LHAP: A Lightweight Network Access Control Protocol for Ad-Hoc Networks

Sencun Zhu [a,*], Shouhuai Xu [b], Sanjeev Setia [c], Sushil Jajodia [c]

[a] *Department of Computer Science and Engineering and School of Information Science and Technology, The Pennsylvania State University, University Park, PA 16802*

[b] *Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249*

[c] *Center for Secure Information Systems, George Mason University, Fairfax, VA 22030*

Abstract

Most ad hoc networks do not implement any *network access control*, leaving these networks vulnerable to *resource consumption* attacks where a malicious node injects packets into the network with the goal of depleting the resources of the nodes relaying the packets. To thwart or prevent such attacks, it is necessary to employ authentication mechanisms to ensure that only authorized nodes can inject traffic into the network. We propose LHAP, a hop-by-hop authentication protocol for ad hoc networks. LHAP resides in between the network layer and the data link layer, thus providing a layer of protection that can prevent or thwart many attacks from happening, including outsider attacks and insider impersonation attacks. Our detailed performance evaluation shows that LHAP incurs small performance overhead and it also allows a tradeoff between security and performance.

Key words: Network Access Control, Hop-by-hop Authentication, One-way Key Chain, Resource Consumption Attacks

* Corresponding author. (Tel) 1-814-865-0995, (Fax) 1-814-865-3176.
Email addresses: szhu@cse.psu.edu (Sencun Zhu), shxu@cs.utsa.edu (Shouhuai Xu), setia@gmu.edu (Sanjeev Setia), jajodia@gmu.edu (Sushil Jajodia).

1 Introduction

In ad hoc wireless networks, no base stations exist and every mobile node acts as both a router and a host. Nodes in an ad hoc network can communicate with each other at any time, subject to connectivity limitations. Currently, most ad hoc networks do not have any provisions for restricting or regulating the traffic that flows through a node, i.e., they do not implement any *network access control*. This leaves these networks vulnerable to *resource consumption* attacks where a malicious node injects packets into the network with the goal of depleting the resources of the nodes relaying the packets. For example, since mobile hosts are usually battery powered, they are susceptible to battery exhaustion attacks [36].

A resource consumption 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 [40], the packet injection attack using broadcast messages may be more favorable to an attacker due to its network-wide harm.

Most of the routing protocols [19,29,28,34,39] 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 authentic. 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 prevent such attacks, recently researchers have proposed several security extensions [8,13,14,41] to the existing routing protocols, which include mechanisms for authenticating the routing control packets in the network.

To the best of our knowledge, however, none of the proposed secure routing protocols include any provisions for authenticating data packets. One reason for this is that data packets are typically unicast; therefore, simply restricting the number of hops a packet can take will limit the effectiveness of a resource consumption attack. However, this argument does not apply to multicast data packets. As such, we believe that it is important to provide network access control for both data and control packets.

To provide full network access control, every node has to verify the authenticity of every packet it receives before forwarding the packet. The simplest solution 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 transmits 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 card.

A well known technique for providing strong source authentication is to attach a digital signature to a packet. However, signing every packet can be prohibitively expensive since the computational capacity and battery power of mobile nodes are quite constrained. Therefore, the challenge is to design a lightweight authentication protocol for the more vulnerable yet more resource-constrained environment of an ad hoc network.

In this paper, we present LHAP, a scalable and efficient authentication protocol for ad hoc networks. To prevent resource consumption attacks, LHAP implements lightweight hop-by-hop authentication, i.e., intermediate nodes authenticate all the packets they receive before forwarding them. Using LHAP, a node joining an ad hoc network only needs to perform some inexpensive authentication operations to bootstrap a trust relationship with its neighbors. It then switches to a very lightweight protocol for subsequent traffic authentications.

LHAP is transparent to and independent of the network routing protocols. It resides between the data link layer and the network layer, providing a layer of protection that can prevent or thwart many attacks, including attacks on ad hoc routing protocols made possible by the lack of support for packet authentication in these protocols. LHAP can be seamlessly integrated with secure routing protocols to provide a more secure ad hoc network. Our detailed performance evaluation shows that LHAP is a very lightweight protocol. Further, LHAP allows a tradeoff between security and performance, and can therefore be configured to provide the desired levels of security and performance for a specific application or network.

The rest of this paper is organized as follows. In Section 3, we provide some background on one way hash chains and TESLA that are needed for under-

[1] A network-wide key is used in the WEP algorithm in 802.11 standard [18] for confidentiality.

standing the techniques used in our protocol. We present the details of the LHAP protocol in Section 4, and analyze its security in Section 5. In Section 6, we analyze the performance of our protocol, and show several possible optimizations for its real deployment in Section 7. Finally, we discuss related work in Section 8, and present our conclusions in Section 9.

2 Assumptions and Design Goals

This section describes our network, node, and security assumptions as well as our design goals.

2.1 Assumptions

Network and Node Assumptions First, we assume that the wireless network links are bidirectional, i.e., if node A can hear node B, node B can also hear node A. This is generally true when the nodes use omnidirectional antennas and have similar power levels. Second, we assume loose time synchronization among all the nodes in a network such that the difference between any two nodes does not exceed a certain value. This is because we use TESLA for providing strong broadcast source authentication and TESLA requires loose time synchronization.

Security Assumptions We assume that each node possesses a public key certificate issued by a trusted certificate authority (CA) as well as the authenticated public key of the CA. The distribution of certificates and keys can be done in any reliable way. LHAP relies on these public keys to bootstrap trust among the nodes in an ad hoc network. We believe this is a reasonable assumption if all the nodes in the network belong to the same autonomous system or administrative unit (e.g., a university). An advantage of using public key certificates is scalability with respect to storage. Unlike a symmetric key based scheme where every node needs to be preloaded with $N-1$ pairwise keys shared with $N-1$ other nodes for a network size of N, in a public key based scheme, a node only needs to remember its own certificate and the public key of the CA, irrespective of the network size N. However, we will not discuss the issue of certificate management and revocation; several previous works [6,17] have studied this issue.

Attack Models We consider *resource consumption attacks* in which an attacker injects a huge number of spurious 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 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 may launches resource consumption attacks because it has been compromised or it is malicious; we do not distinguish between these two cases. The attacker may impersonate another node by inserting that node's id in the source id field of the packets it is injecting. 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 are flooded in the entire network. To achieve its goals, an attacker may eavesdrop on other nodes, reorder or drop packets, replay packets, or modify overheard packets and re-inject them into network.

Note that we do not address attacks against the physical layer and the media access control layer. Techniques such as spread spectrum [33], frequency hopping, or spatial retreat [40] can be employed to prevent physical jamming attacks if necessary. Cardenas et al [5] and Gupta et al [10] have studied techniques for detecting and preventing media access control layer attacks. Note that an attacker does not necessarily favor these lower layer attacks over the resource consumption attacks addressed by LHAP. A lower layer attack may jam the channel or interrupt the communications in a small area, whereas in a resource consumption attack an attacker may inject one spurious broadcast or multicast packet which is replicated throughout the network, thus consuming the energy of all the nodes in the network.

2.2 Design Goals

Generally, there are three approaches for dealing with security attacks: *prevention*, *detection*, and *reaction*. Prevention aims at thwarting security breaches from occurring in the first place, whereas the other two approaches are necessary when prevention fails. Ideally, a defense system integrates all these approaches, but the cost of such a system may be too expensive for the low-end nodes under consideration. As such, in defending against resource consumption attacks, we mainly take the prevention approach.

The main objective of our protocol is to prevent outsider nodes from launching resource consumption attacks. A secondary goal of our protocol is to provide a mechanism that helps identify insider attackers, thus deterring insider attacks. We note, however, that preventing insider attacks is a very difficult problem that is outside the scope of this paper.

To prevent resource consumption attacks, 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 resource consumption attacks. Although security does not come for free, the protocol should incur as little overhead as possible. More specifically, computation-intensive operations such as those based on public key techniques should be minimized, if they cannot be avoided altogether. Since packet transmissions result in the largest contribution to the energy expenditure of a wireless node, the protocol should not incur large 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; for example, requiring that a node shares a pairwise key with every other node in the network.

Immediate Authentication The protocol should provide immediate authentication, i.e., there should be no delay in authenticating a packet that is received; 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 can be turned on or turned off without affecting the functionalities of other protocols such as routing protocols or application layer protocols.

Independence The protocol should be independent of the routing protocol. It is possible to design a specific and more efficient network access control protocol that works with a specific routing protocol; however, this would require the design of a new customized protocol for every routing protocol, which is clearly undesirable.

LHAP achieves all these design goals. Note that our goal is *not* to design another secure routing protocol. Instead, our protocol and a secure routing protocol are complementary to each other, and they could be employed at the same time to make a network stronger against various security attacks.

3 Background

LHAP uses one-way key chains [22] for traffic authentication and TESLA [31,30] for bootstrapping trust. We briefly review these techniques in this section.

3.1 One-way Hash Chain

Since its first use by Lamport [22] for password authentication, one-way key chain has been widely used in cryptography. A one-way key chain is a chain of keys generated through repeatedly applying a one-way hash function on a random number. For instance, if a node wants to generate a key chain of size N, it first randomly chooses a key, say $K(N)$, then computes $K(N-1) = F(K(N))$, $K(N-2) = F(K(N-1))$,..., repeatedly until it gets $K(0) = F(K(1))$. Here F is a cryptographically secure hash function with the one-way property, that is, if $y = F(x)$, it is computationally infeasible to compute x given y and F. As such, given $K(i)$, any others can compute $K(i-1)$, $K(i-2)$,...,$K(0)$ independently, but they cannot compute any keys in $K(i+1), K(i+2), ..., K(N)$.

To use a one-way key chain for authentication, a sender first signs the last value (called the *commitment*) in the chain, i.e., $K(0)$ above, with its private key so that anybody who knows its public key can verify the signature and hence the authenticity of $K(0)$. Then the sender discloses keys in the chain in an order reverse to that of its generation. A receiver can verify $K(j)$ by checking if $K(j-1) = F(K(j))$, if it has $K(j-1)$. Furthermore, if a receiver did not receive $K(j-1)$ and the last key it has verified is $K(i)$, where $i < j-1$, it can still verify $K(j)$ by checking if $F^{j-i}(K(j)) = K(i)$. This property is very useful because it allows the authentication scheme to work in the event of packet losses.

3.2 TESLA

TESLA [31] is a broadcast authentication scheme that uses one-way key chain along with message authentication code (MAC). In the basic scheme of TESLA, a sender uses a key K from its key chain to compute a MAC over packet $P(i)$, and then attaches the MAC to $P(i)$. A receiver cannot verify packet $P(i)$ immediately. Indeed, the key K is disclosed in the next packet $P(i+1)$, which allows the receivers to verify the authenticity of K and hence the MAC of $P(i)$. If both K and the MAC are correct, and if the packet $P(i)$ is guaranteed to be received before $P(i+1)$ was sent, the receivers will accept $P(i)$.

From this description, we can see clearly that the key security issue in TESLA is a receiver's ability to determine the sending time of each packet. This is called security condition in TESLA. TESLA solves this issue through periodical key disclosure and loose time synchronization. Specifically, time is divided into many equal length intervals. In each time interval the sender discloses

one MAC key from its key chain. For example, if the start time is s, the time interval is T, the sender can publish $K(i)$ at time $s + i * T$. The receivers are required to be loosely synchronized with the sender.

One drawback of the original TESLA protocol is that a receiver cannot authenticate a packet immediately. Its variant [30] enables immediate authentication in the receiver side, but it requires packet buffering on the sender side.

4 A Lightweight Hop-by-hop Authentication Protocol (LHAP)

The resource constraints of wireless nodes effectively preclude the use of digital signatures based on public key cryptography for signing each and every packet. Therefore, we seek solutions that use symmetric key techniques for packet authentication. As described in Section 3, TESLA is an efficient source authentication protocol that provides strong security guarantees. However, it does not provide immediate packet authentication to the nodes that relay packets.

The simplest approach to achieve our design goal is to employ a network-wide key shared by all nodes. Every node uses this shared key to compute MACs [2] on the packets it sends. Nodes receiving these packets can use the same key to verify the MACs and hence the authenticity of the packets. Despite its simplicity, this scheme has several disadvantages, as discussed in Section 1.

An approach that avoids the security issues in a network-wide key based scheme is one based on pairwise key sharing. Specifically, a node attaches to every packet multiple MACs, one for each of its immediate neighbors. Each MAC is computed from the pairwise key shared with a neighbor node. Since the number of MACs attached to each packet is equal to the number of immediate neighbors of the transmitting node, this approach becomes very inefficient for networks with high node density.

Therefore, as a tradeoff between the performance and the security of these two approaches, our network access control protocol is based on a *localized* broadcast authentication mechanism. In the following section, we will discuss the protocol and its operation in detail.

[2] In this paper, we use MAC to denote message authentication code, not medium access control

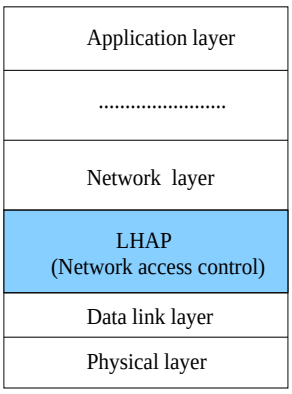

Fig. 1. The protocol stack with LHAP between the network layer and the data link layer

4.1 LHAP Overview

4.1.1 The Architecture

To achieve transparency and independence, we design LHAP as an independent layer, as depicted in Figure 1. LHAP resides between the data link layer and the network layer. For every traffic packet received from the network layer, LHAP adds its own header, which includes its node id, a packet type field indicating it is a traffic packet, and an authentication tag. Both routing control packets and application data packets are referred to as *traffic* packet because LHAP does not distinguish between them. After that, LHAP passes the packet to the data link layer. LHAP also generates its own control packets for establishing and maintaining trust relationship with neighboring nodes.

For a received traffic packet, LHAP verifies its authenticity based on the authentication tag in the LHAP header of the packet. If the packet is valid, LHAP removes the LHAP header and then passes the packet to the network layer; otherwise, it discards the packet. For a received LHAP control packet, LHAP processes the packet and then discards it. That is, a LHAP control packet is not passed to the network layer. This design has the advantage that LHAP can be turned on or off without affecting the operations of the other layers.

LHAP provides a protection mechanism that can prevent or thwart many attacks from happening. In LHAP, every node in the network verifies every packet it receives from a neighbor before forwarding it (if it is not the destination node). Packets from unauthorized nodes are dropped immediately, thus preventing them from propagating through the network. A packet that needs multiple hops before reaching its destination is thus authenticated by each node on its path. We refer to this as *hop-by-hop* authentication.

LHAP's efficiency gains over traditional authentication protocols are derived from two techniques: (i) lightweight packet authentication, and (ii) lightweight trust management. Since all packets are authenticated on every hop on their paths, it is important that the packet authentication technique used by LHAP be as inexpensive as possible. LHAP employs a packet authentication technique based on the use of one-way hash chains. Secondly, LHAP uses TESLA to reduce the number of public key operations for bootstrapping trust between nodes, and also use TESLA for maintaining the trust between nodes. We briefly discuss both these issues below before discussing the operations of the protocol in more detail.

Lightweight Traffic Authentication In LHAP, each node generates a one-way key chain that is used for authenticating traffic to its immediate neighbors. We use the term TRAFFIC key to refer to the keys in a one-way key chain. Every neighbor of a node obtains an authenticated TRAFFIC key in this node's TRAFFIC key chain when it establishes trust relationship with the node for the first time. When transmitting a packet, a node appends a new TRAFFIC key to the packet. All the neighboring nodes receiving this packet can verify the authenticity of this packet by verifying the attached TRAFFIC key.

Trust Management Nodes can bootstrap their trust relationship, i.e., exchange authentic TRAFFIC keys, by using a public key based technique. A simple approach that can be used is one in which a node signs its most recently released TRAFFIC key and sends it to every new neighbor. However, this approach does not scale well for resource-constrained mobile nodes which may encounter a large number of nodes during their lifetime. To address this issue, LHAP uses TESLA to reduce the number of signature operations to one. Specifically, in LHAP every node only uses digital signatures to bootstrap a TESLA key chain, and TESLA keys are then used to provide authenticated TRAFFIC keys.

To maintain its trust relationships, a node periodically announces its most recently released TRAFFIC keys, authenticated by its TESLA keys. Its neighbors will drop any received packets that are authenticated by an old TRAFFIC key. This thwarts replay attacks by both outsider and insider nodes. We call this periodic message a KEYUPDATE message. If a node does not receive a valid KEYUPDATE message from a neighbor within a certain number of TESLA intervals, it infers that that neighbor is most likely out of its transmission range. Therefore, it temporarily terminates its trust with this neighbor. When the two nodes come within each other's transmission range again in the future, they reestablish their trust relationship by exchanging KEY UPDATE messages.

4.2 Notation

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

- A, B are principals, the identities of mobile nodes.
- $Cert_A$ is node A's public-key certificate issued by a trusted CA.
- $Sign_A(M)$ denotes the digital signature of message M, signed with node A's private key. Note that M is also included in the signature.
- $M1|M2$ denotes the concatenation of message $M1$ and $M2$.
- $MAC(K, M)$ denotes the computation of MAC over message M with key K.
- $K_A^T(i)$ denotes node A's i'th key in its TESLA key chain, while $K_A^F(i)$ denotes its i'th key in its TRAFFIC key chain.

4.3 LHAP in Detail

LHAP consists of two security building blocks: *traffic authentication* and *trust management*.

4.3.1 Traffic Authentication

Since LHAP authenticates all traffic packets, we must use a computationally inexpensive technique that can provide immediate authentication. Like TESLA, the traffic authentication technique used by LHAP is based upon the use of one-way key chains. Unlike TESLA, however, our authentication technique does not use periodic and delayed key disclosure. Delayed authentication (as in TESLA) is not appropriate for LHAP since a packet would be delayed at each node in the path from the source to the destination. For example, a traffic packet sent to another node ten hops away will take at least ten seconds if we use TESLA for authentication and the key disclosure period in TESLA is one second. Moreover, each node has to buffer the received traffic packets until they are verifiable, this may require a large memory space at every node. We note that the variant of TESLA [30] that switches the buffer overhead to senders cannot be employed to address this problem because every forwarding node in an ad hoc network is both a sender and a receiver.

Every node uses TRAFFIC keys for authenticating the traffic packets originating from itself or received from its neighbors. Consider a node A that wants to broadcast a message M. Let $K_A^F(i)$ be its next TRAFFIC key. It sends the

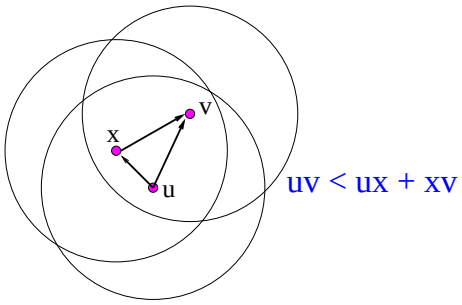

Fig. 2. Triangular Inequality

following packet:

$$A \longrightarrow * : M, K_A^F(i). \tag{1}$$

Every receiving node verifies the authenticity of this packet by verifying the TRAFFIC key $K_A^F(i)$, based on the most recent TRAFFIC key, $K_A^F(j), j < i$, that it received from node A. It then replaces $K_A^F(j)$ with $K_A^F(i)$ in its record.

In LHAP, a node only authenticates traffic packets to its immediate neighbors; thus it is very difficult for an attacker to launch replay attacks. This is due to the *triangular inequality* that applies to the distances of the involved nodes. When a node sends a packet, a neighbor will normally receive the packet directly before it receives a copy forwarded by a third node. Fig. 2 illustrates this property. When node u sends a packet, node v will receive the packet before it receives a forwarded copy from node x because $|uv| < |ux| + |xv|$. We will discuss several security issues related to this property in Section 5.

Using TRAFFIC keys for traffic authentication has several performance advantages. First, it enables immediate verification of traffic packets. Second, unlike TESLA keys, TRAFFIC keys are not disclosed periodically. Disclosing keys periodically wastes TRAFFIC keys when a node has no packets to transmit. Indeed, in LHAP the rate at which a node consumes its TRAFFIC keys is determined by the actual traffic rate. Third, it only requires computing a hash over a key of a small fixed size (e.g., 8 bytes); thus it is more computationally efficient than computing HMAC over the entire message. As a tradeoff between security and performance, on the other hand, our scheme does not provide the same level of security as TESLA.

4.3.2 Trust Management

Trust management includes *trust bootstrapping*, *trust maintenance* and *trust termination*.

Trust Bootstrapping When a node joins an ad hoc network, it first com-

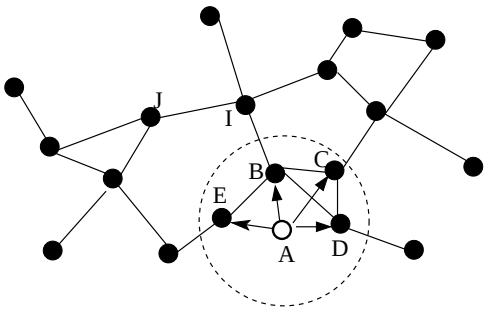

Fig. 3. A scenario when node A joins the ad hoc network. B, C, D, E are its current neighbors.

putes two key chains: a TRAFFIC key chain and a TESLA key chain. Then it signs the commitments of these key chains, and broadcasts them to its neighbors. Fig. 3 shows a scenario where node A is joining a network where its neighbors are B,C,D, and E. A broadcasts a JOIN message to its neighbors:

$$Cert_A, Sign_A\{A|K_A^T(0)|K_A^F(0)|T_A^T(0)|T_A^F(0)\}, \tag{2}$$

where $T_A^T(0)$ and $T_A^F(0)$ are the starting times for its TESLA and TRAFFIC key chains, respectively. Each neighbor first verifies node A's certificate using the CA's public key, then uses node A's public key in the certificate to verify the signature in the message. It records the commitments of node A's key chains in its TRUST table and their starting times if all the verifications succeed. To authenticate itself to node A, each of its neighbors will send an ACK message to A presenting their credentials. For example, node B sends the following ACK message to A:

$$\begin{aligned} &Cert_B, Sign_B\{B|K_B^T(0)|K_B^F(0)|T_B^T(0)|T_B^F(0)\}, \\ &K_B^F(j), K_B^T(i-1), MAC(K_B^T(i), K_B^F(j)). \end{aligned} \tag{3}$$

Note that here node B does not compute a new digital signature because the included signature was generated when node B first joined the network. $K_B^F(j)$ and $K_B^T(i-1)$ are node B's most recently released TRAFFIC key and TESLA key, respectively, and $K_B^T(i)$ is node B's next TESLA key to be released.

Upon receiving this message, node A first verifies the signature in the same way as its neighbors did earlier to verify its JOIN message. Then it records node B's key chain commitments and their starting times in its TRUST table. Based on this information, node A can verify $K_B^F(j)$ and $K_B^T(i-1)$ using a hash function. An attacker cannot forge arbitrary keys to deceive node A, although node A's computational resources will be expended while verifying false keys. In Section 7 we discuss a Merkle hash tree based scheme [26] that speeds up the verification process, thus mitigating this attack. Also, an attacker gains no security privileges by replaying any older keys that node B had previously

released.

The MAC in the message certifies node B's latest TRAFFIC key so that node A will not be deceived by forged packets that contain older TRAFFIC keys. However, node A cannot verify the MAC immediately because $K_B^T(i)$ is not known yet. After node B discloses $K_B^T(i)$ (on average half a TESLA interval later), node A can verify the MAC and then updates $K_B^T(0)$ and $K_B^F(0)$ to $K_B^T(i)$ and $K_B^F(j)$, respectively. After that, node A starts to forward valid traffic packets from node B.

Trust Maintenance Periodically, each node broadcasts a KEYUPDATE message to its immediate neighbors, which contains its most recently disclosed TRAFFIC key. The KEYUPDATE message is authenticated with the next TESLA key in its key chain. As an example, the KEYUPDATE message node A sends is

$$A \rightarrow * : K_A^T(i-1), K_A^F(j), MAC(K_A^T(i), K_A^F(j)). \tag{4}$$

We can see that this message is in the same format as an ACK message except that it does not contain a certificate and a signature. Therefore, the motivation for constructing this message is clear.

In its TRUST table, a node has one or more $(from, to)$ pairs for every node that it has established a trust relationship with, which record the time periods during which they were neighbors. Every node reports its node encounter information to a central authority after it leaves the ad hoc network. If multiple nodes roaming in different locations of the network assume the same identity (of a compromised node), the trusted server may be able to identify this node after analyzing the collected encounter information. When a node receives a KEYUPDATE message from a neighbor, it sets the latest to field to the current time if it has previously received a KEYUPDATE message from that neighbor one TESLA interval ago; otherwise it adds a new $(from, to)$ pair with both $from$ and to being the current time.

Trust Termination In LHAP, there are two scenarios under which the trust relationship between nodes will be terminated. First, when a compromised node is detected and announced, all the nodes will terminate their trust relationship with that node permanently. Second, if a node does not receive a valid KEYUPDATE message from a neighbor for more than one TESLA interval, it will terminate its trust relationship with this neighbor temporarily. As a result, it will not forward any packets for this neighbor until it receives a most recent KEYUPDATE message from this neighbor. The motivation for this approach is to ensure that a node knows the most recent TRAFFIC key of a neighbor in order to detect forged packets that reuse older TRAFFIC keys.

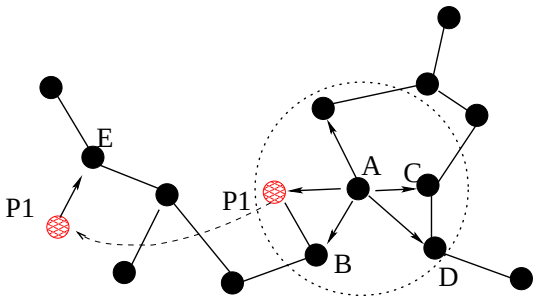

Fig. 4. Single outsider attack. $P1$ is a malicious node. The dashed line indicates node $P1$'s movement.

5 Security Analysis

In this section, we discuss several attacks against LHAP. These attacks are against traffic key chains, not against TESLA key chains, because TESLA [30,31] has been shown to be secure given loose time synchronization in the network.

5.1 Outsider Attacks

Outsider attacks are attacks launched by nodes that do not possess a valid certificate. We discuss two types of outsider attacks.

5.1.1 Single Outsider Attack

Fig. 3 showed a situation where node E received node A's JOIN message when node A joined the network. From the JOIN message, node E obtained the commitments of node A's key chains. Fig. 4 depicts a scenario where node E has moved out of node A's transmission range for longer than one TESLA interval. During this time period, node A has disclosed several TESLA keys and TRAFFIC keys. Assume that an outside attacker, node P_1, eavesdropped on node A's transmissions and recorded these keys. It then moves to the neighborhood of node E and sends spurious packets to node E while impersonating node A. For instance, suppose node A has broadcast a packet with content M and TRAFFIC key $K_A^F(i)$. Node P_1 changes the content M to M' and sends it to node E. Node E might not be able to detect this attack because it has not seen this TRAFFIC key yet.

The *trust termination* phase in LHAP was designed to thwart this attack. Since node E has not heard from node A for longer than one TESLA interval, it will not forward any traffic packets for node A until it receives a valid KEYUPDATE message. Since TESLA keys are disclosed periodically, node E knows which TESLA keys node A has released. Therefore, node P_1 cannot reuse the

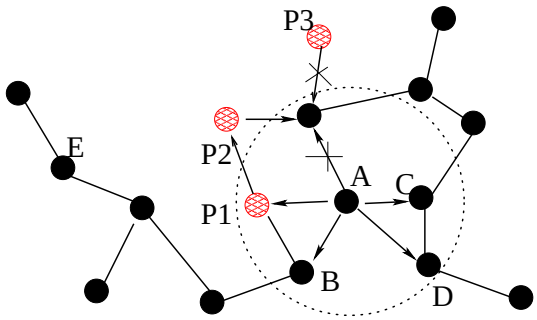

Fig. 5. Hidden terminal attacks. $P1$, $P2$,$P3$ are malicious nodes.

TESLA keys that node A has disclosed to forge a KEYUPDATE message. Node P_1 cannot forge a valid KEYUPDATE message using any TESLA keys that node A has not released yet due to the one-way property of a hash function. As such, we can see that in the worst case P_1 can impersonate node A for one TESLA interval, which can happen when node E moves out of node A's transmission range just after it receives a KEYUPDATE message from A. The number of forged packets is upper bounded by the number of packets node A has transmitted during this TESLA interval, because an attacker cannot derive the TRAFFIC keys that have not been disclosed by node A yet.

5.1.2 Collaborative Outsider Attack

A *collaborative outsider attack* is launched by multiple colluding outsider nodes. We discuss several such attacks below.

Hidden Terminal Attacks Hidden-terminal problem [37] is a unique problem in wireless networks. IEEE 802.11 solves the problem using CSMA/CD with ACKs and optional RTS/CTS control packets. For this scheme to work, however, it is assumed that contending nodes cooperate. Figure 5 shows an attack that tries to disrupt this cooperation, which we call a *hidden-terminal attack*. Note that this attack also applies to the triangular inequality property used in LHAP.

Suppose that node A is broadcasting a traffic packet that includes a TRAFFIC key $K_A^F(j)$ for packet authentication. To attack node E, a malicious node P_3 transmits a packet to node E at the same time, which causes node E to drop both packets. Next node P_1 sends $K_A^F(j)$ to node P_2. Now node P_2 can send an erroneous packet, which includes $K_A^F(j)$, to node E to impersonate node A. If this packet arrives at E before E receives a retransmission from node A, node E will accept the forged packet while dropping the retransmitted authentic packet from node A.

It is not easy for this attack to succeed. Since the packet retransmission interval is very small (tens of microseconds), the attackers have to cause continuous

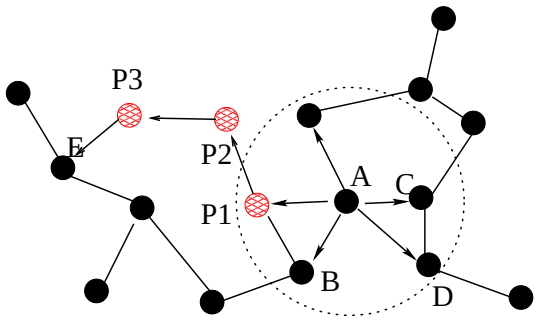

Fig. 6. Wormhole and rushing attacks. $P1$, $P2$,$P3$ are malicious nodes.

collisions on E to prevent a retransmitted packet from being received prior to a forged one. This can be easily detected because the hidden-terminal problem does not happen that frequently in a network where RTS/CTS control packets are used. In addition, impersonating a node within a range of two hops is very likely to be detected by other neighboring nodes.

Wormhole and Rushing Attacks Wormhole attacks [15] and rushing attacks [16] are also launched by multiple colluding nodes. Fig. 6 shows such an attack in which three attacker nodes P_1, P_2, and P_3 collude to tunnel packets between node A and node E with the goal of convincing A and E that they are still neighbors. More specifically, P_1 forwards every message it overhears from node A, including KEYUPDATE messages and traffic packets, towards node E through P_2 and P_3. P_3 retransmits the KEYUPDATE messages to node E, but modifies the traffic packets to deceive node E. Similarly, P_3 forwards every message from node E towards node A through P_1 and P_2. Due to time synchronization errors, if nodes A and E receive replayed KEYUPDATE messages from each other without a significant time delay, they will forward those forged traffic packets.

To prevent these attacks, we need to employ some more sophisticated techniques [15,16] that require the deployment of GPS devices or tight time synchronization. In general, LHAP is vulnerable to these attacks because it does not place these stringent requirements on the nodes and the networks.

Since our goal is to prevent resource consumption attacks, we stress that from resource consumption perspective, an attacker does not benefit much from its attack when the hijacked packets are multicast or broadcast packets. An attacker can increase the size of a hijacked packet, but it cannot inject extra packets. Therefore, the hijacked packets are more likely to be unicast packets.

5.2 Insider Attacks

Insider attacks are attacks launched by one or more compromised nodes that possess valid certificates. An insider node could try to inject spurious packets while impersonating other nodes or in its own name. Multiple insider could also collaborate to launch more sophisticated attacks. In general, there is no way to prevent insider attacks if the compromised node cannot be detected, and it is also very difficult to detect insider attacks, especially by multiple colluding insider nodes.

LHAP cannot prevent an insider node from maliciously injecting packets into a network, but it can restrict an insider node from impersonating other nodes due to our trust management mechanism. Moreover, LHAP can help detect some insider attacks and identify the malicious insider nodes. This is due to the neighbor encounter time periods every node records in its trust maintenance phase, which serves as encounter evidence. A node submits this information to a central authority off-line. From this encounter information reported by all or a fraction of users, the authority might detect some inconsistencies and then identify the malicious nodes. Since compromise detection is an independent research topic, we will not discuss it in detail in this paper.

6 Performance Evaluation

In this section, we evaluate the performance overhead of LHAP. The ability of LHAP to filter unauthenticated data packets and prevent other security attacks has been analyzed in Section 5, so we will not examine this ability through simulation. The overall goal of our experiments is to measure the performance overhead of adding a network access control protocol when the network is *not* under attack. Specifically, we want to answer such questions as the following. How much bandwidth overhead does LHAP introduce? How many authentic packets does LHAP drop? What is the impact of LHAP on the upper layer protocols?

6.1 Performance Analysis

We consider the following performance metrics in evaluating LHAP.

Computational Overhead The numbers and types of cryptographic operations a node has to compute in a time unit.

Latency The time delay introduced by LHAP during the forwarding of a packet and the impact on the other network layers.

Traffic Delivery Ratio γ γ denotes the fraction of authenticated traffic packets that a node accepts among all the authenticated traffic packets it has received from its neighbors. Clearly, $1 - \gamma$ is the fraction of authenticated packets discarded by a node in error. The larger γ, the smaller impact of LHAP on the upper layer protocols; $\gamma = 1$ means there is no impact.

Traffic Overhead Ω We define traffic overhead Ω as the bandwidth overhead (in bytes per second per node) for transmitting LHAP control packets.

In the rest of this section, we evaluate the computational overhead and latency through detailed analysis and evaluate traffic delivery ratio and traffic overhead through detailed simulations.

6.2 Simulation Methodology

We use the GloMoSim 2.0 simulator [11]. The default scenario considered in our simulations is as follows. The physical layer uses the two-ray ground reflection model [35] as the radio propagation model. The medium access control layer uses the IEEE 802.11 Distributed Coordination Function (DCF) [18]. We use the random waypoint mobility model [19]. Each node moves at a speed uniformly distributed between 0 and v_{max} towards a location randomly selected in a square environment space of $2000m \times 2000m$. When it reaches the destination, it waits for a *pause time* and then moves towards a new location. Each node joins the network at a time uniformly distributed between the simulation time 0 and $5s$. The number of nodes is 100. Each simulation runs for 900 seconds of simulated time.

To demonstrate that LHAP is independent of the routing protocol, we insert LHAP beneath three different routing protocols – the unicast routing protocols DSR [19] and AODV [34] and the multicast routing protocol ODMRP [25]. Since our goal is not to evaluate these routing protocols, we simply use the default parameters for these protocols in GloMoSim.

We generate different scenario files with varying traffic patterns. More specifically, a source node generates and transmits constant bit rate (CBR) data packets to a randomly picked node. The size of a CBR packet is 512 bytes. The intervals between two CBR packets vary from 0.1s to 1.0s and the durations of connections vary from 10s to 850s.

When the underlying routing protocol is AODV or DSR, we select 13 source-destination pairs for unicast communication. The setting for ODMRP is a

little bit more complex. Of 100 nodes, 13 nodes form a multicast group, and 12 nodes form another group. Each group has three CBR data sources. The remaining 75 nodes do not belong to either of the two groups, but they can forward packets for the group members when needed.

Table 1 lists the default parameters used in the simulation (unless otherwise mentioned).. The values for TESLA parameters are the same as those used in [14].

Physical Link Bandwidth	2 Mbps
Transmission range	250 m
TESLA interval	2 s
Maximum time synchronization error	0.1 s
Pessimistic end-to-end propagation time	0.2 s
Hash length	10 bytes
Time for signature generation	46 ms
Time for signature verification	1 ms
Key length (including a key id)	10 bytes
Signature size (1024-bit RSA)	128 bytes

Table 1
Simulation Parameters

6.3 Performance Results

6.3.1 Computational Overhead

LHAP introduces two types of computational overhead – for traffic authentication and for trust management respectively. The computational overhead for verifying a traffic packet is usually negligible due to the efficiency of computing a hash function, although a node might need to compute multiple hashes to verify a packet from a neighbor occasionally when the neighbor has recently released several TRAFFIC keys that this node was unaware of due to packet loss or because they were not in each other's transmission range. Note that in LHAP this verification cost is independent of packet size.

The computational overhead for trust management arises from multiple operations. The first operation is a digital signature generation for bootstrapping a TESLA key chain. In LHAP, digital signature generation is the most expensive operation, but it is only performed once. Hence, the cost is negligible when amortized over all packets. The second operation is signature verification. A

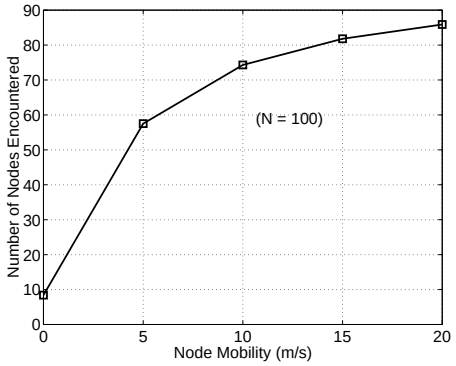

Fig. 7. The impact of node velocity on the number of nodes a node encountered

node verifies two signatures for every JOIN message received from a newly encountered neighbor. The overall verification cost is determined by the number of nodes encountered, which in turn is determined by network size, network density, and node velocity. The larger the network size and the network density are, the more nodes a node will encounter. Fig. 7 shows that the number of encountered nodes increases with node velocity. Therefore, the signature verification cost also increases with node velocity. We observe that the number of signature verifications could be more than a hundred. However, a signature verification is about 40 times faster than a signature generation [1]; therefore, this computational cost is also negligible when amortized over all packets a node receives and transmits. In addition, a node computes two hashes for every KEYUPDATE message. This computational cost is also negligible due to the efficiency of hash function.

6.3.2 Latency

In LHAP, a node normally verifies a received traffic packet by computing one hash, the time for which is less than one millisecond even for handhold PDAs [1] 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 μs [38].

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, which we do not know. Let us just consider the time spent in MAC layer and routing layer and also ignore the computational overhead, then the main processing time includes the random backoff and the jitter time in both the routing protocol and the MAC protocol. In AODV, the average random backoff time is 250ms and the average broadcast jitter time is 5ms; in 802.11 MAC protocol, the average random backoff time is 25ms. Thus, the delay caused by LHAP is very small in comparison to the delay introduced in each hop.

6.3.3 Traffic Overhead

This subsection examines the impact of node mobility and TESLA interval on the traffic overhead. All the results are averaged over 64 independent runs.

Impact of Node Mobility Fig. 8 depicts the impact of node mobility on traffic overhead Ω. We can make four observations here. First, Ω increases with node velocity v, independent of the routing protocol deployed. One reason is that a node encounters a larger number of nodes when it moves at a higher velocity, as shown in Fig. 7. In LHAP, a node sends an ACK packet (of size 300 bytes) to every new neighbor. Another reason is that the increase of node mobility results in more control packets in the routing protocols such as AODV, DSR, and ODMRP. LHAP authenticates every traffic packet, includes every routing control packet in the network layer, leading to larger traffic overhead. Since traffic overhead is application dependent, the values shown in the figure should not be interpreted as the absolute performance of LHAP. In the case of no traffic packets, the main traffic overhead of LHAP is a KEYUPDATE message per TESLA interval, the overhead of which is less than 10 byte/s.

Second, the traffic overhead grows at a lower rate as node velocity increases, because the chance that a node meets new nodes does not increase linearly with its velocity due to the limited size of a network. The figure also shows the traffic overhead in the case of ODMRP is larger than in AODV and DSR. This is due to different traffic patterns used in the simulations.

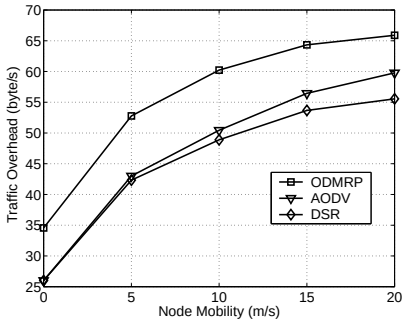

Fig. 8. The impact of node mobility on traffic overhead

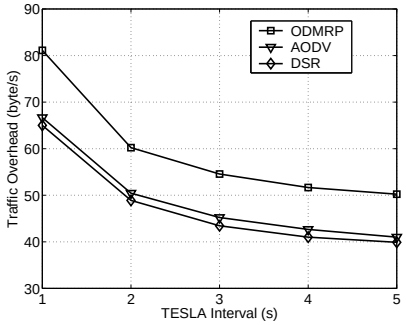

Fig. 9. The impact of TESLA interval on the control packet overhead of LHAP

Impact of TESLA Interval Fig. 9 shows that traffic overhead decreases with TESLA interval. This is because a node sends fewer KEYUPDATE messages when a larger TESLA interval is used.

6.3.4 Traffic Delivery Ratio

Impact of Node Mobility Two observations can be made from Fig. 10. First, the traffic delivery ratio is close to 100%. In LHAP, a node *normally*

never discards any packets from a neighbor with which it has established a trust relationship, although it may fail to drop forged data packets under the collusion attacks discussed in Section 5. When a node receives a packet from a new neighbor, it immediately sends back an ACK message to that neighbor to bootstrap their trust. Therefore, it might drop one or more packets from that neighbor during this process. Note that in our simulations, a node immediately drops any unverifiable packets from a new neighbor. Another strategy could be that a node buffers these unverifiable packets temporarily until the trust bootstrapping process is completed; thus no such packets will be dropped. Moreover, since the first packet a node receives from a new neighbor is normally a broadcast packet (e.g. a routing request packet or a KEYUPDATE message), dropping a few of these packets has a negligible impact on the delivery ratio of application data (i.e., the CBR data packets). In our simulations, we did not see any evidence that this type of rare packet dropping by LHAP reduces the data packet delivery ratio. Indeed, we found that in some cases the data packet delivery ratio increases a little bit as a result of the rare events of packet dropping. For example, if a node is not on the route between a source and destination pair, its dropping of a routing request packet has no effect at all; if it is supposed to be a node on the route, its packet dropping might change the route, but not necessarily leading to a less reliable one.

Second, traffic delivery ratio decreases slightly with node velocity. This is because a node receives more unverifiable packets from more new neighbors. But again, the impact is very small. In the worst case, γ is less than 0.2%.

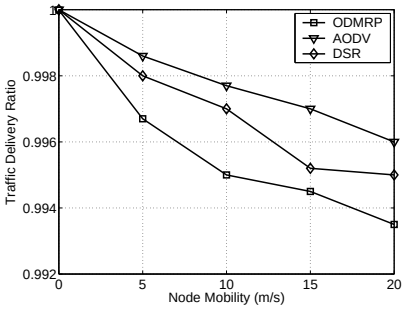

Fig. 10. The impact of node mobility on traffic delivery ratio

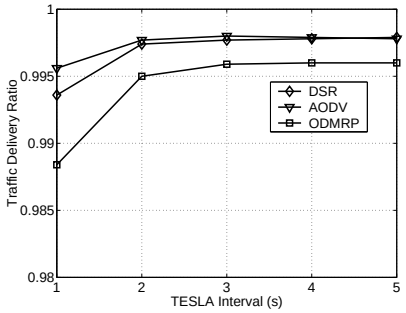

Fig. 11. The impact of TESLA interval on the traffic delivery ratio of LHAP

Impact of TESLA Interval Fig. 11 shows that traffic delivery ratio increases slightly with TESLA interval (node velocity is 10m/s). This is mainly due to the trust termination process, whereby a node temporarily terminates its trust on another node if it has not heard from that node for one TESLA interval. Two cases may cause two nodes to temporarily lose contact in one TESLA interval but regain contact later on. The first case is when a HELLO message is lost; the second case is when a node moves out of the transmission range of the other but moves in again quickly. In our simulation, a node drops packets from

another node when their old trust has been terminated while their new trust has not been reestablished yet. The larger the TESLA interval, the less likely that two nodes completely lose their contact in the interval, thus the larger the packet delivery ratio. Note that packet delivery ratio could be further improved if a node temporarily buffers the unverifiable packets until their trust (re)establishment process is completed.

Based on the above performance evaluation and the security analysis in Section 5, we can see that a larger TESLA interval improves the performance but also enlarges the vulnerability window for impersonation attacks. Therefore, the choice of TESLA interval should be based on the requirements of an application under consideration. Nevertheless, the above performance analysis shows LHAP is a light-weight security protocol with respect to both computation and communication overhead, and yet has very small effect on the performance of other layer protocols.

7 Deployment Issues

We discuss the issues related to the deployment of LHAP in this section.

7.1 Interaction With Routing Protocols

LHAP is independent of the underlying (secure) routing protocols. In practice, it could take advantage of the deployed routing protocol to achieve better efficiency. Some ad hoc routing protocols, e.g., AODV, TORA [28], AMRIS [39] *et al*, require that nodes periodically exchange routing information or beaconing messages with their neighbors. LHAP can piggyback its KEYUPDATE information in these messages to avoid transmitting separate KEYUPDATE packets. This combination does not affect the transparency of LHAP with respect to the routing protocol, because the LHAP layer in a receiving node removes the piggybacked KEYUPDATE message prior to submitting the message to the network layer.

Several secure routing protocols [8,14] use public key certificates to bootstrap trust between nodes. When employed under these protocols, LHAP can use the certificates used by these protocols. Since the security services provided by LHAP are complementary to those provided by secure routing protocols, they can be employed at the same time to provide stronger security.

7.2 Supporting Very Long Key Chains

In LHAP, TESLA keys are disclosed periodically. For a network with a lifetime of five hours and TESLA interval of one second, the TESLA key chain will have a size of $5 \times 3600 = 18,000$ keys. On the other hand, TRAFFIC keys are usually consumed at a much higher rate, depending on the application under consideration. As a result, very long key chains are needed.

If a node stores all the keys in a key chain, it can use any key immediately. This approach however is not scalable with respect to its storage requirements. Another approach is one in which a node only keeps the last key of the key chain and derives every key from the last key every time. This scheme has the minimal storage requirement, but it is not scalable in terms of computational cost. Coppersmith and Jakobsson [3] propose an optimization algorithm that can make tradeoff between storage and computation cost. Their algorithm performs $\lceil d\log_2 N \rceil$ hashes per output element, and uses $\lceil d\log_2 N \rceil$ memory cells, where the size of each cell is slightly larger than that of a key and N is the length of the key chain.

The multilevel key chain scheme proposed by Liu and Ning [23] can also be used to provide a very long TESLA key chain with small storage and computation overhead. The main idea is to construct multiple level key chains of different disclosure intervals such that a low-level key chain can be derived on the fly from a key in a high-level key chain. Both the high-level and the low-level key chains can use the optimization scheme in [3] to enable further tradeoffs between storage and computation.

7.3 Supporting Fast Hash Verification

Consider the scenario in which two nodes encounter each other again T seconds after their last encounter. If $T = 3600$ and the TESLA interval is 1 second, they need to compute 3600 hashes to verify each other's current TESLA keys. Although computation of a hash function is very efficient, it is still undesirable to have such a big number of hash computations. On the other hand, an attacker may exploit this feature to consume the computational resource of a node by engaging the node in computing a large number of hash evaluations. For example, a malicious node may replay another node's initial bootstrapping message and replay it to another node a couple of hours later, forcing that node to perform thousands of hash computations. This is actually a resource consumption attack.

To support fast hash verification, LHAP uses a tree-based authentication scheme, which is also known as Merkle hash tree [26] and is further stud-

ied in [15]. The maximum number of verifications a receiver has to perform is $log_2 N$, where N is the length of a TESLA key chain. Suppose $N = 10,800$, the number of verifications is no more than 14. Thus, although an attacker may launch the aforementioned replay-based resource consumption attack, the consequence is still limited.

Note that this verification algorithm only works for TESLA key chains, not for TRAFFIC key chains because TRAFFIC keys are not disclosed periodically. Actually, fast verification is normally not necessary for a TRAFFIC key chain. Because the most recently disclosed TRAFFIC keys are announced in KEYUPDATE messages, the number of hashes a node has to compute to verify a TRAFFIC key is upper bounded by the number of traffic packets transmitted in one TESLA interval.

8 Related Work

Stajano and Anderson [36], Balfanz et al. [2] proposed schemes to establish trust and keys between nodes through physical contact in the absence of an online authentication server. Zhou and Hass [42] propose to use threshold signature schemes to prevent one or a few compromised nodes from signing messages for the group; this approach was later extended by Kong et al. [21] to a distributed scheme.

To setup a secret key for a pair of nodes, one approach is to preload each of them with the key, but it requires a node to store $N-1$ keys for a network with N nodes. To address this issue, Eschenauer and Gligor [9] proposed a probabilistic key pre-deployment scheme in which every node only stores a subset of keys selected randomly from a large key pool. This scheme was extended by several other schemes [4,7,24,46] to provide stronger security under node compromises. Zhu et al also proposed a key management framework consisting of a suite of keying mechanisms for sensor networks [44] and an efficient group key management scheme for mobile ad hoc networks [45].

Secure routing for ad hoc networks has been extensively studied recently. Dahill et al [8] identified several security vulnerabilities in AODV and DSR, and proposed to use asymmetric cryptography for securing ad hoc routing protocols. Yi, Naldurg, and Kravets [41] presented a security-aware routing protocol that uses security (e.g., trust level in a trust hierarchy) as the metric for route discovery between pairs. Papadimitratos and Hass [32] proposed a secure routing discovery protocol that assumes a security association (SA) between a source and a destination, whereas the intermediate nodes are not authenticated. Hu, Perrig and Johnson designed SEAD [13] which uses one-way hash chains for securing DSDV, and Ariadne [14] which uses TESLA and

HMAC for securing DSR.

LHAP also uses the techniques such as one-way hash chains and TESLA due to their efficiency. The main difference between LHAP and the secure routing protocols is in their design goals. While those protocols are designed for securing *specific routing* protocols, LHAP is a *general* network access control protocol for preventing resource consumption attacks.

Zhang and Lee [43], Marti et al. [27] have studied the intrusion and misbehavior detection issue in mobile networks. It is suggested [43] that every node be installed with an intrusion detection system (IDS) which collects trace data imported from all the layers, because compromised nodes could launch attacks against multiple layers, such as routing layer and application layer. Therefore, the encounter evidence provided by LHAP will help in node compromise detection.

9 Conclusions

In this paper, we have presented LHAP, a lightweight hop-by-hop authentication protocol for network access control in ad hoc networks. LHAP is based on two techniques: (i) hop-by-hop authentication for verifying the authenticity of all the packets transmitted in the network and (ii) one-way key chain and TESLA for packet authentication and for reducing the overhead for establishing trust among nodes. The design of LHAP is transparent to and independent of the routing protocols. Through a detailed simulation study, we show that LHAP is efficient and allows a tradeoff between security and performance. In the future, we will investigate new solutions that do not rely on TESLA.

References

[1] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes. PGP in Constrained Wireless Devices. In Proc. of the 9th USENIX Security Symposium, pages 247-261, August 2000.

[2] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In Proc. of Symposium on Network and Distributed Systems Security (NDSS'02), 2002.

[3] D. Coppersmith, M. Jakobsson, Almost Optimal Hash Sequence Traversal, In Proc. of Finanical Cryptography (FC) 02.

[4] H. Chan, A. Perrig, D. Song. Random Key Predistribution Schemes for Sensor Networks. In Proc. of the IEEE Security and Privacy Symposim 2003, May

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 on Security of Ad Hoc and Sensor Networks (SASN '04).

[6] C. Davis and C. Crepeau. A Certificate Revocation Scheme for Wireless Ad hoc Networks. In Proc. of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03).

[7] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In Proc. of the 10th ACM conference on Computer and communication security, pp.42-51, 2003.

[8] B. Dahill, B. Levine, E. Royer, C. Shields, A Secure Routing Protocol for Ad-Hoc Networks. In Proc. of the 10 Conference on Network Protocols (ICNP), November 2002.

[9] L. Eschenauer and V. Gligor. A Key Management Scheme for Distributed Sensor Networks. In ACM CCS2002, Washington D.C., 2002.

[10] V. Gupta, S. Krishnamurthy, and M. Faloutsos, Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks. In Proc. of Milcom 2002, Anaheim.

[11] URL: http://pcl.cs.ucla.edu/projects/glomosim/.

[12] J. Hubaux, L. Buttyan, S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. In Proc. of ACM MobicHoc 2001.

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

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

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

[16] Y. Hu, A. Perrig, and D. Johnson. Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols. In Proc. of ACM Workshop on Wireless Security (WiSe 2003).

[17] H. Huang and S. Wu. An Approach to Certificate Path Discovery in Mobile Ad Hoc Networks. In Proc. of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03).

[18] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11, 1997.

[19] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T.Imielinski and H. Korth, editors, Mobile Computing , Pages 153-181. Kluwer Adcademic Publishers, 1996.

[20] Vesa Karpijoki. Signaling and Routing Security in Mobile and Ad-hoc Networks. http://www.hut.fi/ vkarpijo/iwork00/.

[21] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks, In Proc. of the 9th Internation Conference on Network Protocols (ICNP'01).

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

[23] D. Liu and P. Ning, Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks In Proc. of NDSS'03.

[24] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In Proc. of the 10th ACM conference on Computer and communication security, pp. 52-61, 2003.

[25] 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 Communications in Wireless Mobile Networks, 2001.

[26] Ralph Merkle. A certified digital signature. In Gilles Brassard, editor, Advances in Crypto-89, pages 218-238, Berlin, 1989.

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

[28] V. Park and M. Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, In Proc. of IEEE INFOCOM '97, Kobe, Japan (April 1997) .

[29] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In Proc. of SIGCOMM'94. 1994.

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

[31] A. Perrig, R. Canetti, J. Tygar, D. Song. Efficient authentication and signing of multicast streams over lossy channels. In Proc. of IEEE Symposium on Security and Privacy. May 2000.

[32] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad hoc Networks. In SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), 2002.

[33] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of Spread Spectrum Communications A Tutorial. IEEE Transactions on Communications, 30(5):855884, May 1982.

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

[35] T. Rappaport. Wireless Communications: Principles and Practice. Prentice-Hall, 1996.

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

[37] F. Tobagi and L. Kleinrock. Packet switching in radio channels: part ii - the hidden terminal problem in carrier multiple-access and the busy-tone solution. IEEE Transactions on Communications, vol. com-23, no.12, pp. 1417–1433, 1975.

[38] http://www.eskimo.com/*sim*weidai/amd64-benchmarks.html.

[39] C. Wu, Y. Tay, and C. Toh, Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification, Internet-Draft, draft-ietf-manet-amris-spec-00.txt, Nov. 1998, Work in progress.

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

[41] S. Yi, P. Naldurg, R. Kravets. Security-Aware Ad-Hoc Routing for Wireless Networks. Technical Report, UIUCDCS-R-2001-2241, UILU-ENG-2001-1748.

[42] L. Zhou and Z. Hass. Securing Ad Hoc Networks. IEEE Network Maganize, 13(6), November/December 1999.

[43] Y.Zhang and W.Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In Proc. of ACM MOBICOM 2000.

[44] S. Zhu, S. Setia and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03), 2003.

[45] 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 Mobile and Ubiquitous Systems (Mobiquitous'04), 2004.

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