LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks

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

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, which replays it locally. This can have an adverse effect in route establishment by preventing nodes from discovering routes that are more than two hops away. In this paper, we present a lightweight countermeasure for the wormhole attack, called LITEWORP, which does not require specialized hardware. 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 encountered when the method is not applied.

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

1. Introduction

Ad-hoc and sensor networks are emerging as a promising platform 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, that have been addressed by customized cryptographic primitives in the wired domain. Alternately, the attacks may be targeted to the control or the 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 wired networks.

A particularly severe security attack, called the wormhole attack, has recently been introduced in the context of ad-hoc networks [5], [7], [8]. During the 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, such as through an out-of-band hidden channel (e.g., a wired link), packet encapsulation, or high powered transmission. This tunnel makes the tunneled packet arrive either sooner or with 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 affect network routing, 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.

In this paper, we present a simple lightweight protocol, called LITEWORP, to detect and mitigate wormhole attacks in 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

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

features that make it especially suitable for resource-constrained wireless environments, such as sensor networks. LITEWORP does not require any specialized hardware, such as directional antennas or fine granularity clocks. It does not require any 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], and clock synchronization [7]) that often make them impractical for sensor networks and infeasible for many classes of ad-hoc networks.

We present a coverage analysis of LITEWORP and show the relation between the number of nodes required for local monitoring, called the *guards*, and the probability of false or missed detection. Also, we build a simulation model for LITEWORP using the network simulator *ns-2* and perform comparative evaluation of a network with and without the technique. The results show that LITEWORP can achieve 100% detection of the wormholes for a wide range of network densities. They also show that detection and isolation of the nodes involved in the wormhole can be achieved in a negligible time after the attack starts, and that the cumulative number of lost packets and malicious routes established do not grow because wormholes are identified and isolated. Finally, we provide analysis for 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 taxonomy of the wormhole attack modes. Section 4 describes the LITEWORP protocol and its defenses against the various modes of the wormhole attack. Section 5 presents coverage and cost analysis of LITEWORP. Section 6 presents simulation results. Finally, Section 7 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]. Initial proposals to thwart wormhole attacks suggest using secure modulation of bits over the wireless channel that can be demodulated only by authorized nodes. This only defends against outside attackers who do not possess cryptographic keys. A similar 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. When sending a packet, the sending node includes in the packet an authenticated version of its own location and the time at which it sent the packet. The receiving node uses these values, in addition to its own location and the time at which it receives the packet, to compute an upper bound on the distance to the sender. 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. The sender includes in each packet an authenticated version of the time of sending. The receiver compares this value to the time at which it received the packet. Based on the time delay and the speed of light, the receiver can determine if the packet has traveled too far. 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 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* the wormhole attack. To thwart the wormhole, each node shares a secret key with every other node and maintains an updated list of its neighbors. To discover its neighbors, a node, called the announcer, uses its directional antenna to broadcast a HELLO message in every direction. Each node that hears the HELLO message sends its identity and an encrypted message, containing the identity of the announcer and a random challenge nonce, back to the announcer. Before the announcer adds the responder to its neighbor list, it verifies the message authentication using the shared key, and that it heard the message in the opposite directional antenna to that reported by the neighbor. This approach is suitable for secure dynamic neighbor detection. 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. The requirement of directional antennas on all nodes may be infeasible for some deployments. Finally, the protocol may degrade the

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

connectivity of the network by rejecting legitimate neighbors in their conservative approach to prevent wormholes from materializing. 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.

3. Wormhole Attack Modes

In this section we classify the wormhole attack based on the techniques used for launching it.

3.1. Wormhole using Encapsulation

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 *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. However, in a malicious environment, this protocol will 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 for two colluding malicious nodes can involve themselves in a route is by simply giving the false illusion that the route through them is the shortest, even though they may be many hops away. 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.

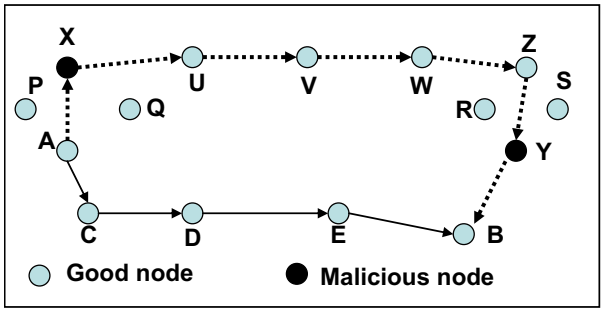

Figure 1: Wormhole through packet encapsulation

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.

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 is sending a route request to node B, nodes X and Y are malicious 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*. Node *B* gets two route requests — *A-X-Y-B* and *A-C-D-E-F-B*. The first route is both shorter and faster than the second, and is thus chosen by *B*.

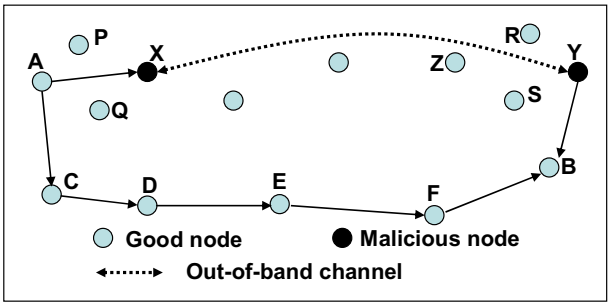

Figure 2: Wormhole through out-of-band channel

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

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 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 purpose is to let the request packet it forwards arrive first at the destination and it is therefore included in the path to the destination. 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. Defenses

In this section, we describe the method for wormhole detection in LITEWORP followed by the method 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 the 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 links are bi-directional, which means that if a node *A* can hear node *B* then *B* can hear *A*. We assume that there is a certain amount of time from a node's deployment, called the *compromise threshold time* (T_{CT}) that is minimally required to compromise the node. We have a protocol presented in Section 4.2.1 for discovery of first and second hop transmission neighbors of a node. We define the maximum time required for the neighbor discovery protocol to complete as T_{ND}. Our assumption is that for a given node n_i, all its first and second hop neighbors are deployed within T_{CT}-$2T_{ND}$ of the deployment of n_i. This assumption implies that there can be no malicious insider node within two hops of n_i within T_{ND} time units from its deployment. Note however, that this assumption allows insider malicious nodes, that are greater than two hops from n_i to exist any where in the network. We assume that the network has a static topology, though the functional roles a node plays (e.g., cluster head, data aggregator, etc.) may change. Finally, LITEWORP requires a pre-distribution pair-wise key management protocol (e.g. [11] for ad-hoc networks and [12],[13] for sensor networks).

4.2. Local Monitoring for Wormhole Defense

4.2.1. Information Structures

Building neighbor lists: 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 an authenticated reply to *A*, using the shared key. Node *A* accepts all the replies that arrive within a timeout.

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

For each reply, A verifies the authenticity of the reply and adds the responder to its neighbor list R_A. Then A does a one-hop broadcast of a message containing the list R_A. This broadcast is authenticated individually by the shared key with each member in R_A. When B hears the broadcast, it verifies the authenticity of R_A, and stores R_A if correctly verified. Hence, at the end of this neighbor discovery process, each node has a list of its direct neighbors and the neighbors of each one of its direct neighbors. This process is performed only once in the lifetime of a node and is secure because of the system model assumptions. 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 packet is legitimate or not. 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. After building its first and second hop neighbor list, node A activates local monitoring.

Local monitoring: A collaborative detection strategy for wormholes is used, where a node monitors the traffic going in and out of its neighbors. For a node, say α, to be able to watch a node say, β, two conditions are required: (i) each packet forwarder must explicitly announce the immediate source of the packet it is forwarding, i.e., the node from which it receives the packet, and (ii) α must be a neighbor of both β and the previous hop from β, say δ. If the second condition is satisfied, we call α the guard node for the link from δ to β. This implies that α is the guard node for all its outgoing links. For example, in Figure 3, nodes M, N, and X are the guard nodes of the link from X to A. Information from 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 towards 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.

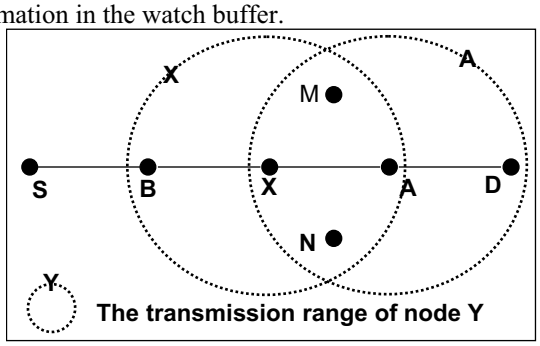

Figure 3: X, M, N are guards of the link from X to A

A malicious counter ($Mal_C(i,j)$) is maintained at each guard node, i, for a node, j, at the receiving end of each link that i is monitoring. $Mal_C(i,j)$ is incremented for any malicious activity of j that is detected by i. The increment to Mal_C depends on the nature of the malicious activity detected, e.g., V_f for fabricating and V_d for dropping a control packet.

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 $Mal_C(\alpha,A)$ crosses a threshold, $\boldsymbol{C_t}$, 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 guard to 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* of D. The detection confidence represents the minimum number of guard nodes that must report that a certain node is malicious for a neighbor of that node to isolate it. A small value for γ may result in blackmailing of good nodes, while a large value of γ may result in missed detection. Note that the number of alerts is cumulative over time.
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.

4.2.3. Detecting Different Modes of Wormhole Attacks

Detecting out-of-band and packet encapsulation wormholes

A guard node α for a 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 A claims has 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 must have fabricated the packet. Node α increments $Mal_C(\alpha,A)$ by V_f. If an entry for a packet sent from X to A stays in the watch buffer beyond τ, then A is accused of dropping the corresponding packet. Node α increments $Mal_C(\alpha,A)$ by V_d.

Consider the scenario in Figure 4. M_1 and M_2 are two malicious nodes wishing to establish a wormhole between

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

the two 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, all the guards of 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 Mal_C 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 the link from *D* to M_2 (*D*, *N*, and *W*) overhear the *REP* and save an entry in their watch buffers.

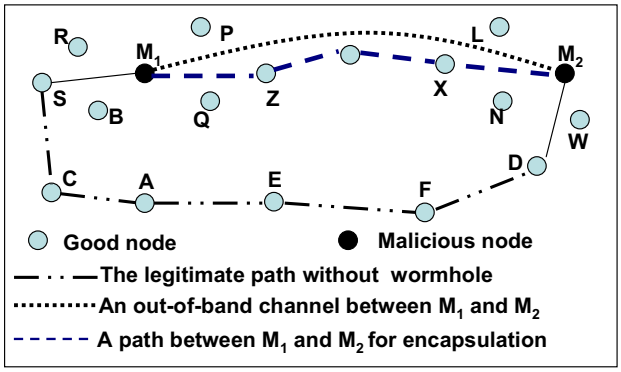

Figure 4: Wormhole detection for out-of-band and packet encapsulation modes

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 Mal_C 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, Mal_C 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. In the second case, all the guards of 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.

Detecting high power transmission wormhole

This mode is detected using the assumption of symmetric bi-directional channels. Suppose a malicious node, say *X*, tries to use high power transmission to forward a packet P_1 to it is 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.

Detecting packet relay wormhole

This mode is detected using the stored neighbor lists at each node. Suppose a malicious node *X* 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.

Detecting protocol deviation wormhole

This mode *can not* 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].

5. LITEWORP Analysis

5.1. 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 5(a). *S* and *D* are separated by a distance *x,* and the communication range is *r*. The value of *x* follows a random variable with probability density function of $f(x) = 2x/r^2$ with range (0,*r*). This follows from the assumption of uniform distribution of the nodes.

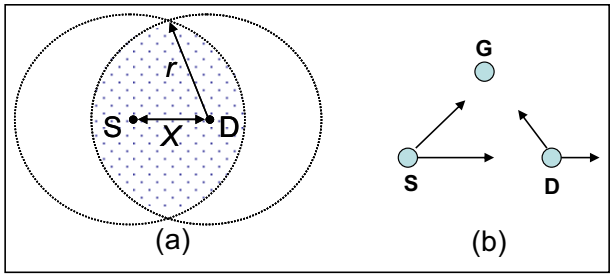

Figure 5: (a) The area where a node can guard the link S→D; (b) Illustration for detection accuracy

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 5(a). This area is given by

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

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

The minimum value of *Area(x), Area*$_{min}$, is when $x = r$. Therefore, the minimum number of guards is $g_{\min} = Area_{\min} d = 0.36r^2 d$. The expected value of the area is

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

$$= \left(\frac{2\pi}{3} - \frac{1}{2}\right) r^2 \approx 1.6r^2$$

Therefore, the expected number of guards $g = E[Area(x)]d = \lfloor 1.6r^2 d \rfloor$. The number of neighbors of a node is given by $N_B = \pi r^2 d$, thus

$$g = \left(\frac{2}{3} - \frac{1}{2\pi}\right) N_B \approx \lfloor 0.51 N_B \rfloor \qquad \text{——————(I)}$$

Now, as in [26] where IEEE 802.11 was analyzed, we assume that each packet collides on the channel with a constant and independent probability, P_C. As shown in Figure 5(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 μ packet fabrications occur within a certain time window, *T*. Also assume that a guard must detect at least β fabrications to cause the Mal_C for a node to cross the threshold, and thus generate an alert. Then, the alert probability at a guard is given by $P_{\beta|\mu} = \sum_{i=\beta}^{\mu} \binom{\mu}{i} (1-P_C)^i (P_C)^{\mu-i}$. Thus, assuming independence of collision events among the different guards, the probability that at least γ of the guards generate an alert

$$p_{\geq\gamma} = \sum_{i=\gamma}^{g} \binom{g}{i} \left(P_{\beta|\mu}\right)^i \left(1 - P_{\beta|\mu}\right)^{g-i} = \frac{B(P_{\beta|\mu}, \gamma, g-\gamma+1)}{B(\gamma, g-\gamma+1)}$$

$$= \frac{g!}{(\gamma-1)!(g-\gamma)!} \int_0^{P_{\beta|\mu}} u^{\gamma-1}(1-u)^{g-\gamma}\, du$$

Where, $B(\gamma, g-\gamma+1)$ is the Beta function and $B(P_{\beta|\mu}; \gamma, g-\gamma+1)$ is the incomplete Beta function.

Figure 6 shows the probability of detecting wormholes with $\mu = 7$, β=5, γ=3, 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 (I). 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 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 detection starts to fall rapidly beyond a point. Figure 11 shows that for the same μ, β, and P_C, the probability of wormhole detection as a function of γ when $N_B = 15$ and $M = 2$. As γ increases, the probability of detection ($P_{\geq\gamma}$) decreases.

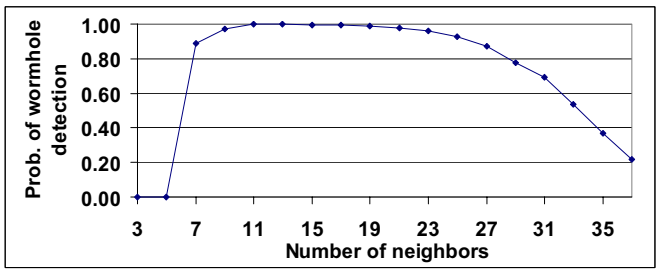

Figure 6: (a) Probability of wormhole detection

As shown in Figure 5(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 μ packets to *D* for forwarding, within a certain time window, *T*. The probability that *D* is falsely accused is the probability that β or more packets are falsely suspected as fabricated. This is given by $P_{FA(\beta|\mu)} = \sum_{i=\beta}^{\mu} \binom{\mu}{i} (P_{FA})^i (1-P_{FA})^{\mu-i}$, and the probability that at least γ guards generate false alarms is given by

$$p_{FA\geq\gamma} = \sum_{i=\gamma}^{g} \binom{g}{i} \left(P_{FA(\beta|\mu)}\right)^i \left(1 - P_{FA(\beta|\mu)}\right)^{g-i} = \frac{\beta(P_{FA(\beta|\mu)}, \gamma, g-\gamma+1)}{\beta(\gamma, g-\gamma+1)}$$

$$= \frac{g!}{(\gamma-1)!(g-\gamma)!} \int_0^{P_{FA(\beta|\mu)}} u^{\gamma-1}(1-u)^{g-\gamma}\, du$$

Figure 7 shows the probability of false alarm as a function of the number of nodes for the same parameters as in Figure 6. 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 increased false detection at these γ guards. 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 negligible (less than 0.3×10^{-6}).

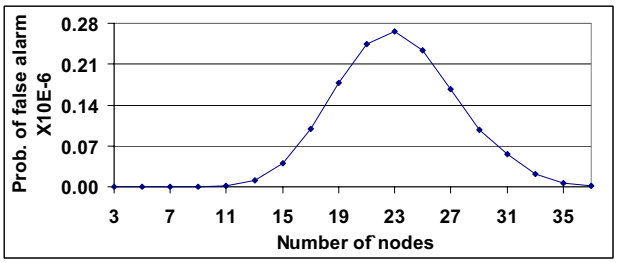

Figure 7: Probability of false alarm

5.2. Cost Analysis

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

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 **IEEE**

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 Mal_C 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. 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 8). 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.

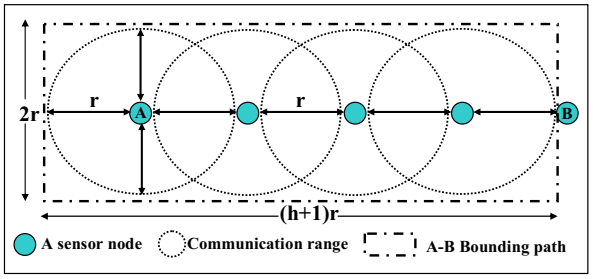

Figure 8: The average number of nodes involved in the watch of a route reply

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 is $f + (N_{REP} / N)f$. 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 one lookup for the current source and the current destination in the neighbor list, adding an entry to the watch buffer (incoming) or deleting an entry from the watch buffer (outgoing), and 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.

Thus, due to the low resource overheads, LITEWORP is suitable for use in resource-constrained wireless networks.

6. 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 compromised 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.

Each node acts as a data source and generates data using an exponential random distribution with inter-arrival rate of μ. The destination is chosen at random and is changed using an exponential random distribution with rate ξ. The input parameters with the experimental values are given in Table 2. The output parameters include the isolation latency, the number of data packets dropped due to the wormhole, the number of routes established, and the number of routes affected by the wormhole. The simulation also accounts for losses due to natural collisions. The isolation latency is calculated from the time a malicious node starts a wormhole attack until it is completely isolated by all of its neighbors. The guards inform all the neighbors of the detected malicious node through multiple unicasts. The output parameters that we present here are obtained by averaging over 30 runs. For each run, the malicious nodes are chosen at random such that they are more than 2 hops away from each other.

Table 2: Input parameters for LITEWORP simulation

Param.	Value	Param.	Value
Tx Range (r)	30 m	γ	2-8
N_B	8	μ	1/10 sec
$TOut_{Route}$	50 s	M	0-4
β	5	τ	0.5 sec
# nodes (N)	20,50,100,150	BW	40 kbps
ξ	1/200 sec	T	200

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

Figure 9 shows the number of packets dropped as a function of simulation time for the 100-node setup with 2 and 4 colluding nodes both with LITEWORP and without LITEWORP. 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 at 75 sec, due to the cached routes that contain the wormhole and continue to be used till route timeout occurs.

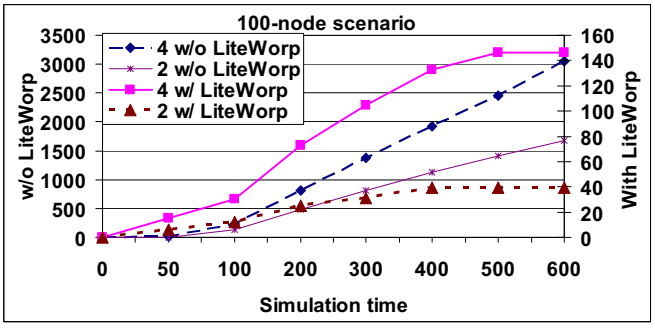

Figure 9: Cumulative number of dropped packets with and without LITEWORP

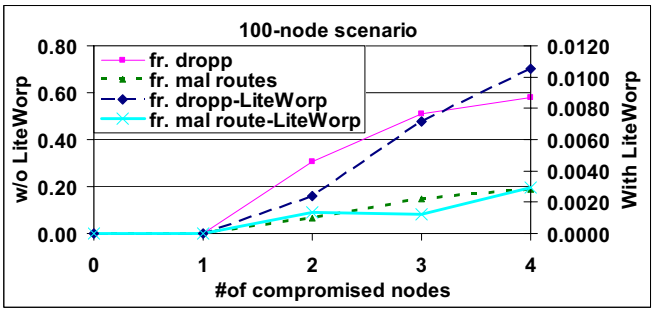

Figure 10: Fraction of dropped packets and malicious routes with and without LITEWORP

Figure 10 shows a snapshot, at simulation time of 2000 sec, of the fraction of the total number of packets dropped to the total number of packets sent, and the fraction of the total number of routes that involve wormholes to the total number of routes established. This is shown for 0-4 compromised nodes for the baseline and with LITEWORP. With 0 or 1 compromised 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.

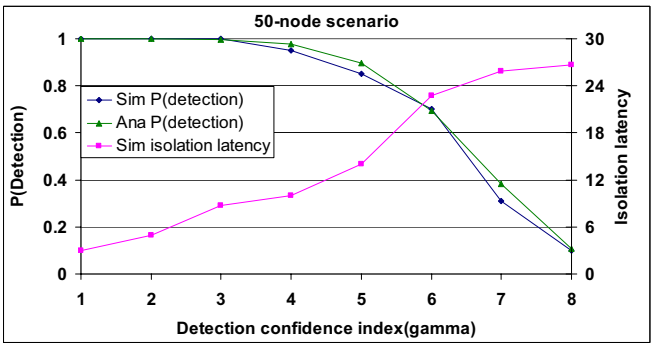

Figure 11: Detection probability & latency with varying γ

Figure 11 bears out the analytical result for the detection probability as γ is varied with N_B = 15 and M= 2. As γ increases, the detection probability goes down due to the need for alarm reporting by a larger number of guards, in the presence of collisions. Also the isolation latency goes up, though it is very small (less than 30 s) even at the right end of the plot.

7. Conclusion and Future Work

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. 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. 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.

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

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

network density. 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 ideally suited to resource-constrained wireless networks, such as sensor networks.

8. 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, November 4-7, 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), Washington D.C., USA October 27-30, 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. Available at http://www.darpa.mil/ato/solicit/baa01_01faqv4.doc, October 2000.

[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/

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE