
LIP: a lightweight interlayer protocol for preventing packet injection attacks in mobile ad hoc network

Hung-Yuan Hsu,* Sencun Zhu and Ali R. Hurson

Department of Computer Science and Engineering,
The Pennsylvania State University, USA

E-mail: hhsu@cse.psu.edu

E-mail: szhu@cse.psu.edu

E-mail: hurson@cse.psu.edu

*Corresponding author

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 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 authorised nodes can inject traffic into the network. We design a Lightweight Inter-layer Protocol (LIP) for preventing packet injection attacks based on an efficient local broadcast authentication mechanism. In addition to preventing attacks by unauthorised nodes, LIP can also detect and minimise the *impersonation attacks* by compromised nodes. Through detailed simulation study, we show that LIP is scalable, and it incurs small bandwidth overhead as well as 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.

Keywords: ad hoc network; injection attack; hop-by-hop authentication.

Reference to this paper should be made as follows: Hsu, H-Y., Zhu, S. and Hurson, A.R. (2007) 'LIP: a lightweight interlayer protocol for preventing packet injection attacks in mobile ad hoc network', *Int. J. Security and Networks*, Vol. 2, Nos. 3/4, pp.202–215.

Biographical notes: Hung-Yuan Hsu is a PhD candidate in the Department of Computer Science and Engineering at The Pennsylvania State University, USA. Since 2005 he has been researching in Mobile Ad hoc Networks (MANETs) security.

Sencun Zhu is an Assistant Professor in the Department of Computer Science and Engineering at The Pennsylvania State University, USA. His research interests are in network and system security, especially key management, ad hoc and sensor network security, DDoS attack prevention and worm detection.

Ali R. Hurson is a Professor in the Department of Computer Science and Engineering at The Pennsylvania State University, USA. He is the co-founder of the *IEEE Symposium on Parallel and Distributed Processing* (currently IPDPS) and *IEEE Conference on Pervasive Computing and Communications*.

1 Introduction

Most ad hoc networks do not have any provisions for restricting the traffic that flows through a node, that is, 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, for example, 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 (Xu et al., 2004), the packet injection attack using broadcast messages may be more favourable 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. 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 neighbours will forward packets for it and also assumes that the packets it receives from its neighbours are authenticated. This naive trust model allows a malicious node to inject erroneous routing requests or routing updates into a network, which can paralyse the entire network. To deal with such attacks, recently several security extensions (Hu et al., 2002a,b; Papadimitratos and Haas, 2002; Sanzgiri et al., 2002; Yi et al., 2001) 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 key¹ shared by all nodes. Every node uses this shared key to compute Message Authentication Codes (MACs)² on the packets it sends and verify packets from its neighbours. Despite its simplicity, this scheme has several disadvantages. Firstly, an attacker only needs to compromise one node to break the security of the system. Secondly, 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. Finally, 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 attach n MACs to the packet, where n is the number of its immediate neighbours. 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.

1.1 Contribution

We present Lightweight Interlayer Protocol (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 *localised* broadcast authentication mechanism using which a node authenticates its packets only to its immediate neighbours. 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. A location-aware version of LIP can also prevent sophisticated attacks such as Wormhole attacks (Hu et al., 2003). Another unique feature of LIP is its *transparency* and *independence* with respect to the network routing protocols due to its interlayer 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. Furthermore, the per-hop per-packet authentication capability provided by LIP is critical not only for many civilian applications of ad hoc networks that do pricing or require cooperation stimulation and incentive (Buttyán and Hubaux, 2003; Luo et al., 2003; Zhong et al., 2002), but also for enabling the traceback of DoS attackers in mobile ad hoc networks.

Note that our contribution does not lie in using the technique of one-way key chain as one of the security mechanisms, as this technique has been extensively adopted by hundreds of protocols since its invention by Lamport (1981); instead, we contribute to the literature by proposing a practical solution to a significant problem that is interesting enough by itself, regardless of the underlying building blocks.

The rest of this paper is organised as follows. We first discuss related work in Section 2. Then we present the details of the protocol in Section 3 and analyse its security in Section 5. In Section 6, we analyse the performance of our protocol. Finally, we conclude this paper in Section 7.

2 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. Sanzgiri et al. (2002) identify several security vulnerabilities in AODV (Perkins et al., 2003) and DSR (Jonhson et al., 2001) and proposed to use asymmetric cryptography for securing ad hoc routing protocols. Yi et al. (2001) 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 et al. (2002a) designed SEAD for securing DSDV (Perkins and Bhagwat, 1994) and Ariadne (Hu et al., 2002b) for securing DSR. Both SEAD and Ariadne use a one-way key chain. SEAD uses one-way key chain for authentication of hop count. Ariadne uses a TESLA (Perrig et al., 2001) key chain, a variant of one-way key chain, for authentication of routing control packets. 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 Aad et al. (2004) quantitatively study the DoS resilience of an ad hoc network under Jellyfish attack and Black-hole attack. In Zhu et al. (2003) propose LHAP, a protocol for preventing resource consumption attacks in ad hoc networks. LHAP uses TESLA (Perrig et al., 2001) for bootstrapping one-way key chains. However, the use of TESLA in LHAP leads to some inherent difficulty. Firstly, TESLA requires *periodic* key disclosure, thus introducing some constant bandwidth overhead that is independent of the actual traffic rate. Secondly, 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 impersonation attacks as well. Moreover, LIP incurs smaller bandwidth overhead than LHAP does. With velocity varying from 0 to 20 m/s, in LHAP the per node bandwidth overhead is between 25 and 65 bytes/s, whereas in LIP, as shown in Section 6, the bandwidth overhead is between 8 and 23 bytes/s.

Based on threshold cryptography, Zhou and Haas (1999) and Luo et al. (2004) 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 authorisation and node revocation. Since these schemes and LIP address different issues, they can be employed in parallel.

3 Assumptions and design goal

3.1 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,³ for example, preloading pairwise keys or probabilistic-polynomials (Liu and Ning, 2003) 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 online 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 (Hu et al., 2002b; Sanzgiri et al., 2002) 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 (Xu et al., 2004) can be employed to prevent physical jamming attacks if necessary. Cardenas et al. (2004) have studied techniques for detecting and preventing media access control layer attacks.

3.2 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 (unauthorised) node that does not possess a valid credential, or an insider (authorised) node that possesses a valid credential. An insider node launches the impersonation attack because it has been compromised or it intentionally does it; we do not distinguish the attack motivation here. We use the term 'impersonation'

to refer to the case when compromised nodes impersonate non-compromised nodes, not when compromised nodes impersonate each other.⁴ To achieve the attack goal, an attacker may eavesdrop, reorder, and drop packets, fabricate packets, replay older packets or modify overheard packets and reinject 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 (Marti et al., 2000)) for each node. The violation of the regulation indicates the compromise of the node and rekeying schemes such as GKMPAN (Zhu et al., 2004) may then be applied to revoke the compromised node.

3.3 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 minimise 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, that is, there should be no latency in authenticating a received packet; otherwise, the latency of packet delivery will be unacceptably high in a multihop 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, that is, 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.

4 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.

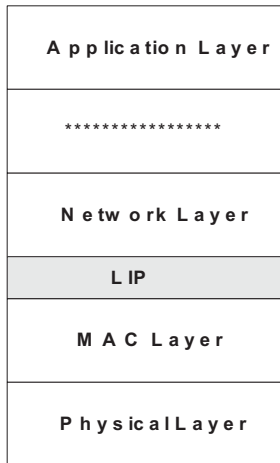
Notation: we use the following notation to describe security protocols and cryptography operations in this paper:

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

4.1 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.

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



To minimise packet overhead, we design LIP based on a *localised* broadcast authentication mechanism in which a node only computes and attaches one MAC to each traffic packet it is forwarding (or originated from it). For its neighbours to verify its packets, a node must share its MAC keys (referred to as *one-time authentication* keys hereafter) with its neighbours. Hence, we introduce the *one-time authentication key management* process for a node to establish and maintain its one-time authentication keys. The one-time authentication keys of a node should only be used by the node to authenticate its packets to its neighbours while its neighbours use the same one-time

authentication keys only for verification purpose. However, due to the symmetry nature of one-time authentication keys, a malicious neighbour may impersonate the node by using the node's one-time authentication keys to generate MACs over injected packets. To thwart this impersonation attack, we propose three techniques: *one-time authentication key*, *random neighbourhood verification* and *location-aware verification*. The use of one-time authentication keys builds the first defence 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 defence line. By assuming the availability of node location and velocity information, location-aware verification can further reduce bandwidth overhead. The details of these techniques are presented below.

4.2 Scheme I: basic scheme

4.2.1 Using one-time authentication keys

The basic scheme uses one-time authentication keys; that is, a node uses every authentication key only once to thwart an attacker from reusing its authentication keys. One-time authentication keys are provided by the technique of one-way key chains (Lamport, 1981). 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 one-time authentication keys and every one-time authentication key is used for authenticating one packet. To enable its neighbours to verify an one-time authentication key in its key chain, a node first bootstraps its key chain by sending the commitment of its key chain, that is, $K(0)$, to each of its current neighbours, encrypted with their pairwise key. The node then uses an one-time authentication key in its key chain to compute the MAC of a packet it is transmitting. Note that the one-time authentication 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 $K(i) = H^{j-i}(K(j))$.

Consider the scenario where node u wants to authenticate a packet $P(i)$ to its neighbours v_1, v_2, \dots, v_m , using $K(i)$ as the MAC key. In the message M that contains $P(i)$, the node embeds its next one-time authentication key $K(i+1)$ and attaches a MAC of $P(i)$ computed with $K(i)$. Assuming that the key chain size is n where $i = 1, 2, \dots, n$, we present the message M as follows:

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

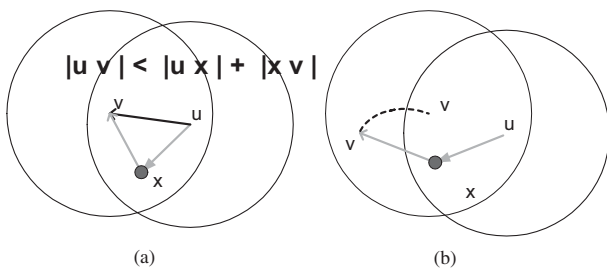
Here, we assume the size of a key is the same as the output of a MAC, for example 8 bytes. When a neighbour node v receives the message, it performs three operations. Firstly, it

computes $\text{MAC}(K(i), P(i))$ based on $K(i)$, which it derives from the previous message. Secondly, 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 an one-time authentication key prior to $K(i+1)$ will be discarded. As a result, an attacker cannot simply reuse the previous one-time authentication keys of node u to deceive a neighbour 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 neighbours, its LIP protocol will use node v_j 's own one-time authentication key to authenticate the packet to others. As such, a packet is authenticated in a *hop-by-hop* fashion.

Now we consider the synchronisation issue due to unreliable transmission. From Message(1) we can observe that if a neighbour 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 neighbour 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.

The above authentication scheme is motivated by two observations. Firstly, since packets are authenticated hop-by-hop, a node only needs to authenticate a packet to its *immediate* neighbours. Secondly, when a node sends a packet, a neighbour will normally receive the packet before it receives a copy forwarded by any other nodes. This is due to the *triangle inequality* among the distances of the involved nodes, as shown in Figure 2(a). When node u sends a packet that is authenticated with an one-time authentication 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 one-time authentication keys.

Figure 2 Impersonation attacks (a) triangle inequality and (b) impersonation attack by x



The one-time key-based scheme however cannot completely prevent the impersonation attack. In Figure 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 one-time authentication key disclosed by u . Therefore, node x may reuse u 's old one-time authentication keys $K(i)$ and $K(i+1)$ to inject a false packet $P'(i)$ to node v using a directional antenna.

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

More complicated attacks may involve more than one colluding nodes. For example, even when node v is still in the range of u , another malicious node could manage to jam node v when node u is transmitting. Later node x could send a modified packet to v impersonating u as in the previous case. We note, however, 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 one-time authentication keys which node u has not disclosed yet due to the one-wayness property of a hash function. Moreover, reusing node u 's one-time authentication keys within the one-hop range of node u is subject to detection by u and the other neighbours. Therefore, this scheme provides reasonably strong source authentication and an attacker takes a high risk of being detected when it reuses the one-time authentication keys of other nodes.

4.2.2 Random neighbourhood verification

We now discuss a random neighbourhood verification scheme to further deter the impersonation attack. The main idea is that a node challenges its neighbourhood with another node with certain probability. In the example shown in Figure 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 :

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

where i is the packet index and K_{vu} is the pairwise key shared between v and u . To save computational overhead, here node v refers to packet $P(i)$ by $\text{MAC}(K(i), P(i))$, which is the MAC contained in $P(i)$ (refer to Message(1)). Also, node u is required to keep the MAC of every transmitted packet for a short time interval to recognise any challenged packet. If node u can hear this CHALLENGE message, it replies with the following ACK message:

$$u \rightarrow v : i, \text{MAC}(K_{vu}, i \parallel \text{FLAG}), \text{FLAG} \quad (4)$$

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} ; thus, it will not be able to pass the inspection. However, the attack node x may maliciously drop the CHALLENGE message pretending the message was lost in transmission. In this case, after a timeout, node v increases the probability p_c to 1 and starts to resend the *same* CHALLENGE message whenever receiving a packet from a claimed source u . Then node v starts to count challenge failure times; once the challenge failure times exceeds a threshold or it keeps greater than zero after a timeout,

node v drops the neighbourhood of node u and further forensic analysis may be taken to identify the attacker. Contrarily, if node v receives an ACK message from node u before the challenge failure times reaches the threshold. Node v resets p_c and clears the challenge failure times. Note that node v must not change the CHALLENGE message in the following challenges since the first challenge failed in order to prevent the attack node x using new packets from u as a *credential* to regain the trustworthiness of node v . Moreover, node v should not *actively* resend the CHALLENGE message to node u in order to avoid the attack node x luring node v into consuming network resource.

In this way, the neighbourhood verification process is symmetric in the sense each of the two nodes will be convinced the other is its current neighbour. Hence, the number of verification processes is reduced by one half. Also, the choice of p_c should make a trade-off 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 neighbour 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.

4.3 Scheme II: a location-aware verification scheme

In this scheme, instead of challenging each other randomly, two nodes do not verify their neighbourhood *when they believe they are highly likely in neighbourhood*, 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 \vec{V}_u . When it bootstraps its key chain commitment to a neighbour v , it also sends its location parameter $LP_u = (X_u, Y_u, V_u)$ to v in an authenticated way.

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

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

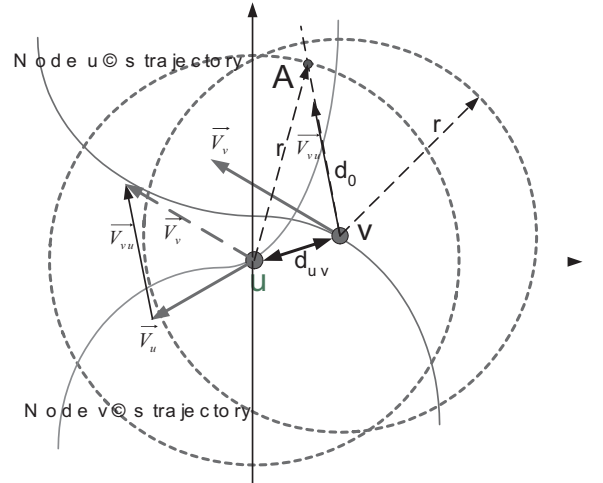
$$v \rightarrow u: LP_v, i, \{K_v(i)\}_{K_{uv}}, MAC(K_{vu}, i \parallel \{K_v(i)\}_{K_{uv}} \parallel LP_v) \quad (6)$$

After this message exchange, both u and v know the location parameter and the previous one-time authentication key from each other. An attacker cannot replay the above message. 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 Figure 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 \vec{V}_u and node v is moving in \vec{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 \vec{V}_{vu} under this new 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} .

Figure 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



Consider the triangle ΔAuv in Figure 3. Since we already know the transmission range r and the distance d_{uv} , we can acquire d_0 by applying Equation (7).

$$r^2 = d_0^2 + d_{uv}^2 + 2d_0d_{uv} \cos \theta \quad (7)$$

where θ is the degree of the angle on vertex v in ΔAuv . The value of $\cos \theta$ can be obtained from the following equation.

$$\cos \theta = \frac{|\vec{V}_{vu} \vec{vu}|}{|\vec{V}_{vu}| |\vec{vu}|} \quad (8)$$

where \vec{vu} is a relative position vector pointing from v to u and $|\vec{vu}| = d_{uv}$. By rearranging Equation (7), we deduce a quadratic polynomial shown in Equation (9). By solving this equation, we can get two roots, which represents d_0 and the distance from v to the transmission boundary of u along the opposite direction of \vec{V}_{vu} respectively.

$$d_0^2 + 2d_{uv} \cos \theta d_0 + (d_{uv}^2 - r^2) = 0 \quad (9)$$

After getting d_0 , the time t_u^1 can be calculated by dividing the d_0 by the relative speed $|V_{vu}|$ as the following equation:

$$t_u^1 = t_u^0 + \frac{d_0}{|V_{vu}|} \quad (10)$$

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. Note that this inaccuracy does not introduce new security vulnerability if two nodes have moved out of each other's transmission range before their REDs have expired. However, if they are still in each other's transmission range after their REDs have expired, they should not discard the data packets from each other immediately; otherwise, a large number of legitimate packets will be falsely dropped. To address this issue, we adopt a buffering strategy, in which a node first temporarily buffers a data packet from a node whose RED has expired and then challenges that node for reauthentication, thus minimising the impact of RED estimation errors on data delivery ratio. The required buffer size is evaluated in Section 6.

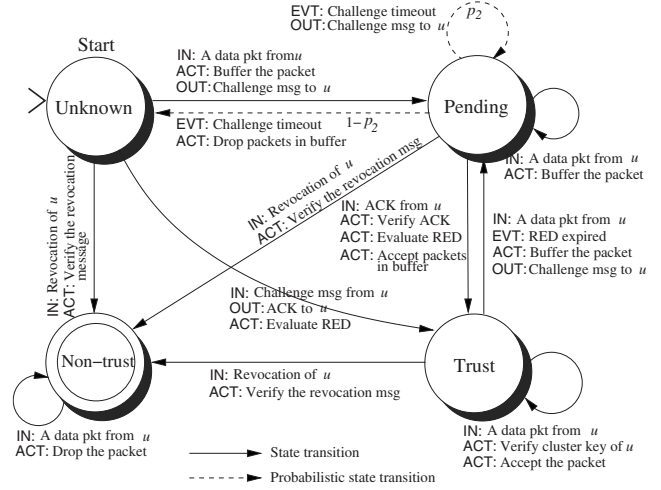
Now we show the basic steps involved in the authentication process. Suppose node v receives a data packet authenticated with node u 's one-time authentication key $K_u(i)$, it will perform one of the following steps:

- 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 neighbourhood with node u . The messages exchanged between them are similar to that in messages (5) and (6) except that they both exchange the most recently released one-time authentication keys. After successfully exchanging their messages, node u and node v update their REDs and renew their trust relationship. Finally it goes to Step 1.
- 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 updates the corresponding RED and goes to Step 1.

Figure 4 depicts the state transition diagram used by node v when it receives different types of packets from node u . In the diagram there is no transition from 'non-trust' to other states, which indicates that any packets from a revoked node will be dropped immediately. However, if the enforced

security policy allows a revoked node to come back for whatever reasons, the network controller could broadcast a reauthorisation message. In this case, a transition from 'non-trust' to 'unknown' may be added.

Figure 4 Node v 's state transition diagram. Each neighbour u belongs to one of the four states



Note: IN: indicates receiving a message which could be a traffic packet, a revocation notice, a challenge message 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 neighbour u or drop u 's packets in buffer in case of a challenge timeout.

This challenge and acknowledgement activity can be taken as a kind of synchronisation. However, the frequency of synchronisation 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 synchronise. 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 Equation (10) to become very large. For security reason, we set an upper bound time period RED^0 on RED. Two nodes will verify their neighbourhood within RED^0 even if their estimated RED is larger than RED^0 .

Finally, we mention that there exists a DOS attack against the above scheme because of the buffering strategy. For instance, an attacker may impersonate a legitimate node u , which has moved out of neighbourhood and whose RED has also expired, by injecting a large number of data packets to node v . Given a fixed buffer space, this attack could cause node v to drop data packets from other legitimate neighbouring nodes whose REDs have expired. To address this attack, node v could allocate a fixed buffer size for each pending node and enforce a drophead or droptail packet dropping strategy. If the number of buffers is also limited, node v could first recycle the buffers for the nodes who were challenged earlier but still have not responded yet.

The above attack seems to also result in service denial to node u . Because node v challenges the impersonated

node u with a decreasing probability p_2 after the first challenge message, when node u moves into the neighbourhood again later, many of its legitimate data packets may be dropped. However, this situation does not really happen. When node u moves into the neighbourhood, it will first resynchronise with node v by sending a challenge message, which node v will process immediately. If the challenge message is valid, node v will move u to the 'trust' state, thus no data packets will be dropped by mistake.

4.4 Further discussions

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

4.4.1 Interaction with routing protocols

LIP is independent of the (secure) routing protocols. It requires that a node use its one-time authentication keys to authenticate all its packets to its direct neighbours despite the type of transmission (e.g. unicast, multicast or broadcast) of each packet. In practice, it could take advantage of the deployed routing protocol to provide stronger security. For example, 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.

4.4.2 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 one-time authentication keys because every authentication key is only used once. The issue of providing a sufficient number of one-time keys has been addressed recently based on techniques such as Merkle-hash tree and multilevel key chains (Coppersmith and Jakobsson, 2002; Liu and Ning, 2003). 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 neighbour and knows one or some of its one-time authentication keys, node u should discard any future keys in its key chain and bootstrap a new key chain to its current neighbours other than v (e.g. GKMPAN (Zhu et al., 2004)). This completely prevents node v from impersonating u .

4.4.3 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, Schemes I and 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. Firstly, how can a node distinguish these two kinds of neighbours 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. Secondly, 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 neighbourhood 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 neighbours is also different. The performance of this hybrid scheme is studied in Section 6.

4.4.4 Potential applications

The per-hop per-packet authentication capability provided in LIP is not only critical for building any applications that are open to attackers, but also necessary for many civilian applications of ad hoc networks that do pricing or require cooperation stimulation and incentive (Buttyán and Hubaux, 2003; Luo et al., 2003; Zhong et al., 2002). To promote cooperation, a node must forward packets for other nodes to receive credits to enable the forwarding of its own packets. For example, in Buttyán and Hubaux (2003), a node is required to pass each packet to its security module. The security module maintains a counter, which is decreased when the node wants to send a packet as originator and increased when the node forwards a packet. The value of the counter must remain positive so that the node can send its own packets. Clearly, the secure module or the similar stimulation mechanisms can make use of our per-packet authentication protocol to prevent selfish nodes from getting free rides. Hence, our mechanisms might be utilised as an enforcing technique for the system.

Another potential application is for providing traceback services in mobile ad hoc network. IP traceback has been extensively studied as a way to identify the attack source that launches distributed DoS attacks (Savage et al., 2000; Snoren et al., 2001; Song and Perrig, 2001). The basic idea is to mark a packet probabilistically (Song and Perrig, 2001) or record a digest of a packet (Savage et al., 2000; Snoren et al., 2001) such that the information of the intermediate links or nodes is preserved and recoverable, thus tracing back to the source step-by-step. Here a fundamental security assumption in most of these schemes is that the intermediate routers are cooperative so that the reconstructed link or node information is trustworthy. While this assumption is generally believed to hold in the internet environment, in mobile ad hoc networks with malicious nodes the per-link information can no longer be assumed unless security mechanisms such as our schemes are employed to enforce per-link authenticity. Although traceback in mobile ad hoc networks is still an open problem, our mechanism at least provides a valid building block for the future research on this topic.

5 Security analysis

This section analyses the security of our schemes based on the security threat models in Hu et al. (2003). 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. As long as the underlying encryption algorithm and authentication algorithm are secure, an attacker cannot break the one-time authentication keys of the legitimate nodes.
- *Active I*: an active attacker attempts to inject malicious packets into the network although it has no cryptographic keys from the network controller. Since our protocol performs hop-by-hop authentication of every packet, without knowing a valid one-time authentication key, the attacker cannot inject its own packets into the network. Therefore, our schemes can prevent the ActiveI 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
 - The attacker can only replay the packets transmitted by a node u to the other nodes that possess node u 's one-time authentication keys.
 - If the nodes having node u 's one-time authentication key are node u 's current neighbours, they will drop the duplicated packets based on packet sequence numbers or one-time authentication key versions. For example, if an attacker replays node u 's ROUTE REQUEST packets to node u 's neighbours in DSR (Jonhson et al., 2001), 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.
 - If the nodes possessing some one-time authentication keys previously released by node u are not neighbours 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 broadcast messages.
 - If time synchronisation 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 ActiveI model. An ActiveX attack is not more severe than a single ActiveI attack for LIP except when multiple attackers collude to launch a *wormhole* attack (Hu et al., 2003). Scheme I cannot prevent the wormhole attack. In Scheme II, two neighbouring 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 neighbourhood 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/misbehaviour detection techniques (Marti et al., 2000; Zhang and Lee, 2000) 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. Both Schemes I and 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 neighbourhood information. Zhang and Lee (2000) and Marti et al. (2000) have studied the intrusion and misbehaviour 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.

6 Performance analysis

Since, we have discussed the capability of LIP in filtering unauthenticated data packets and other threat models in Section 5, 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? What is the impact of node mobility model? 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.

6.1 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 neighbours. 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.

6.2 Simulation methodology

This simulation utilises GloMoSim 2.02 (GloMoSim, 2001) and sets the default configuration as follows. There are 100 nodes distributed in a square environment space of 2000 m × 2000 m. Each node joins the network at a time uniformly distributed between the simulation time 0 and 5 sec and each simulation runs for 900 sec 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 (Jonhson et al., 2001) and AODV (Perkins et al., 2003) and the multicast routing protocol ODMRP (Lee et al., 2002). 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 0.1 to 1.0 sec and the durations of connections from 10 to 850 sec.

Table 1 summarises the default parameters used in our simulation unless otherwise mentioned.

6.3 Evaluation results

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

6.3.1 Computational overhead and latency

In LIP, a node normally verifies a received traffic packet by computing one HMAC and one hash, the time for which is less than 1 ms even for handheld PDAs (Brown et al., 2000) that have very constrained computational capability. On a AMD Opteron 1.7 GHz processor, a hash over 8 bytes can be computed in about 0.08 μ s (Dai, 2004). The other computational load would be the infrequent estimation of REDs, which is very fast to compute.

Table 1 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	-81dBm
Mobility model	Waypoint
RED ⁰	200 sec
Threshold time for waiting responses	2 sec
Random neighbour 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 μ s
HMAC key size (including a key id)	10 bytes

Consider the total processing delay of a packet at one hop (node). It includes the time spent in all the layers. The time spent in transport layer or application layer is application dependent. Let us just consider the time spent in the medium access control layer and network layer while ignoring the computational overhead, then the main processing time includes the random backoff and the jitter time in both the routing protocol and the protocol. In AODV, the average random backoff time is 250 ms and the average broadcast jitter time is 5 ms; in the 802.11 medium access control protocol, the average random backoff time is 25 ms. Thus, the delay caused by LIP is several-order smaller in comparison to the delay introduced in each hop.

6.3.2 The impact of node mobility

(a) *Control overhead*: Figure 5(b) 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. Firstly, with higher node mobility, RED is normally smaller. In other words, nodes have to synchronise with each other more frequently, which in turn increases the number of challenge and ACK messages. Secondly, with higher mobility, a node will encounter more nodes. Chances are many of them have never contacted before or their neighbourhood have expired. When a node receives a packet from its neighbour that is not in its trust list, it will automatically send a challenge to the neighbour and the neighbour will also respond with an ACK message. This round of challenge and ACK alone costs 58 bytes of control overhead. Finally, 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.

(b) *Traffic delivery ratio*: Figure 5 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 five 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 neighbour which failed to respond to a challenge. Since our scheme only makes proximate estimation of RED, it is possible that the RED of a neighbour node expires while it is still within the transmission range. In this case, if that neighbour 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 neighbour will the traffic packets from it be accepted and passed to the routing protocol. If the neighbour 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 neighbours know its existence until it broadcasts a message, which causes many neighbours to challenge it at the same time and hence more challenge packets will be lost.

(c) *Buffer size*: Figure 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. Firstly, the average and maximum buffer length grows with the node mobility. The reason is that a node encounters more neighbours when it moves at a higher speed. For each neighbour with an expired RED, the packets from it will be temporarily stored in the buffer until an ACK from it is received and verified. Secondly, 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.

6.3.3 Impact of node density

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

Control overhead: Figure 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.

Traffic delivery ratio: Figure 7(b) the traffic delivery ratios are above 99.85% for all three routing protocols when the number of nodes is 160 or less.

6.3.4 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.

Figure 5 The impact of node mobility on control overhead and traffic delivery ratio (a) control overhead and (b) traffic delivery ratio

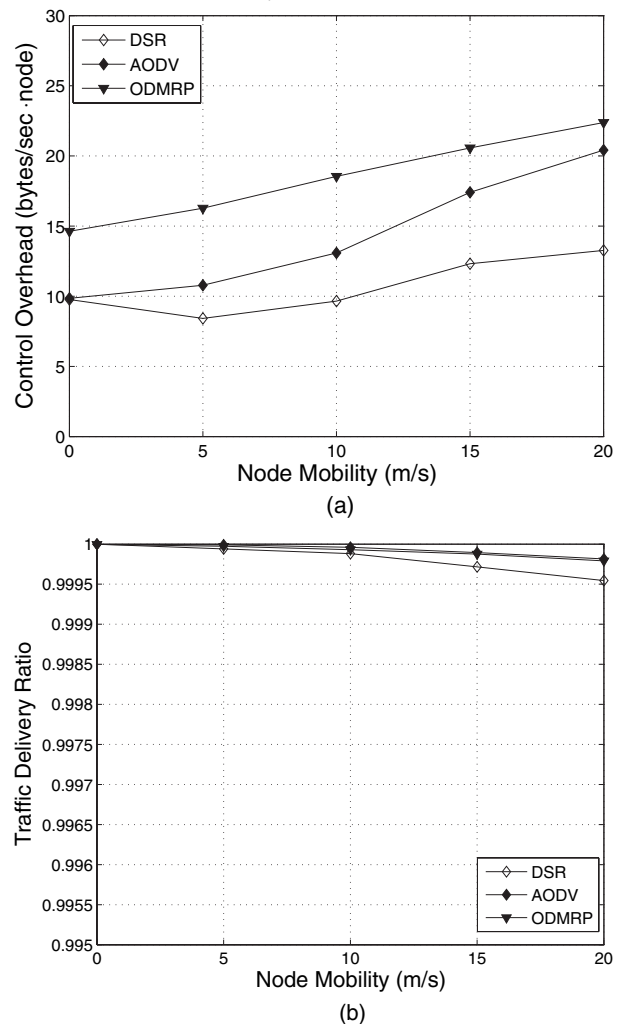


Figure 6 The impact of node mobility on the length of buffer

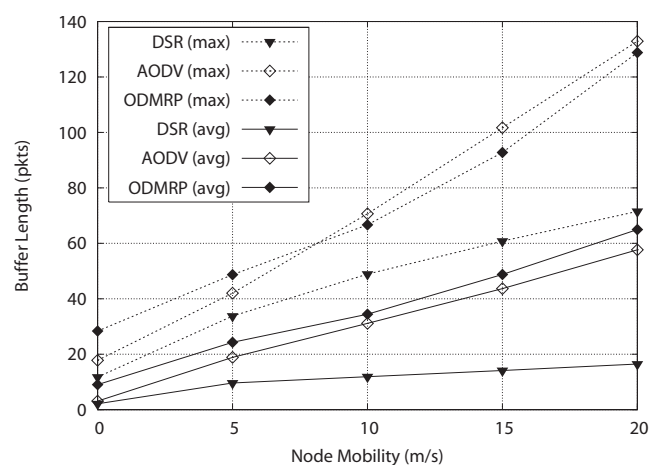
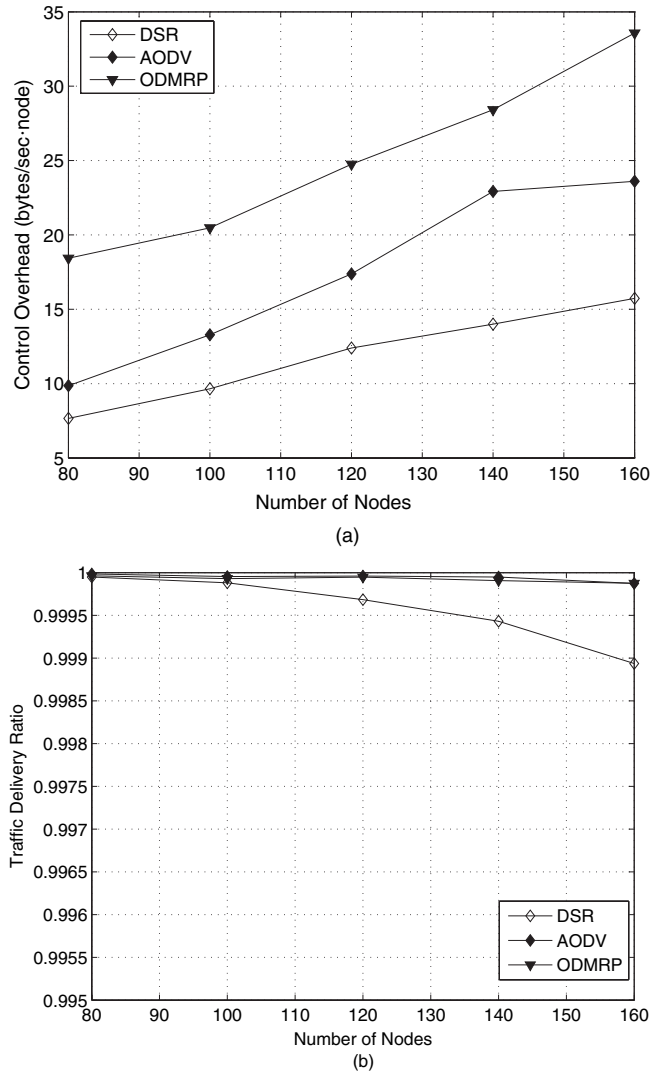


Figure 8 illustrates that with challenge probability $p_c = 0.1$, control overhead decreases with α (except slightly different in the case of DSR). The reason is that, instead of randomly challenging a neighbour, a location-aware node knows more

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. On the other hand, our simulation results show that the traffic delivery ratios are above 99.99% under all the α values.

Figure 7 The impact of node density on control overhead and traffic delivery ratio (node mobility 10 m/s)
(a) control overhead and (b) traffic delivery rate



6.3.5 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) (Hong et al., 1999) and Manhattan (Bai et al., 2003). We compare these two mobility models with the default random waypoint mobility model at a maximum speed of 10 m/s. We utilise BonnMotion (BonnMotion, 2004) to generate mobility scenarios. Table 2 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 neighbourhood among nodes is expected. This results in larger REDs. Figure 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 models, as shown in Figure 9(a) conform to the observations on the different REDs shown in Figure 9(b).

Figure 8 The impact of location-awareness on control overhead ($p_c = 10\%$, node speed 10 m/s)

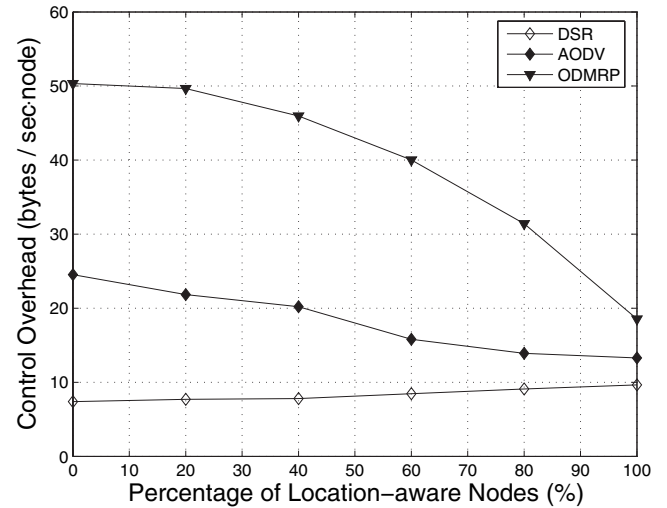


Table 2 The mobility model parameters

Parameters	Values
<i>RPGM</i>	
2 Average number of nodes per group	20
Group number	5
Group change probability	0
Maximum distance to group centre	50 m
Minimum speed	1 m/s
Maximum speed	10 m/s
Maximum pause	30 sec
<i>Manhattan</i>	
Minimum speed	1 m/s
Mean speed	10 m/s
Maximum pause	30 sec
Pause probability	0.001
Turn probability	0.2
Update distance	10 m
Number of blocks along x -dimension	10
Number of blocks along y -dimension	10

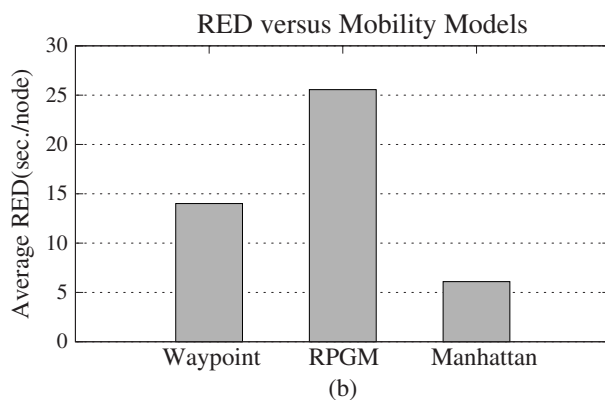
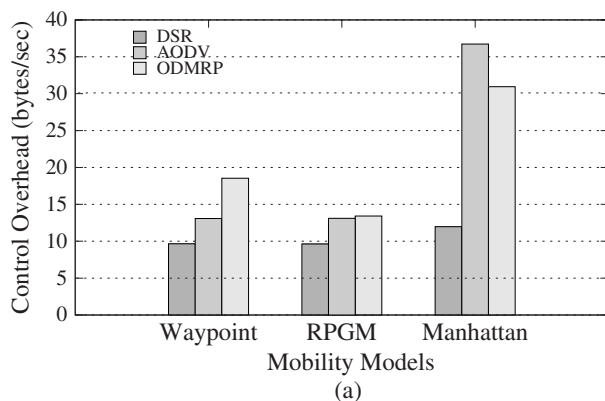
Overall, the above performance analysis shows that LIP is a lightweight protocol and it provides traffic delivery ratio close to 100% with different node speeds, network densities and mobility models.

7 Conclusion

We have presented LIP, a lightweight hop-by-hop authentication protocol for preventing unauthorised nodes from injecting spurious packets into ad hoc networks. The protocol is transparent to and independent of the network

routing protocols. The protocol relies solely on symmetric key operations and is based on a *localised* broadcast authentication technique whereby each packet is efficiently authenticated at each hop by the immediate neighbour(s) of the node transmitting the packet. In addition to being able to prevent resource consumption attacks by unauthorised nodes, the protocol can mitigate impersonation attacks by compromised nodes. We proposed three different schemes for achieving this goal. Our performance analysis showed that the protocol has low communication overhead and high traffic delivery ratio.

Figure 9 Control overhead and REDs under different mobility models: (a) control overhead and (b) REDs



Acknowledgement

This work was supported in part by National Science Foundation under contract IIS-0324835. Additionally supported in part by Army Research Office (W911 NF-05-1-0270) and the National Science Foundation (CNS-0524156).

References

- Aad, I., Hubaux, J. and Knightly, E. (2004) 'Denial of service resilience in ad hoc networks', *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking*, pp.202–215.
- Bai, F., Sadagopan, N. and Helmy, A. (2003) 'IMPORTANT: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks', *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, Vol. 2.
- Boneh, D. and Franklin, M. (2003) 'Identity-based encryption from the Weil pairing', *SIAM Journal of Computing*, Vol. 32, No. 3, pp.586–615.
- BonnMotion (2004) Available at: <http://web.informatik.uni-bonn.de/iv/mitarbeiter/dewaal/bonnmotion/>.
- Brown, M., Cheung, D., Hankerson, D., Hernandez, J., Kirkup, M. and Menezes, A. (2000) 'PGP in constrained wireless devices', *Ninth USENIX Security Symposium*, pp.247–261.
- Buttyán, L. and Hubaux, J. (2003) 'Stimulating cooperation in self-organizing mobile ad hoc networks', *Mobile Networks and Applications*, Vol. 8, No. 5, pp.579–592.
- Cardenas, A., Radosavac, S. and Baras, J. (2004) 'Detection and prevention of mac layer misbehavior for ad hoc networks', *SASN*.
- Coppersmith, D. and Jakobsson, M. (2002) 'Almost optimal hash sequence traversal', *Proceedings of the Fourth Conference on Financial Cryptography (FC'02)*.
- Dai, W. (2004) Available at: <http://www.eskimo.com/~weidai/amd64-benchmarks.html>.
- GloMoSim (2001) Available at: <http://pcl.cs.ucla.edu/projects/gloimosim/>.
- Hong, X., Gerla, M., Pei, G. and Chiang, C. (1999) 'A group mobility model for ad hoc wireless networks', *Proceedings of the Second ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile systems*, pp.53–60.
- Hu, Y., Johnson, D. and Perrig, A. (2002a) 'SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks', *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, pp.3–13.
- Hu, Y., Perrig, A. and Johnson, D. (2003) 'Packet leases: a defense against wormhole attacks in wireless ad hoc networks', *Proceedings of INFOCOM*.
- Hu, Y.-C., Perrig, A. and Johnson, D.B. (2002b) 'Ariadne: a secure on-demand routing protocol for ad hoc networks', *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*.
- Johnson, D., Maltz, D., Hu, Y. and Jetcheva, J. (2001) *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks*.
- Lampert, L. (1981) 'Password authentication with insecure communication', *Communications of the ACM*, Vol. 24, No. 11, pp.770–772.
- Lee, S.J., Su, W. and Gerla, M. (2002) 'On-demand multicast routing protocol in multihop wireless mobile networks', *Mobile Network Application*, Vol. 7, No. 6, pp.441–453.
- Liu, D. and Ning, P. (2003) 'Establishing pairwise keys in distributed sensor networks', *CCS '03: Proceedings of the Tenth ACM Conference on Computer and communications security*, New York, NY: ACM Press, pp.52–61.
- Luo, H., Kong, J., Zerfos, P., Lu, S. and Zhang, L. (2004) 'URSA: ubiquitous and robust access control for mobile ad hoc networks', *Networking, IEEE/ACM Transactions on*, Vol. 12, No. 6, pp.1049–1063.
- Luo, H., Ramjee, R., Sinha, P., Li, L. and Lu, S. (2003) *Ucan: A Unified Cellular and Ad-Hoc Network Architecture*.
- Marti, S., Giulì, T.J., Lai, K. and Baker, M. (2000) 'Mitigating routing misbehavior in mobile ad hoc networks', *MobiCom '00: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, New York, NY: ACM Press, pp.255–265.

- Papadimitratos, P. and Haas, Z. (2002) 'Secure routing for mobile ad hoc networks', *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*.
- Perkins, C., Belding-Royer, E. and Das, S. (2003) 'RFC3561: ad hoc on-demand distance vector (AODV) routing', *Internet RFCs*.
- Perkins, C. and Bhagwat, P. (1994) 'Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers', *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pp.234–244.
- Perrig, A., Canetti, R., Song, D. and Tygar, J. (2001) 'Efficient and secure source authentication for multicast', *Network and Distributed System Security Symposium, NDSS*, Vol. 1, pp.35–46.
- Sanzgiri, K., Dahill, B., Levine, B., Shields, C. and Belding-Royer, E. (2002) 'A secure routing protocol for ad hoc networks', *Network Protocols, 2002. Proceedings of Tenth IEEE International Conference on*, pp.78–87.
- Savage, S., Wetherall, D., Karlin, A.R. and Anderson, T. (2000) 'Practical network support for IP traceback', *SIGCOMM*, pp.295–306.
- Snoren, A., Partridge, C., Sanchez, L., Jones, C., Tchakountio, F., Kent, S. and Strayer, W. (2001) 'Hash-based IP traceback', *Proceedings of ACM SIGCOMM'2001*.
- Song, D. and Perrig, A. (2001) 'Advanced and authenticated marking schemes for IP traceback', *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 2, pp.878–886.
- Xu, W., Wood, T., Trappe, W. and Zhang, Y. (2004) 'Channel surfing and spatial retreats: defenses against wireless denial of service', *Proceedings of the 2004 ACM Workshop on Wireless Security*, pp.80–89.
- Yi, S., Naldurg, P. and Kravets, R. (2001) 'Security-aware ad hoc routing for wireless networks', *Proceedings of the Second ACM international symposium on Mobile Ad hoc Networking and Computing*, pp.299–302.
- Zhang, Y. and Lee, W. (2000) 'Intrusion detection in wireless ad-hoc networks', *ACM MobiCom'2000*, pp.275–283.
- Zhong, S., Yang, Y. and Chen, J. (2002) *Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks*.
- Zhou, L. and Haas, Z.J. (1999) 'Securing ad hoc networks', *IEEE Network*, Vol. 13, No. 6, pp.24–30.
- Zhu, S., Setia, S., Xu, S. and Jajodia, S. (2004) 'GKMPAN: an efficient group rekeying scheme for secure multicast in ad-hoc networks', *MobiQuitous*, IEEE Computer Society, pp.42–51.
- Zhu, S., Xu, S., Setia, S. and Jajodia, S. (2003) *Lhap: A Lightweight Hop-By-Hop Authentication Protocol For Ad-Hoc Networks*.

Notes

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

²To avoid confusion, we do not use MAC as the abbreviation for medium access control.

³The term 'id-based scheme' does not refer to the id-based encryption scheme, for example, the Boneh–Franklin scheme (Boneh and Franklin, 2003).

⁴Note that LIP only thwarts the impersonation attack launched by an insider node; the *general* insider attacks pose more severe threats, and have to be dealt with a more sophisticated scheme.