LIP: A Lightweight Inter-layer Protocol for Network Access Control in Mobile Ad-Hoc Networks

Hungyuan Hsu Sencun Zhu Ali Hurson
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802

Abstract—
Most ad hoc networks do not implement any *network access control*, leaving these networks vulnerable to packet injection attacks where a malicious node injects a large number of packets into the network with the goal of depleting the resources of the nodes relaying the packets. To prevent such attacks, it is necessary to employ authentication mechanisms that ensure that only authorized nodes can inject traffic into the network. We design a Lightweight Inter-layer Protocol (LIP) for network access control based on efficient local broadcast authentication mechanisms. In addition to preventing attacks by unauthorized nodes, LIP can also detect and minimize the *impersonation attacks* by compromised insider nodes. Through detailed simulation study, we show that LIP incurs small bandwidth overhead and has little impact on the traffic delivery ratio even in the case of high node mobility. Moreover, the *transparency* and *independence* properties of LIP allows it to be turned on/off as desired and to be integrated *seamlessly* with secure routing protocols, providing stronger security services for ad hoc networks.

I. Introduction

Most ad hoc networks do not have any provisions for restricting the traffic that flows through a node, i.e., they do not implement any network access control. This leaves these networks vulnerable to packet injection attacks where a malicious node injects a large number of packets into the network with the goal of depleting the resources of the nodes relaying the packets. The packet injection attack must be addressed for the successful deployment of ad hoc networks due to the constrained resources of mobile nodes.

A packet injection attack can be especially effective if a packet injected into an ad hoc network by a malicious node ends up being multicast or broadcast throughout the network. For example, the operation of most routing protocols involves steps in which a control packet, e.g., a route request packet, is broadcast to all nodes. Moreover, many applications for ad hoc networks are group-oriented and involve collaborative computing; thus multicast communication is likely to increase in importance as multicast routing protocols for ad hoc networks become more mature. Compared to the channel jamming attack, which only affects a relative small area around the malicious node and could be addressed by techniques such as spread spectrum, channel surfing, or spatial retreat [24], the packet injection attack using broadcast messages may be more favorable to an attacker due to its network-wide harm.

Clearly, a network access control capability is essential for preventing packet injection attacks in an adversarial environment such as a battlefield. It is also necessary for ad hoc networks that do pricing [2]. Most of the routing protocols that have been proposed for ad hoc networks do not address the issue of network access control. In these protocols, a node trusts that its neighbors will forward packets for it and also assumes that the packets it receives from its neighbors are authenticated. This naive trust model allows a malicious node to inject erroneous routing requests or routing updates into a network, which can paralyze the entire network. To deal with such attacks, recently several security extensions [7], [10], [11], [22], [25] have been proposed for authenticating the routing control packets in the network. We note, however, that none of the proposed secure routing protocols include any provisions for authenticating data packets although data packets are the main traffic in an ad hoc network.

The simplest approach to provide network access control is to employ a network-wide key[1] shared by all nodes. Every node uses this shared key to compute message authentication codes (MACs) on the packets it sends and verify packets from its neighbors. Despite its simplicity, this scheme has several disadvantages. First, an attacker only needs to compromise one node to break the security of the system. Second, if the global key is divulged, it is difficult to identify the compromised node. A compromised node may launch various attacks impersonating other nodes due to the lack of source authentication. Third, it is expensive to recover from a compromise because it usually involves a group key update process. In practice, a system administrator might have to manually reset the group key in the configuration of every user's wireless NIC.

Instead of using a network-wide key, one may use pairwise keys for authenticating every packet. However, when a node broadcasts a packet, it has to attack n MACs to the packet, where n is the number of its immediate neighbors. Thus, this approach becomes very inefficient for networks with high node density. On the other hand, source node signing *every* packet based on public key cryptography can provide network access control; however, its large overhead has even prohibited per-packet signature in wired networks, not mentioning ad hoc networks that are generally more scarce on resources.

Contribution We present LIP, an efficient, scalable, and general-purpose network access control protocol for preventing packet injection attacks in ad hoc networks. LIP is based on a lightweight *localized* broadcast authentication mechanism using which a node authenticates its pack-

[1] A network-wide key is also used in the WEP algorithm in the 802.11 standard.

ets only to its immediate neighbors. It can provide much stronger network access control capability than a network-wide key based scheme, and does not involve computing digital signatures over traffic packets. In addition to preventing outsider attack, LIP can also minimize insider impersonation attacks. A location-aware version of LIP can also prevent sophisticated attacks such as Wormhole attacks [12]. Another unique feature of LIP is its *transparency* and *independence* with respect to the network routing protocols due to its inter-layer design principle. It can be thought of as residing in between the data link layer and the network layer, providing a layer of protection that can prevent many attacks from happening. Our scheme can be integrated *seamlessly* with secure routing protocols to provide strong security services for an ad hoc network. Through extensive simulation study, we show that LIP incurs very small performance overhead even in the case of high node mobility.

The rest of this paper is organized as follows. We first discuss related work in Section II. Then we present the details of the protocol in Section III, and analyze its security in Section V. In Section VI, we analyze the performance of our protocol. Finally, we conclude this paper in Section VII.

II. Related Work

The work mostly close to ours includes secure routing, DoS or resource consumption attacks, and network access control. Secure routing for ad hoc networks has been extensively studied. Dahill et al. [7] identify several security vulnerabilities in AODV [21] and DSR [14], and proposed to use asymmetric cryptography for securing ad hoc routing protocols. Yi, Naldurg, and Kravets [25] present a security-aware routing protocol which uses security (e.g., trust level in a trust hierarchy) as the metric for route discovery between pairs. Hu, Perrig and Johnson designed SEAD [10] for securing DSDV, and Ariadne [11] for securing DSR. Our protocol differs with these work on design goals. These previous work are designed to secure specific routing protocols, whereas our protocol focuses on designing a transparent layer between the data linker layer and the network layer, and it does not distinguish between data and routing control packets.

In [1], Aad, Hubaux, and Knightly quantitatively study the DoS resilience of an ad hoc network under Jellyfish attack and Black-hole attack. In [28], Zhu *et al* propose LHAP, a protocol for preventing resource consumption attacks in ad hoc networks. LHAP uses TESLA [20] for bootstrapping one-way key chains. However, the use of TESLA in LHAP leads to some inherent difficulty. First, TESLA requires *periodic* key disclosure, thus introducing some constant bandwidth overhead that is independent of the actual traffic rate. Second, since TESLA introduces *delayed* packet verification to forwarding nodes, the use of TESLA makes LHAP vulnerable to an outsider attack for up to one TESLA period. In contrast, our protocol provides immediate packet authentication. It can prevent outsider attacks and thwart insider attacks as well. Moreover, LIP incurs much smaller bandwidth overhead than LHAP does.

Based on threshold cryptography, Zhou and Haas [26] and Luo et al. [16] have proposed hierarchical and distributed schemes, respectively, for network access control in ad hoc networks. However, the focus of their work is on membership management regarding node join authorization and node revocation. Since these schemes and LIP address different issues, they can be employed in parallel.

III. Assumptions and Design Goal

A. Security Assumptions

We assume that every pair of mobile nodes can establish a pairwise key on the fly based on an appropriate id-based scheme, e.g., preloading pairwise keys or probabilistic-polynomials [17], or using standard public key cryptography (if the computational resources of nodes are less constrained). The id-based scheme allows two nodes knowing each other's id to establish a pairwise key on-the-fly without requiring the existence of an on-line key server. Moreover, the id-based scheme prevents a node from impersonating another node because it does not possess the keys for that node. We note that secure routing protocols [7], [11] also assume these similar ways to bootstrap trust between nodes. Hence, when employing LIP together with secure routing protocols, we do not need to add another mechanism for establishing pairwise keys between nodes.

We do not address attacks against the physical layer and the media access control layer. Techniques such as spread spectrum, frequency hopping, spatial retreat [24] can be employed to prevent physical jamming attacks if necessary. Cardenas et al [5] have studied techniques for detecting and preventing media access control layer attacks.

B. Attack Models

We mainly consider the *packet injection attack* in which an attacker injects a huge number of junk packets into an ad hoc network with the goal of depleting the resources of the nodes that relay the packets. In addition, these packets could introduce severe wireless channel contention and network congestion. The packets could be unicast packets, local (one-hop) broadcast packets, or network-wide broadcast packets. Clearly, the attack is the most effective if the injected packets end up being flooded in the entire network.

The attacker could be an outsider (unauthorized) node that does not possess a valid credential, or an insider (authorized) node that possesses a valid credential. An insider node launches the attack because it has been compromised or it intentionally does it; we do not distinguish the attack motivation here. To achieve the attack goal, an attacker may eavesdrop, reorder, and drop packets, replay older packets, or modify overheard packets and re-inject them into network.

An attacker may use its own id, fabricated ids, or spoofed ids as the sources of the injected packets; however, in this paper we do *not* prevent the attack where an insider attacker directly uses its own id. To prevent this type of insider attack, we have to regulate the normal traffic pattern

(e.g., the maximum Route Request rate [19]) for each node. The violation of the regulation indicates the compromise of the node and rekeying schemes such as GKMPAN [29] may then be applied to revoke the compromised node.

C. Design Goal

The goal of this work is to provide an efficient network access control mechanism for preventing packet injection attacks. To achieve this goal, it is essential that a node is able to verify the authenticity of *every* packet received from other nodes. As a result, the protocol should meet the following requirements:

Efficiency The protocol must be very resource efficient since every packet will need to be authenticated; otherwise, the amount of resources it consumes may be equivalent to that caused by packet injection attacks. Since packet transmission contributes to the main portion of energy expenditure of a wireless node, the protocol should minimize the additional bandwidth overhead.

Scalability The performance of the protocol, in terms of computational and communication cost, should not degrade with the network size. The scheme should not require every node to have the global knowledge of a network;

Immediate Authentication The protocol should provide immediate authentication, i.e., there should be no latency in authenticating a received packet; otherwise, the latency of packet delivery will be unacceptably high in a multi-hop communication setting and a node might have to dedicate a large memory space for buffering those temporarily unverifiable packets.

Transparency It is very undesirable that the deployment of a protocol requires modification or redesign of other protocols in the protocol stack. Therefore, the protocol should work transparently with other protocols, i.e., the protocol may be turned on or turned off without affecting the functionality of other protocols such as routing protocols or application layer protocols.

Independency The protocol should work regardless of the deployed routing protocol. It is possible to design a specific and more efficient protocol that works with a specific routing protocol; however, this is not an efficient way given so many routing protocols in the literature. Especially, so far little work has been done to secure multicast and broadcast routing protocols.

In the following sections, we show LIP achieves all these design goals.

IV. LIP: A Lightweight Inter-layer Protocol

We first present an overview of LIP, then discuss two schemes in detail—a basic scheme, followed by a location-aware version of this scheme.

Notation We use the following notation to describe security protocols and cryptography operations in this paper:

- u, v (in lower case) are the identities of mobile nodes.
- $M1 \parallel M2$ denotes the concatenation of message $M1$ and $M2$.
- $MAC(K, M)$ denotes the computation of MAC over message M with key K.

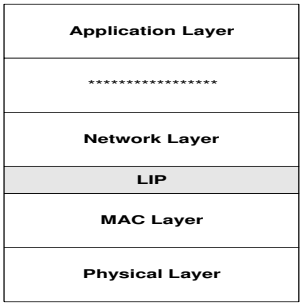


Fig. 1. The protocol stack in which LIP is between the network layer and the data link layer.

- $\{M\}_K$ is encrypting message M with key K.

A. Overview

The goal of our protocol is to provide full network access control. As such, the protocol does not distinguish between data packets and routing control packets for authentication purposes. For simplicity, we call all these packets *traffic* packets. The protocol is transparent to and independent of the network routing protocol. It can be thought of as residing between the data link layer and the network layer, providing a protection mechanism that can prevent many attacks from happening. This transparency and independence allows the protocol to be turned on or turned off without affecting the operations of other layers. Figure 1 shows the protocol stack.

To minimize packet overhead, we design LIP based on a *localized* broadcast authentication mechanism in which a node only computes and attaches one message authentication code (MAC) to each traffic packet it is forwarding (or originated from it). For its neighbors to verify its packets, a node must share its MAC keys (referred to as *cluster* keys hereafter although there are no topological clusters here) with its neighbors. Hence, we introduce the *cluster key management* process for a node to establish and maintain its cluster keys. The cluster keys of a node should only be used by the node to authenticate its packets to its neighbors while its neighbors use the same cluster keys only for verification purpose. However, due to the symmetry nature of clusters keys, a malicious neighbor may impersonate the node by using the node's cluster keys to generate MACs over injected packets. To thwart this impersonation attack, we propose three techniques: *one-time cluster key*, *random neighborship verification*, and *neighborship estimation*. The use of one-time cluster keys builds the first defense line to prevent the impersonation attack, making the attack very difficult to succeed. The random verification process can further detect such attack in case that sophisticated attacks cross the first defense line. By assuming the availability of node location and velocity information, neighborship estimation can further reduce bandwidth overhead. The details of these techniques are presented below.

B. Scheme I: Basic Scheme

B.1 Using One-time Cluster Keys

The basic scheme uses one-time cluster keys; that is, a node uses every cluster key only once to thwart an at-

tacker from reusing its cluster keys. One-time cluster keys are provided by the technique of one-way key chains [15]. A one-way key chain is an ordered set of keys generated through repeatedly applying a one-way hash function H on a random number. For instance, if a node wants to generate a key chain of size $l+1$, it first randomly chooses a key, say $K(l)$, then computes $K(l-1) = H(K(l))$, $K(l-2) = H(K(l-1))$,..., repeatedly until it obtains $K(0) = H(K(1))$.

Once a node has generated its key chain, it can use the keys in its key chain as cluster keys and every cluster key is used for authenticating one packet. To enable its neighbors to verify a cluster key in its key chain, a node first bootstraps its key chain by sending the commitment of its key chain, i.e., $K(0)$, to each of its current neighbors, encrypted with their pairwise key. The node then uses a cluster key in its key chain to compute the MAC of a packet it is transmitting. Note that the cluster keys are consumed in an order reverse to that of their generations. A receiver can authenticate $K(j)$ by verifying $K(j-1) = H(K(j))$ if it has $K(j-1)$. Furthermore, if a receiver did not receive $K(j-1)$ and the last key it authenticated is $K(i)$, where $i < j-1$, it can still authenticate $K(j)$ by verifying $K(i) = H^{j-i}(K(j))$. This property is very useful because it means the authentication scheme can tolerate packet losses.

Consider the scenario where node u wants to authenticate a packet $P(i)$ to its neighbors $v_1, v_2, ..., v_m$, using $K(i)$ as the MAC key. In the message M that contains $P(i)$, the node embeds its next cluster key $K(i+1)$, and attaches a MAC of $P(i)$ computed with $K(i)$.

$$M : P(i), K(i+1) \oplus MAC(K(i), P(i)) \tag{1}$$

Here we assume the size of a key is the same as the output of a MAC, for example 8 bytes. When a neighbor node v receives the message, it performs three operations. First, it computes $MAC(K(i), P(i))$ based on $K(i)$, which it derives from the previous message. Second, after computing a MAC over $P(i)$ based on $K(i)$, it derives $K(i+1)$ by a bitwise-XOR operation. Finally, it checks if $H(K(i+1)) = K(i)$. If the verification succeeds, it sets $K(i+1)$ as node u's next valid MAC key. In addition, it adds u into its local *trust list*. Any future packets that are authenticated with a cluster key prior to $K(i+1)$ will be discarded. As a result, an attacker cannot simply reuse the previous cluster keys of node u to deceive a neighbor node v_j. Finally, the LIP protocol of node v_j passes the verified packet to the routing protocol for process. If node v_j decides to forward this packet to one or more neighbors, its LIP protocol will use node v_j's own cluster key to authenticate the packet to others. As such, a packet is authenticated in a *hop-by-hop* fashion.

The above authentication scheme is motivated by two observations. First, since packets are authenticated hop-by-hop, a node only needs to authenticate a packet to its *immediate* neighbors. Second, when a node sends a packet, a neighbor will normally receive the packet before it receives a copy forwarded by any other nodes. This is due to the

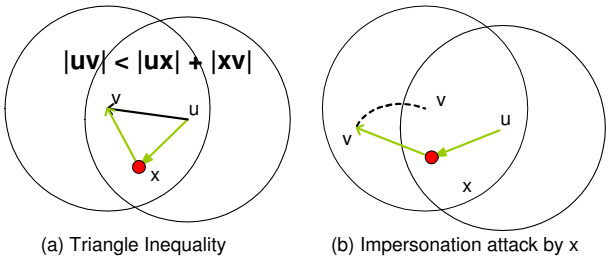


Fig. 2. Impersonation Attacks

triangular inequality among the distances of the involved nodes, as shown in Fig. 2(a). When node u sends a packet that is authenticated with a cluster key $K(i)$ in its key chain, node v normally receives the packet before it receives a forwarded copy from node x because $|uv| < |ux| + |xv|$, unless the packet is lost. Thus, it is difficult for an adversary x to impersonate node u to v by reusing node u's cluster keys.

The one-time key based scheme however cannot completely prevent the impersonation attack. In Fig. 2(b), after node v has moved out of the transmission range of node u (but v is unaware of it), it cannot know the most recent cluster key disclosed by u. Therefore, node x may reuse u's old cluster keys $K(i)$ and $K(i+1)$ to inject a false packet $P'(i)$ to node v.

$$M' : P'(i), K(i+1) \oplus MAC(K(i), P'(i)) \tag{2}$$

We note that the number of impersonated packets in this attack is bounded by the actual transmission rate of node u, because node x cannot use the cluster keys which node u has not disclosed yet due to the one-wayness property of a hash function. Moreover, reusing node u's cluster keys within the one-hop range of node u is subject to detection by u and the other neighbors. Therefore, this scheme provides reasonably strong source authentication and an attacker takes a high risk of being detected when it reuses the one-time cluster keys of other nodes.

B.2 Random Neighborship Verification

We now discuss a random neighborship verification scheme to further deter the impersonation attack. The main idea is that a node challenges its neighborship with another node with certain probability. In the example shown in Fig. 2(b), when node v receives a packet $P(i)$ from a claimed source u, it responds with a CHALLENGE message at probability p_c:

$$v \xrightarrow{p_c} u : i, MAC(K_{vu}, MAC(K(i), P(i))), \tag{3}$$

where i is the packet index. To save computational overhead, here node v refers to packet $P(i)$ by $MAC(K(i), P(i))$, which is the MAC contained in $P(i)$ (referred to Message(1)). K_{vu} is the pairwise key shared between v and u. If node u can hear this CHALLENGE message, it replies with the following ACK message:

$$u \longrightarrow v : i, MAC(K_{vu}, i|FLAG), FLAG \tag{4}$$

If FLAG is TRUE, the message proves to v that node u has really sent the packet $P(i)$; otherwise, if FLAG is FALSE,

it denies. The attack node x cannot forge the response impersonating node u because it does not have the pairwise key K_{vu}. On the other hand, the neighborship verification process is symmetric in the sense each of the two nodes will be convinced the other is its current neighbor. Hence, the number of verification processes is reduced by one half.

Clearly, the choice of p_c should make a tradeoff between security and performance. A larger p_c leads to stronger security, but it incurs larger overhead due to the exchange of challenges and responses. To control the probability p_r that a node receives a challenge, every neighbor sets its probability to challenge the node as $p_c = p_r/d$, where d is the estimated network node density. Finally, this scheme requires a node to buffer the MACs of several packets it has recently transmitted to answer possible challenges.

C. Scheme II: A Location-Aware Verification Scheme

In this scheme, instead of challenging each other randomly, two nodes do not verify their neighborship *when they believe they are highly likely in neighborhood*, thus further reducing message overhead. This scheme assumes that every node knows its own current location and velocity because of a GPS and the transmission range r of a legitimate node is a fixed system parameter known to all the nodes in the network.

Let node u's current coordinate be (X_u, Y_u), and its velocity $\overrightarrow{V_u}$. When it bootstraps its key chain commitment to a neighbor v, it also sends its location parameter $LP_u = (X_u, Y_u, V_u)$ to v in an authenticated way.

$$u \rightarrow v : LP_u, \{K_u(0)\}_{K_{uv}}, MAC(K_{uv}, K_u(0) \parallel LP_u) \quad (5)$$

Without loss of generality, let node v's location parameter be $LP_v = (X_v, Y_v, V_v)$ and its next key to be disclosed in its key chain $K_v(i+1)$. Node v responds with the following message.

$$v \rightarrow u : LP_v, \{K_v(i+1)\}_{K_{uv}}, MAC(K_{vu}, K_v(i+1) \parallel LP_v) \quad (6)$$

After this message exchange, both u and v know the location parameter and the next cluster key from each other. In addition, they record the message exchange time and estimate the time when they will move out of each other's transmission range. For instance, node v records the time when it receives the response message from u as t_u^0 and the estimated time that node u will move out of its transmission range t_u^1. It keeps the time period $[t_u^0, t_u^1]$, referred to as *Radio Effective Duration*(RED), during which node u is very possible within its transmission range. The RED is maintained as part of the record corresponding to node u in the trust list of node v.

The problem remained is how to evaluate t_u^1. Consider Fig. 3, which depicts a snapshot of the moment when node u and node v are exchanging their location parameters. Node u is moving in velocity $\overrightarrow{V_u}$, and node v is moving in $\overrightarrow{V_v}$. To simplify this problem, we focus on node u, and translate the coordinate such that node u stays still and node v is moving in a relative velocity $\overrightarrow{V_{vu}}$ under this new

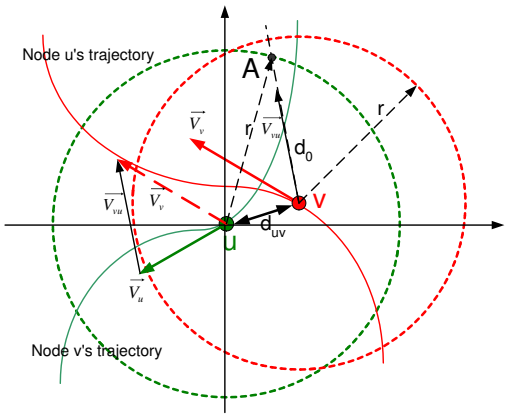


Fig. 3. Nodes u and v move at velocities $\vec{V_u}$ and $\vec{V_v}$, respectively, along two different trajectories. Their distance is d_{uv}. Using node u as the reference, we can consider that node v moves at velocity $\vec{V_{vu}}$ and will intersect with the circle centered at u at point A. After that, u and v will move out the transmission range of each other. d_0 is the distance between node v and point A.

coordinate. Our goal is to find out d_0, which is the distance between node v and the transmission boundary of node u along the extension line of $\vec{V_{vu}}$. Due to space limit, here we omit the mathematical details for acquiring d_0. After getting d_0, the time t_u^1 can be calculated by dividing the d_0 by the relative speed $|\vec{V_{vu}}|$ as the following equation:

$$t_u^1 = t_u^0 + d_0/|\vec{V_{vu}}|. \quad (7)$$

Once calculating its RED for node u, node v will use it to decide whether or not to challenge node u when it receives packets from node u later.

To make the scheme as general as possible, we have estimated REDs based on the *current* location parameters of mobile nodes. In practice, however, the estimated REDs may become invalid due to the rapid change of node velocities. If two nodes are still in each other's transmission range after their REDs have expired, they should not discard the packets from each other immediately. Therefore, we adopt a buffering strategy to minimize the impact of RED estimation errors on data delivery ratio. The required buffer size is evaluated in Section VI.

Now we show the basic steps involved in the authentication process. Suppose node v receives a packet authenticated with node u's cluster key $K_u(i)$, it will perform one of the following steps:

Step 1 if the current time is between $[t_u^0, t_u^1]$, and to its knowledge that K_i has not been released yet, and K_i is valid, it accepts the packet and passes the packet to the routing protocol.

Step 2 if the current time is larger than t_u^1, it temporarily buffers the packet. In addition, it marks the sender u *expired*, moves u from its trust list to a pending list, and then verifies its neighborship with node u. The messages exchanged between them are similar to that in messages (5) and (6). After successfully exchanging their messages, node u and node v update their RED and renew their trust relationship. Finally it goes to Step 1.

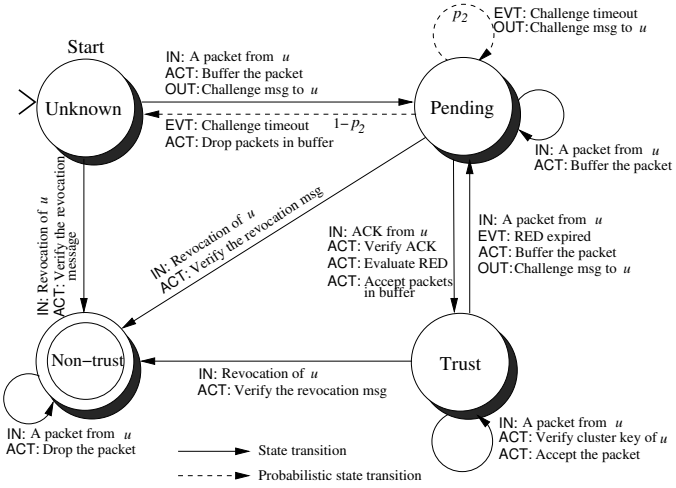


Fig. 4. Node v's state transition diagram. Each neighbor u belongs to one of the four states. In this diagram, IN: indicates receiving a message which could be a traffic packet, a revocation notice, or an ACK message from u, OUT: indicates sending a challenge to u, ACT: indicates the actions that node v takes to accomplish this transition, and EVT: indicates an event which is either timeout or RED expired. State transition is triggered by IN or EVT. Dashed arrows depict that node v can either rechallenge neighbor u or drop u's packets in buffer in case of a challenge timeout.

Step 3 if it challenges node u but no reply has been received within a threshold time, it either drops the packet and removes node u from its pending list or challenges node u again with probability p_2 (here p_2 may decrease with the number of previously failed challenges to mitigate active attacks). If a challenge succeeds, it goes to Step 1.

Fig. 4 depicts the state transition diagram used by node v when it receives different types of packets from node u. The above four steps correspond to the state "Trust".

This challenge and acknowledgement activity can be taken as a kind of synchronization. However, the frequency of synchronization in LIP depends on the relative mobility between the two nodes. As a result, if the relative positions of the two nodes are small, either moving in the same direction or in slow speeds, there would be a long period that the two nodes have no need to synchronize. Thus, we take the advantage of saving the control overhead. In some cases the relative speed of two nodes may be too small, causing t_u^1 in Eqn. 7 very large. For security reason, we set an upper bound time period RED^0 on RED. Two nodes will verify their neighborship within RED^0 even if their estimated RED is larger than RED^0.

D. Further Discussions

Below we discuss several issues related to the implementation and deployment of LIP.

D.1 Interaction With Routing Protocols

LIP is independent of the (secure) routing protocols. In practice, it could take advantage of the deployed routing protocol to provide stronger security. Previously, LIP requires that a node uses its cluster keys to authenticate all its packets to its direct neighbors despite the type of transmission (e.g., unicast, multicast, or broadcast) of each packet. However, if LIP can infer from the header of the routing protocol that a (data) packet is to be unicast (e.g., in a unicast-based application), it can use its pairwise key shared with the next hop to authenticate the packet because using pairwise key can prevent impersonation attacks. On the other hand, since the security services provided by LIP are complementary to those provided by secure routing protocols, they can be employed at the same time to provide stronger security.

D.2 Adding Robustness to Packet Losses

Now we consider the issue due to unreliable transmission. From Message(1) we can observe that if a neighbor v_j of node u lost the previous packet $P(i-1)$ that contains $K(i)$, it will not be able to verify the current packet $P(i)$ or to derive $K(i+1)$ because it does not know $K(i)$. In this case, a simple solution is that the neighbor requests $K(i)$ or $K(i+1)$ from node u. A better solution is to use a *self-healing* key distribution mechanism. Instead of using $K(i)$ for authenticating $P(i)$, we can use an earlier key $K(i-m), m > 0$. If node v_j has $K(i-m)$, it can verify $P(i)$ and derive $K(i+1)$, and then compute all the keys between $K(i-m)$ and $K(i+1)$ based on the one-way hash function even if it has missed these intermediate keys. Here the choice of m should be determined by the packet loss rate in the network.

D.3 Key Chain Generation and Renew

Subject to the network traffic patterns and the network lifetime, a node may need to transmit (forward or originate) a very large number of traffic packets. This will consume a large number of cluster keys because every cluster key is only used once. The issue of providing a sufficient number of one time keys has been addressed recently [6], [17]. We note that a key chain may need to be discarded without being used up upon a node revocation. When a node u knows that another node v is being revoked (e.g., announced by a trusted authority), if node v has been its neighbor and knows one or some of its cluster keys, node u should discard any future keys in its key chain and bootstrap a new key chain to its current neighbors other than v. This completely prevents node v from impersonating u.

D.4 Legacy Issue

Scheme II requires every node to be equipped with a GPS; this requirement may not be easily met in the very near future for every application of ad hoc network. To support incremental deployment, we consider the case where only a fraction α of nodes are equipped with GPS devices (referred to as *GPS nodes*) while the rest do not have GPS devices (referred to as *non-GPS nodes*). Clearly, Scheme I and Scheme II are special cases of this hybrid scheme when $\alpha = 0$ and $\alpha = 1$, respectively.

The coexistence of GPS nodes and non-GPS nodes poses several questions. First, how can a node distinguish these two kinds of neighbors and then response with the correct control message? The way to distinguish between GPS and non-GPS nodes is to incorporate one flag bit in each packet

telling which type of node the sender belongs to. Second, which scheme should two nodes use to authenticate their packets? They run Scheme II (with the location-aware verification) only when they both are GPS nodes; in other cases they apply Scheme I (with the random neighborship verification process) due to the lack of information to compute their distance and relative velocity. As such, α^2 fraction of node pairs run Scheme II and $(1-\alpha^2)$ fraction of pairs run Scheme I. In addition, the recorded information in a node's trust list for two types of neighbors is also different. The performance of this hybrid scheme is studied in Section VI.

V. Security Analysis

This section analyzes the security of our schemes based on the security threat models in [12]. The security threat of these models, from *Passive* to *ActiveCCX*, increases with strength.

- *Passive*: An attacker, without cryptographic keys from the network controller, only passively eavesdrop on the traffic in the network. In LIP, as long as the underlying encryption algorithm and authentication algorithm are secure, an attacker cannot break the cluster keys of the legitimate nodes.
- *Active1*: An active attacker attempts to inject malicious packets into the network although it has no cryptographic keys from the network controller. Since LIP performs hop-by-hop authentication of every packet, without knowing a valid cluster key, the attacker cannot inject its own packets into the network. Therefore, our schemes can prevent the Active1 attack. We note that it is possible that the attacker replays another node's packets, however, the attacker achieves little by doing this. This is because (i) the attacker can only replay the packets transmitted by a node u to the other nodes that possess node u's cluster keys. (ii) If the nodes having node u's cluster key are node u's current neighbors, they will drop the duplicated packets based on packet sequence numbers or cluster key versions. For example, if an attacker replays node u's ROUTE REQUEST packets to node u's neighbors in DSR [14], the replay attack does not lead to multiple flooding of the same packet in the entire network because of the *request id* in the packet. (iii) If the nodes possessing some cluster keys previously released by node u are not neighbors of u, they may accept the packet if the packet is a broadcast packet and they have not received it before. This however actually increases the reliability on the delivery of broadcast messages. (IV) If time synchronization is provided (e.g., using GPS in Scheme II), a timestamp can be used to further prevent replay attacks.
- *ActiveX*: This threat model consists of multiple instances of the Active1 model. An ActiveX attack is not more severe than a single Active1 attack for LIP except when multiple attackers collude to launch the *wormhole* attack [12]. Scheme I cannot prevent these attacks. In Scheme II, two neighboring nodes exchange their location parameters once they receive the first packet from each other, whereby determining their RED, say $[t_0, t_1]$. If they are in neighborhood before t_1, they will be able to detect and prevent the wormhole attack. However, if they move out of each other's transmission range at t_m before t_1, they cannot completely prevent the attack during $[t_m, t_1]$. We note that this vulnerability can be addressed by letting a node include its current location in each packet, but this incurs larger bandwidth overhead.
- *ActiveC*: One active attacker node has all the cryptographic keys of a compromised node. Since an ActiveC attacker takes over the node, it can do whatever a node is allowed to do in the system on behalf of the compromised node. For most security systems, we have to resort to using some intrusion/misbehavior detection techniques [19], [27] to defend against this type of attack. Another attack an ActiveC attacker can launch in our protocol is the impersonation attack due to the use of MAC-based broadcast authentication schemes. Indeed, both Scheme I and Scheme II are designed to mitigate this attack.
- *ActiveCX* and *ActiveCCX*: In the ActiveCX threat model, multiple active attacker nodes have all the cryptographic keys of one compromised node, whereas in the ActiveCCX model, multiple active attackers have all the keys of multiple compromised nodes. To reduce the risk of being detected because of using the same identity, ActiveCX attacker nodes are usually distributed in different locations of the network. ActiveCCX attacks can be even more sophisticated and difficult to detect. Our protocol alone does not have a solution for addressing this attack. We note a better solution is that every node is installed with an intrusion detection system (IDS). Moreover, multiple nodes could also perform cooperative detection, for example, by recording and exchanging their neighborship information. Zhang and Lee [27], Marti et al. [19] have studied the intrusion and misbehavior detection issue in mobile networks. We believe this is still an open area, and a study that identifies the possible attack patterns is the first step towards addressing these attacks.

VI. Performance Analysis

Since we have discussed the capability of LIP in filtering unauthenticated data packets and other threat models in Section V, we will not examine this capability through simulation. The goal of our experiments is to measure the performance overhead introduced by LIP when the network is *not* under adversary attack. In particular, we want to answer the following questions: *how much bandwidth overhead does LIP introduce*? *how much packets does LIP drop*, and *what is the impact of location-awareness on LIP*? The simulation results are based on Scheme II, the *location-aware* scheme unless otherwise mentioned.

A. Metrics

We mainly consider the following performance metrics in this scheme.

- **Control Overhead** We define control overhead as the transmission overhead (in bytes per second per node) introduced by our scheme, which includes one MAC attached

to each packet and all the challenge-response messages in our protocol.

• **Traffic Delivery Ratio** We define the traffic delivery ratio as the fraction of traffic packets that a node *accepts* to the total number of packets it *receives* from its legitimate neighbors. The higher the traffic delivery ratio, the smaller impact on the upper layer protocols.

Note that here we do not consider computational overhead because the scheme mainly involves several symmetric key operations (MAC and hash computations), which are all computationally efficient. The other computational load would be the infrequent estimation of REDs, which is very fast to compute.

B. Simulation Methodology

This simulation utilizes GlomoSim 2.02 [8] and sets the default configuration as follows. There are 100 nodes distributed in a square environment space of 2000m × 2000m. Each node joins the network at a time uniformly distributed between the simulation time 0 and 5s, and each simulation runs for 900 seconds of simulated time. In physical layer, the radio propagation model is two-ray ground reflection model. In medium access control layer, we use IEEE 802.11 Distributed Coordination Function (DCF).

To demonstrate that LIP is independent of the routing protocol, we insert LIP beneath three different routing protocols the unicast routing protocols DSR [14] and AODV [21] and the multicast routing protocol ODMRP [18]. When the routing protocol is either AODV or DSR, we pick up 13 source-destination pairs for unicast communication. The setting for ODMRP is that, of 100 nodes, 13 nodes form one multicast group, and 12 nodes form another multicast group. The rest of 75 nodes do not belong to either of these two groups. The rest parameters of these protocols are simply the default values in GloMoSim.

In application layer, the traffic pattern is constant bit rate (CBR). The size of a CBR packet is 512 bytes. We vary the intervals between two CBR packets from 0.1s to 1.0s and the durations of connections from 10s to 850s.

Table I summarizes the default parameters used in our simulation unless otherwise mentioned.

TABLE I
THE SIMULATION PARAMETERS

Parameters	Values
Physical Link Bandwidth	2 Mbps
Radio Frequency	2×10^9 Hz
Radio Transmission Power	15 dBm
Radio Transmission Sensitivity	-91 dBm
Radio Transmission Threshold	-81 dBm
mobility model	Waypoint
RED^0	200 sec.
Threshold time for waiting responses	2 sec.
Random neighbor challenge probability p_c	0.1
Rechallenge probability p_2	0
Location information size	8 bytes
Velocity information size	8 bytes
Time for MAC verification	1 μs
HMAC Key size (including a key id)	10 bytes

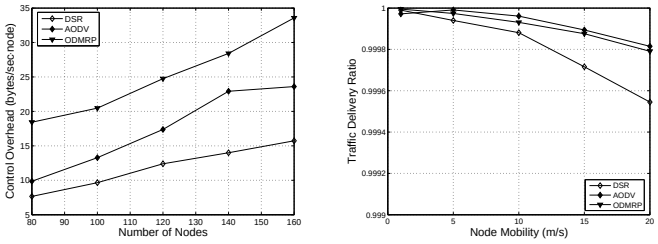
C. Evaluation Results

All the simulation results in this subsection are averaged over 40 independent runs.

C.1 The Impact of Node Mobility

a) Control Overhead: Fig. 5(a) shows that control overhead increases with node mobility. The control overhead is normally between 10 and 20 bytes/sec·node for all three routing protocols. This indicates that our scheme has low bandwidth overhead.

The impacts of node mobility on control overhead are three folds. First, with higher node mobility, RED is normally smaller. In other words, nodes have to synchronize with each other more frequently, which in turn increases the number of challenge and ACK messages. Second, with higher mobility, a node will encounter more nodes. Chances are many of them have never contacted before or their neighborship have expired. When a node receives a packet from its neighbor that is not in its trust list, it will automatically send a challenge to the neighbor, and the neighbor will also respond with an ACK message. This round of challenge and ACK alone costs 58 bytes of control overhead. Third, the rise of node mobility causes the routing protocols such as AODV, DSR, and ODMRP, to transmit more control packets to maintain network connectivity. Since LIP authenticates every traffic packet, the increase of the amount of control packets also implies the enlarged control overhead. Since DSR has lower control packet overhead than AODV and ODMRP, the overhead of LIP with DSR is also the lowest.



(a)Control Overhead (b)Traffic Delivery Ratio

Fig. 5. The impact of node mobility on control overhead and traffic delivery ratio.

b) Traffic Delivery Ratio: Fig. 5(b) indicates that the traffic delivery ratio of LIP is close to 1.0 though it goes down slightly when the node mobility increases. In the worst case in DSR, a node drops about 5 traffic packets out of 10,000 packets it receives. The packet loss is mainly due to normal network packet loss when the network topology changes. Another source of packet loss is dropping packets from a neighbor which failed to respond to a challenge. Since our scheme only makes proximate estimation of RED, it is possible that the RED of a neighbor node expires while it is still within the transmission range. In this case, if that neighbor broadcasts traffic packets for forwarding, those packets will be temporarily stored and a challenge will be

sent to it. Not until an ACK is received from the neighbor will the traffic packets from it be accepted and passed to the routing protocol. If the neighbor moves out of transmission range before it can respond to this challenge, all its packets temporarily stored will be dropped. We can also notice that DSR has slightly lower traffic delivery ratio than the other two. This is mainly because of the feedback implosion problem. In DSR a node may keep silent at most of the time. It is possible none or few of its neighbors know its existence until it broadcasts a message, which causes many neighbors to challenge it at the same time and hence more challenge packets will be lost.

c) Buffer Size: Fig. 6 depicts the temporary storage cost of LIP in terms of the number of packets. In this figure, the maximum buffer length means the largest storage space that a node had ever used throughout the simulation. We can make two observations from the figure. First, the average and maximum buffer length grows with the node mobility. The reason is that a node encounters more neighbors when it moves at a higher speed. For each neighbor with an expired RED, the packets from it will be temporarily stored in the buffer until an ACK from it is received and verified. Second, LIP under DSR has a lower storage overhead than under AODV or ODMRP. This is because, under DSR, LIP has smaller trust lists. In the worst case in the figure about 135 packets are buffered, which account for 67.5 KB. For many wireless devices such as PDAs, this storage overhead is not a bottleneck.

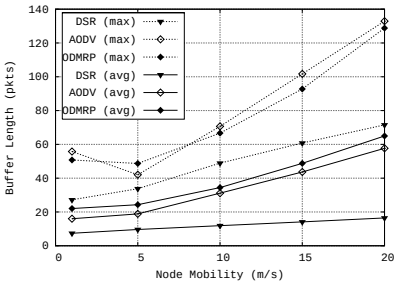


Fig. 6. The impact of node mobility on the length of buffer

C.2 Impact of Node Density

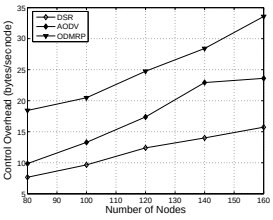
To examine the impact of node density, we vary the number of nodes in the 2000m × 2000m field from 60 to 160. We also increase the number of source-destination pairs (approximately) linearly.

a) Control Overhead: Fig. 7(a) shows the impact of node density on control overhead. We observe that the control overhead increases with node density under every routing protocol. This is because the control overhead of LIP depends on the communication load. The MAC attached to each packet for authentication accounts for the majority of the control overhead.

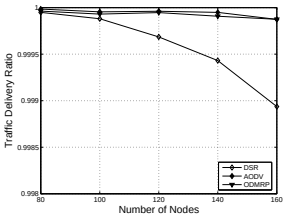
b) Traffic Delivery Ratio: Fig. 7(b) shows the impact of node density on traffic delivery ratio. Still, the traffic delivery ratio is close to 100%.

C.3 Impact of Location Awareness

As we discussed earlier, to support incremental deployment, we need to consider the case where only a fraction α of nodes are *GPS nodes* while the rest are *non-GPS nodes*). Now we examine the impact of α on the performance of LIP.



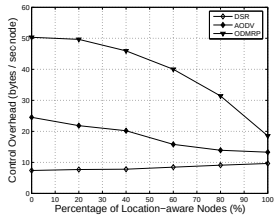
(a)Control overhead



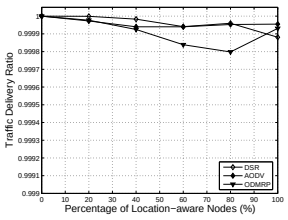
(b)Traffic delivery rate

Fig. 7. The impact of node density on control overhead and traffic delivery ratio (node mobility 10 m/s)

Fig. 8 illustrates that with challenge probability $p_c = 0.1$, control overhead decreases with α (except slightly different in the case of DSR). The reason is that, instead of randomly challenging a neighbor, a location-aware node knows more precisely when to challenge based on its RED. The control overhead in LIP under ODMRP is relatively large, 50 bytes/sec·node, because of the large number of control packets in ODMRP. To reduce the control overhead, one may reduce p_c as a tradeoff between security and performance. Fig. 9 shows that the traffic delivery ratios are close to 100% under all the α values.



(a)control overhead



(b)traffic delivery ratio

Fig. 8. The impact of location-awareness on control overhead and traffic delivery ratio ($p_c = 10\%$, node speed 10 m/s).

C.4 Impact of Mobility Models

In LIP, node mobility model affects REDs. For example, nodes moving in a group will have on average larger REDs (although bounded by RED^0) because their relative speeds are smaller. To examine the impact of mobility models, next we introduce two more mobility models: Reference Point Group Mobility (RPGM)[9] and Manhattan[4]. We compare these two mobility models with the default random waypoint mobility model at a maximum speed of 10 m/s. We utilize BonnMotion [3] to generate mobility scenarios. Table II lists the parameters used to generate scenario files.

In RPGM, nodes are divided into logical groups. For each group, there is a reference point. All of the nodes belonging to that group move according to that point. Since the nodes of one group generally move together around, more stable neighborship among nodes is expected. This results in larger REDs. Fig. 9 (b) confirms it. In Manhattan mobility model, although some geographic restrictions are imposed on node mobility, we find that its REDs are the smallest. The control overhead in different mobility

TABLE II
THE MOBILITY MODEL PARAMETERS

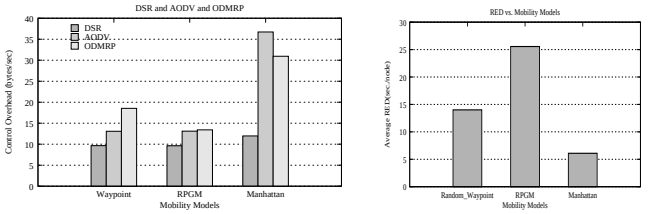
RPGM

Average number of nodes per group	20
Group number	5
Group change probability	0
Maximum distance to group center	50 m
Minimum speed	1 m/s
Maximum speed	10 m/s
Maximum pause	30 sec.

Manhattan

Minimum speed	1 m/s
Mean speed	10 m/s
Maximum pause	30 sec.
Pause probability	0.001
Turn probability	0.2
Update distance	10m
Number of blocks along x-dimension	10
Number of blocks along y-dimension	10

models, as shown in Fig. 9(a) conform to the observations on the different REDs shown in Fig. 9 (b).



(a)Control Overhead (b)REDs

Fig. 9. Control overhead and REDs under different mobility models

Overall, the above performance analysis shows that LIP is a lightweight protocol and it provides traffic delivery ratio close to 100% with different node speeds, network densities, and mobility models.

VII. CONCLUSIONS

We have presented LIP, a lightweight network access control protocol for preventing unauthorized nodes from injecting spurious packets into ad hoc networks. The protocol is *transparent* to and *independent* of the network routing protocols. In addition to being able to prevent resource consumption attacks by unauthorized nodes, LIP can mitigate impersonation attacks by compromised nodes. Our detailed simulations showed that LIP has low communication overhead and traffic delivery ratio close to 100%.

REFERENCES

[1] I. Aad, J. Hubaux, and E. Knightly. Denial of Service Resilience in Ad Hoc Networks. In Proc. of ACM Mobicom 2004.

[2] L. Buttyan and J. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. In ACM/Kluwer Mobile Networks and Applications (MONET), vol. 8, no. 5, Oct. 2003.

[3] BonnMotion, http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/

[4] F. Bai, N. Sadagopan, and A. Helmy IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. INFOCOM 2003.

[5] A. Cardenas, S. Radosavac, and J. Baras. Detection and Prevention of MAC Layer Misbehavior for Ad Hoc Networks. In Proc. of ACM workshop SASN '04.

[6] D. Coppersmith, M. Jakobsson, Almost Optimal Hash Sequence Traversal, Finanical Cryptography (FC) 02

[7] B. Dahill, B. Levine, E. Royer, C. Shields, A Secure Routing Protocol for Ad-Hoc Networks, In Proc. of IEEE ICNP 2002.

[8] GloMoSim, http://pcl.cs.ucla.edu/projects/glomosim/

[9] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, A group mobility model for ad hoc wireless networks, in ACM/IEEE MSWiM, Augest 1999.

[10] Y. Hu, D. Johnson, A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In Proc. of IEEE Workshop WMCSA 2002.

[11] Y. Hu, A. Perrig, D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In Proc. of ACM Mobicom 2002

[12] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In Proc. of INFOCOM 2003.

[13] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks In proceedings of the 5th International Conference on Mobile Computing and Networking (ACM MOBICOM 99), August 1999, pages 195- 206.

[14] D. Johnson, D. Maltz, Y. Hu, J.Jetcheva. The Dynamic Souce Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-07.txt, February 2002.

[15] Leslie Lamport. Password authentication with insecure communication communication. Communications of the ACM, 24(11):770-772, Nov., 1981.

[16] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. IEEE/ACM Transactions on Networking, December, 2004.

[17] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In Proc. of ACM CCS 2003.

[18] S. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communication in Wireless Mobile Networks, to appear, 2001.

[19] S. Marti, T. Giuli, K. Lai, M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. ACM MOBICOM, 2000.

[20] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secuire source authentication for multicast. In Network and Distributed System Security Symposium, NDSS'01, Feb. 2001.

[21] Charles Perkins. Ad hoc On Demand Distance Vector (AODV) Routing, Internet draft, draft-ietf-manet-aodv-00.txt.

[22] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad hoc Networks. In SCS Conference (CNDS 2002), 2002.

[23] F. Stajano and R. Anderson. The Resurrecting Ducking: Security Issues for Ad-hoc Wireless Networks. In Security Protocols, 7th International Workshop, 1999.

[24] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. Proceedings of the 2004 ACM Workshop on Wireless Security.

[25] S. Yi, P. Naldurg, R. Kravets. Security-Aware Ad-Hoc Routing for Wireless Networks. In Proc. of Mobihoc 2001.

[26] L. Zhou and Z. Haas. Securing Ad Hoc Networks. IEEE Network, 13(6):2430, 1999.

[27] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad-Hoc Networks. MOBICOM 2000.

[28] S. Zhu, S. Xu, S. Setia, and S. Jajodia. LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks. In Proc. of ICDCS Workshop MWN 2003, May 2003.

[29] S. Zhu, S. Setia, S. Xu, and S. Jajodia. GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks. In Proc. of Mobiquitous 2004.