Detecting and Avoiding Wormhole Attacks in Wireless Ad Hoc Networks

Farid Naït-Abdesselam, University of Sciences and Technologies of Lille
Brahim Bensaou, The Hong Kong University of Science and Technology
Tarik Taleb, Tohoku University

ABSTRACT

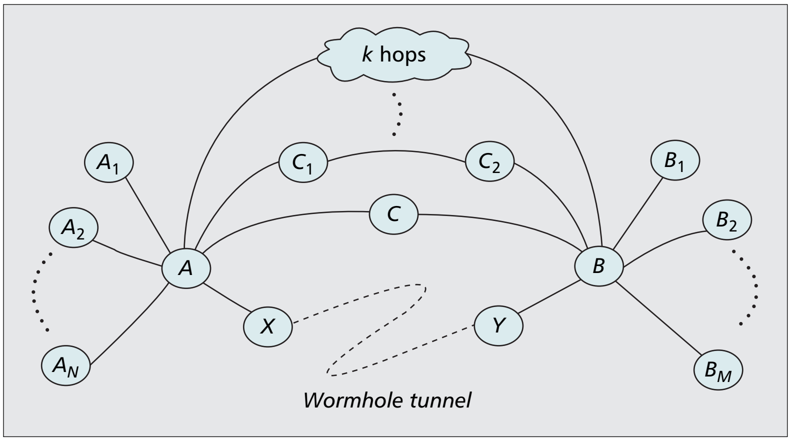
A particularly severe attack on routing protocols in ad hoc networks is the so-called wormhole attack in which two or more colluding attackers record packets at one location, and tunnel them to another location for replay at that remote location. When this attack targets specifically routing control packets, the nodes that are close to the attackers are shielded from any alternative routes with more than one or two hops to the remote location. All routes are thus directed to the wormhole established by the attackers. In the optimized link state routing protocol, if a wormhole attack is launched during the propagation of link state packets, the wrong link information percolates throughout the network, leading to routing disruption. In this article we devise an efficient method to detect and avoid wormhole attacks in the OLSR protocol. This method first attempts to pinpoint links that may potentially be part of a wormhole tunnel. Then a proper wormhole detection mechanism is applied to suspicious links by means of an exchange of encrypted probing packets between the two supposed neighbors (endpoints of the wormhole). The proposed solution exhibits several advantages, among which are its nonreliance on any time synchronization or location information, and its high detection rate under various scenarios.

INTRODUCTION

Due to the versatile nature of their application domain, mobile ad hoc networks are very likely to often be deployed in hostile environments. Due to numerous constraints such as lack of infrastructure, dynamic topology, and lack of pre-established trust relationships between nodes, most of the envisioned routing protocols for ad hoc networks are vulnerable to a number of disruptive attacks. In this article we focus on the so-called wormhole attack, which is known to be particularly challenging to defend against [1], and has been shown to be potentially damaging to a wide range of ad hoc routing protocols.

In a wormhole attack a hostile node constantly monitors the channel, records packets overheard in its vicinity, and tunnels them to a remotely located colluding node, who will replay them in its floor. When this tunneling particularly targets routing control packets such as HELLO messages and route requests (RREQs), nodes close to the attackers are unable to discover the legitimate routes that originate and end in the vicinity of the two attackers: in the typical wormhole attack scenario, such legitimate routes would span more hops than the one or two hops declared by the wormhole attackers. This severely disrupts the network operation. For example, when used against an on-demand routing protocol, such as Ad Hoc On Demand Vector (AODV) or DSR [2], this attack prevents any node from discovering routes of more than two hops. This can be done by tunneling each RREQ message, originating from a node close to the attacker, directly to the target node of the route request. Periodic protocols such as Optimized Link State Routing (OLSR) and TBRPF [2] are also vulnerable to this attack. For example, OLSR uses HELLO packets for neighbor discovery. Considering the scenario in Fig. 1, if the two colluding attackers X and Y tunnel to B all HELLO packets transmitted by A and tunnel to A all HELLO packets transmitted by B, A and B will believe that they are direct neighbors and select each other to route all ensuing data packets. The result of this is that a large number of data packets are directed to the wormhole, with ultimately all the side effects that this may induce such as congestion, packet loss, eavesdropping, spoofing, and so on.

In this article we introduce an efficient method to detect and prevent wormhole attacks in OLSR. Our solution first tries to pinpoint links that may possibly belong to wormhole tunnels, and then applies to such suspicious links an appropriate wormhole detection mechanism by means of an exchange of encrypted probing



■ **Figure 1.** *Wormhole attack model.*

packets between the two supposed neighbors (endpoints of the wormhole) to distinguish between wormhole links and other legitimate ones. Our solution has several advantages among which is its nonreliance on any time synchronization or location information.

The remainder of this article is organized as follows. We present some related work on the wormhole attack problem in ad hoc networks. We describe the features of the wormhole attack and how it works in OLSR. We present our method to detect suspicious links and wormhole tunnels in OLSR. Some simulation results are given to characterize the performance of our proposed method. We finally draw our conclusions.

Related Work

Several approaches have been developed to defend against wormhole attacks in mobile ad hoc networks. In [1] packet leashes are used to protect reactive routing protocols against wormhole attacks. A leash is defined as any information appended to a packet to restrict the maximum transmission distance of the packet. Two kinds of leashes have been proposed: *geographical* and *temporal*. In the geographical leash, the sender appends its location and the sending time to a packet. Based on this information, the receiving node computes an upper bound on the distance to the sender. This solution in fact requires location information and coarse synchronization of all nodes in the network. In the temporal leash, the sender appends the sending time to the packet, and the receiving node computes a traveling distance of that packet assuming propagation at the speed of light, and using the difference between the packet sending time and packet receiving time. This solution requires fine-grained synchronization among all nodes. In [3] directional antennas are used to prevent against wormhole attacks. Each node in the network shares a secret key with every other node and broadcasts HELLO messages to discover its neighbors using directional antennas in each direction.

The SECTOR protocol [4] presents a countermeasure against wormhole attacks by allowing nodes to prove their encounters with other nodes. However, several hypotheses are needed for this protocol to work correctly. Among these are the necessity for coarse synchronization, the ability of nodes to measure their local timing with nanosecond precision, the pre-establishment of security associations between each pair of nodes, and the presence of a central authority that controls the network membership.

So-called disjoint-path-based approaches have been adopted recently. In [5] a statistical approach based on multipath routing is proposed. This approach uses the relative frequency of each link when discovering routes within the network. The main idea beneath this approach resides in the fact that the relative frequency of a link which is part of a wormhole tunnel is much higher than other normal links.

In [6] the proposed DelPHI protocol allows a sender to observe the delays associated with the different paths to a receiver. Therefore, a sender can check whether there are any malicious nodes sitting along its paths to a receiver trying to launch wormhole attacks. The obtained delays and hop count information of some disjoint paths are used to decide whether a certain path among these disjoint paths is under a wormhole attack.

There are also some other methods proposed in the literature [7–9] to defend against wormhole attacks. However, most of these methods require fine-grained time synchronization between nodes in the network or special hardware to prevent against the wormhole attack.

Attack Model

Description of Wormhole Attacks

A wormhole attack is composed of two attackers and a wormhole tunnel. To establish a wormhole attack, attackers create a direct link, referred to as a *wormhole tunnel*, between them. Wormhole tunnels can be established by means of a wired link, a high-quality wireless out-of-band link, or a logical link via packet encapsulation. After building a wormhole tunnel, one attacker receives and copies packets from its neighbors and forwards them to the other colluding attacker through the wormhole tunnel. This latter node receives these tunneled packets and replays them into the network in its vicinity. In a wormhole attack using a wired link or a high-quality wireless out-of-band link, attackers are directly linked to each other, so they can communicate swiftly. However, they need special hardware to support such communication. On the other hand, a wormhole using packet encapsulation is relatively much slower, but it can be launched easily since it does not need any special hardware or special routing protocols.

Wormhole Attack in OLSR

Since a wormhole attack can heavily affect topology construction, it may be lethal to many ad hoc routing protocols, especially proactive routing protocols such as OLSR, which periodically exchange control packets for neighbor discovery and topology construction. Figure 1 depicts an ad hoc network including a wormhole tunnel. When node A broadcasts its HELLO message, node X (an attacker) copies this HELLO message and tunnels it to node Y (the colluding attacker) through the constructed wormhole. Y receives A's HELLO message and replays it in its floor. When node B receives the replayed HELLO message, B deems node A to be its one-hop neighbor. Following a similar procedure, node A may be brought to assume node B to be its one-hop neighbor. After a certain time, a symmetric link can be established between A and B according to the OLSR mechanism. Once this spoofed symmetric link is established, A and B are very likely to choose each others as multipoint relays (MPRs), which then leads to an exchange of some topology control (TC) messages and data packets through the wormhole tunnel. In our example of Fig. 1, B can reach A's one-hop neighbors, which are part of B's two-hop neighbors, only through A. Therefore, B has to select A as its MPR to reach A's one-hop neighbors. Although there are other routes to A and A's one-hop neighbors, because of the wormhole, other routes are certainly more than two hops long. Moreover, in OLSR only MPR nodes can forward TC messages, so selecting MPRs that forward flawed topology information will result in the spread of incorrect topology information throughout the network. This leads to routing disruption and ultimately results in significant performance degradation of the ad hoc network as a whole.

Detecting Wormhole Attacks

In this section we describe our proposed method for detecting and preventing wormhole attacks against OLSR. In our approach the nodes first try to detect links suspected to be part of a wormhole. They then try to ascertain such information through a judicious exchange of newly defined control packets.

Detecting Suspicious Links

In OLSR each node periodically broadcasts a HELLO message to discover its own one-hop neighbors. Upon reception of a HELLO message, a node regards the originator of the HELLO message as a neighbor. However, in a wormhole attack this HELLO message can be replayed from afar (more than one hop away). While this operation does not compromise any nodes, it can give wrong information to the underlying routing protocol and may ultimately cause its failure in finding adequate routes. Two nodes are regarded as neighbors if and only if they are within transmission range of each other.

In our proposed approach we first detect network links with high probability to be involved in a wormhole attack. One commonly accepted and invoked representative feature of wormhole attacks consists of relatively longer packet latency than the normal wireless propagation latency on a single hop. This is typically because, in a wormhole attack, many other multihop routes are channeled to the wormhole. The load on the single route increases, leading to typically longer queuing delays in the wormhole. Nevertheless, this is not a sufficient condition for the existence of a wormhole, because packet transmission is affected by various factors like congestion and intranodal processing. So delay alone may lead to false identification of wormholes. Instead, in our approach links that experience long delays are treated as suspicious links. As such, wormhole verification must be performed only on such suspicious links.

To infer suspicious links, we define two new control packets for the OLSR protocol: $HELLO_{req}$ and $HELLO_{rep}$. The $HELLO_{req}$ message supersedes the standard HELLO message in OLSR, and depending on the option used, it can bear one of two meanings. In the standard option it functions as the original message. In another option a node uses the HELLO message to request an explicit reply from its neighbors. In this option, upon receiving a $HELLO_{req}$ message, the neighbors must respond with a $HELLO_{rep}$ message. $HELLO_{req}$ and $HELLO_{rep}$ have exactly the same format, and the three packet types (standard HELLO, $HELLO_{req}$, and $HELLO_{rep}$) are distinguished by using two of the unused bits in the original message.

After each N standard HELLO message transmissions, a node must send one $HELLO_{req}$ message (requesting thereby explicit HELLO replies from its neighbors) and set an expiry *timeout* for the transmitted $HELLO_{req}$. The value of N can be adjusted according to the desired security level. If the application needs a high security level and has to detect launched attackers rapidly, N should be set to an adequately small value.

When a node receives a $HELLO_{req}$, it records the sender's address i and the time Δ_i left until it is scheduled to send its next HELLO message. The default HELLO message transmission interval is 2 s in OLSR [2]. To avoid overloading the network with too many HELLO replies, a receiver delays the replies of multiple requests until it is scheduled to send its normal HELLO message, and piggybacks the replies to this HELLO message. For each piggybacked reply, the node attaches the recorded address of the sender of the corresponding $HELLO_{req}$ and the respective values of Δ_i. Figure 2 shows an example of a timing diagram where a $HELLO_{rep}$ aggregates replies of three previously received $HELLO_{req}$ messages.

When a node receives a $HELLO_{rep}$, it checks whether this $HELLO_{rep}$ contains information related to any of its outstanding requests. If there is no information about its previous requests, the node treats the received $HELLO_{rep}$ as any normal HELLO message. Otherwise, the node checks the $HELLO_{rep}$'s arrival time to see whether the $HELLO_{rep}$ has arrived within its scheduled timeout interval while taking into account the corresponding delay Δ_i incurred at the receiver. If $HELLO_{rep}$ arrives within its timeout, the originator ranks the link between itself and the node that sent $HELLO_{rep}$ as proven safe. In this case the originator updates its data on the neighbor relationship with that node and neighbors advertisement from that node (see [2] for the details). If $HELLO_{rep}$ does not arrive within its scheduled timeout, the originator ranks the link between itself and the node that has sent the $HELLO_{rep}$ as suspicious and stops communicating with that node until the end of the wormhole verification procedure.

Delay, alone, may lead to false identification of wormholes. Instead, in our approach, links that experience long delays are treated as suspicious links. As such, wormhole verification must be performed only on such suspicious links.

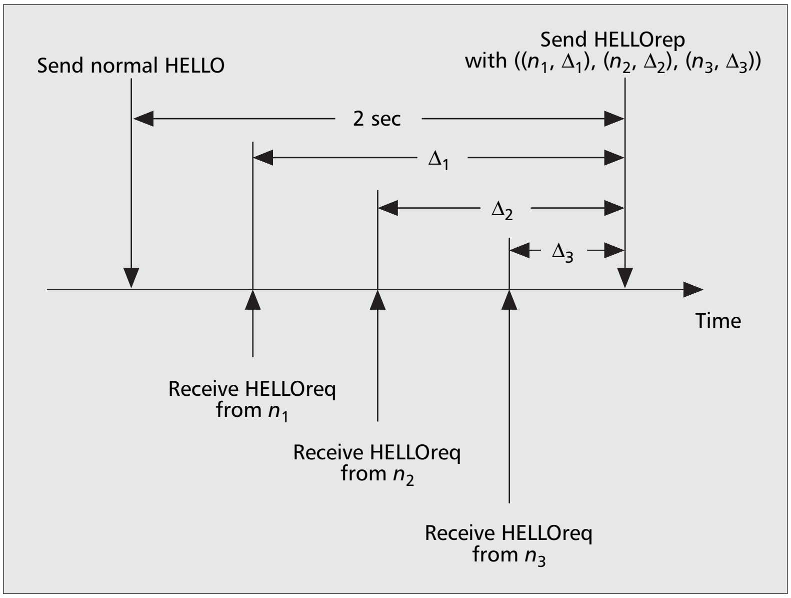


Figure 2. *$HELLO_{rep}$ aggregation.*

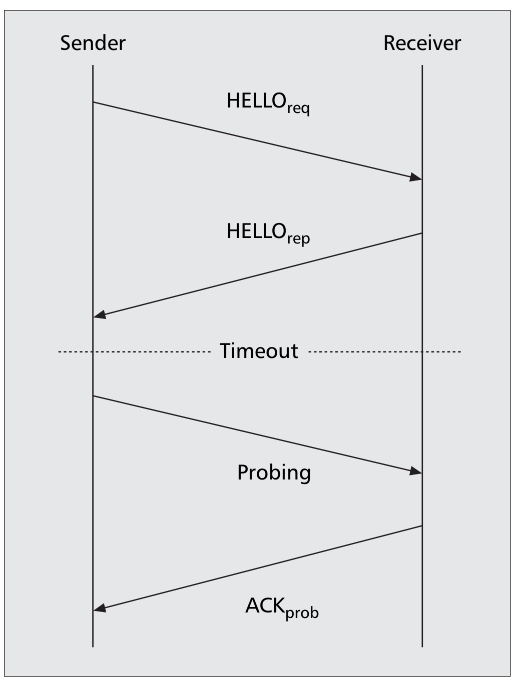


Figure 3. *Message exchange for detecting a wormhole.*

WORMHOLE VERIFICATION

After detecting suspicious links, the originator of $HELLO_{req}$ performs a verification procedure for each suspicious link to check whether there is any wormhole tunnel sitting along the path between itself and the other endpoint of the suspicious links. For this purpose, two new messages are added to the protocol. To detect the wormhole tunnel, the node sends a *Probing* packet to all of its suspect nodes. When a node receives the *Probing* packet, it replies with an ACK_{prob} message to the originator of the *Probing* packet after stopping all transmissions of data packets. Let a *Probing* packet be sent by a node *i* to query a node *j* about its own wormhole status reputation. Node *j* replies with an ACK_{prob} packet where it piggybacks its own opinion about the status of node *i*. The reputation state of a node that has been inferred in the previous exchange of $HELLO_{req}$ and $HELLO_{rep}$ procedure can be either "proved" or "suspicious" depending on the conclusions derived from the suspicious link detection procedure. The ACK_{prob} also contains the processing taken by the receiver of the *Probing* packet until the time it responded with the ACK_{prob}. This timing information is used to tune an accurate timeout. If the node that receives a *Probing* packet does not have any information about the state of the source node, it omits sending the ACK_{prob} and starts collecting the desired information by means of $HELLO_{req}$ and $HELLO_{rep}$ exchanges. When the originator of the *Probing* packet receives $HELLO_{req}$ instead of ACK_{prob}, it immediately sends a $HELLO_{rep}$ and initializes a new timeout only for this node. The timeout of other nodes is not changed. When the node receives $HELLO_{rep}$, it decides the state of the node that sends $HELLO_{rep}$ and sends this information to the originator of the *Probing* packet through ACK_{prob}. If a node has to send both a *Probing* packet and ACK_{prob}, each packet can piggyback another packet.

Figure 3 shows an example of a timing diagram of the exchange of these messages. To ensure the security of exchanging a *Probing* packet and ACK_{prob}, end-to-end authentication is needed as in [10]. A sender chooses a large random number, sufficiently large that an attacker cannot guess, and concatenates it to the *Probing* packet. After that, the sender hashes the *Probing* packet and encrypts that message. If nodes use digital signatures, the sender sends the encrypted message with its certificate. Otherwise, if two nodes share a secret key, we can use symmetric key cryptography instead. When the node receives an encrypted *Probing* packet, first it decrypts that packet and then verifies the sender's identity. If the authentication is successful, the node builds an ACK_{prob} that contains the state of the sender and the large random number that is chosen by the sender. In the same way the node hashes the ACK_{prob} and encrypts it before sending it. After its reception, the sender verifies the validity of the ACK_{prob} message before using the information it contains.

Once again, the originator of the *Probing* packet checks whether the ACK_{prob} arrived within the required timeout. Similar to the $HELLO_{req}$ and $HELLO_{rep}$ procedure, the originator also decides in this exchange about possible suspicious links. To decide whether a suspicious link is traversing a wormhole tunnel, the node compares its evaluation of the reputation of the other endpoint of the suspicious link with the other node's evaluation of its own reputation status:

(Proved, Proved): If the result of the reputation of the remote node is *proved* and the contents of the encrypted ACK_{prob} is proved, the originator concludes that the link between itself and the suspicious node does not contain a wormhole tunnel. The originator maintains the neighbor relationship with this node and accepts information from that node.

• **(Suspicious, Proved) or (Proved, Suspicious):** If one of the two nodes judges the remote node or the content of ACK_{prob} as *suspicious*, the originator concludes that the link is still suspicious. In this case the originator restarts communication with that node after a randomly chosen time. When this time expires, the originator proceeds again with the exchange of *Probing* and ACK_{prob} packets. If the result of this exchange leads to the conclusion of at least one suspicious state, the originator treats this link as a wormhole tunnel.

(Suspicious, Suspicious): If the reputation of the remote node and the contents of the ACK_{prob} are suspicious for both nodes, the originator concludes that the link contains a wormhole tunnel. As a result, the originator removes that node from its one-hop neighbors list and the two-hop neighbors which are one-hop neighbors to that node. If the suspected node has been chosen as an MPR, the originator moves it to a list of forced non-MPR nodes. The originator does not use that link, and packets arriving via that link are dropped until the next $HELLO_{req}$-$HELLO_{rep}$ exchange procedure. If the originator has packets to send to the suspicious node, it has to find another path to reach that node excluding the wormhole link. If there is no other path to that node, the originator waits for the next $HELLO_{req}$-$HELLO_{rep}$ exchange procedure to discover alternate paths.

Timeouts

The value of the timeout has to be calculated carefully in order to avoid false decisions. If the timeout value is set too small, legitimate nodes can be mistakenly suspected. On the other hand, if the timeout is set to a very large value, it becomes hard to detect almost any wormhole attack. The timeout setting is related to whether it can distinguish the normal wireless transmission range of a single hop. Timeout can be then defined as follows:

$$Timeout = \frac{2R}{V} + T_{proc}, \quad (1)$$

where R denotes the maximum transmission range of each node or radio coverage. V is the propagation speed of the wireless signal (e.g., light speed C). In our solution, if a link is regarded as suspicious, the link is given another chance to prove its legitimacy rather than being subject to immediate coercive measures. The parameter T_{proc} denotes the packet processing time and queuing delays within nodes. Usually, T_{proc} is hard to calculate by formulation as it heavily relies on topology, the amount of traffic sent/received, and link conditions (with many collisions or not). In our solution a sender uses an approximation of a receiver's T_{proc} because it is not used for any authentication in the $HELLO_{req}$-$HELLO_{rep}$ exchange procedure. When the originator sends normal HELLO messages and $HELLO_{req}$ messages, it records the difference between packet scheduling time and real transmission time. An average of the latest three records is calculated and is used as T_{proc} in the $HELLO_{req}$-$HELLO_{rep}$ exchange procedure. However, an approximation of T_{proc} is not needed in the *Probing*-ACK_{prob} exchange procedure due to the used end-to-end authentication. Therefore, the sender uses T_{proc} from the receiver, the difference between the *Probing* packet receiving time, and the ACK_{prob} sending time to decide whether there is a wormhole link or not.

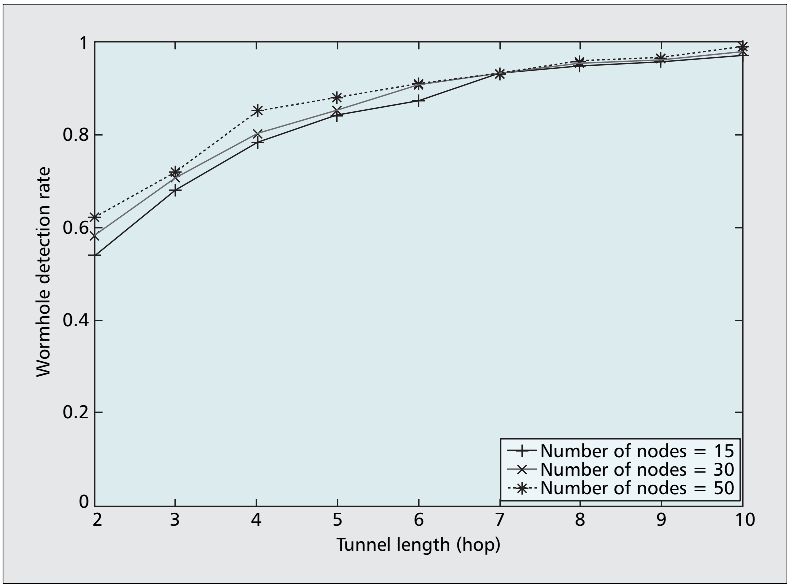


Figure 4. *Wormhole link detection rate for different network sizes ($HELLO_{req}$ emission interval N = 5, wormhole attack duration = 30sec).*

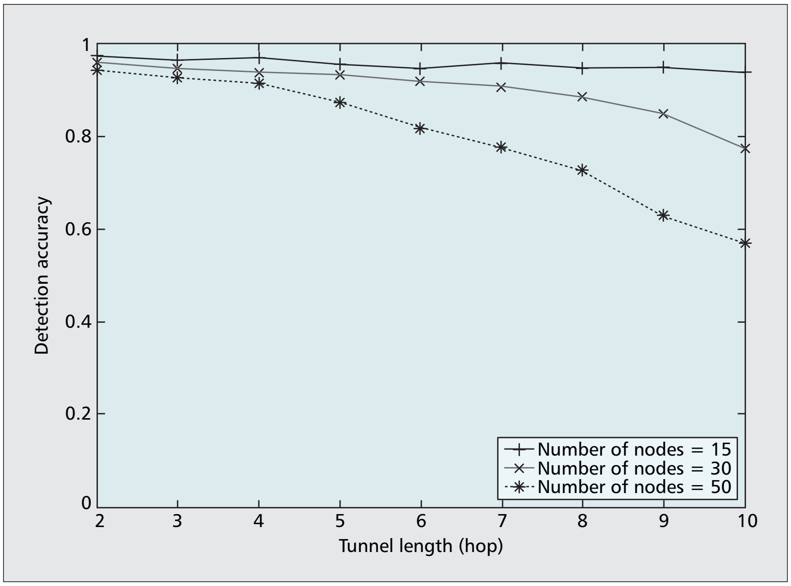
Performance Evaluation

In this section we evaluate the performance of our scheme using the ns-*2* simulator. We generated a number of random topologies with M nodes over a square field, where M ranges from 10 to 50. The square field size is varied from 300 × 300 m to 1500 × 1500 m depending on network size (i.e., number of nodes). The maximum transmission range of each node is set to 250 m. The malicious node pair is selected randomly among the nodes in the formed network. To prevent statistical biases, the presented results are the average of 100 simulation runs. Every node, including the malicious nodes, and control messages such as HELLO or TC messages follow the default settings in the specifications of the OLSR protocol [2].

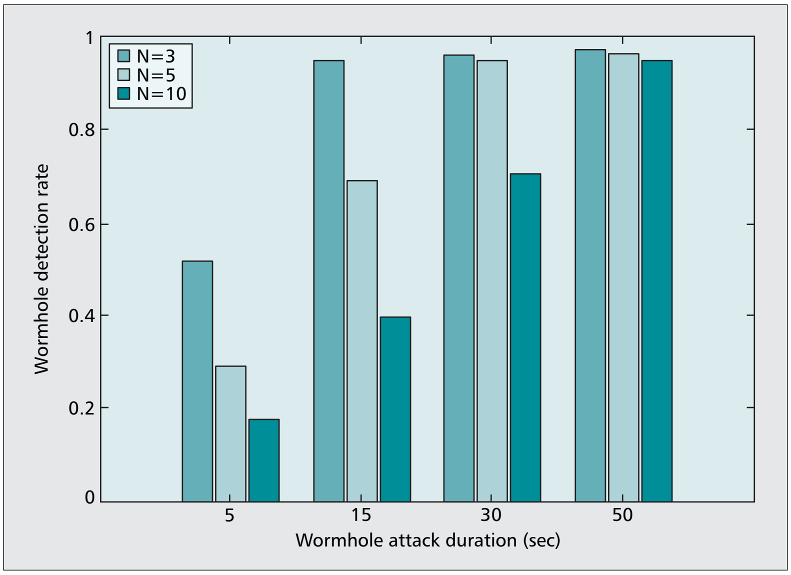
Figure 4 shows the wormhole link detection rate as a function of the tunnel length for different network sizes. Tunnel length refers to the number of hops between the malicious nodes. The $HELLO_{req}$ emission interval is equal to 5 (which means that after sending five normal HELLOs, a $HELLO_{req}$ is sent), and the duration of the wormhole attack is set to 30 s. We define a wormhole link detection rate as the proportion of the number of detected links that contain wormhole tunnels to all links that contain wormhole tunnels. The results show that wormholes are detected more in the configuration where this attack is launched over a longer hop count. This result is quite obvious, since through a wormhole tunnel packets are encapsulated and decapsulated repeatedly, which leads to more delayed transmissions. In the case of less than three hops, the detection rate is relatively low.

This can be explained by the effect of an overestimated T_{proc}. In fact, when the sender has many packets to send, T_{proc} can erroneously be set to a large value. Therefore, as the sender's T_{proc} can be overestimated, some wormhole attacks go undetected. However, we can notice that this overestimated T_{proc} does not affect the detection rate of wormhole attacks over a path with a length exceeding four hops. We conclude here that the number of nodes constructing the wormhole tunnel more or less affects its detection.

Figure 5 shows results on detection accuracy. Detection accuracy is measured as the ratio of links that effectively contain wormhole tunnels to the links that are judged suspicious by our solution. The results show that detection accuracy depends on the correlation between the number of nodes and the tunnel length. In a network of 15 nodes, the detection accuracy rarely decreases as the tunnel length increases. However, in larger networks (e.g., 30 and 50 nodes), the detection accuracy decreases dramatically as the tunnel length increases. This can be explained by the number of neighbors that can be selected to form wormhole tunnels by malicious nodes. When the number of nodes in the network is equal to 15, the number of any node's neighbors is more likely to be small; as the tunnel length increases, it becomes rarely obvious to find another route similar to that of the detected wormhole tunnel. However, if the number of nodes in the network becomes larger, malicious nodes are more likely to have many neighbors even though they are far away from each other and connected through a longer wormhole tunnel. Moreover, in OLSR each node periodically sends routing control messages, which increases the load in dense networks. As these routing control messages are tunneled through the wormhole tunnel, the traffic increases dramatically, and congestion becomes inevitable through the path of that wormhole tunnel. This makes the legitimate nodes suspect and faultily identify some links as containing wormhole tunnels because of the increased delays.



■ **Figure 5.** *Wormhole link detection accuracy (HELLO$_{\mathrm{req}}$ emission interval* N *= 5, wormhole attack duration = 30sec).*



■ **Figure 6.** *Wormhole link detection rate for different HELLO$_{\mathrm{req}}$ emission interval and different wormhole attack durations (network size = 30 nodes).*

Figure 6 presents the wormhole link detection rate for different HELLO$_{req}$ emission intervals and different wormhole attack durations when the number of nodes is 30. The graph elucidates the correlation between HELLO$_{req}$ emission intervals and wormhole attack durations. If the wormhole attack duration is shorter than the HELLO$_{req}$ emission interval, the wormhole link detection rate becomes poor (i.e., less than 0.5). This is due to the fact that there are some nodes that do not perform the HELLO$_{req}$-HELLO$_{rep}$ exchange procedure. Our approach shows a good detection rate after two HELLO$_{req}$ emission intervals. This result demonstrates the impact of the HELLO$_{req}$ emission interval on detection time. If the HELLO$_{req}$ emission interval is long enough, it takes more time to detect any wormhole tunnel. Therefore, an application that needs a high security level has to use small HELLO$_{req}$ emission intervals.

Conclusion

Wormhole attacks are severe attacks that can easily be launched even in networks with confidentiality and authenticity. Malicious nodes usually target the routing control messages related to topology or routing information. In this article we have presented an effective method for detecting and preventing wormhole attacks in OLSR. To detect wormhole tunnels, we use a simple four-way handshaking message exchange. The proposed solution is easy to deploy: it does not require any time synchronization or location information; nor does it require any complex computation or special hardware. The performance of this approach shows a high detection rate under various scenarios.

References

[1] Y. C. Hu, A. Perrig, and D.B. Johnson, "Wormhole Attacks in Wireless Networks," *IEEE JSAC*, vol. 24, no. 2, Feb. 2006, pp. 370–80.

[2] IETF MANET Working Group, http://www.ietf.org/html.charters/manet-charter.html
[3] L. Hu and D. Evans, "Using Directional Antennas to Prevent Wormhole Attacks," *Proc. Network and Distrib. Sys. Sec. Symp.*, San Diego, CA, Feb. 2004.
[4] S. Capkun, L. Buttyan, and J.-P. Hubaux, "SECTOR: Secure Tracking of Node Encounters in Multihop Wireless Networks," *Proc. ACM Wksp. Sec. of Ad Hoc and Sensor Networks*, Fairfax, VA, Oct. 2003.
[5] L. Qian, N. Song, and X. Li, "Detecting and Locating Wormhole Attacks in Wireless Ad Hoc Networks through Statistical Analysis of Multi-path," *Proc. IEEE WCNC*, New Orleans, LA, Mar. 2005.
[6] H.S. Chiu and K.S. Lui, "DelPHI: Wormhole Detection Mechanism for Ad Hoc Wireless Networks," *Proc. Int'l. Symp. Wireless Pervasive Comp.*, Phuket, Thailand, Jan. 2006.
[7] L. Lazos *et al.*, "Preventing Wormhole Attacks on Wireless Ad Hoc Networks: A Graph Theoretic Approach," *Proc. IEEE WCNC*, New Orleans, LA, Mar. 2005.
[8] I. Khalil, S. Bagchi, and N. B. Shroff, "LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks," *Proc. Int'l. Conf. Dependable Sys. and Networks*, Yokohama, Japan, July 2005.
[9] Y. Zhang *et al.*, "Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks," *IEEE JSAC*, vol. 24, no. 2, Feb. 2006, pp. 247–60.
[10] Y. C. Hu, D. Johnson, and A. Perrig, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," *Proc. ACM Wksp. Wireless Sec.*, San Diego, CA, Sept. 2003.

BIOGRAPHIES

FARID NAÏT-ABDESSELAM [M] (farid.nait-abdesselam@lifl.fr) obtained his engineering degree in computer science from the University of Sciences and Technologies Houari Boumediene (USTHB), Algiers, Algeria, in June 1993 and a Master's degree in computer science from the University of Paris Descartes, France, in September 1994. After two years spent in industry working as a software engineer, he joined the University of Versailles San Quentin, France, in January 1996, and got his Ph.D. degree in computer science in January 2000. Duriing 1998 he worked as an associate researcher at the University of Western Ontario, London, Canada, working on distributed interactive virtual environments and multimedia communications over ATM networks. From September 1999 to August 2000 he worked as an assistant professor at the University of Sciences and Technologies of Lille, France. From September 2000 to August 2003 he worked as an associate professor at INSA of Lyon and a research member of INRIA Rhone Alpes. Since September 2003 he has been an associate professor at the University of Sciences and Technologies of Lille, and until September 2007 as a research member of INRIA Lille Nord Europe. His research interests lie in the field of computer and communication networks with emphasis on architectures and protocols for quality of service and security in IP-based networks, mobile ad hoc, sensor, vehicular, and mesh networks, and overlay networks. He has been on the technical program committees of different IEEE and ACM conferences, including GLOBECOM, ICC, LCN, nd MSWiM, and regularly invited to chair some of their sessions. He is chairing or has chaired the IEEE International Workshop on Wireless Local Networks, IEEE/ACS International Workshop on Internet Services, and International Workshop on Peer-to-Peer Networking. He is currently serving as Editorial Liaison chair of the IEEE LCN Conference, Exhibit & Sponsorship Co-Chair of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, and Publicity Co-Chair of many conferences.He is a member of the IEEE Communications and Computer Societies.

BRAHIM BENSAOU [SM] (brahim@cse.ust.hk) received an engineering degree in computer science (with distinction) from USTHB in 1982 and a D.E.A. degree from the University of Paris XI in computer science in 1988. He earned his doctorate degree in computer science from the University of Paris VI in 1993. From 1990 to 1994 he was a research assistant at France Telecom Research Laboratories near Paris, where he was involved in the early designs and study of ATM technology. In mid-1995 he joined the Hong Kong University of Science and Technology (HKUST) as a research associate, where he spent nearly two years working on various problems in congestion and traffic control. In 1997 he joined the Centre for Wireless Communications, a national R&D center in Singapore (now known as the Institute of Infocomm Research, I2R A-Star) as a member of technical staff, where he worked as a system architect on the design of QoS enabled MAC protocolsand scheduling algorithms in a wireless prototype. ATM network prototype.In 1998 he was promoted to senior member of technical staff, and was instrumental in forming a small R&D group in the area of wireless networking at the CWC. He led the group for a year and a half, and then moved to academia at HKUST in fall 2000, where he is now a faculty member in the Department of Computer Science and Engineering. His general areas of research are in QoS enabled wired/wireless networks, including ad hoc networks, sensor networks, and wireless LANs, on which he has published more than 80 research papers in prominent conferences and journals, received numerous research grants, graduated nearly 20 postgraduate students, four of which are Ph.D.s, and invented three U.S. patents, one of which is licensed. He is an Associate Editor of *IEEE Communications Letters* and a member of the ACM.

TARIK TALEB [M] [S'04, M'05] (taleb@aiet.ecei.tohoku.ac.jp) is currently working as an assistant professor with the Graduate School of Information Sciences (GSIS), Tohoku University, Japan. From October 2005 to March 2006 he was working as a research fellow with the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B.E. degree in information engineering with distinction, and M.E. and Ph. D. degrees in computer sciences from GSIS, Tohoku University, in 2001, 2003, and 2005, respectively. His research interests lie in the field of wireless networking, satellite and space communications, congestion control protocols, mobility and handoff management, on-demand media transmission, and network security. He is on the editorial board of *IEEE Wireless Communications*. He also serves as Secretary of the Satellite and Space Communications Technical Committee of IEEE ComSoc. He has been on the technical program committee of different IEEE conferences, including GLOBECOM, ICC, and WCNC, and chaired some of their sessions. He is a recipient of the 2007 Funai Foundation Award (March 2007), the 2006 IEEE Computer Society Japan Chapter Young Author award (December 2006), the Niwa Yasujirou Memorial award (February 2005) and the Young Researcher's Encouragement award from the Japan chapter of the IEEE Vehicular Technology Society (October 2003).