WORMEROS: A New Framework for Defending against Wormhole Attacks on Wireless Ad Hoc Networks

Hai Vu, Ajay Kulkarni, Kamil Sarac, and Neeraj Mittal

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA

Abstract. Wormhole attack is a type of replay attack in wireless networks that has serious consequences and is hard to defend against. This is because the attacker does not need to modify packets or compromise wireless nodes. This paper introduces Wormeros, a new framework to detect wormhole attacks in wireless networks. The framework contains two phases namely suspicion and confirmation. Our solution does not require any special hardware (such as GPS) or expensive mechanisms (such as time synchronization) added to the wireless nodes. Using analysis and simulation, we show that our solution is effective in detecting and defending against wormhole attacks.

1 Introduction

Wireless ad hoc networks consist of wireless devices that do not require an infrastructure to operate on. Such networks are vulnerable to many security threats, as data is broadcasted over a shared channel. We focus on wormhole attacks, wherein an attacker captures data in one region of the network, tunnels it to another region (possibly using higher transmission power or an out-of-band channel), and injects it back into the network. We illustrate a wormhole attack in ad hoc networks (see Figure 1) below.

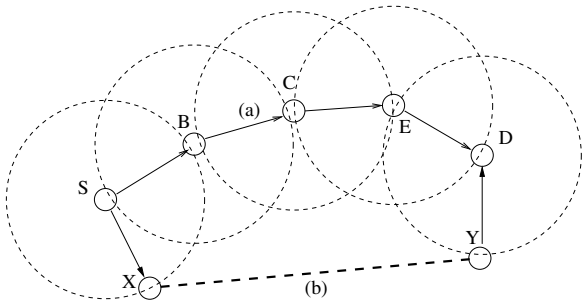

Fig. 1. A wormhole attack in ad hoc networks: (a) Normal link, (b) Wormhole link

In the ad hoc network in Figure 1, assume that S, B, C, D, E are legitimate nodes and S can communicate with D using the path $S - B - C - E - D$. An attacker

Y. Li et al. (Eds.): WASA 2008, LNCS 5258, pp. 491–502, 2008.

places two nodes in the network, X and Y, which are close to S and D, respectively. The attacker can make S and D think that they are neighbors by having X capture the packets sent by S, tunnel the packets to Y, which replays the packets to D. From then on, S and D use the erroneous direct link for communication. Thus, the attacker has successfully mounted the wormhole attack. He now controls the link and can mount further attacks such as flow analysis, blackhole attack, etc [1].

The attacker can initiate a wormhole attack in a passive mode without modifying any packet and thus neither needs to break the authentication scheme nor needs to have the knowledge of encryption keys that are used between the two nodes in the network. As a result, the attacker does not need to compromise a node in the network to successfully launch the wormhole attack. Hence, a solution that depends only on cryptographic techniques is clearly not effective enough to defend against wormhole attacks.

In this paper we present a novel framework that combines multiple simple techniques which help detect wormhole attacks on wireless ad-hoc networks. Our main contribution is to show how simple protocols can be jointly applied to first suspect a link and then challenge the link to confirm this suspicion. The specific techniques we present in the paper are just examples of how to employ different protocols in each phase of our framework.

The rest of the paper is structured as follows. In Section 2, we describe the previous work in wormhole detection. In Section 3, we describe our ideas and how our solution differs from other solutions. In Section 4, we present Wormeros framework in details. Security analysis and experiment results of our framework are presented in Section 5 and Section 6, respectively. We conclude the paper in Section 7.

2 Related Work

Packet Leashes [1] is among the first work in defending against wormhole attacks. The authors introduce two solutions: *Geographical Leashes* and *Temporal Leashes*. In the first one, location information (e.g. from GPS devices), included in the packets, is used to detect the presence of wormhole link. In the latter, nodes are tightly time synchronized and the packet being transmitted between a source node and a destination node contains the time at which it is sent. The destination node will not accept if the packet is expired.

Hu and Evans [2] use directional antennas (introduced in [3]) to detect wormhole attacks while SECTOR [4] uses another type of special hardware and depends on the fact that the distance between two nodes can be measured accurately based on the speed of data transmitted between them.

TrueLink [5] depends on the RTS-CTS-Data-ACK mechanism of IEEE 802.11 MAC protocol to defend against wormhole attacks. In this solution the nodes have a time constraint to authenticate each other and thus wormhole attacks are prevented.

LITEWORP [6] introduces a notion of *guard node*, which is a common neighbor of two nodes to detect a legitimate link between them. The guard node can detect the wormhole if one of its neighbor is behaving maliciously. In a sparse network, however, it is not always possible to find a guard node for a particular link.

Maheswari et al. [7] propose a different approach to detect wormhole attacks. Their idea is to look for some *forbidden substructures* in the connectivity graph that should not exist in a network that has no wormhole.

The authors in [8,9] use RTT (round-trip-time) to detect wormhole attacks. Chiu et al. [8] measure the delay per hop in the whole path while Tran et al. [9] measure the RTT for each successive links in the whole path. If this measurement is higher than some threshold value, then the alarm is raised. These solutions, however, require the cooperation of all nodes in the path.

3 Wormeros Framework

Our main work in the paper is to introduce and analyze Wormeros(the name is inspired from Kerberos), a framework that consists of two phases to detect wormhole attacks. The first phase applies inexpensive techniques and utilizes local information that is available during the normal operation of wireless nodes. Advanced techniques in the second phase are applied only when a wormhole attack is suspected. Thus, in case there are no wormholes in the network, the wireless nodes do not need to waste computation and communication resources.

Following are two phases of Wormeros:

- **Phase I - Suspicion**: We use two techniques to examine the existence of the wormhole attack in the network. First, we measure the RTT between a node S and all of its immediate neighbors. If RTT(S, D), where D is one of S's neighbors, is abnormally higher than the average RTT of all links from S to its neighbors, then there might be a wormhole between S and D. This technique is different from work in [8,9] in that we do not require the cooperation of all nodes in the path between S and D. Second, we use an observation that in a dense network, two neighbors S and D are likely to share some common neighbors. This technique is similar to work in [6,7] but we only use local information instead of global information. If any of the techniques in the Suspicion phase detects the existence of a suspicious link, then we move to the second phase of Wormeros to confirm the wormhole.
- **Phase II - Confirmation**: Having suspected a possible wormhole link in the network, Wormeros launches a series of challenges to make sure that the wormhole is correctly identified. In this phase, the two legitimate nodes being attacked by the wormhole link collaborate to challenge the attacker. We propose to use frequency hopping for this purpose. TrueLink [5] can also be used in this phase.

To the best of our knowledge, this paper is the first to introduce a framework consisting of more than one techniques to defend against wormhole attacks. Our solution differs from work of [1,2,4] in that we do not require additional hardware devices or expensive time synchronization. Work in [6,7] are not efficient in sparse networks, but in the Suspicion phase of our solution we overcome this by adding techniques based on RTT measurements which work well in sparse networks. TrueLink [5] is efficient but expensive if applied all the time. In our framework, expensive techniques such as TrueLink are only used in the second phase where a link is suspected and this helps reduce overhead for the wireless nodes in the network.

4 Details of Wormeros Framework

4.1 System Model and Notation

We consider a wireless ad hoc network with no centralized server. We assume that none of the nodes in the network is compromised. The links are assumed to be bidirectional and all nodes use the same transmission power. We assume that the topology does not change rapidly and there is at least one RTS/CTS/Data/Ack period of time that a pair of nodes can communicate. We further assume that any pair of nodes in the network share two cryptographic keys K_1 and K_2. This assumption is common in the area and can be achieved by applying a key exchange scheme like Diffie-Hellman or a scheme by Eschenauer and Gligor [10].

In the remaining sections of the paper, we use the following notations:

- p: the propagation delay of a legitimate link
- $RTT_{(S,D)}$: RTT between node S and node D
- $RTT_{wormhole}$: RTT of a link under wormhole attack
- Avg^S_{All}: the average RTT of all links from S to its neighbors
- w: the time to tunnel a packet between two wormhole ends
- d: the number of neighbors of a node
- E(K, M): the message M is encrypted using secret key K
- HMAC(K, M): the message digest of M, using a secure hash function with secret key K

4.2 Suspicion Phase

In this phase, we use simple triggers to find out if a link should be suspected and challenged. One of the simplest triggers is based on the round trip delay of a link. Let a node S communicate with a neighbor node D. During peace time, the RTT between S and D is $2p$. If the direct link (S, D) is formed as a result of a wormhole attack, then the round trip time would be $RTT_{wormhole} = 2(p + w + p) = 2(2p + w)$. Thus we believe the RTT of the wormhole link should be at least two times the RTT of a normal link, even though w can be smaller than p. In Section 6 we conduct experiments to confirm this fact.

We develop a simple scheme for wormhole suspicion based on RTT in Algorithm 1 as illustrated in Figure 2. Let node S communicate with node D through a wormhole link XY. Node S knows that A, B, C, D, E are its neighbors and S can measure the RTT with all of the links $(S, A), (S, B), (S, C), (S, D)$ and (S, E). If the $RTT_{(S,D)}$ is at least k times the average RTT between S and all its neighboring nodes, then the link (S, D) may be a wormhole. The value of k is the system parameter which depends on d and w. In Section 5.1 we explain how the value of k is determined. Algorithm 1 is similar to the scheme proposed in [9] which detects the presence of wormhole by measuring the RTT during route discovery. However, the difference is that we define deterministic threshold value while the scheme in [9] decides the threshold value based on simulations.

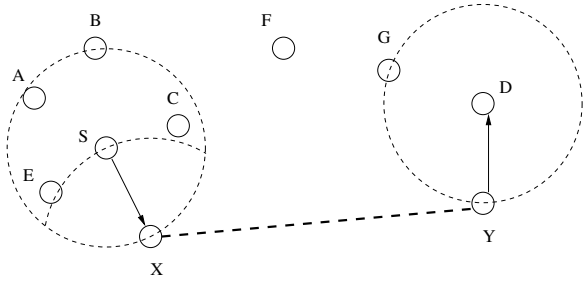

Fig. 2. Probing neighbors to find out suspicious link

Algorithm 1. RTTDetection(S, D)

1: S calculates the average RTT of each link, based on n (system parameter) samples
2: S calculates the average RTT of all d neighbors, Avg^S_{All}
3: **if** $RTT_{(S,D)} \geq k * Avg^S_{All}$ **then**
4: Confirm the link (S, D) is suspicious and execute Challenge phase.
5: **end if**

Other techniques can also be applied simultaneously to Algorithm 1 in Suspicion phase. For example, Algorithm 2 uses the neighbor information available at each node to detect a suspicious link. The idea of Algorithm 2 is that if S and D are far away, then it is very likely that D will not be in the neighbor list of any of S's neighbors. In Figure 2, S's neighbors are A, B, C and E; none of these nodes have D as their neighbor. By applying Algorithm 2, S will find the link (S, D) suspicious.

Algorithm 2. NeighborDetection(S, D)

1: S collects identity of its one-hop and two-hop neighbors and creates the Neighborhood Set.
2: S checks the Neighborhood Set to see if D belongs to the set.
3: **if** D is not an element of the set **then**
4: Confirm the link (S, D) is suspicious and execute the Challenge phase.
5: **end if**

Note that the more techniques we apply in the Suspicion phase, the more accurate it is to detect wormhole attacks. This is because each technique is applied independently and thus the wormhole links that are bypassed by one technique are likely to be detected by another technique. However, there is a trade-off to consider: the higher number of techniques is applied, the higher cost it is. Therefore, we suggest to use a small number of techniques in the Suspicion phase.

4.3 Confirmation Phase

In this phase we use frequency hopping for confirming the existence of a wormhole. The pseudo-code is presented in Algorithm 3.

Algorithm 3. FrequencyHoppingChallenge(S, D)

1: S sends an encrypted message to D requesting to challenge the link (S, D). The message is transmitted using frequency f_1.
2: D replies with an encrypted message accepting the challenge, and specifies a random frequency f_2. The message is transmitted in frequency f_1.
3: D switches its receiver to f_2 and waits for $2 * RTT_{(S,D)}$ time.
4: After receiving the reply message, S transmits a message in frequency f_2 and starts waiting for acknowledgment in f_2.
5: **if** S does not receive an acknowledgment from D in frequency f_2 within a duration of $2 * RTT_{(S,D)}$ time, **then**
6: Confirm link (S, D) is a wormhole link
7: **end if**

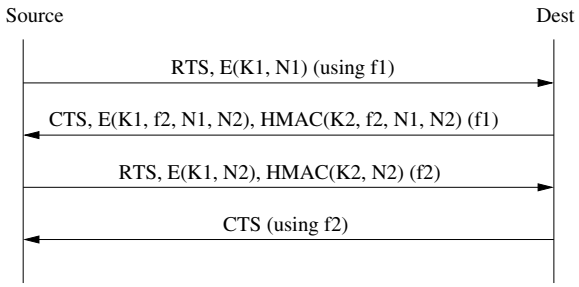

Fig. 3. Frequency hopping challenge

We illustrate the implementation of Algorithm 3 using RTS/CTS mechanism of IEEE 802.11b standards in Figure 3. In the first message, S sends RTS and a nonce N_1 (encrypted using K_1) to D using a frequency f_1 being used for communication between them. Upon receiving this message from S, D replies in frequency f_1 with a CTS message that contains the frequency f_2 (picked from the set of common frequencies shared by S and D), the nonce N_1 received previously and a new nonce N_2, also encrypted with K_1. To protect the integrity of the packet, D can optionally compute a message digest using HMAC function with key K_2.

After replying to S with CTS packet, D switches its receiver to frequency f_2 and starts waiting for a packet from S. Here we assume the CTS always gets through if the environment conditions are stable. Later in the analysis section we discuss this assumption in depth. Immediately after receiving CTS, S switches its transmitter to frequency f_2 and sends a new RTS message to D that contains N_2 for the sake of authentication. Finally D replies with a CTS packet to finish the challenge.

If S and D are far away and become direct neighbors due to the wormhole, then by switching to the new frequency they will not be able to receive messages from each other. This is because the attacker does not know the new frequency and thus cannot forward the messages between S and D.

The use of nonces N_1 and N_2 is to avoid the replay attacks. Without the nonces, the attacker can launch the attack as follows. Suppose that the attacker has captured a

CTS packet which contains an encrypted frequency f_2 that he does not know. He can store the message and try to scan all the frequencies to find out the one in which S and D are communicating. On correctly identifying the frequency, he can replay the same message for any new challenge between the same pair S and D, thus effectively breaking the solution. This attack is not possible if we use nonces because they can help detect replayed messages. We can further improve the security for these messages by including the expiry time for each message (for example, each message is expired after 10 seconds).

5 Security Analysis

5.1 Analysis of Suspicion Phase

In Algorithm 1 we require that $RTT_{(S,D)}$ be at least k times Avg^S_{All} so that S can start suspecting the link (S, D) to be a wormhole. Now we show how each node can determine the value of k. Let d be the number of neighbors a node has and assume that among d neighbors there exists at most m ($m < d$) wormhole link. We have:

$$RTT_{(S,D)} = 2(2p + w)$$

$$Avg^S_{All} = \frac{(d-m)2p + 2(2p+w)m}{d}$$

$$Test = \frac{RTT_{(S,D)}}{Avg^S_{All}} = \frac{2(2p+w)d}{(d-m)2p + 2(2p+w)m} \geq k$$

Observe that $Test$ increases when w increases. Thus, to avoid detection, the attacker should try to decrease the value of $Test$ by decreasing w. However, w is always greater than 0. Thus, if we set the threshold value k for $w = 0$ then the attacker will very likely be detected. In that case, $k = \frac{2d}{d+2m}$ and can easily be computed by each wireless node. For example, if $d = 6$ and $m = 1$, then the threshold value k will be $12/7 = 1.7$. This is a deterministic value, contradicting with the one in [9], where the threshold value varies in different networks.

5.2 Analysis of Confirmation Phase

802.11 RTS-CTS Mechanism. The wireless channel is unreliable and nodes use the CSMA/CA mechanism protocol for transmitting. The unreliability is caused by two factors: noise and collisions. We assume that during one execution of RTS-CTS-Data-ACK the environment is stable, thus loss of packets due to noise spike can be ignored. Hence, if the sender has successfully sent the RTS to the receiver, all of its neighbors would have received the RTS and would not contend for the channel. Therefore, the CTS will be received correctly at the sender.

Attacking the Confirmation Phase. The attacker has two options to respond to the challenge: either to drop the RTS packet or to allow the packet to pass through to D. We now show that using any of these options is not helpful to the wormhole attack and it will eventually be discovered.

a) Dropping the RTS Packet

In our solution if S does not get the CTS reply in a finite amount of time it will timeout and resend the RTS. In 802.11 mechanism each node retries r times (typically $r = 7$) before declaring a transmission failure. If a transmission failure occurs our solution considers that to be a missed challenge. If a link has M such continuous missed challenges, our solution declares that link to be malicious.

If node S is sending an RTS frame then the probability that collisions occurs is given by:

$$\text{P[collision]} = 1 - (1 - \tau)^{d-1}$$

where τ is the probability of transmission at a moment t of each node and d is the number of neighbors of a node. If S does not get the CTS reply within a finite amount of time it times out and resends the RTS frame. If all these r RTS frames were to collide with transmissions from other node then the probability of that happening is:

$$\text{P[Losing r RTS]} = [1 - (1 - \tau)^{d-1}]^r$$

The probability of failing M challenges due to wireless issues rather than wormhole is:

$$\text{P[Failing M challenges]} = [1 - (1 - \tau)^{d-1}]^{rM}$$

Using $M = 5, r = 7, d = 10$ and $\tau = 0.1$ we get

$$\text{P[Failing M challenges]} = 5.3 * 10^{-8}.$$

This probability of failing M challenges without the existence of wormhole is thus negligible. Hence the strategy of dropping RTS packets is not in the interest of the wormhole.

b) Allowing the RTS Packet Through

The other option for the wormhole is to allow the RTS to go through. We assume that (i) it is too expensive for the attacker to listen on all the available channels and (ii) it is computationally infeasible for the attacker to break the encryption to obtain f_2 in a short duration. Therefore, by allowing the RTS get through the attacker has to guess the frequency f_2, because the content of the message is encrypted and integrity protected.

The probability of correctly guessing the right frequency is $1/N$, where N is the number of channels. If we further force each node to pass the challenge for δ times this probability of guessing the correct frequency every time is reduced to $1/N^{\delta}$. Using appropriate values of δ and N this probability can be made very small. For example if $N = 11$ (802.11b network) and $\delta = 2$ the probability is less than 1%. The wormhole thus is unlikely to pass the Confirmation phase.

6 Performance Evaluation

First, we conduct experiments to study the impact of wormhole links on the RTT values. In the first set of experiments, we verify if the RTT of a wormhole link is twice as much

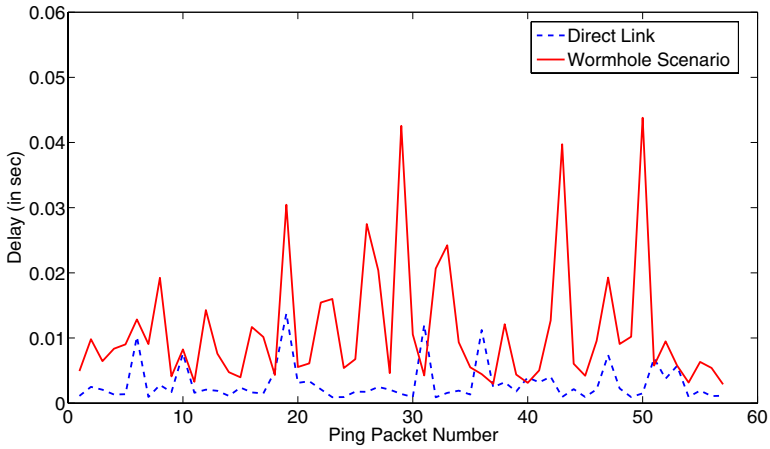

Fig. 4. Round trip time (wormhole link and normal link)

as that of a normal link. We set up a simple ad hoc network consisting of two PDAs running in ad hoc mode. We measure the average RTT when sending an ICMP ping packet from one PDA to another and receive an acknowledgment back for the same packet. In the second set of experiments, we mimic a wormhole attack where a packet sent from one PDA is captured at the first laptop, tunneled to the second laptop, and replayed at the second PDA. The rationale behind using a laptops is to mimic the high resource capability of a wormhole node.

We conduct both experiments for two minutes continuously and take the average of the results. Figure 4 show that the round trip time when the wormhole existed is much higher than that in normal case. The average RTT of sending a packet through wormhole link and a legitimate link was observed to be 11.09msec and 4.9msec, respectively. Thus the node can use the delay as an indicator to suspect any link.

Next, we use ns-2 [11] simulator to implement two algorithms in Suspicion phase. For Algrithm 1 we create a network topology and randomly pick a node S. We then create a wormhole link between S and a distant node D. Repeating the experiment many times we can select S with varying degree of neighbors. We then measure the RTT between the neighbors of S and calculate k (threshold) as described in Section 5.1. Comparison of the simulated values to the analytical value is shown in Figure 5. We observe that the ratio of the wormhole RTT to average RTT is always above the calculated threshold and hence we conclude that the threshold value we suggested is effective.

As for Algorithm 2, we design two tests to evaluate its performance:

- Test 1: The percentage of wormhole detection. Specifically, for all the wormhole links, how many of them will be detected by Algorithm 2. Note that if q is the percentage of wormhole links being detected then $1 - q$ is the percentage of false negative, i.e. number of links that escaped an algorithm.
- Test 2: The percentage of false positive, i.e. the percentage of direct links which are falsely detected as wormhole link.

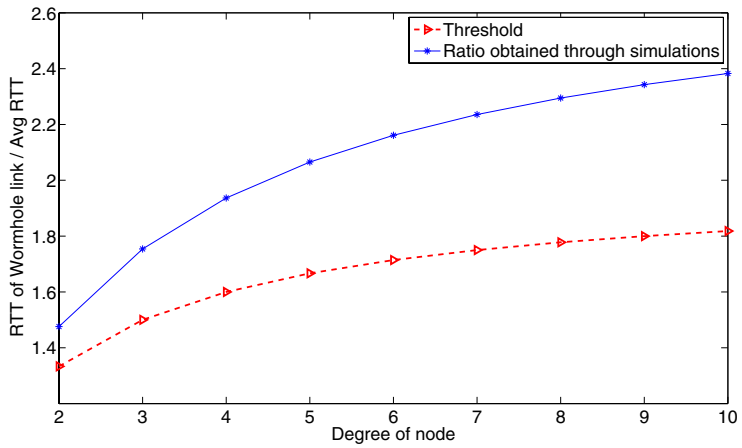

Fig. 5. Round trip time (simulated and analytical comparison)

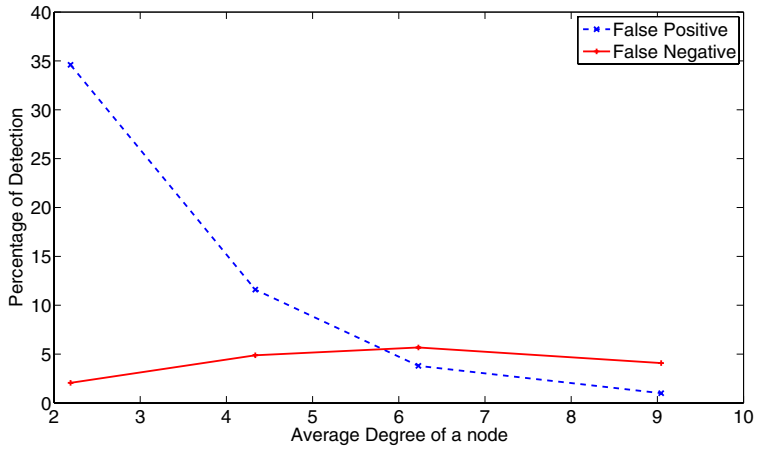

Fig. 6. Results of NeighborDetection algorithm

We set up a fixed network area of 2000m x 2000m with the transmission range of a node set to 250m. We conduct experiments by varying network density from 50 to 200 nodes. For Test 1, we randomly create a number of wormhole links in the network and apply Algorithm 2 to detect wormhole links. For Test 2, we pick a pair of directed neighbors (which is not under a wormhole attack) and apply Algorithm 2.

For each of the network size, we conduct multiple runs and take the average of the results. The final result is shown in Figure 6. The results show that in a dense network (200 nodes of 9.044 average degree), the chances of detecting wormhole is very high (92.92%) while the ratio of false positive is reduced closely to 1%. Our results in Figure 6 also show that Algorithm 2 has a high percentage of detecting wormhole links in

Test 1. These links will then be used in the Confirmation phase. While for Test 2 we observe a low result, which means that if a link is a real direct link, the algorithm is not likely to falsely interpret as a wormhole link. This would help in minimizing the resource (e.g. energy) consumption for communicating in the Confirmation phase.

Let the set of wormhole links that fail to be detected by Algorithm 1 and Algorithm 2 be $Fail_{Alg1}$ and $Fail_{Alg2}$, respectively. Since we move to Confirmation phase if either of the algorithms in Suspicion phase suspect a link, the set of false negative links of the whole Suspicion phase, i.e. the wormhole links that escape all algorithms, $Fail_{Suspicion}$, can be computed as:

- If $Fail_{Alg1}$ and $Fail_{Alg2}$ are mutually disjoint, then for the whole Suspicion phase, none of the wormhole links will be missed. Thus, $Fail_{Suspicion}$ will be empty.
- If $Fail_{Alg1}$ and $Fail_{Alg2}$ overlap, then some of wormhole links are missed by both algorithms. In this case, $Fail_{Suspicion}$ will be $Fail_{Alg1} \cap Fail_{Alg2}$. In the worst case, Sizeof($Fail_{Suspicion}$) = Min{Sizeof($Fail_{Alg1}$), Sizeof($Fail_{Alg2}$)}.

Given that one detection algorithm may work better than another in different situations, with the introduction of Wormeros, only a small fraction of wormhole will be able to escape all techniques applied in Suspicion phase. This is because, as we already show, Sizeof($Fail_{Suspicion}$) = Min{Sizeof($Fail_{Alg1}$), Sizeof($Fail_{Alg2}$), ..., Sizeof($Fail_{AlgN}$)} where N is the number of techniques.

As for false positive, if one of the techniques wrongly concludes that a normal link is suspicious, we move to Confirmation phase. Thus, the probability of false positive of the whole phase will be the sum of that of each technique. This is the trade-off of Detection phase in Wormeros: minimizing the probability of false negative while accepting higher probability of false positive.

Let us consider the performance of the whole framework. Let p_i be the probability of false positive for phase i ($i \in \{1, 2\}$). Similarly, let q_i be the probability of false negative for phase i ($i \in \{1, 2\}$). Then we have:

$$\text{P[fp]} = \text{P[false positive for framework]} = p_1 * p_2$$
$$\text{P[fn]} = \text{P[false nagative for framework]} = q_1 + (1 - q_1) * q_2$$

Interestingly enough, the probability of false positive is reduced significantly when combining two phases of Wormeros while the probability of false negative is almost the same as that in the first phase, because as analyzed the probability of false negative in the second phase is negligible. For example, let $p_1 = 0.1, p_2 = 0.01, q_1 = 0.03$ and $q_2 = 0.01$, then Pr[fp] $= 0.001$ and Pr[fn] $= 0.0397$. We can conclude that Wormeros handles very well both false negative and false positive.

7 Conclusions

The wormhole attack is considered to be a difficult attack to defend against. We propose Wormeros, a framework that uses simple techniques to identify the wormhole and then perform proper actions to confirm the existence of the attack. Through experiment and simulation, we make a compelling argument showing the ability of Wormeros to detect the wormhole attack. Our analysis further confirms the effectiveness of our framework.

References

1. Hu, Y.C., Perrig, A., Johnson, D.B.: Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In: Proceedings of IEEE InfoCom. (April 2003)
2. Hu, L., Evans, D.: Using Directional Antennas to Prevent Wormhole Attacks. In: Network and Distributed System Security Symposium (February 2003)
3. Ko, Y.B., Shankarkumar, V., Vaidya, N.: Medium access control protocols using directional antennas in ad hoc networks. In: Proceedings of IEEE InfoCom., pp. 13–21 (March 2000)
4. Čapkun, S., Buttyán, L., Hubaux, J.P.: SECTOR: secure tracking of node encounters in multi-hop wireless networks. In: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, pp. 21–32 (October 2003)
5. Eriksson, J., Krishnamurthy, S.V., Faloutsos, M.: Truelink: A practical countermeasure to the wormhole attack in wireless networks. In: Proceedings of IEEE ICNP, pp. 75–84 (November 2006)
6. Khalil, I.: LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multi-hop Wireless Networks. In: Proceedings of IEEE DSN, pp. 612–621 (June 2005)
7. Maheshwari, R., Gao, J., Das, S.R.: Detecting Wormhole Attacks in Wireless Networks Using Connectivity Information. In: Proceedings of IEEE InfoCom. (May 2007)
8. Chiu, H.S., Lui, K.S.: DelPHI: wormhole detection mechanism for ad hoc wireless networks. In: 1st International Symposium on Wireless Pervasive Computing (January 2006)
9. Tran, P.V., Hung, L.X., Lee, Y.K., Lee, S., Lee, H.: TTM: An Efficient Mechanism to Detect Wormhole Attacks in Wireless Ad-hoc Networks. In: 4th IEEE Consumer Communications and Networking Conference (January 2007)
10. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proceedings of ACM CCS, pp. 41–47 (November 2002)
11. The Network Simulator - ns-2, `http://www.isi.edu/nsnam/ns/`