NRTT/2) \times hop\_count$. For a wormhole path, with the claimed small *hop_count*, the value remains small, but the *DPH* is increasing with the path length (actual hop count). For a normal path, on the other hand, the value is increasing with the path length, while the *DPH* remains small independently. The above observation is justified by simulations. The *DPH* and T_2 relationships of wormhole path and normal path are shown in Fig. 6. Therefore, the threshold (T_2) is designed as:

$$T_2 = \frac{NRTT}{2} \times hop\_count_i \times \alpha \qquad (2)$$

In case of a single hop transmission, or it is a hidden wormhole attack (such that *hop_count* = 1), we need a weighting factor α to avoid *DPH* equals to T_2. The weighting factor is defined as:

$$\alpha = \begin{cases} 1.1 & \text{if } hop\_count = 1 \\ 1 & \text{if } hop\_count > 1 \end{cases}$$

Therefore, if DPH_i is larger than T_2, the path through node *i* can be concluded, with high certainty, as a wormhole path.

V. Performance Evaluations

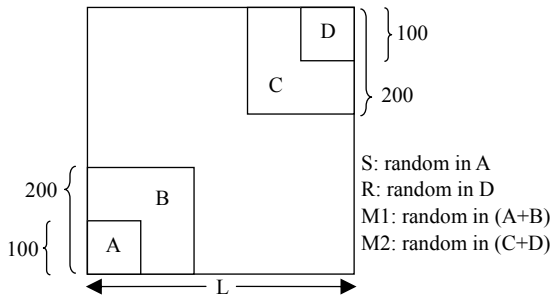

Fig. 4. Simulation topology

In this section, the performance of DelPHIX is evaluated by simulation using the LBNL network simulator *ns* [18]. Random topologies with *N* nodes and size $L \times L$ are randomly generated by a random generator provided by *ns*. The value of *num_repeats* is set to 3. 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 100×100 in the lower left hand corner.

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, and we obtained the average among all the 1000 topologies.

We first conducted simulations with different settings against different threshold values (T_1), the results are shown in Tables I – III. Column "Normal path" refers to the percentage of normal paths (paths that are not attacked) that are detected correctly. Column "Wormhole path" is defined in a similar way. As expected, the smaller the threshold (T_1), the easier to detect wormhole attacks, as fewer paths are treated as normal path in the first test. However, it also leads to a higher rate of treating a normal path as a wormhole path.

TABLE I
DETECTION RATE WITH DIFFERENT THRESHOLD (L=500)

Threshold (T) /ms	Normal path /%	Wormhole path /%
5	98.14	50.48
4	96.55	51.18
3	93.89	51.75
2	87.58	52.08
1	86.92	60.55

TABLE II
DETECTION RATE WITH DIFFERENT THRESHOLD (L=750)

Threshold (T) /ms	Normal path /%	Wormhole path /%
5	98.16	92.38
4	97.78	92.66
3	97.78	93.50
2	97.70	93.50
1	97.62	93.22

TABLE III
DETECTION RATE WITH DIFFERENT THRESHOLD (L=1000)

Threshold (T) /ms	Normal path /%	Wormhole path /%
5	99.66	96.47
4	99.58	96.47
3	99.58	96.79
2	99.58	96.79
1	99.58	96.79

It is found that the detection rate is higher in a larger topology. This can be explained by the fact that smaller L has a larger chance that the tunnel is short, the *DPH* values of wormhole paths and normal paths become comparably close, and may be treated as normal paths (in the first test), 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_1 = 3ms in the following simulations, which evaluate the performance of DelPHIX in the absence of background traffic, and study the effect of background traffic on DelPHIX.

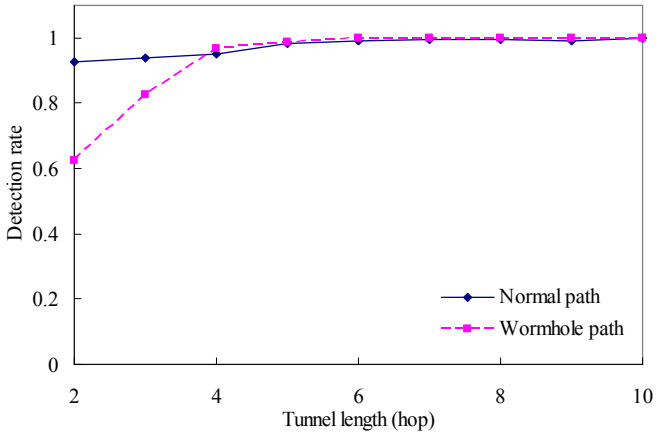

Fig. 5. No background traffic

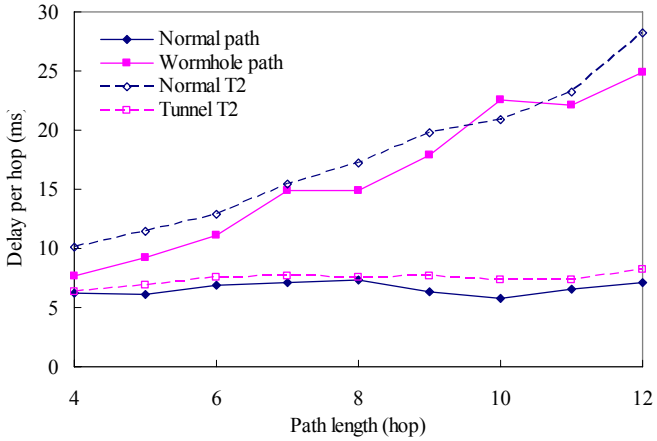

Fig. 6. DPH when no background traffic

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. For normal path, M1 and M2 simply act as legitimate nodes. It is found that the longer the tunnel, the higher the detection rate of wormhole attack, and the detection rate of normal path is independent to the tunnel length.

To explain this phenomenon, the *DPH* values are plotted against different path length, with the corresponding threshold T_2, as shown in Fig. 6. The "path length" in the figure indicates the actual *hop_count* of the path from the sender to the receiver, including the nodes within the tunnel. It is shown that the *DPH* of wormhole path increases with increasing path length, while the *DPH* of normal path remains in a similar level. 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 path gives a larger *RTT*. Therefore, the longer the path (as well as the tunnel), the larger the *DPH*.

T_2 has an opposite behavior. For a normal path, T_2 is directly proportional to the path length. However, for a wormhole path, since the *hop_count* is small, T_2 remains at a small value. This behavior of *DPH* and T_2 leads to a high detection rate of DelPHIX wormhole detection. Noted that the detection rate of wormhole path drops when the path is short (2 to 3 hops). This is because in this case, the *DPH* values of normal path and wormhole path are similar, some wormhole paths are mistakenly treated as normal paths while passing the first test.

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 receiver and no node is a receiver of more than 1 sender.

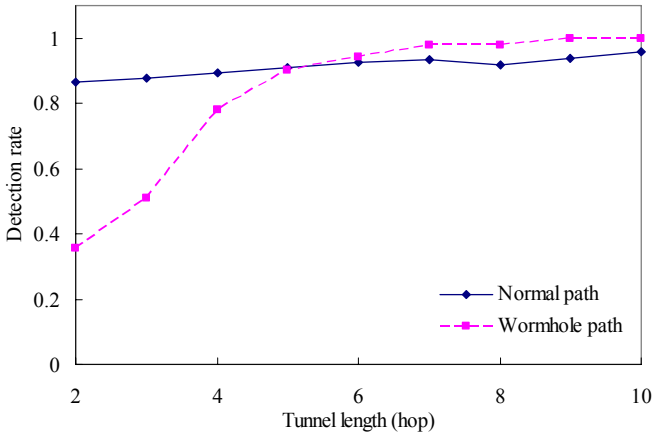

Fig. 7. Light background traffic

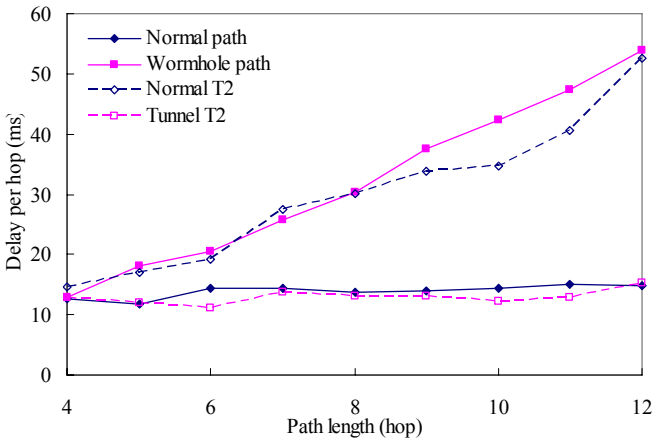

Fig. 8. DPH when light background traffic

Fig. 7 presents the detection rate of DelPHIX when there is light background traffic. We define *light* as 20% of nodes are having connections, i.e. the number of connections equals to 10% of the number of nodes. The result shows similar pattern when compared as Fig. 5. However, it is found that when there is light background traffic, the detection rate is slightly decreased.

According to the *DPH* and T_2 behavior in Fig. 8, when there is background traffic, the *DPH* of normal path is increased. For wormhole path, with the claimed small hop count, the light background traffic is attracted by the

wormhole, and makes the tunneled link congested, which in turn increases the *DPH*. On the other hand, the variation (due to delay jitter) of the *DPH* becomes larger in normal path due to the background traffic. Since the background traffic is light, the difference between the *DPH* values of wormhole path and normal path may not be larger than the variation of the *DPH* of normal path. This makes some wormhole paths become mistakenly treated as normal paths and slightly lowers the detection rate.

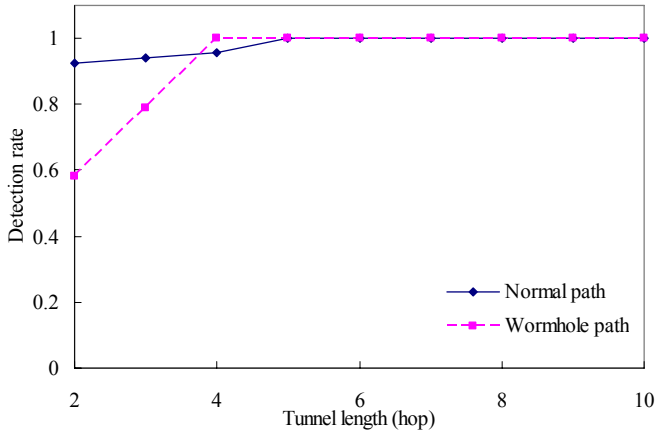

Fig. 9. Heavy background traffic

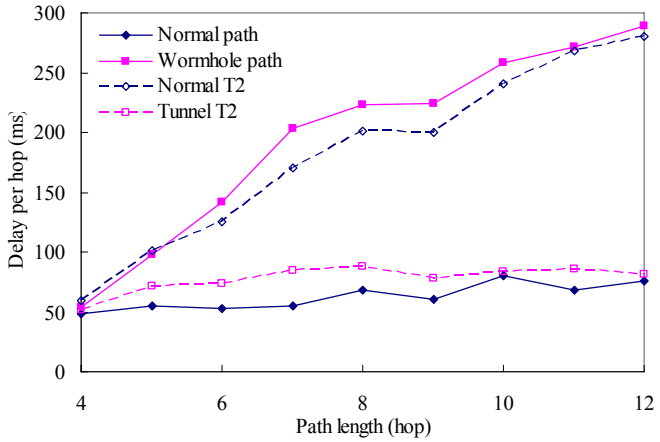

Fig. 10. DPH when heavy background traffic

When there is heavy background traffic, the detection rate of both normal path and wormhole path can achieve 100% when the tunnel path is long as shown in Fig. 9. We define *heavy* as all nodes are having connections, i.e. the number of connections equals to 50% of the number of nodes. Fig. 10 shows the *DPH* behavior under heavy background traffic. It is found that the *DPH* of wormhole path is increased significantly, while that of normal path remains small when compared with wormhole path.

The reason is that, 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 wormhole path is dramatically increased. The difference between the *DPH* of wormhole path and that of normal path becomes more obvious and larger than the variation (due to delay jitter) of the *DPH* of normal path. Therefore all paths are put in the suspected set for the second test. Since the network is congested, *NRTT* is large, and the resulted T_2 is also increased for normal path, but it remains small for wormhole path, as shown in Fig. 10. This makes the second test successful, and leads to a higher detection rates than the pervious 2 scenarios.

Finally, we also conducted simulations with uneven background traffic by randomly locate the light background traffic in the upper-left quarter of the topology. The detection rate is similar to the case of light background traffic but with a larger fluctuation. Due to space limitation, the result is not shown here.

VI. Discussion

In this section, the overhead of DelPHIX is addressed. The notations are defined as follows:

N : Number of mobile nodes

P : Number of found disjoint paths

h_i : hop count of path 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 equal to the number of disjoint paths P. In DelPHIX, P is bounded by the number of neighbors of the sender and the number of neighbors of the receiver. Therefore we have

$$P << N.$$

Since the receiver replies to all DREQ packets, the number of DREP packet is given by

$$\sum_{i=1}^{P} h_i \cdot$$

Hence the total number of packets transmitted in the network for one DelPHIX request can be calculated as the sum of DelPHIX request and DelPHIX reply packets, which is

$$N - 1 + \sum_{i=1}^{P} h_i \cdot$$

Note that the detection procedure consists of *num_repeats* requests, therefore the total message overhead for DelPHIX wormhole detection is given by

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

AODV is chosen to provide comparison because the route setup procedure is similar to DelPHIX. For AODV route setup, since the sender only broadcasts the RREQ once, and the receiver only replies to one RREP, therefore the message overhead is given by

$$N - 1 + h. \quad (4)$$

If we set N = 50 and *num_repeats* = 3, by simulation, we find that

$$\mathrm{E}[P] = 3.06537,$$
$$\mathrm{E}[h] = 4.16280.$$

Substituting them into equations 3 and 4, we obtain the message overhead of DelPHIX wormhole detection is 185.28157, while that of AODV route setup is 53.16280. The

major factor is the *num_repeats* request procedures in providing reliability. In other words, there is a tradeoff between providing reliability and minimizing the message overhead. To reduce message overhead in highly congested situation, DelPHIX sender can launch DREQ for once only.

VII. Conclusion and Future Directions

In this paper, we have described an efficient algorithm for detecting wormhole attack in mobile ad hoc networks. We call it DelPHIX. The advantages of DelPHIX 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. DelPHIX does not collect single-hop information or link-based information, which have been shown to be insufficient to detect both hidden wormhole attacks and exposed wormhole attacks. To be more specific, DelPHIX collects path-based information with DelPHIX Request (DREQ) and DelPHIX Reply (DREP) packets in a similar way of AODV route setup procedure. And we use the delay per hop (DPH) values as an indicator for detecting a wormhole attack.

The performance of DelPHIX has been evaluated by conducting various simulations using the *ns* simulator. It has been shown that DelPHIX can achieve higher than 95% in detecting both normal paths and wormhole paths, in the absence of background traffic. Simulation results have also shown that DelPHIX can maintain above 90% of detection rate for both normal and wormhole paths given that there is background traffic.

The message overhead of DelPHIX has also been addressed in this paper. We compared it with AODV route setup procedures and found that the major factor is the *num_repeats* request procedures in providing reliability. There is a tradeoff between providing reliability of DelPHIX and minimizing the message overhead, and may need further investigation. In the future, we would also like to work on a theoretical analysis of our protocol.

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] 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.

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

[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] T. Korkmaz, "Verifying Physical Presence of Neighbors against Replay-based Attacks in Wireless Ad Hoc Networks," *Proc. of ITCC'05*, vol. 2, 2-6 April 2005, pp. 704-709.

[14] L. Qian, N. Song, and X. Li, "Detecting and Locating Wormhole Attacks in Wireless Ad Hoc Networks through Statistical Analysis of Multi-path," *Proc. of WCNC 2005*, vol. 4, 13-17 March 2005, pp. 2106-2111.

[15] N. Song, L. Qian, and X. Li, "Wormhole Attacks Detection in Wireless Ad Hoc Networks: A Statistical Analysis Approach," *Proc. of IPDPS'05*, 4-8 April 2005, pp. 289a.

[16] H. S. Chiu and K. S. Lui, "DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks," *Proc. of the International Symposium on Wireless Pervasive Computing (ISWPC) 2006*, Phuket, Thailand, 16-18 Jan. 2006, pp. D1-1.

[17] 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.

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

Hon Sun Chiu received his B.Eng. degree in Information Engineering in 2002, and M.Phil. degree in Electrical and Electronic Engineering (major in computer networks) in 2004, both were from The University of Hong Kong. He is currently a Ph.D. candidate in the Department of Electrical and Electronic Engineering of The University of Hong Kong. His major research area is protocol and algorithm design in the wireless ad hoc networks and wireless mesh networks.

King-Shan Lui obtained her B.Eng. (first class honors) and M.Phil. degrees in computer science from the Hong Kong University of Science and Technology. She then received her Ph.D. degree, also in computer science, from the University of Illinois at Urbana-Champaign, USA, in 2002. She joined the Department of Electrical and Electronic Engineering, the University of Hong Kong, as an assistant professor in August 2002. Her research interests include QoS issues, protocol and algorithm design in the Internet, ad hoc networks, and sensor networks. She is a member of IEEE.

.

Kwan L. Yeung received his B.Eng. and Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong in 1992 and 1995, respectively. He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong in July 2000, where he is currently an Associate Professor. His research interests include next-generation Internet, active queue management, packet switch/router design, all-optical networks and wireless data networks.