LITEWORP: Detection and Isolation of the Wormhole Attack in Static Multihop Wireless Networks

Issa Khalil, Saurabh Bagchi, Ness B. Shroff
Dependable Computing Systems Lab (DCSL) & Center for Wireless Systems and Applications (CWSA)
School of Electrical & Computer Engineering, Purdue University
Email: {ikhalil, sbagchi, shroff}@purdue.edu

Corresponding author: Saurabh Bagchi, sbagchi@purdue.edu

Contact Info for Corresponding Author: 465 Northwestern Avenue, West Lafayette, Indiana 47907. USA.

Phone: 765-494-3362 Fax: 765-494-2706

Abstract

In multihop wireless systems, such as ad-hoc and sensor networks, the need for cooperation among nodes to relay each other's packets exposes them to a wide range of security attacks. A particularly devastating attack is known as the wormhole attack, where a malicious node records control and data traffic at one location and tunnels it to a colluding node far away, which replays it locally. This can either disrupt route establishment or make routes pass through the malicious nodes. In this paper, we present a lightweight countermeasure for the wormhole attack, called LITEWORP, which relies on overhearing neighbor communication. LITEWORP is particularly suitable for resource-constrained multihop wireless networks, such as sensor networks. Our solution allows detection of the wormhole, followed by isolation of the malicious nodes. Simulation results show that every wormhole is detected and isolated within a very short period of time over a large range of scenarios. The results also show that the fraction of packets lost due to the wormhole when LITEWORP is applied is negligible compared to the loss in an unprotected network. Simulation results bring out the configuration where no framing is possible, while still having high detection rate. Analysis is done to show the low resource consumption of LITEWORP, the low detection latency, and the likelihood of framing by malicious nodes.

Keywords: Wireless sensor and ad-hoc networks, neighbor watch, wormhole attack, malicious node detection, malicious node isolation.

1 Introduction

Ad-hoc and sensor networks are emerging as promising platforms for a variety of application areas in both military and civilian domains. These networks are especially attractive for scenarios where it is infeasible or expensive to deploy significant networking infrastructure. Initial research efforts have focused on the realization and practical implementation of these networks by focusing on their functional attributes, such as data aggregation protocols and routing protocols. However, the open nature of the wireless communication channels, the lack of infrastructure, the fast deployment practices, and the hostile environments where they may be deployed, make them vulnerable to a wide range of security attacks. These attacks could involve eavesdropping, message tampering, or identity spoofing, which have been addressed by customized cryptographic primitives in the wired domain. Alternately, attacks may be targeted at control or data traffic in wireless networks, such as the blackhole attack [5] and the rushing attack [9]. Since many multihop wireless environments are resource-constrained (e.g., bandwidth, power, or processing), providing detection and countermeasures to such attacks often turn out to be more challenging than in their wired counterparts.

A particularly severe security attack, called the wormhole attack, has been introduced in the context of ad-hoc networks [5], [7], [8], [30]. During this attack, a malicious node captures packets from one location in the network, and "tunnels" them to another malicious node at a distant point, which replays them locally. The tunnel can be established in many different ways, e.g., through an out-of-band hidden channel (e.g., a wired link), packet encapsulation, or high powered transmission. This makes the tunneled packet arrive either sooner or with a lesser number of hops compared to the packets transmitted over normal multihop routes. This creates the illusion that the two end points of the tunnel are very close to each other. A wormhole tunnel can actually be useful if used for forwarding all the packets. However, in its malicious incarnation, it is used by attacking nodes to subvert the correct operation of ad-hoc and sensor network routing protocols. The two malicious end points of the tunnel may use it to pass routing traffic to attract routes through them. They can then launch a variety of attacks against the data traffic flowing on the wormhole, such as selectively dropping the data packets. The wormhole attack can prevent two nodes from discovering legitimate routes greater than two hops away and thus disrupt network functionality. In addition, it may affect data aggregation and clustering protocols and location-based wireless security systems. Finally, it is worth noting that the wormhole attack can be launched even without having access to any cryptographic keys or compromising any legitimate node in the network [5], [7].

In previous paper [29], we present a simple lightweight protocol, called LITEWORP, to detect and mitigate wormhole attacks in static ad-hoc and sensor wireless networks. LITEWORP uses secure two-hop neighbor discovery and local monitoring of control traffic to detect nodes involved in the wormhole attack. It provides a countermeasure technique that isolates the malicious nodes from the network thereby removing their ability to cause future damage. We provide a novel taxonomy of the different ways in which wormhole attacks can be launched and show how LITEWORP can be used to handle all but one of these attack modes. LITEWORP has several features that make it especially suitable for resource-constrained wireless environments, such as sensor networks. LITEWORP does not require specialized hardware, such as directional antennas or fine granularity clocks. It does not require time synchronization between the nodes in the network. It does not increase the size of the network traffic, and incurs negligible bandwidth overhead, only at initialization and on detection of a wormhole. The lightweight feature of LITEWORP is in contrast to other countermeasures for wormhole attacks, which have requirements (e.g. directional antennas [8], highly accurate time measurement [21], specialized trusted nodes [30], and clock synchronization [7]) that often make them impractical for sensor networks and other classes of ad-hoc networks. Finally, in LITEWORP, detection and isolation are done judiciously to minimize the possibility of victimizing innocent nodes due to false alarms caused by natural collisions in the wireless medium or due to malicious framing.

In this paper, we present a coverage analysis of LITEWORP and show the relation between the number of nodes required for local monitoring, called *guards*, and the probability of false or missed detection. Moreover, we present an analysis for the isolation latency and the framing probability with various parameters such as the number of malicious nodes. We build a simulation model for LITEWORP using the network simulator *ns-2* and perform a comparative evaluation of a network with and without the technique. The results show that with a large number of guards, LITEWORP can achieve 98.9% non-malicious routes, with 12% of the network nodes compromised. For this configuration, the possibility of false detection (due to natural collisions) or framing (due to malicious reporting) is negligible. Further, the detection and isolation of the nodes involved in the wormhole can be achieved in a negligible time after the attack starts, and the cumulative number of lost packets and malicious routes established saturates with

time because wormholes are identified and isolated. Finally, we analyze the storage, computational, and bandwidth overheads incurred by LITEWORP, and demonstrate its lightweight nature.

The rest of the paper is organized as follows. Section 2 presents related work in the field of wormhole detection and mitigation. Section 3 describes the taxonomy of the wormhole attack modes. Section 4 describes local monitoring and isolation. Section 5 describes the LITEWORP defenses against the various modes of the wormhole attack. Section 6 presents analysis of the framing probability, isolation latency, coverage, and cost of LITEWORP. Section 7 presents simulation results. Finally, Section 8 discusses some extensions and concludes the paper.

2 Related Work

The wormhole attack in wireless networks was independently introduced by Dahill [1], Papadimitratos [2], and Hu [7]. An approach called RF watermarking [17] modulates the radio waveform in a specific pattern and any change to the pattern is used as the trigger for detection. This mechanism will fail to prevent a wormhole if the waveform is accurately captured at the receiving end of the wormhole and exactly replicated at the transmitting end.

Hu *et al.* [7] introduce the concept of geographical and temporal packet leashes for detecting wormholes. They define a leash to be any added information to the packet for the purpose of defending against the wormhole. The geographical leashes ensure that the recipient of the packet is within a certain distance from the sender. They require each node to know its own location and require all the nodes to have loosely synchronized clocks. The temporal leashes ensure that the packet has an upper bound on its lifetime, which restricts the maximum travel distance. They require that all nodes have tightly synchronized clocks. An implicit assumption is that packet processing, sending, and receiving delays are negligible. Both geographical and temporal leashes need to add authentication data to each packet to protect the leash, which add processing and communication overhead. In addition, a large amount of storage is needed at each node since a hash tree based authentication scheme (Merkle hash trees) is used [25]. Capkun *et al.* [21] present SECTOR, a set of mechanisms for the secure verification of the time of encounters between nodes in multihop wireless networks. They show how to detect wormhole attacks without requiring any clock synchronization through the use of MAD (Mutual Authentication with Distance-Bounding). Each node u estimates the distance to another node v by sending it a one bit challenge, which node v responds to instantaneously. Using the time of flight, node u detects if node v is a neighbor or not. The approach uses special hardware for the challenge request-response and accurate time measurements. Neither of the above two techniques nullifies the capacity of the compromised nodes from launching attacks in the future.

Hu and Evans [8] use directional antennas [18],[19] to *prevent* wormhole attacks. To thwart the wormhole, each node shares a secret key with every other node and maintains an updated list of its neighbors. Neighbor lists are built in a secure manner by using the direction in which a signal is heard from a neighbor with the assumption that the antennas on all the nodes are aligned. However, it only partially mitigates the wormhole problem. Specifically, it only prevents the kind of wormhole attacks in which malicious nodes try to deceive two nodes into believing that they are neighbors. This is only one of the five wormhole attack modes that we describe in Section 3. Moreover, the requirement of directional antennas on all nodes may be infeasible for certain deployments. Finally, the protocol may degrade the connectivity of the network by rejecting legitimate neighbors in their conservative approach to prevent wormholes from materializing.

Wang *et al.* [28] present a method for graphically visualizing the occurrence of wormholes in static sensor networks by reconstructing the lay-out of the sensors using multi-dimensional scaling. However, their approach is centralized and only detects the existence of wormholes but does not isolate malicious nodes involved in the attack. Lazos *et al.* [30] propose a technique for neighbor discovery that prevents external nodes from forming wormholes by using the references to *trusted specialized* guards (the guards are trusted, higher range, know their locations) and it prevents local nodes from forming the wormhole attack using a global preloaded key in the sensors.

Awerbuch *et al.* [20] present a protocol called ODSBR that does not prevent the wormhole from happening but tries to mitigate its consequences through discovery and avoidance. The technique suffers from the drawback that every single packet needs to be acknowledged by the destination and many packets could be lost before the wormhole is discovered.

Note that if it is possible for each node in the network to securely verify the locations of all its first-hop and second-hop communicating nodes, then the wormhole attack can be defeated. There are few solutions proposed in the literature for secure neighbor discovery. The approach by Evans [8] uses directional antennas on each node with precise alignment of the nodes. The approach by Perrig [9] is presented in the context of designing a route discovery component that is secure to the rushing attack. The approach relies on the time of flight and thus assumes very accurate time measurement and disregards all sources of delay other than the propagation delay. The MAC delay in

networks of even moderate density can make this assumption dubious. Many schemes use beacons sent by powerful nodes to enable location determination by other nodes. Sastry *et al.* [34] tackle the problem of a node securely verifying the location of possibly malicious beacon nodes that send spurious information about their own location. Their approach uses a very fast (e.g., radio frequency) and a relatively slow (e.g., ultrasound) signal to derive distance from the time delay. While this kind of capability can be mounted on a limited set of beacon nodes, it is infeasible to do this on all the nodes in the network.

This paper builds on our previous work in [29] and [32]. In [29], we introduced the concept of local monitoring and applied it to detection and isolation of wormhole attacks in static networks, the problem addressed in the current paper. In [32], we showed how local monitoring can be applied to detect the basic primitives (drop, delay, fabricate, and modify) that form the basis for many control attacks in ad hoc networks. In this paper, we introduce the concept of framing of good nodes through local monitoring and show through analysis and simulation the parameter settings that can be used to minimize framing. This paper also provides the analysis for the detection latency, both with and without independence of observations and modifies the analysis of missed and false isolation from [29] to take into account the possibility that evidence of misbehavior can come from direct observations or second hand reporting.

3 Wormhole Attack Modes

Wormhole attacks are particularly severe against many ad-hoc and sensor network routing protocols, such as the two ad-hoc on-demand routing protocols DSR [4] and AODV [14], and the sensor TinyOS beaconing routing protocol [5]. First, we demonstrate how a generic wormhole attack is launched against such routing protocols, using DSR as an example. In DSR, if a node, say *S*, needs to discover a route to a destination, say *D*, *S* floods the network with a route request packet. Any node that hears the request packet transmission, processes the packet, adds its identity to the source route, and rebroadcasts it. To limit the amount of flooding through the network, each node broadcasts only the first route request it receives and drops any further copies of the same request. For each route request that *D* receives, it generates a route reply and sends it back to *S*. The source *S* then selects the best path from the route replies; the best path could be either the path with the shortest number of hops or the path associated with the first arrived reply. In a malicious environment, this protocol may fail. When a malicious node at one part of the network hears the route request packet, it tunnels it to a second colluding party at a distant location near the destination. The second party then rebroadcasts the route request. The neighbors of the second colluding party receive the route request and drop any further legitimate requests that may arrive later on legitimate multihop paths. The result is that the routes between the source and the destination go through the two colluding nodes that will be said to have formed a wormhole between them. This prevents nodes from discovering legitimate paths that are more than two hops away. One way in which two colluding malicious nodes can involve themselves in a route is by giving the false illusion that the route through them is the shortest, even though they may be many hops away.

In the following subsections we classify the wormhole attack based on the techniques used for launching it.

3.1 Wormhole using Encapsulation

Consider Figure 1 in which nodes *A* and *B* try to discover the shortest path between them, in the presence of the two malicious nodes *X* and *Y*. Node *A* broadcasts a route request (*REQ*), *X* gets the *REQ* and encapsulates it in a packet destined to *Y* through the path that exists between *X* and *Y* (*U*-*V*-*W*-*Z*). Node *Y* demarshalls the packet, and rebroadcasts it again, which reaches *B*. Note that due to the packet encapsulation, the hop count does not increase during the traversal through *U*-*V*-*W*-*Z*. Concurrently, the *REQ* travels from *A* to *B* through *C*-*D*-*E*. Node *B* now has two routes, the first is four hops long (*A*-*C*-*D*-*E*-*B*), and the second is apparently three hops long (*A*-*X*-*Y*-*B*). Node *B* will choose the second route since it appears to be the shortest while in reality it is seven hops long. Any routing protocol that uses the metric of shortest path to choose the best route is vulnerable to this mode of wormhole attack.

This mode of the wormhole attack is easy to launch since the two ends of the wormhole do not need to have any cryptographic information, nor do they need any special capabilities, such as a high speed wire line link or a high power source. A simple way of countering this mode of attack is a by-product of the secure routing protocol ARAN [10], which chooses the fastest route reply rather than the one which claims the shortest number of hops. This was not a stated goal of ARAN, whose motivation was that a longer, less congested route is better than a shorter and congested route.

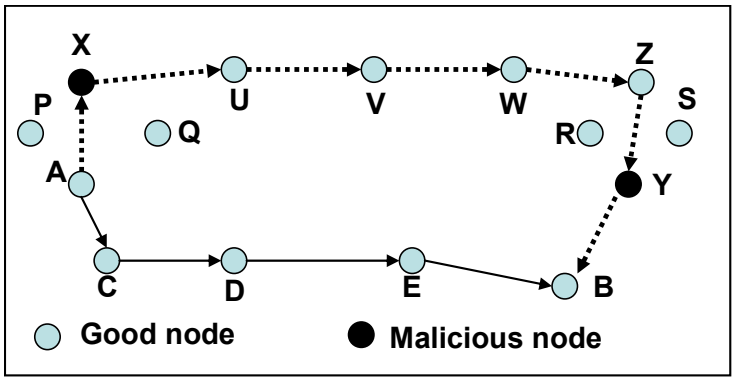

Figure 1: Wormhole through packet encapsulation

3.2 Wormhole using Out-of-Band Channel

This mode of the wormhole attack is launched by having an out-of-band high-bandwidth channel between the malicious nodes. This channel can be achieved, for example, by using a long-range directional wireless link or a direct wired link. This mode of attack is more difficult to launch than the previous one since it needs specialized hardware capability. Consider the scenario depicted in Figure 2. Node *A* sends a route request to node *B*, and nodes *X* and *Y* are malicious nodes having an out-of-band channel between them. Node *X* tunnels the route request to *Y*, which is a legitimate neighbor of *B*. Node *Y* broadcasts the packet to its neighbors, including *B*. *B* gets two route requests—*A*-*X*-*Y*-*B* and *A*-*C*-*D*-*E*-*F*-*B*. The first is both shorter and faster than the second, and is thus chosen by *B*.

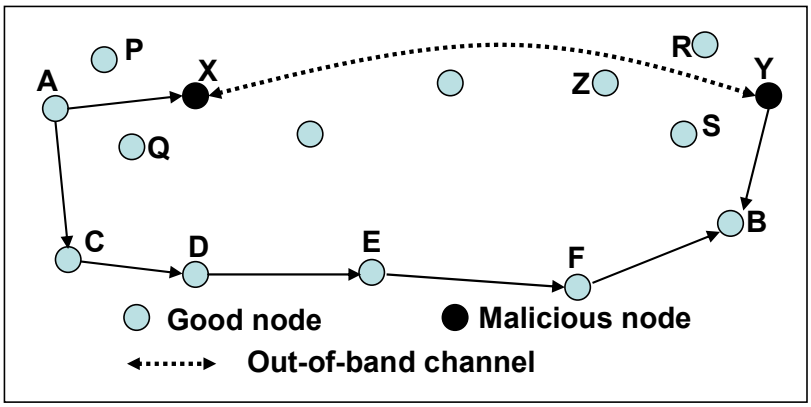

Figure 2: Wormhole through out-of-band channel

3.3 Wormhole with High Power Transmission

In this mode, when a single malicious node gets a route request, it broadcasts the request at a high power level, a capability which is not available to other nodes in the network. Any node that hears the high-power broadcast rebroadcasts it towards the destination. By this method, the malicious node increases its chance to be in the routes established between the source and the destination even without the participation of a colluding node. A simple method to mitigate this attack is possible if each node can accurately and securely measure the received signal strength and has models for signal propagation with distance. In that case, a node can independently determine if the transmission it receives is at a higher than allowable power level. However, this technique is approximate at best and dependent on environmental conditions. The local monitoring approach used in LITEWORP provides a more feasible defense against this mode.

3.4 Wormhole using Packet Relay

In this mode of the wormhole attack, a malicious node relays packets between two distant nodes to convince them that they are neighbors. It can be launched by even one malicious node. Cooperation by a greater number of malicious nodes serves to expand the neighbor list of a victim node to several hops. For example, assume that node

A and node *B* are two non-neighbor nodes with a malicious neighbor node *X*. Node *X* can relay packets between nodes *A* and *B* to give them the illusion that they are neighbors.

3.5 Wormhole using Protocol Deviations

Some routing protocols, such as ARAN [10], choose the route with the shortest delay in preference to the one with the shortest number of hops. During the route request forwarding, the nodes typically back off for a random amount of time before forwarding. This is motivated by the fact that the request forwarding is done by broadcasting and hence, reducing MAC layer collisions is important. A malicious node can create a wormhole by simply not complying with the protocol and broadcasting without backing off. The adversary's purpose is to let the request packet it forwards arrive first at the destination thereby increasing the chances of being included in the path. This is a special form of the rushing attack described in [9].

Table 1: Summary of wormhole attack modes

Mode name	Minimum # adversary nodes	Special requirements
Packet encapsulation	Two	None
Out-of-band channel	Two	Out-of-band link
High power transmission	One	High energy source
Packet relay	One	None
Protocol deviations	One	None

Summarizing, the different modes of the wormhole attack along with the associated requirements are given in Table 1. Many routing protocols, including secure ones [2], [6], are vulnerable to the wormhole attack (see [7] for review). Moreover, all the protocols that are used in building neighbor lists and, by extension, the routing protocols (e.g. DSDV [3], OLSR [15], and TBRPF [16]) that use these lists, are vulnerable as well.

4 Local Monitoring & Isolation

In this section, we describe the process for wormhole detection in LITEWORP followed by the process for isolation of the malicious nodes.

4.1 System Model and Assumptions

Attack Model: The wormhole is launched by a malicious node, which may be either an external node that does not have cryptographic keys, or an insider node, that possesses the keys. The insider node may be created, for example, by compromising a legitimate node. All these malicious nodes can exhibit Byzantine behavior and can collude amongst themselves. The malicious node can be a powerful entity that can establish out-of-band fast channels or have high-powered transmission capability.

System assumption*:* We assume that the communication links are bi-directional which means that if a node *A* can hear node *B* then *B* can hear *A*. We assume that a finite amount of time is required from a node's deployment for it to be compromised. We further assume that no external or internal malicious node exists before the completion of the first- and second-hop neighbor discovery. However, we can remove this assumption and use one of the protocols for secure neighbor discovery such as the one by Hu and Evans using directional antennas [8] or by using trusted and more powerful nodes as in [30]. There is an obvious tradeoff here between cost (advanced hardware resources) and benefit (more relaxed set of assumptions). We assume that the network has a static topology. This does not rule out route changes due to node failures, malicious node isolation, route evictions from the routing cache, or the change in the role that a node practices (e.g., cluster head, data aggregator, etc.). From the point of view of LITEWORP, incremental deployment of a node in the network is identical to having a mobile node move to its location and thus is not addressed here. The interested reader may refer to [33] for techniques to handle a mobile adversary node. LITEWORP requires each packet forwarder to explicitly announce the immediate source of the packet it is forwarding, i.e., the node from which it receives the packet. Finally, LITEWORP assumes a pre-distribution pair-wise key management protocol (e.g. [11] for ad-hoc networks and [12],[13] for sensor networks) such that any two nodes can acquire a key for secure communication.

4.2 Local Monitoring for Wormhole Defense

4.2.1 Information Structures

Building neighbor lists: This protocol is used to build the data structure of the first-hop neighbors of each node and the neighbors of each neighbor. The data structure is used in local monitoring to detect malicious nodes and in local response to isolate these nodes. A neighbor of a node, *X*, is any node that lies within the transmission range of *X*. As soon as a node, say *A*, is deployed in the field, it does a one-hop broadcast of a HELLO message. Any node, say *B*, that hears the message, sends back a reply to *A*. Node *A* accepts all the replies that arrive within a timeout. For each reply, *A* adds the responder to its neighbor list R_A. Then, *A* does a one-hop broadcast of a message containing the list R_A. When *B* hears the broadcast, it stores R_A. Hence, at the end of this neighbor discovery process, each node has a list of its direct neighbors and the neighbors of each of its direct neighbors. This process is performed only once in the lifetime of a node and is assumed to be secure. Henceforth, a node will not accept a packet from a node that is not a neighbor, nor forward to a node that is not a neighbor. Also, second-hop neighbor information is used to determine if a forwarded packet comes from a neighbor of the forwarder. .If a node *C* receives a packet forwarded by *B* purporting to come from *A* in the previous hop, *C* discards the packet if *A* is not a second-hop neighbor. Finally, *A* activates local monitoring immediately after building its first and second-hop neighbor lists.

Local monitoring: This module detects the wormhole attack and diagnoses the malicious nodes involved in launching it. Local monitoring starts immediately after the completion of neighbor discovery. It uses a collaborative detection strategy, where a node monitors the traffic going in and out of its neighbors.

For a node, say α, to be able to monitor a node say, β, α must be a neighbor of both β and the previous hop from β, say δ. If this is satisfied, we call α the guard node of β over the link from δ to β. This implies that α is the guard node for its entire outgoing links. For example, in Figure 3, nodes *M*, *N*, and *X* are the guard nodes of *A* over the link from *X* to *A*. Information for each packet sent from *X* to *A* is saved in a *watch buffer* at each guard. The information includes the packet identification and type, the packet source, the packet destination, the packet's immediate sender (*X*), and the packet's immediate receiver (*A*). The guards expect that *A* will forward the packet toward the ultimate destination, unless *A* is itself the destination. Each entry in the watch buffer is time stamped with a time threshold, τ, by which *A* must forward the packet. Each packet forwarded by *A* with *X* as a previous hop is checked for the corresponding information in the watch buffer.

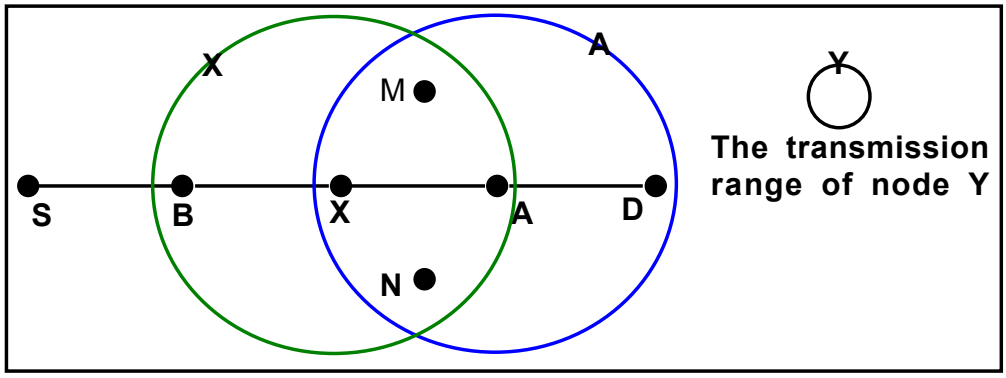

Figure 3: X, M, and N are guards of node A over the link from X to A

A malicious counter (*MalC(i,j)*) is maintained at each guard node, *i*, for a node, *j*, at the receiving end of each link that *i* is monitoring over a sliding window of length T_{win} that slides by δ units. *MalC(i,j)* is incremented for any malicious activity of *j* that is detected by *i*. The increment to *MalC* depends on the nature of the malicious activity detected, e.g., V_f for fabricating and V_d for dropping a control packet, being higher for more severe infractions. To account for intermittent natural failures that can occur at legitimate nodes, a node is determined to be misbehaving, only if the *MalC* goes above a threshold (C_t) over T_{win} time units. Of course, it is possible that there may not be any guard node for a given link. In that case, malicious behavior cannot be detected.

Now, we present the detection algorithm individually for each of the first four wormhole attack modes and show how existing approaches can be used to detect the fifth mode. However, prior to that, we give the isolation and the response algorithm that applies across all the attack modes

4.2.2 Response and Isolation Algorithm

1. When *MalC(α,A)* crosses C_t, α revokes A from its neighbor list, and sends to each neighbor of A, say D, an authenticated alert message indicating A is a suspected malicious node. This communication is authenticated using the shared key between α and D to prevent false accusations. Alternately, if the clocks of all the nodes in the network are loosely synchronized, α can do authenticated local two-hop broadcast as in [22] to inform the neighbors of A.
2. When D gets the alert, it verifies the authenticity of the alert message, that α is a first-hop neighbor of node A, and that A is D's neighbor. It then stores the identity of α in an *alert buffer* associated with A.
3. When D gets enough alert messages, γ, about A, it isolates A by marking A's status as revoked in the neighbor list. We call γ the *detection confidence index* (Section **Error! Reference source not found.**) of D. The detection confidence represents the minimum number of guard nodes that must report that a certain node, *j,* is malicious for a neighbor, *i*, of that node to isolate it, if *i* does not directly detect *j*. Note that the number of guards that report malicious activity is cumulative over time. A single node, due to the authentication mechanism, cannot generate more than one acceptable alert. *Framing* is the process by which an innocent node is proved to be malicious by a quorum of malicious nodes. A small value for γ increases the chance of successful framing of good nodes, while a large value of γ increases the rate of harm a malicious node causes in the network before being locally detected. If we set γ to be infinity it means that a node only trusts itself in revoking a suspicious node and thus the local framing probability goes to zero. *False alarm*, distinct from framing, is caused by a (legitimate) node mistaking another (legitimate) node to be malicious because of imperfections in the wireless channel, e.g., node *i* does not observe node *j* dutifully forwarding a packet.
4. After isolation, D does not accept or send any packet to a revoked node.

Note that this isolation is performed locally within the neighbors of the malicious node. This makes the response process quick and lightweight, and has the desired effect of removing the malicious nodes from the network.

5 Detecting Different Modes of Wormhole Attacks

5.1 Detecting out-of-band and packet encapsulation wormholes

A guard α of a node A over the link, say from X to A, saves information from the packet header of each control packet going over the link and time stamps it with the deadline τ. Node α overhears every packet going out of the receiver end of the link, A. For all the packets that node A claims have come from X, α looks up the entry in its watch buffer. If an entry is found, α drops that entry since the corresponding packet has been correctly forwarded. If an entry is not found, then A is accused of fabricating the packet. Therefore, α increments *MalC (α,A)* by V_f. If an entry for a packet sent from X to A stays in the watch buffer of α beyond τ, then A is accused of dropping the corresponding packet. Therefore, α increments *MalC(α,A)* by V_d.

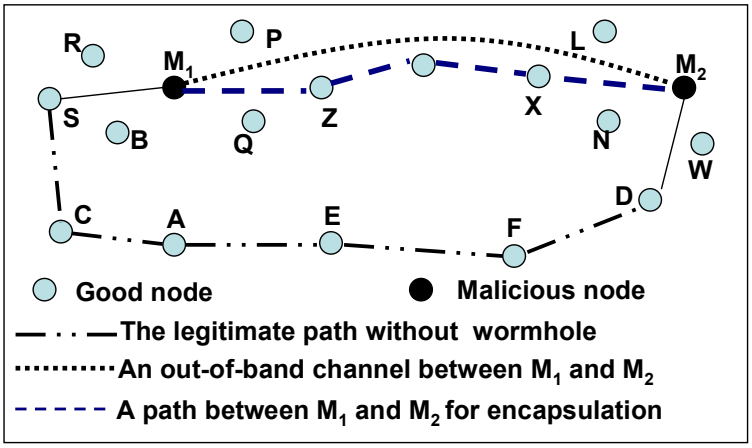

Figure 4: Wormhole detection for out-of-band and packet encapsulation modes

Consider the scenario in Figure 4 above. M_1 and M_2 are two malicious nodes wishing to establish a wormhole between the nodes S and D. When M_1 hears the *REQ* packet from S, it directs the packet to M_2. Node M_2 rebroadcasts the *REQ* packet after appending the identity of the previous hop from which it got the *REQ*. Node M_2 has two choices for the previous hop—either to append the identity of M_1, or append the identity of one of M_2's neighbors, say X. In the first choice all the neighbors of M_2 will reject the *REQ* because they all know, from the stored data structure of the two-hop neighbors, that M_1 is not a neighbor to M_2. In the second case, the knowledge of the first-hop and second-hop neighbor lists is not sufficient for all the guards to detect the attack. However, using local monitoring, all the guards of M_2 over the link from X to M_2 (X, N, and L) will detect M_2 as fabricating the route request since they do not have the information for the corresponding packet from X in their watch buffer. In both cases M_2 is detected, and the guards increment the *MalC* value of M_2.

In addition, the *REP* packet may also be used for detection of M_1 and M_2. When D gets the *REQ*, it generates a route reply packet, *REP*, and sends it back to M_2. The guards of M_2 over the link from D to M_2 (D, N, and W) overhear the *REP* and save an entry in their watch buffers. Node M_2 sends the route reply back to M_1 using the out-of-band channel or packet encapsulation. After τ time units, the timers in the watch buffers of the guards D, N, and W run out, and thus the guards detect M_2 as dropping the *REP* packet and increment the *MalC* of M_2. However, if M_2 is smarter, it can forward another copy of the *REP* through the regular slower route. In this case, *MalC* of M_2 is not incremented. When M_1 gets the *REP* from M_2, M_1 forwards it back to S after appending the identity of the previous hop. As before, M_1 has two choices—either to append the identity of M_2, or append the identity of one of M_1's neighbors, say Z. In the first choice, node S rejects the *REP* because it knows that M_2 is not a neighbor to M_1. Also, all the neighbors of M_1 know that M_2 is not a neighbor to M_1 and therefore increment the *MalC* of M_1. In the second case, all the guards of M_1 over the link from Z to M_1 detect M_1 as forging the *REP* since they don't have the corresponding entry from Z in their watch buffers.

5.2 Detecting high power transmission wormhole

This mode is detected using the first-hop neighbor list. Suppose a malicious node, say X, tries to use high power transmission to forward a packet P_1 to its final destination, or to cross multiple hops to introduce itself in the shortest path. Then all the nodes for which X is not in their neighbor lists detect the malicious behavior of X and reject P_1. For example, in the scenario shown in Figure 5, the inner circle represents the legitimate neighborhood of X. When X uses high power transmission, its coverage (the outer circle) reaches non-neighbor nodes such as A and B. Based on the absence of X in their neighbor lists, both A and B detect the malicious behavior of X.

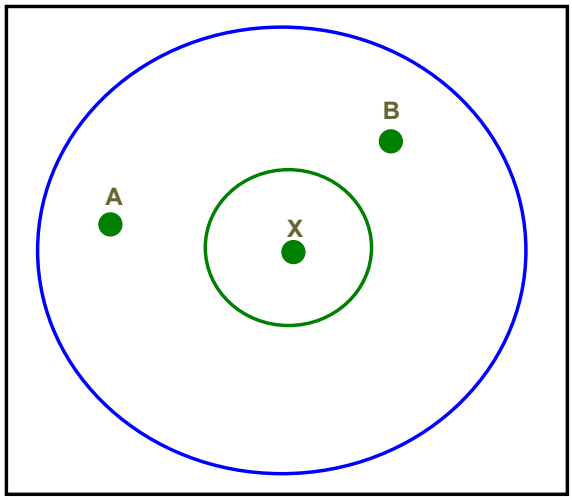

Figure 5: Single-node high power transmission mode of the wormhole attack

5.3 Detecting packet relay wormhole

This mode is detected using the stored neighbor lists at each node. Suppose a malicious node *X* (Figure 6) is a neighbor to two non-neighbor nodes *A* and *B* and tries to deceive them by relaying packets between them. Both *A* and *B* detect the malicious behavior of *X* since they know that they are not neighbors and reject the relayed packet.

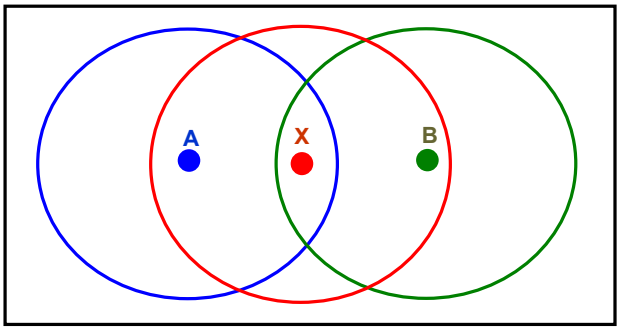

Figure 6: Single-node packet relay mode of the wormhole attack

5.4 Detecting protocol deviation wormhole

This mode *cannot* be detected using LITEWORP. Researchers have proposed techniques for countering selfish behavior in specific protocols. Selfishness refers to the property that nodes may tend to deny providing cooperating services to other nodes in order to save their own resources, e.g., battery power. Kyasanur *et al.* have addressed the problem of greediness at the MAC layer [23], while Buttyán *et al.* have addressed the problem in packet forwarding [21]. Hu *et al.* have proposed a solution to an attack, called the rushing attack, in which nodes greedily forward the route request passing through them without back off [9].

6 LITEWORP Analysis

6.1 Selection of the Detection Confidence Index (γ) Value

The value of γ is application-specific and may range between one and infinity. A small value for γ increases the chance of successful framing, while a large value of γ increases the rate of harm a malicious node causes the network before being locally detected and isolated. If we set γ to be infinity it means that a node only trusts itself in revoking a suspicious node, thus the local framing probability goes to zero. Any malicious node may be fully isolated as long as γ or more good-guards detect it. If the number of good guards is less than γ, then the node is only partially isolated from the network. Only the good guards that directly detect the malicious activity of the node isolate the malicious node. However, other neighbors of the malicious node continue to consider the malicious node as a legitimate node. The effect of partial or full isolation of malicious nodes is studied through the simulations (Figure 23, Figure 27, and Figure 28). In the simulations, the nodes are distributed randomly with a given density and the malicious nodes are also distributed randomly in the sensor field. The simulations include the case when the number of good guards of a node may fall below γ which negatively impacts isolation latency and the delivery ratio. Thus, the output metrics such as the drop ratio and the detection coverage are affected when number of guards drop below γ. However, the protocol as a whole does not break since this case may rarely occur for a reasonable ratio of malicious nodes. Moreover, based on our mathematical analysis presented in Section 6, we examine the effect of changing the detection confidence (γ) in the network. Our simulation and analytical results indicate that a value of γ equal to infinity provides a relatively high performance. Our recommendation of infinite value of γ is dependent on the experimental condition, specifically that the volume of traffic on the different outgoing links is statistically equal. Hence, γ can be looked upon as a design parameter in LITEWORP to tune its performance according to the application needs.

6.2 Coverage Analysis

In this section, we characterize the probability of missed detection and false detection as the network density increases and as the detection confidence index γ varies. The results provide some interesting insights. For example, we are able to compute the required network density d to detect p% of the wormhole attacks for a given γ.

Consider a homogeneous network of nodes where the nodes are uniformly distributed in the field. For simplicity, we assume that the field is large enough that edge effects can be neglected in our analysis. Consider any two randomly selected neighbor nodes, *S* and *D*, as shown in Figure 7(a). Nodes *S* and *D* are separated by a distance *X*, and the communication range is *r*. *X* is a random variable that has the probability density function of $f_X(x) = 2x/r^2$ with range (0,*r*). This follows from the assumption of uniform distribution of the nodes.

The guard nodes for the communication between *S* and *D* are those nodes that lie within the communication range of *S* and *D*, the shaded area in Figure 7(a). This area is given by

$$Area(X) = 2r^2 \cos^{-1}\left(\frac{X}{2r}\right) - X\sqrt{r^2 - \frac{X^2}{4}} \tag{1}$$

The minimum value of the *Area(X), Area*$_{min}$, is when $X = r$. Therefore, the minimum number of guards is

$$g_{\min} = Area_{\min} d = 1.23 r^2 d \tag{2}$$

The expected value of the area is

$$E[Area(X)] = \int_0^r \left\{ 2r^2 \cos^{-1}\left(\frac{x}{2r}\right) - (2x)\sqrt{r^2 - \frac{x^2}{4}} \right\} \left(\frac{2x}{r^2}\right) dx \approx 1.84 r^2 \tag{3}$$

Therefore, the expected number of guards is

$$g = E[Area(X)]d = \lfloor 1.84 r^2 d \rfloor \tag{4}$$

The number of neighbors of a node is given by $N_B = \pi r^2 d$, therefore,

$$g \approx \lfloor 0.59 N_B \rfloor \tag{5}$$

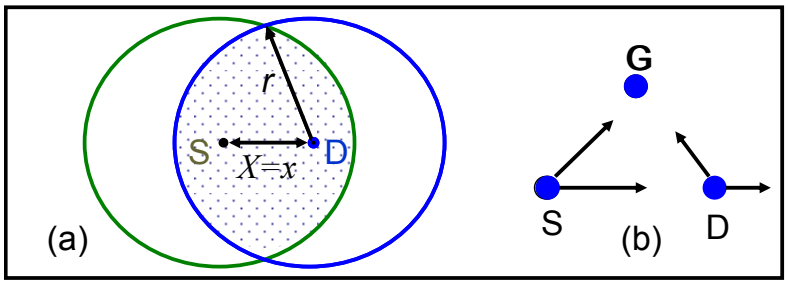

Figure 7: (a) The area where a node can guard the link S→D; (b) Illustration for detection accuracy

Now, as in [26] where IEEE 802.11 was analyzed, we assume that each packet collides on the channel independently with a constant probability P_C. As shown in Figure 7(b), a guard *G* will not detect a fabricated packet sent by *D*, claiming it was received from *S*, if *G* experienced a collision at the time that *D* transmits. Thus, the probability of missed detection is P_C. Assume that *S* sends ψ packets to be forwarded by *D* within a time window T_{win}. Assume that *D* selectively fabricates (to evade detection) packets with probability P_{fab}. Then, the number of packet fabrications (μ) that occur within T_{win} is $\psi \bullet P_{fab}$. Also assume that the *MalC* threshold over time window of T_{win} is β and each malicious activity increases the *MalC* by one. Then, using the binomial distribution, the probability of detection by direct observation at a guard (henceforth shortened as "direct detection") is given by,

$$P_{direct}(\beta \mid \mu) = \sum_{i=\beta}^{\mu} \binom{\mu}{i} (1 - P_C)^i (P_C)^{\mu - i} \tag{6}$$

Now consider the case of detection through evidence furnished by γ or more guards, shortened as "indirect detection". Assuming independence of collision events among the different guards, the probability that at least γ of the guards generate an alert is given by

$$P_{indirect}(\gamma) = \sum_{i=\gamma}^{g} \binom{g}{i} \left(P_{direct}(\beta \mid \mu)\right)^i \left(1 - P_{direct}(\beta \mid \mu)\right)^{g-i} = \frac{B(P_{direct}(\beta \mid \mu), \gamma, g - \gamma + 1)}{B(\gamma, g - \gamma + 1)}$$
$$= \frac{g!}{(\gamma - 1)!(g - \gamma)!} \int_0^{P_{direct}(\beta \mid \mu)} u^{\gamma - 1} (1 - u)^{g - \gamma} du \tag{7}$$

Where, $B(\gamma, g-\gamma+1)$ is the Beta function and $B(P_{direct}(\beta \mid \mu); \gamma, g-\gamma+1)$ is the incomplete Beta function. This gives the probability of indirect detection at a guard. Therefore, the probability of detection at a guard is the sum of the probability of direct detection and the conditional probability of indirect detection given that no direct detection occurs,

$$P_{detect} = P_{direct}(\beta \mid \mu) + P_{indirect}(\gamma) - P_{direct}(\beta \mid \mu) P_{indirect}(\gamma) \quad (8)$$

Based on Equation (8), Figure 8 shows the probability of detection at a guard as a function of the average number of neighbors with $\mu = 7$, $\beta = 4$, $\gamma = 3$, $P_{fab} = 1$, the number of compromised nodes $M = 2$, and $P_C = 0.05$ at $N_B = 3$. The number of guards is determined from N_B using Equation 5. Thereafter, P_C is assumed to increase linearly with the number of neighbors. Since the number of guards increases as the number of neighbors increases, the probability of indirect detection increases since it becomes easier to get the alarm from γ guards. However, the collision probability also increases with the number of neighbors, and thus the probability of direct detection starts to fall rapidly beyond a point which in turn decreases the indirect detection and the overall detection at a guard. However, note that the detection is still high (above 98.5%) at the relatively high density of each node having 35 neighbors since the reduction in the direct detection capability is compensated by the indirect detection.

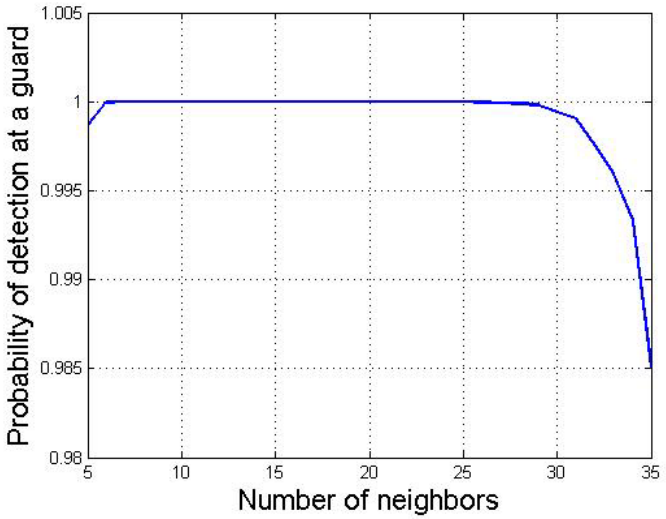

Figure 8: Probability of detection at a guard against N_B

Figure 9 shows the probability of detecting the wormhole attack against γ with $\mu = 7$, $\beta = 4$, $N_B = 20$, the number of compromised nodes $M = 2$, and $P_C = 0.33$. As γ increases, the probability of indirect detection at a guard decreases since it becomes harder to reach consensus among all the γ guard nodes. Therefore, the probability of detection decreases rapidly with increasing γ. However, note that the probability of detection is still high even at the lowest point (above 0.88) since the probability of direct detection is not affected by γ.

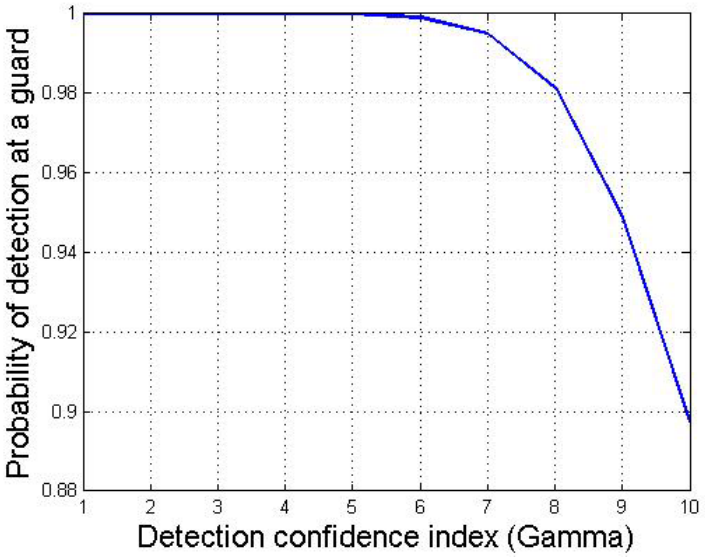

Figure 9: Probability of wormhole detection at a guard against γ

Recollect that false alarm is caused by a (legitimate) node mistaking another (legitimate) node to be malicious because of imperfections in the wireless channel. As shown in Figure 7(b), a false alarm occurs when *D* receives a packet sent from *S*, while *G* does not receive that packet, and later, *G* receives the corresponding packet forwarded by *D*. Thus, the probability of false alarm is $P_{FA} = P_C(1-P_C)^2$. Assume that *S* sends ψ packets to *D* for forwarding, within T_{win}. The probability that *D* is falsely accused directly by a guard is the probability that β or more packets are falsely suspected as fabricated. Therefore, using the binomial distribution, the probability of direct false alarm (P_{DF}) is given by,

$$P_{DF}(\beta \mid \psi) = \sum_{i=\beta}^{\psi} \binom{\psi}{i} (P_{FA})^i (1-P_{FA})^{\psi-i} \tag{9}$$

The probability of indirect false alarm (P_{IF}) is the probability that at least γ guards generate false alarms, which is given by

$$P_{IF}(\gamma) = \sum_{i=\gamma}^{g} \binom{g}{i} (P_{DF}(\beta \mid \psi))^i (1-P_{DF}(\beta \mid \psi))^{g-i} = \frac{\beta(P_{DF}(\beta \mid \psi), \gamma, g-\gamma+1)}{\beta(\gamma, g-\gamma+1)}$$
$$= \frac{g!}{(\gamma-1)!(g-\gamma)!} \int_0^{P_{DF}(\beta \mid \psi)} u^{\gamma-1}(1-u)^{g-\gamma}\, du \tag{10}$$

The probability of false alarm at a guard is given by the sum of the probability of direct false alarm and the conditional probability of indirect false alarm given that no direct false alarm occurs

$$P_{false} = P_{DF}(\beta \mid \psi) + P_{IF}(\gamma) - P_{DF}(\beta \mid \psi) P_{IF}(\gamma) \tag{11}$$

Based on Equation (11), Figure 10 shows the probability of false alarm at a guard as a function of the number of nodes for the same parameters as in Figure 8. The non-monotonic nature of the plot can be explained as follows. As the number of neighbors increases, so does the number of guards. Initially, this increases the probability that at least γ guards miss the packet from *S* to the guard but not from *D* to the guard, leading to increase in indirect false detection. But beyond a point, the increase in the number of neighbors increases the collision probability. This increases the probability that both of these packets are missed at the guard and thus does not lead to false detection. The worst-case false alarm probability is still low (less than 1.2×10^{-3}).

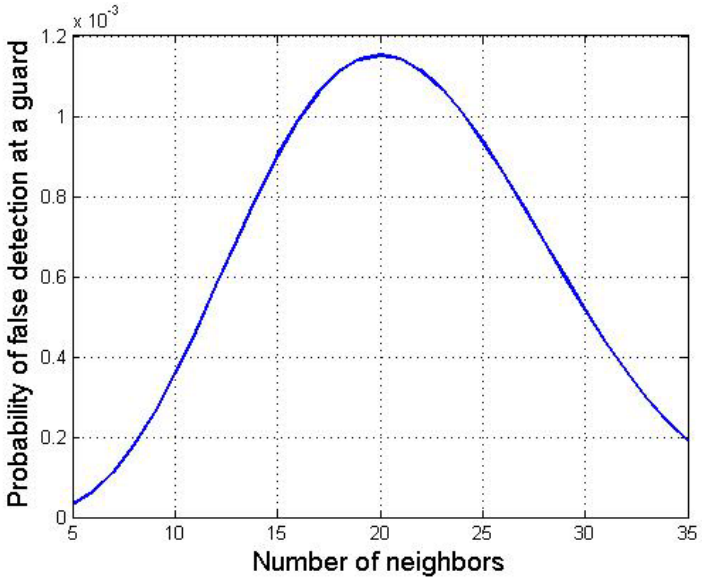

Figure 10: Probability of false alarm at a guard against N_B

Figure 11 shows the probability of false alarm against γ with $P_C = 0.05$, β=4, μ=7, and N_B=20. As γ increases, the probability of false detection decreases since it becomes harder to reach consensus among all the γ guard nodes.

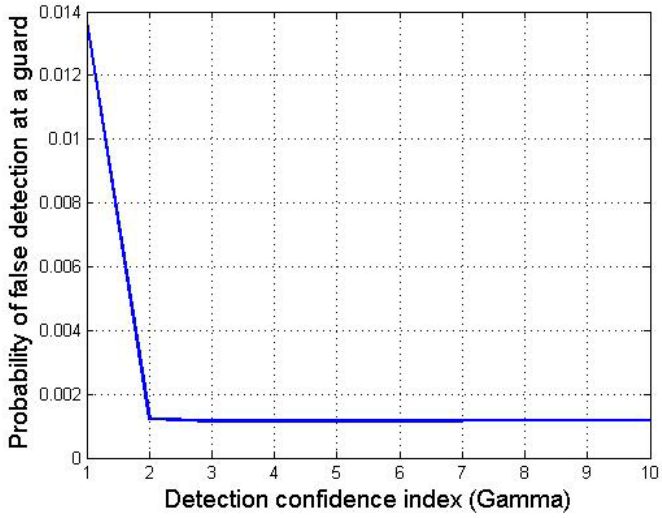

Figure 11: Probability of false alarm at a guard against γ

6.3 Analysis of a Node being Framed

Let N be the total number of nodes in the network, N_m be the number of malicious nodes, P_m=N_m/N be the probability that a node gets compromised, d be the density of nodes in the network, r be the range of communication, and $N_B = \pi r^2 d$ be the number of neighbors of a node.

Using the binomial distribution and assuming that false detection is zero, then, the probability that a good node X is locally framed equals the probability that there are at least γ malicious nodes among X's neighbors which is given by the following equation. Here, we are assuming the worst case in which all the malicious nodes are concurrently trying to frame the good node,

$$P_{frame}(\gamma) = \sum_{i=\gamma}^{N_b} \binom{N_B}{i} P_m^i \left(1 - P_m\right)^{N_B - i} \tag{12}$$

The probability of node framing (P_{frame}) as a function of the probability of node compromise for $\gamma = 5$ and $N_B = 7$ is plotted on Figure 12. From the figure, we see that the probability of framing increases exponentially with the probability of node compromise but up to the upper end of the range, it is still less than 0.03.

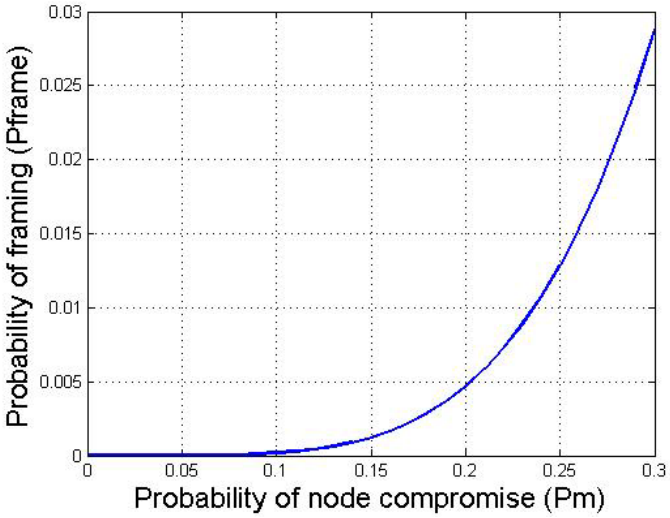

Figure 12: Probability of node framing against the probability of compromising a given node (γ=5, N_B=6)

6.4 Detection Latency Analysis

Here, we analyze the amount of time it takes to detect a malicious node. Assume the traffic distribution and the bandwidth capacity allows a maximum of μ packets to be forwarded by a malicious node M within a time window T_{win}. Assume that M selectively fabricates (to evade detection) packets with probability P_{fab}. Let G be the guard node of M over the link from X to M that collects and keeps a malicious counter ($MalC(G,M)$) for M over a window of length T_{win} which slides by δ units, Figure 13. Assume the $MalC$ threshold C_t over this time window is β and that each malicious activity increases the $MalC$ by one. Let $T_{win}/\delta = \eta$. When $\eta=1$ in Figure 13 (a), the sliding windows are non-overlapping and therefore, the events detected in any two windows are independent.

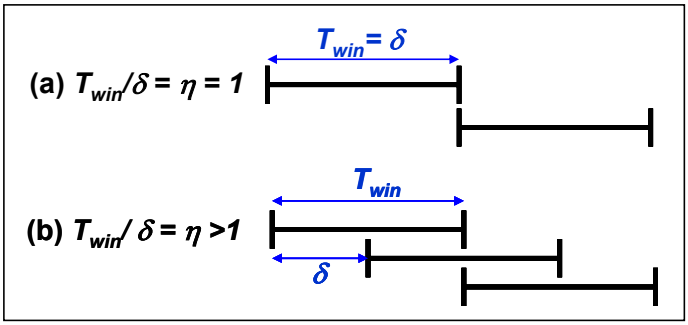

Figure 13: Sliding window illustration

Using the binomial distribution, the probability that G detects M during a certain time window (P_{gdM}) equals the probability that M fabricates at least β packets within T_{win}, which is given by

$$P_{gdM} = \sum_{i=\beta}^{\mu} \binom{\mu}{i} \left(1 - P_{fab}\right)^{\mu - i} P_{fab}^{\ i} \tag{13}$$

The expected time of detection is calculated from the number of T_{win} time slots (N_{ts}) that pass before the guard G detects the malicious node M. The probability that $N_{ts} = k$ is

$$P(N_{ts} = k) = P_{gdM} \left(1 - P_{gdM}\right)^{k-1} \tag{14}$$

Using Bernoulli trials, the expected value for N_{ts} is given by $E[N_{ts}] = 1/P_{gdM}$. The expected number of time slots ($E[N_{ts}]$) before a single guard detects a malicious node is plotted in Figure 14. The plot shows that the latency decreases very fast with increasing probability of malicious behavior.

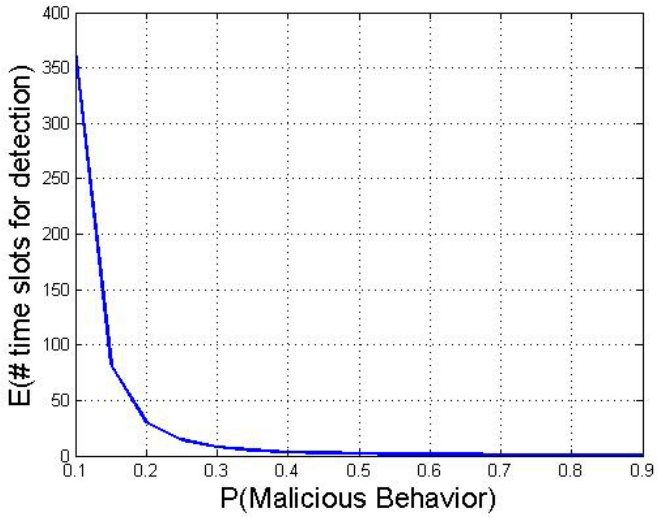

Figure 14: Expected number of time slots $E[N_{ts}]$ before a single guard detects a malicious node

For the case with overlapping sliding windows (η>1), Figure 13(b), the analysis becomes more difficult and we use Martingale Theory [31] to obtain bounds on the delay. Here, we assume rate-based detection, i.e., a node is determined to be malicious if the rate of malicious activities goes above a threshold α (Figure 15). As Figure 15 shows, after each δ time units, each guard checks the total number of malicious events over the past T_{win} time units and if it crosses the threshold rate (α), the guard marks the respective node as malicious. The dots in Figure 15 represent the malicious events rate that is calculated by the guard at the specific time point. We present this analysis for $\gamma = \infty$ since it eliminates framing and is shown to give reasonable detection rates as shown through the simulations (Section 7).

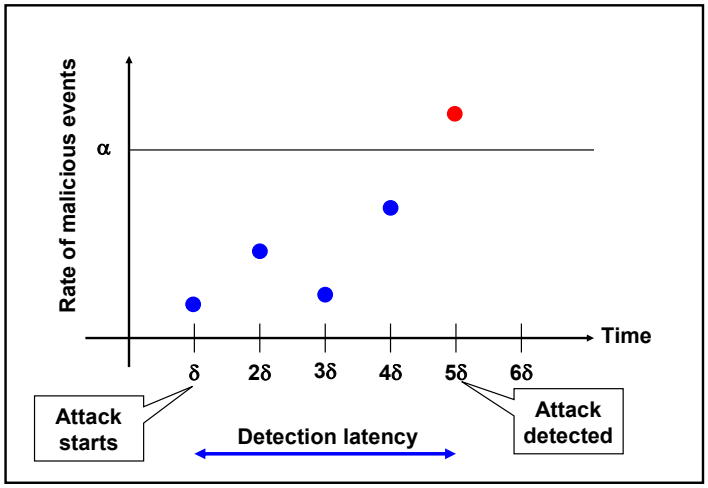

Figure 15: Illustration of rate-based detection with overlapping windows

Let X_i be an i.i.d. Bernoulli random variable that tracks the number of malicious actions by a node, such that X_i=1 (malicious activity) with probability λ and zero, otherwise. Thus, $E[X_i] = \lambda$. Consider that the guard observes the node for N_{act} activities (packet forwarding actions). Define

$$Z_{N_{act}} = \sum_{i=1}^{N_{act}} X_i - N_{act}\lambda \quad (15)$$

Then it can easily be shown that $Z_{N_{act}}$ is a zero-mean *martingale process*. Similarly, $Y_{N_{act}}$ defined below is also a zero-mean martingale process,

$$Y_{N_{act}} = \sum_{i=2}^{N_{act}} X_i - (N_{act} - 1)\lambda \tag{16}$$

Now, let N_0 be the number of activities at which the guard detects the node to be malicious. Then,

$$N_0 = \min_{N_{act}} \sum_{i=1}^{N_{act}} X_i \geq \alpha N_{act} \tag{17}$$

Our goal is to find $E[N_0]$. From elementary probability,

$$E[N_0] = E[N_0 \mid N_0 = 1] \cdot P(N_0 = 1) + E[N_0 \mid N_0 > 1] \cdot P(N_0 > 1) \tag{18}$$

Note that $E[N_0|N_0=1]P(N_0=1) = 1 \times \lambda = \lambda$. Also $P(N_0>1) = P(X_1=0) = 1-\lambda$.

Next we find $E[N_0|N_0>1]$. Note that since $Y_{N_{act}}$ is a martingale, using the Optional Stopping Theorem [31], $E[Y_{N_0}] = E[Y_2] = 0$. Also, note that given $N_0>1$,

$$N_0 = \min_{N_{act}} \sum_{i=2}^{N_{act}} X_i \geq \alpha N_{act} \tag{19}$$

This means that given $N_0>1$,

$$N_0 = \min_{N_{act}} Y_{N_{act}} + (N_{act} - 1)\lambda \geq \alpha N_{act} \tag{20}$$

In other words, $Y_{N_0} \geq (\alpha-\lambda)N_0 + \lambda$. Taking expectations on both sides,

$$E[Y_{N_0} \mid N_0 > 1] \geq (\alpha - \lambda) \cdot E[N_0] + \lambda \Rightarrow (\alpha - \lambda) \cdot E[N_0] + \lambda \leq 0 \Rightarrow E[N_0 \mid N_0 > 1] \geq \lambda / (\lambda - \alpha) \tag{21}$$

Therefore, de-conditioning, we get the lower bound as

$$E[N_0] \geq \lambda + \lambda \cdot (1 - \lambda) / (\lambda - \alpha) \tag{22}$$

For the upper bound we can repeat the arguments. Therefore, define

$$Z_{N_0} + \lambda \cdot N_0 < \alpha \cdot N_0 + 1 \tag{23}$$

The last term is because $X_i \leq 1$. Now, choosing α such that, $\lambda < \alpha$ (i.e., the rate of malicious activity is less than the detection threshold) and taking expectations we obtain

$$0 + E[N_0] \cdot (\lambda - \alpha) < 1 \Rightarrow E[N_0] < 1/(\lambda - \alpha) \tag{24}$$

Therefore, the bounds for the expected number of activities after which the guard will detect the node as malicious is,

$$\lambda \cdot (1 - \alpha) / (\lambda - \alpha) < E[N_0] < 1/(\lambda - \alpha) \tag{25}$$

We plot Equation 23 in Figure 16 and find that the bounds asymptotically converge and exist only for $\lambda > \alpha$.

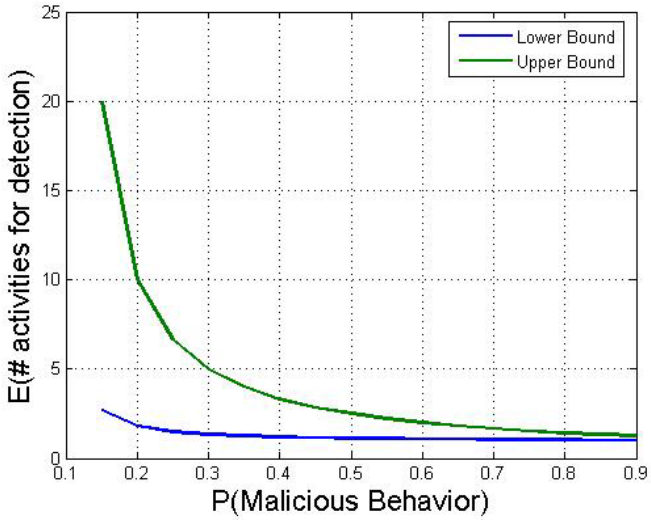

Figure 16: Lower and upper bound for expected number of activities before a malicious node is detected by a guard

6.5 Cost Analysis

In this section, we show the memory, the computation, and the bandwidth overhead of LITEWORP to evaluate its suitability to resource-constrained environments.

Memory overhead: We need to store the first and the second-hop neighbor lists, the watch buffer, and the alert buffer. The identity of a node in the network is 4 bytes. Reusing the notation from the previous section, the size of neighbor list is $NBL = \pi r^2 d$ entries. Each entry in the *NBL* needs 5 bytes; 4 for identity of the neighbor and 1 for the *MalC* associated with that neighbor. So the total *NBL* storage, $NBLS=5(\pi r^2 d)^2$. For example, for an average of 10 neighbors per node, *NBLS* is less than half a kilobyte. The alert buffer has γ number of 4 byte entries. The watch buffer size depends on the average number of hops between a source-destination pair, *h*, the frequency of route establishment, *f*, as well as the density of the nodes, *d*.

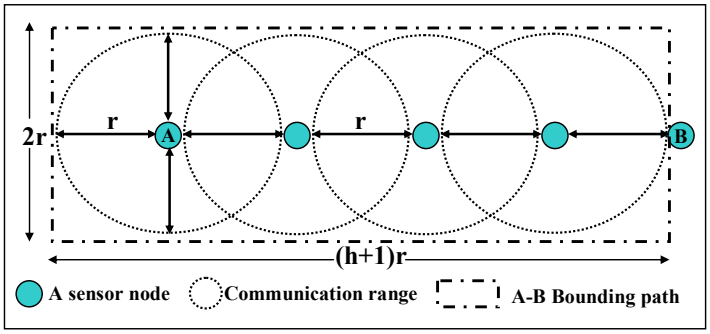

Figure 17: The average number of nodes involved in the watch of a route reply

To find the average number of nodes involved in watching a *REP*, we create a rectangular bounding box containing nodes that may overhear the *REP* sent from *A* to *B* (Figure 17). This is an overestimate since we use a square that circumscribes the circular transmission range. The number of nodes involved in monitoring is $N_{REP} = 2r^2(h+1)d$. Thus, given *N* as the total number of nodes in the network, each node is involved in watching $(N_{REP} / N)f$ route replies per unit time.

For example, if N=100 nodes, h = 4 hops, and f = 1 route every 4 time units, then N_{REP} = 17, and each node watches only 4 route replies every 100 time units. Because the time τ for which the packet is kept in the watch buffer is relatively small (may be less than one time unit), a watch buffer size of 4 entries is more than enough for this example. Each entry in the watch buffer is 20 bytes: 4 bytes each for the immediate source, the immediate destination, and the original source, and 8 bytes for the sequence number of the *REP*. If we include the route request in the watch, then each node will be involved in watching $f + (N_{REP} / N)f$. That requires each node to watch 4 packets every 16 time units; again 4 entries are still sufficient for the watch buffer.

Computation and bandwidth overhead: Each watched route reply requires (i) one lookup for the current source and the current destination in the neighbor list, (ii) adding an entry to the watch buffer (incoming) or deleting an entry from the watch buffer (outgoing), and (iii) may be another addition and deletion from the watch buffer (if a node is a guard for two consecutive links). Since the size of the watch buffer and the neighbor list structure are relatively small, the computation time required for these operations is negligible. For example, a lookup in a 100 entry buffer takes the MICA mote with an Atmega128 4 MHZ processor, about 2μ seconds. The bandwidth overhead is incurred after deployment of a node for neighbor discovery and in the case of wormhole detection for informing the neighbors of the detected node. This is therefore a negligible fraction of the total bandwidth over the lifetime of the network.

From the above analysis, we can conclude that LITEWORP has relatively modest memory, computation, and bandwidth overhead. This makes it especially suitable for resource-constrained sensor and ad-hoc networks.

7 Simulation Results

We use the *ns-2* simulation environment [27] to simulate a data exchange protocol, individually in the baseline case without any protection, and with LITEWORP. We distribute the nodes randomly over a square field with a fixed average node density. Thus, the field size varies (80×80 m to 204×204 m) with the number of nodes. We use a generic on-demand shortest path routing that floods route requests and unicasts route replies in the reverse direction. A route, once established, is not used forever but is evicted from the cache after a timeout period expires ($TOut_{Route}$). When a malicious node hears a route request, it directs the request to all the malicious nodes in the network using an out-of-band channel or using packet encapsulation. For packet encapsulation, we assume that the colluding nodes always have a route between them. We simulate the out-of-band channel by letting the malicious nodes deliver the packets instantaneously to their colluding parties. These two schemes exercise the principal feature of LITEWORP, namely, local monitoring and are the most challenging to mitigate. Hence, we simulate them in preference to other modes of attack. After a wormhole is established, the malicious nodes at each end of the wormhole drop all the packets forwarded to them. Furthermore, a malicious node always frames its good neighbors.

The simulation also accounts for losses due to natural collisions. The guards inform all the neighbors of the detected malicious node through multiple unicasts. For each run, malicious nodes are chosen at random such that they are more than 2 hops away from each other.

Input parameter: Each node acts as a data source and generates data using an exponential random variable with inter-arrival rate ϕ. The destination is chosen at random and is changed using an exponential random distribution with rate ξ. We use N_M for the number of malicious nodes, γ for the detection confidence, and N for the total number of nodes. The input parameters with the experimental values are given in Table 2. A design parameter in LITEWORP is the increment to the malicious counter value upon detecting a malicious event. On the one hand, we want the increment to be large for higher detection probability, fast detection, and small watch buffer size. On the other hand, we want the increment to be small to reduce the percentage of false alarms. We conduct an experiment to design the malicious counter increment. We choose the increment as the lower of the two points–the point where the percentage detection reaches its maxima and the point where the knee of the false detection curve lies. This gives us a reasonable combination of low false alarm rate and high detection rate. The value of the *MalC* increment used for the experiments is given in Table 2.

Output parameters: The output parameters include (i) the *isolation latency*, which is defined as the time between when the node performs its first malicious action to the time by which *all* the neighbors of the node have isolated it (ii) the fraction of data packets dropped/received due to the wormhole to the data packet sent (drop ratio/delivery ratio), (iii) the fraction of malicious routes to the total number of routes established. This parameter quantifies the amount of harm caused by the malicious nodes, (iv) the percentage of framing, which is defined as the percentage of the number of good nodes that could be framed to the total number of nodes, (v) the percentage of false isolation, which is defined as the percentage of the number of nodes that have been isolated due to natural causes to the total number of nodes, (vi) and the percentage of malicious node isolation (% true isolation), which is defined as the number of malicious nodes isolated to the total number of malicious nodes.

All the output parameters that we present here are measured at the end of the simulation time (1500 seconds) unless otherwise stated. The output parameters are obtained by averaging over 30 runs. Finally, the figures we present are for the 100-node scenario unless otherwise stated.

Table 2: Input parameters for LITEWORP simulation

Param.	Value	Param.	Value
Tx Range (r)	30 m	γ	3,5,7,infinity (default = 3)
MalC increment	10	ϕ	0.2
$TOut_{Route}$	50 s	N_M	0-6 (default 4)
C_t	150	τ	0.5 sec
# nodes (N)	20,50,100,150 (default 100)	BW	40 kbps
ζ	0.02	T_{win}	200

Data Packet Drop: Figure 18 shows the number of packets dropped as a function of the simulation time for 2 and 4 colluding nodes both with LITEWORP and without LITEWORP with γ =3. The attack is started 50 sec after the start of

the simulation. Since the numbers are vastly different in the two cases, they are shown on separate Y-axes; the axis on the left corresponds to the baseline case and the axis to the right corresponds to the system using LITEWORP. In the baseline case, since wormholes are not detected and isolated, the cumulative number of packets dropped continues to increase steadily with time. But in the LITEWORP case, as wormholes are identified and isolated permanently, the cumulative number stabilizes. Notice that the cumulative number of packets dropped grows for some time even after the wormhole is locally isolated, due to the cached routes that contain the wormhole and continue to be used till route timeout occurs.

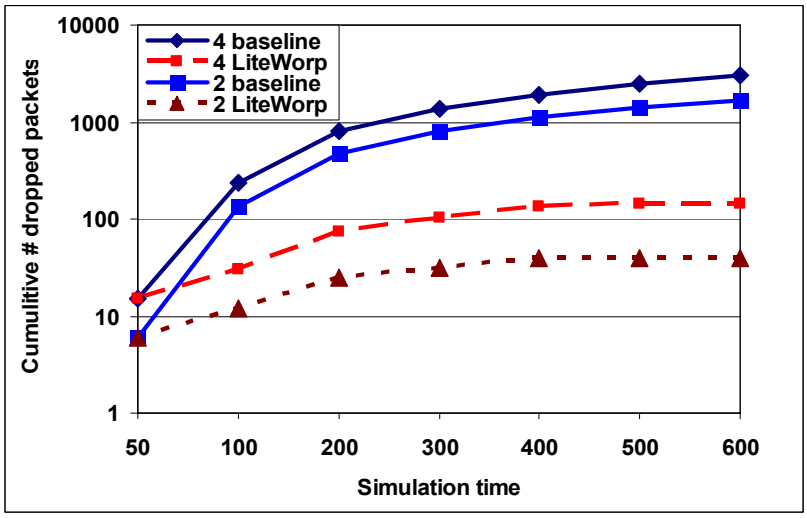

Figure 18: Cumulative number of dropped packets with and without LITEWORP

Figure 19 shows a snapshot, at the end of the simulation time, of (a) the fraction of data packets dropped and (b) the fraction of the malicious routes. This is shown on a log scale for 0-4 malicious nodes for the baseline and with LITEWORP with γ =3. With 0 or 1 malicious node, there is no adverse effect on normal traffic since no wormhole is created. The relationship between the number of dropped packets and the number of malicious routes is not linear. This is because the route established through the wormhole is more heavily used by data sources due to the aggressive nature of the malicious nodes at the ends of the wormhole. If we track these output parameters over time, with LITEWORP, they would tend to zero as no more malicious routes are established or packets dropped, while without LITEWORP they would reach a steady state as a fixed percentage of traffic continues to be affected by the undetected wormholes.

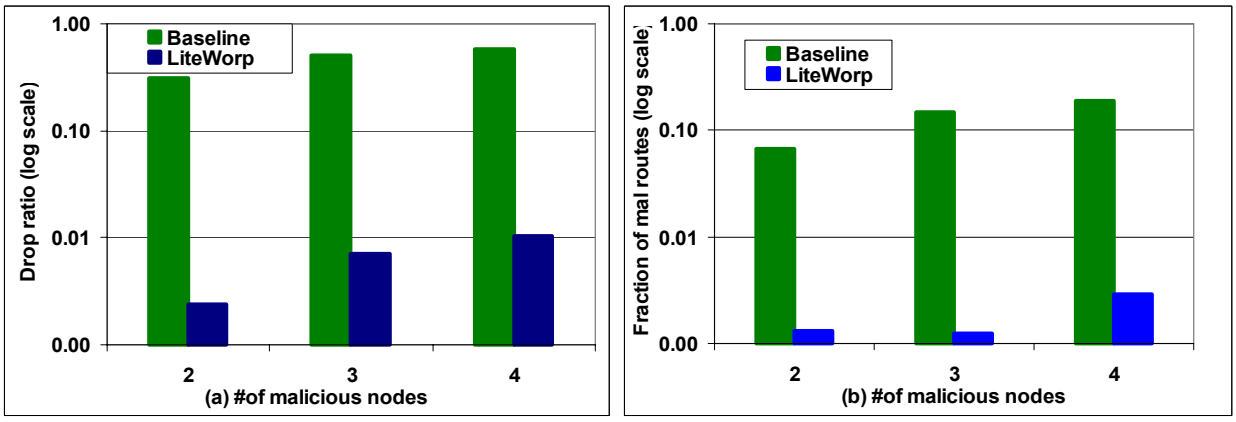

Figure 19: Fraction of dropped packets and malicious routes with and without LITEWORP

Framing: Figure 20 shows the percentage of framing with various values of γ. As the number of malicious nodes increases, the chances of getting γ malicious nodes framing a good node increases and thus the framing percentage increases. As we increase γ, the percentage of framing decreases since it becomes more difficult to get γ malicious nodes to frame a good node. When the value of γ is greater or equal to 7, the probability of framing goes to zero since no node has more than 7 neighbors in this simulation setup, therefore, it is impossible for framing to occur. As

γ increases however, as seen from Figure 21, Figure 23, and Figure 28, the damage done to the network before a malicious node is detected and isolated, increases. In the rest of this section the value of γ =7 is equivalent to γ=infinity since no node has more than 7 neighbors in the simulation setup, therefore, these two values represent the same results.

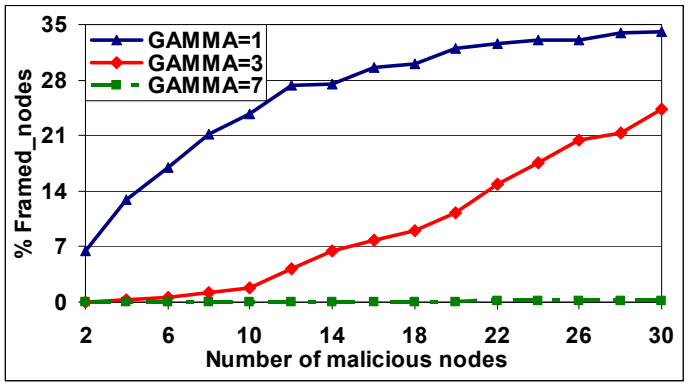

Figure 20: Percentage of framing

Varying the number of malicious nodes: Figure 21 shows the percentage of malicious nodes isolated at the end of the simulation time for three different values of γ. The isolation percentage falls almost linearly as we increase the number of colluding malicious nodes from 2 to 6 due to the decrease in the number of available guards. Note that as γ increases, the percentage of malicious nodes isolated decreases slightly due the requirement of higher number of guards to agree on the detection. However, the percentage of malicious nodes isolated is above 90% for 6 malicious nodes with infinite γ.

Figure 22 shows that the percentage of false isolation increases as the number of malicious nodes increases. This is because not all guard nodes come to the decision to isolate a malicious node at the same time. Therefore, a given guard node may suspect another guard node when the latter isolates a malicious node but the former still has not. For example, a guard node G_1 detects a malicious node M earlier than the other guard nodes for the link to M. Node G_1 subsequently drops all the traffic forwarded to M and is therefore suspected by other guard nodes for M. This problem can be solved by having an authenticated one-hop broadcast whenever a guard node performs a local isolation.

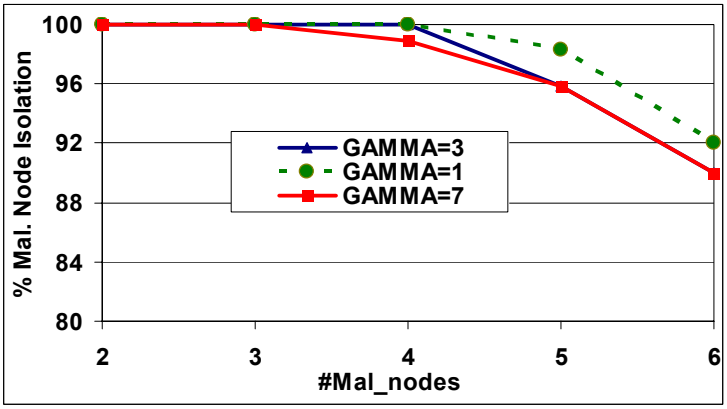

Figure 21: Percentage of malicious node isolation

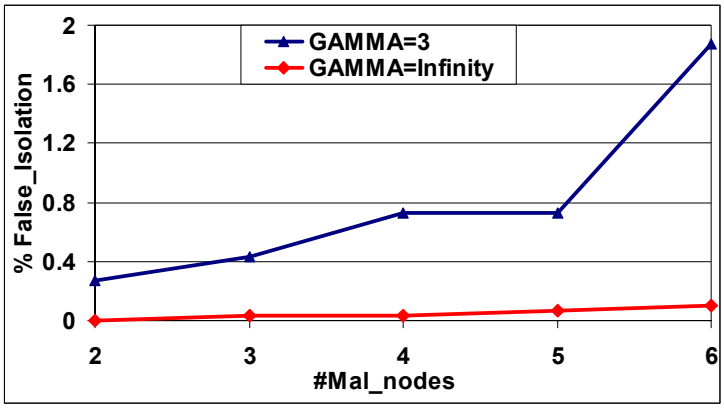

Figure 22: Percentage of false isolation

Figure 23 shows that the percentage of malicious routes increases as we increase the number of malicious nodes. As the number of malicious nodes increases, the percentage of damage that occurs before each of the nodes is detected and isolated increases.

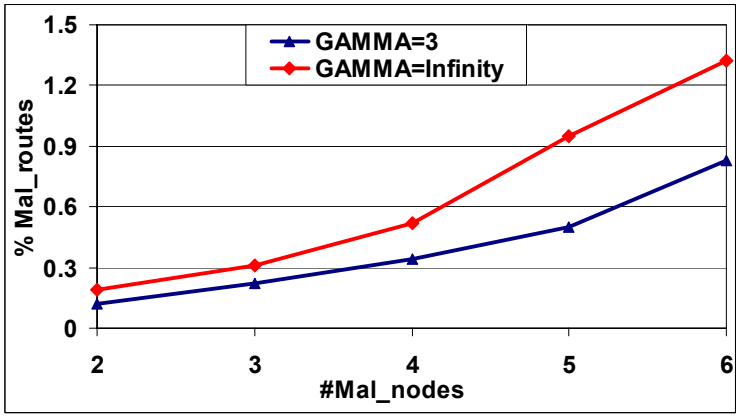

Figure 23: Percentage of malicious routes

Higher number of malicious nodes in the network. The following results are shown for γ=3, f_{drop} = 0.6, μ = 10. Here the goal is to observe the effect of increasing the number of malicious nodes. We can extrapolate the trend for other values of γ from these experiments. Figure 24 shows the variations of the % true isolation as we vary the number of malicious nodes (N_M). Recollect that isolation is defined as all the good neighbors of a malicious node isolating the malicious node, i.e., refusing to relay its traffic. The % true isolation decreases almost linearly as we increase N_M. This is because the number of available guards in the network decreases as more and more nodes get compromised. Furthermore, as N_M increases the local isolation becomes less effective since the number of good guards decreases. Moreover, as N_M increases the data traffic in the network decreases (malicious nodes do not generate their own data) which results in a decrease in the number of packets that a single malicious node may drop. This in turn decreases the chances that the malicious node is detected and isolated.

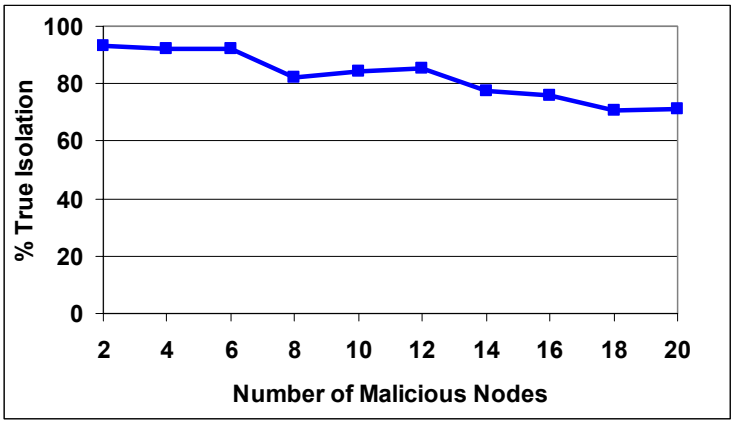

Figure 24: True isolation as a function of the number of malicious nodes in the network

Figure 25 shows the variations of the % delivery ratio as we vary N_M. The % delivery ratio decreases as N_M increases. This is due to the packets dropped before the malicious nodes are detected and isolated. As the number of malicious nodes increases, this total initial drop increases and thus the delivery ratio decreases. Moreover, as Figure 24 shows, as N_M increases, the true isolation decreases. Therefore, the malicious nodes that could not be isolated continue to drop packets and this decreases the delivery ratio.

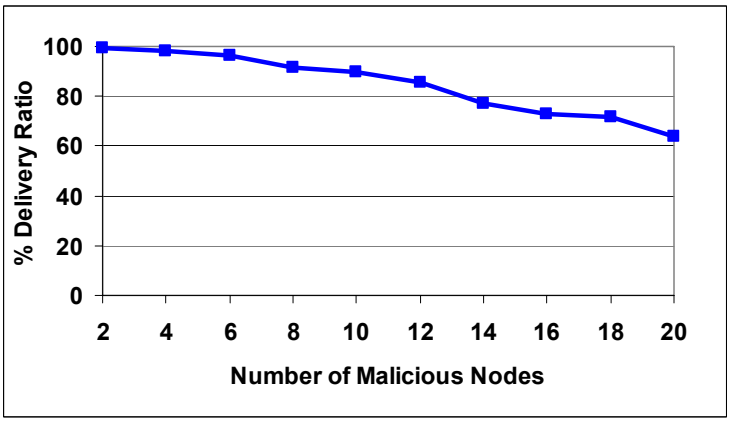

Figure 25: % Delivery ratio as a function of the number of malicious nodes in the network

Figure 26 shows the variations of the % false isolation as we vary N_M. The % false isolation initially increases with N_M and starts to decrease beyond a point. This is because not all guard nodes come to the decision to isolate a malicious node at the same time. Thus, a guard node may suspect another guard node when the latter isolates a malicious node but the former still has not. The occurrence of this situation increases with N_M and hence the % of false isolation increases with N_M. For example, a guard node G_1 detects a malicious node Z earlier than the other guard nodes for the link to Z. G_1 subsequently drops all the traffic forwarded to Z and is therefore suspected by other guard nodes for Z. This problem can be solved by having an authenticated one-hop broadcast whenever a guard node performs a local detection. However, the number of good nodes decreases as we increase N_M. This in turn decreases the indirect false isolation since a node may not have more than γ good nodes to agree on falsely isolating a neighbor. Moreover, as N_M increases, the data traffic decreases since malicious nodes are not generating data. This in turn decreases the chance for collisions and consequently decreases % false isolation. Beyond a point (N_M = 6), the latter factors dominates the first factor and thus the overall result is a decrease in false isolation with increasing N_M.

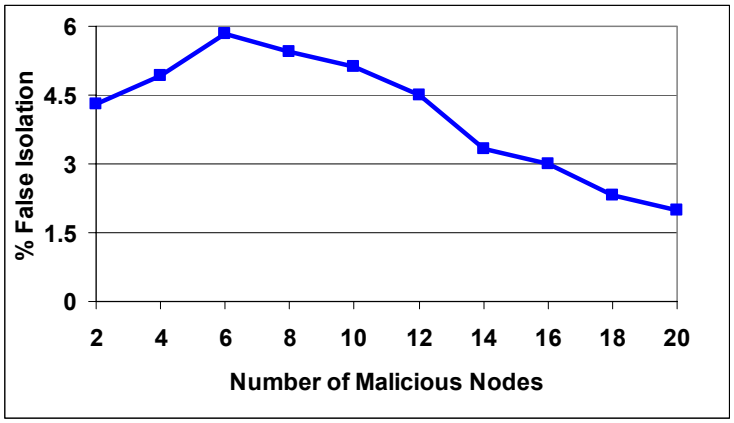

Figure 26: % False isolation as a function of the number of malicious nodes in the network

Varying γ. Figure 27 shows the percentage of false isolation as a function of γ. As γ increases the false isolation decreases since it becomes more and more unlikely to get γ nodes falsely accuse a good node as malicious. As the number of malicious nodes increases the false isolation increases for the same reasoning as in Figure 20.

Figure 28 shows that the percentage of malicious routes increases with γ. As γ increases, the detection and isolation of nodes decreases and takes longer time which gives the malicious nodes more chance to establish more malicious routes. Moreover, as the number of malicious nodes increases, the percentage of damage (malicious routes) increases intuitively.

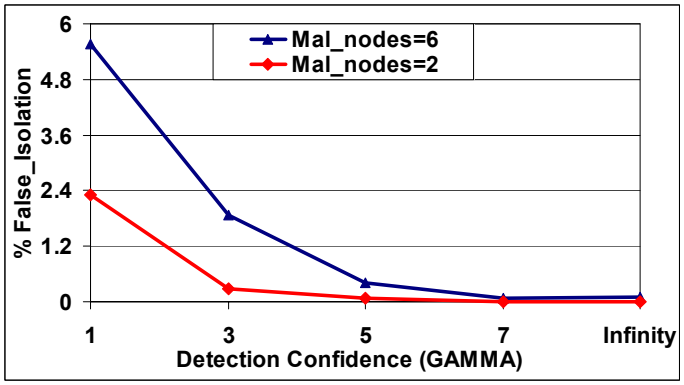

Figure 27: Percentage of false isolation

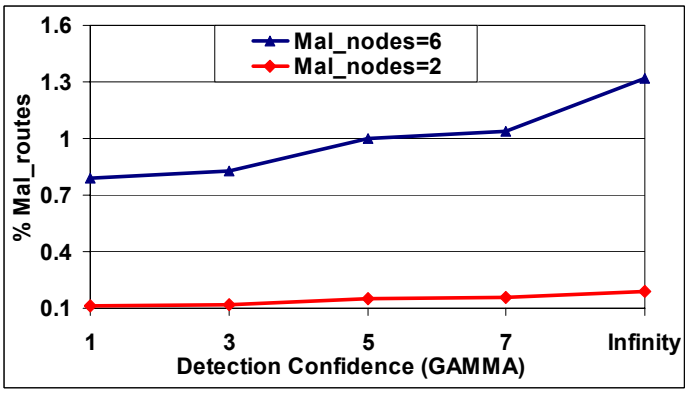

Figure 28: Percentage of malicious routes

The observation from all the experiments is that an infinite value of γ appears to be a desirable operating region. We find that it eliminates framing and minimizes the percentage of false isolation. On the other hand, it only slightly increases the percentage of malicious routes and slightly decreases the percentage of malicious nodes isolated. However, these values are acceptable and close to the case when γ is small. This is because the guards of a

node over a certain link are likely to see the same view of the node and therefore, they are likely to reach to the same reasoning about the monitored node whether individually or through the reports of other guards. This reduces the importance of having guards inform each other of their view about the monitored node which results in little change when we increase the value of γ to infinity.

8 Conclusion and Future Work

In this paper, we have presented taxonomy for attack modes used to launch the wormhole attack in multihop wireless networks. We have presented a protocol called LITEWORP that incorporates a detection protocol and an isolation protocol. The detection protocol can be applied for detecting each mode of the wormhole attack except the protocol deviation. The fundamental mechanism used is local monitoring whereby a node monitors traffic in and out of its neighboring nodes and uses a data structure of first and second-hop neighbors. LITEWORP isolates the malicious node and removes its ability to cause future damage. The coverage analysis of LITEWORP brings out the variation of probability of missed detection and false detection with increasing network density. The framing analysis brings out the conflicting effect of the detection confidence parameter γ. Higher values of γ result in lower framing probability but decrease the detection coverage. However, the degradation of the detection coverage is relatively small for the kind of network we consider and therefore γ set to infinity also performs well. The detection latency analysis provides insights on the time it takes a node to be detected for two different detection strategies (consecutive windows of detection being independent or overlapping). Finally, the cost analysis shows that LITEWORP has low storage, processing, and bandwidth requirements. These, together with the fact that no specialized hardware is required, make the protocol well suited to resource-constrained wireless networks, such as sensor networks.

We propose to investigate the extension of LITEWORP to consider mobile ad-hoc and sensor networks and the effect of local monitoring on sleeping techniques for energy efficiency. For mobility, the fundamental requirement is the ability of a node to securely determine its first-hop and second-hop neighbors in the face of mobility. We can augment LITEWORP with existing work on dynamic secure neighborhood determination protocols, e.g., [8],[9] to achieve the goal as in static networks. However, we are also investigating an alternate design of LITEWORP that is customized to mobile networks. Even though, we show the application of local monitoring for mitigating the wormhole attack, this approach is general and can be extended to detect other control attacks like the Sybil and the sinkhole attacks by changing the kind of information that is maintained in the watch buffers and the checks run on them. We will also investigate the application of local monitoring in mesh networks which have different constraints and different traffic patterns.

9 References

[1] B. Dahill, B. N. Levine, E. Royer, and C. Shields, A secure routing protocol for ad-hoc networks, Electrical Engineering and Computer Science, University of Michigan, Tech. Rep. UM-CS-2001-037, August 2001.

[2] P. Papadimitratos and Z. Haas, Secure routing for mobile ad hoc networks, in SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), 2002.

[3] C. E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, In ACM SIGCOMM on Communications Architectures, Protocols and Applications, 1994.

[4] D. Johnson, D. Maltz, and J. Broch, The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, in Ad Hoc Networking, Addison-Wesley, 2001.

[5] C. Karlof and D. Wagner, Secure Routing in Sensor Networks: Attacks and Countermeasures, at the 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[6] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," at the 6th ACM MobiCOM, 2000.

[7] Y. C. Hu, A. Perrig, and D.B. Johnson, Packet leashes: a defense against wormhole attacks in wireless networks, in Proceedings of the 22nd INFOCOM, pp. 1976-1986, 2003.

[8] L. Hu and D. Evans, Using Directional Antennas to Prevent Wormhole attacks, in Network and Distributed System Security Symposium, 2004.

[9] Y. C. Hu, A. Perrig, and D. Johnson, Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols, in ACM WiSe Workshop, 2003.

[10] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Belding-Royer, A Secure Routing Protocol for Ad hoc Networks, in Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP 02), November 2002.

[11] S. Zhu, S. Xu, S. Setia, and S. Jajodia, Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach, in the 11th IEEE International Conference on Network protocols (ICNP'03), Atlanta, Georgia, 2003.

[12] W. Du, J. Deng, Y. Han, and P. Varshney, A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks, in Proceedings of the 10th ACM conference on Computer and communication security (CCS'03), Washington D.C., USA October 27-30, 2003.

[13] D. Liu and P Ning, Establishing Pair-wise Keys in Distributed Sensor Networks, in Proceedings of the 10th ACM conference on Computer and communication security (CCS'03), 2003.

[14] C. E. Perkins and E. M. Royer, Ad-Hoc On-Demand Distance Vector Routing, in Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99), pp. 90-100, February 1990.

[15] A. Qayyum, L. Viennot, and A. Laouiti, Multipoint Relaying: An Efficent Technique for Flooding in Mobile Wireless Networks, Technical Report Research Report RR-3898, project HIPEERCOM, INRIA, February 2000.

[16] B. Bellur and R. G. Ogier, A Reliable, Efficient Topology Broadcast for Dynamic Networks, in Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99), pp. 178-186, March 1999.

[17] Defense Advanced Research Projects Agency, Frequently Asked Questions v4 for BAA 01-01, FCS Communications Technology. Washington, DC. October 2000. Available at http://www.darpa.mil/ato/solicit/baa01_01faqv4.doc,

[18] Y. Ko, V. Shankarkumar, and N. Vaidya, Medium access control protocols using directional antennas in ad hoc networks, in Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pages 13–21, 2000.

[19] R. Choudhury, X. Yang, R. Ramanathan, and N. Vaidya, Using directional antennas for medium access control in ad hoc networks, at the 8th ACM International Conference on Mobile Computing and Networking (MobiCOM), 2002.

[20] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H Rubens, Mitigating Byzantine Attacks in Ad Hoc Wireless Networks, Department of Computer Science, Johns Hopkins University, Tech. Rep. Version 1, March 2004.

[21] S. Capkun, L. Buttyán, and J.-P. Hubaux, 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 (SASN 03), pp.21-32, 2003.

[22] D. Liu and P. Ning, Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks, in Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS), pages 263-276, February 2003.

[23] P. Kyasanur and N. H. Vaidya, Detection and handling of MAC layer misbehavior in wireless networks, in Proceedings of the International Conference on Dependable Systems and Networks (DSN '03), pp. 173- 182, 2003.

[24] S. Buchegger and J.-Y. Le Boudec, Performance analysis of the CONFIDANT protocol, in Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pages 226-236, June 2002.

[25] Ralph C. Merkle, Protocols for Public Key Cryptosystems, in Proceedings of the IEEE Symposium on Security and Privacy, 1980.

[26] G. Bianchi, Performance analysis of the IEEE 802.11 Distributed Coordination Function, in IEEE Journal on Selected Areas in Communications, 18(3):535-547, March 2000.

[27] "The Network Simulator - ns-2," At: http://www.isi.edu/nsnam/ns/

[28] W. Wang and B. Bhargava, Visualization of Wormholes in Sensor Networks, Proceedings of the 2004 ACM workshop on Wireless security (Wise), pp. 51-60.

[29] I. Khalil, S. Bagchi, and N. B. Shroff, LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks, International Conference on Dependable Systems and Networks (DSN), pp. 612-621, 2005.

[30] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang, Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach, IEEE Wireless Communications and Networking Conference (WCNC05), Volume 2, 13-17 March 2005 Page(s):1193 - 1199 Vol. 2.

[31] R. Durrett, Essentials of Probability, Duxbury Press, 1994.

[32] I. Khalil, S. Bagchi, and C. Nita-Rotaru, DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks, pp. 89-100, SecureComm 2005.

[33] I. Khalil, S. Bagchi, and N. B. Shroff, MOBIWORP: Mitigation of the Wormhole Attack in Mobile Multihop Wireless Networks, IEEE/CreateNet conference on Security and Privacy in Communication networks (SecureComm 2006), Baltimore, MD, August 28th – September 1st 2006, 12 pages.

[34] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in ACM Workshop on Wireless Security (WiSe'03), pp. 1-10, 2003.

Appendix: Notations

This section provides a summary of the notions used throughout the paper.

Table 3: Summary of notations

Acronym	Description	Acronym	Description
REQ	Route Request	REP	Route Reply
R_A	The neighbor list of node A	r	The communication range
MalC(*i*,*j*)	The malicious counter maintained by node *i* for its neighbor *j*	T_{win}	The malicious counter sliding window length
δ	The size of the sliding steps of the malicious counter sliding window	V_f	The amount of increment to the malicious counter if fabrication is detected
V_d	The amount of increment to the malicious counter if drop is detected	C_t	The malicious counter threshold beyond which a node is determined as malicious
γ	The detection confidence index, which is the minimum number of distinct neighbors that report malicious activity before a node isolates its suspected neighbor	τ	The time threshold by which a forwarding node has to forward the packet it has
g	The expected number of guards over a certain communication link	g_{min}	The minimum number of guards over a certain communication link
N_B	The number of neighbors of a node	d	The density of node distribution
P_C	The probability of collision over a certain communication link	P_{fab}	The probability of packet fabrication by a malicious node
P_{direct}	The probability of malicious node detection by direct observation	$P_{indirect}$	The probability of malicious node detection by indirect observation
P_{detect}	The probability of malicious node detection by both direct and indirect observations	P_{FA}	The probability of false alarm, which is the probability that single a packet has been falsely identified as malicious one
P_{DF}	The probability of direct false alarm, which is the probability that a good has been directly identified as malicious by another good node	P_{IF}	The probability of indirect false alarm, which is the probability that a good node has been indirectly identified (through γ other nodes) by another good node
P_{false}	The probability of false alarm, which is the probability that a good node has been identified as malicious by another good node either directly or indirectly	P_{gdm}	The probability that node G detects a node M over a certain time window
N	The total number of nodes	N_M	The number of malicious nodes
P_m	The probability of node compromise	P_{frame}	The probability of node framing
N_{ts}	The number T_{win} time slots	NBL	The size of the neighbor list
h	The number of hops between a source and destination nodes	NBLS	The size of the total storage of the neighbor lists over all the nodes
f	The frequency of route establishment	N_{REP}	The number of nodes involved in monitoring a REP
$TOut_{Route}$	The time out period for unused route	ϕ	The inter-arrival rate of packet generation at each data source
ξ	The rate for changing the destination by a data source		

Issa Khalil received the B.Sc. degree in computer engineering from Jordan University of Science and Technology (JUST), Jordan, in 1994, and the MS degree in computer engineering from JUST in 1996. He is currently pursuing a PhD in the Dependable Computing Systems Lab of Prof. Bagchi S. His research interest includes key-management, secure routing protocols, and intrusion detection in Ad Hoc and Sensor networks. He has worked as the director of computer and communication center of Alquds Open University, West Bank, for more than 6 years.

Saurabh Bagchi joined the department of Electrical and Computer Engineering at Purdue University in West Lafayette, Indiana as an Assistant Professor in August 2002. Before that, he did his Ph.D. from the Computer Science department of the University of Illinois at Urbana-Champaign with Prof. Ravishankar Iyer at the Coordinated Science Laboratory. His Ph.D. dissertation was on error detection protocols in distributed systems and was implemented in a fault-tolerant middleware system called Chameleon.

Ness B. Shroff received his Ph.D. degree from Columbia University, NY in 1994. He is currently a full Professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span the areas of wireless and wire line communication networks, and more recently network security. Dr. Shroff is an editor for IEEE/ACM Trans. on Networking and the Computer Networks Journal, and past editor of IEEE Communications Letters. He was the conference chair for the 14th Annual IEEE Computer Communications Workshop in Estes Park, CO, October 1999) and program co-chair for the symposium on high-speed networks, Globecom 2001 (San Francisco, CA, November 2000). He was the Technical Program co-chair for IEEE INFOCOM'03 and panel co-chair for ACM Mobicom'02. He received the NSF CAREER award in 1996 and the best paper of the year award for Computer Networks, 2003.