LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks

Sencun Zhu[1] Shouhuai Xu[2] Sanjeev Setia[1*] Sushil Jajodia[1,3]

[1]*Center for Secure Information Systems, George Mason University, Fairfax, VA 22030*
[2]*Dept. of Information and Computer Science, Univ. of California at Irvine, Irvine, CA 92697*
[3]*The MITRE Corporation, McLean, VA 22102*
E-mail: {szhu1,setia,jajodia}@gmu.edu,shxu@ics.uci.edu

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 that ensure that only authorized nodes can inject traffic into the network. In this paper, we present LHAP, a scalable and light-weight authentication protocol for 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. We analyze the security of LHAP, and show LHAP is a lightweight security protocol through detailed performance analysis.*

1. Introduction

In ad hoc wireless networks, no base stations exist and each 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.

Clearly, a network access control capability is essential for ad hoc networks in an adversarial environment such as a battlefield. 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.

To deal with such attacks, recently researchers [1, 2, 3] have proposed security extensions to existing routing protocols that include mechanisms for authenticating the routing control packets in the network. However, none of the proposed secure routing protocols include any provisions for authenticating data packets. Thus, a resource consumption attack based on data packets, especially in multicast applications, can be launched easily. 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, an intuitive solution is to authenticate all packets so that a node only forwards packets from authorized nodes. A simple solution is to use a network-wide key shared by all nodes and each node uses this shared key to compute message authentication codes (MACs) on the packets it sends and receives. This scheme, however, requires an expensive global re-key operation if the shared key is compromised. Another option is to use authentication techniques based on asymmetric cryptography. However, these techniques usually do not adapt well to ad hoc networks. In wireless ad hoc networks with high node mobility, the neighbor set of a node may keep changing; therefore, the frequency and hence the cost for performing mutual authentication between nodes is much greater than that in wired networks that do not have node mobility. Further, the resources of a mobile node such as battery power, computational capacity and bandwidth are usually quite constrained. All these facts make most of the

*also with Dept. of Computer Science, George Mason University

Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)
0-7695-1921-0/03 $17.00 © 2003 IEEE

authentication protocols proposed in literature impractical for implementing access control for ad hoc networks.

In this paper, we present LHAP, a scalable and efficient network access control 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 can be thought of as residing in between the data link layer and the network layer, providing a layer of protection that can prevent or thwart many attacks from happening, including attacks on ad hoc routing protocols made possible by the lack of support for packet authentication in these protocols.

The rest of this paper is organized as follows. We present the details of the LHAP protocol in Section 2, and analyze its security in Section 3. In Section 4, we analyze the performance of our protocol, and show several possible optimizations for its real deployment in Section 5. Finally, we discuss related work in Section 6, and present our conclusions in Section 7.

2. A Lightweight Hop-by-hop Authentication Protocol (LHAP)

In this section, we first describe the assumptions we made during the design of our protocol. Next we give an overview of the design goals and basic operation of LHAP. Finally, we discuss the operations of LHAP in detail.

2.1. Assumptions

We make the following assumptions in our work. First, the network links are bidirectional. Second, we assume that a packet sent by a node is received by a neighboring node before a third node can replay the packet to it, unless the neighbor under consideration has dropped the packet. Third, we assume each node has a public key certificate signed by a trusted certificate authority (CA) and also an authentic public key of the CA. Our protocol relies on these public keys to bootstrap trust in the ad hoc network. The distribution of certificates and keys can be done in any reliable way. Fourth, we assume that the mobile nodes under consideration are relatively underpowered. Public-key operations such as digital signatures are relatively expensive to compute. Finally, we assume loose time synchronization in the ad hoc network since LHAP utilizes the TESLA broadcast authentication protocol [8]. (In the discussion below, we assume that reader is familiar with TESLA [8].)

2.2. 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.
- $M1|M2$ 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.

2.3. Protocol Description

The main goal of our protocol is to provide network access control, i.e., to prevent unauthorized nodes from being able to inject traffic into the ad hoc network. To achieve this goal, under LHAP, every node in the network authenticates every packet, irrespective of whether it is a data packet or a routing control packet, received from its neighbors before forwarding it. Packets from unauthorized nodes are dropped, thus preventing them from propagating through the network. LHAP 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 LHAP to be turned on or off without affecting the operations of other layers.

LHAP's efficiency gains over traditional authentication protocols derive from two techniques: (i) lightweight packet authentication, and (ii) lightweight trust management. Since all packets are authenticated on every hop on their paths (we refer to this as *hop-by-hop* authentication), it is necessary 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 relationship between nodes. Below we present these two techniques in more detail.

Lightweight Traffic 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. Moreover, since each node has to buffer the traffic packets it has received until they are authenticated, delayed authentication will lead to large storage requirements at every node.

In LHAP, each node generates a one-way key chain that is used for traffic authentication by its immediate neighbors. We use the term TRAFFIC key to refer to the keys in this one-way key chain. For example, consider a node A that wants to broadcast a packet M. Let its next TRAFFIC key be $K_A^F(i)$. It will send the following message

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

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. In LHAP, a node only authenticates traffic packets from its direct neighbors, thus it is very difficult, if not impossible, for an attacker to launch replay attacks.

Using TRAFFIC keys for traffic authentication has the following benefits. First, it enables instant verification of traffic packets. Second, it is not necessary to disclose TRAFFIC keys periodically; disclosing keys periodically would result in a severe wastage of keys when a node has no packets to transmit. In practice, in LHAP the rate at which a node consumes its TRAFFIC keys can be adapted to the actual traffic rate. Third, it is computationally more efficient than computing HMAC over the entire message, because it only requires computing a hash over a key of a small fixed size (e.g., 8 bytes). However, we also note this scheme does not achieve the same level of security as in TESLA, as a tradeoff between security and performance. We shall discuss the possible attacks on TRAFFIC keys in Section 4.

Trust Management

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

Trust Bootstrapping When a node wants to join an ad hoc network, it first pre-computes a one-way key chain and a TESLA key chain. Then it signs the commitments of these key chains and broadcasts them to its neighbors. In Figure. 1, we show a scenario where node A starts to join a network where its neighbors are B,C,D and E. Node A broadcasts a JOIN message with $TTL = 1$.

$$A \longrightarrow * : Cert_A, Sign_A\{A|K_A^T(0)|K_A^F(0)|T_A^T(0)|T_A^F(0)\}, \quad (2)$$

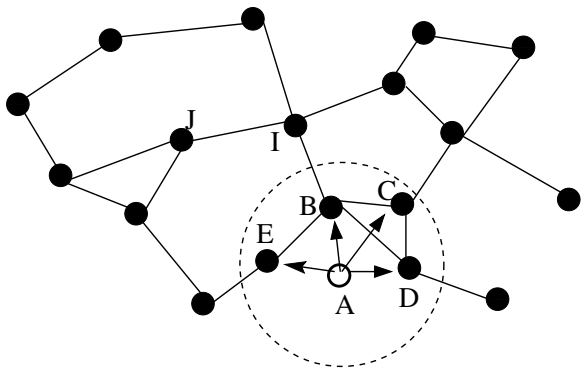

Figure 1. A scenario where node A joins the ad hoc network

where $T_A^T(0)$ and $T_A^F(0)$ are the starting times for its TESLA and TRAFFIC key chains respectively. Every receiving node first verifies the authenticity of node A's certificate using the CA's public key, then uses node A's public key in the certificate to verify the signature on the message. It will record the commitments of node A's key chains and their starting times if all the verifications are successful.

To bootstrap an authentic TRAFFIC key and a TESLA key to node A, each of its neighbors (say B) unicasts the following ACK message to A:

$$B \longrightarrow A : Cert_B, Sign_B\{B|K_B^T(0)|K_B^F(0)|T_B^T(0)|T_B^F(0)\}, MAC(K_B^T(i), K_B^F(j)), \quad (3)$$

where the signature was generated when node B first joined the network, $K_B^F(j)$ is node B's most recently released TRAFFIC key, and $K_B^T(i)$ is node B's next TESLA key to be released. When receiving this message, node A does two verifications and obtain node B's authentic key chain commitments. Note node A cannot verify the MAC until node B releases $K_B^T(i)$. The disclosure delay is half of a TESLA interval on average. After it receives and verifies $K_B^T(i)$ later, node A starts to forward valid traffic from node B, as described in Section 2.3.1.

Trust Maintenance Periodically, each node broadcasts an KEYUPDATE message (with TTL=1) to its 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

$$A \longrightarrow * : A, K_A^T(i-1), MAC(K_A^T(i), K_A^F(j)), \quad (4)$$

where $K_A^F(j)$ is node A's most recently released TRAFFIC key, and $K_A^T(i)$ is node A's next TESLA key to be released.

In addition, node A includes $K_A^T(i-1)$ to allow its neighbors to verify the previous KEYUPDATE messages from node A. The purpose of broadcasting KEYUPDATE messages is for preventing malicious nodes from forging traffic using the TRAFFIC keys node A has already released.

A neighboring node receiving the above KEYUPDATE message can verify the authenticity of $K_A^F(j)$ based on the most recent key in this key chain, even though it cannot verify the MAC immediately. However, the node does not know if $K_A^F(j)$ is the most recent key from node A until it receives the delayed disclosed $K_A^T(i)$. We shall discuss the security of this approach in more detail in Section 4.

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, all the nodes will terminate their trust relationship with that node permanently. Second, when a node does not receive a (valid) KEYUPDATE message from a neighbor within a TESLA interval because the latter has moved out of its transmission range, it will terminate its trust of this neighbor temporarily. If the two nodes move within transmission range again, they can run the trust bootstrapping process again to reestablish their trust relationship, if they do not have any cached commitments of the other's key chain. Otherwise, they can reestablish their trust relationship using TESLA, with a delay of half a TESLA interval on average.

3. Security Analysis

In this section, we discuss some possible attacks against LHAP, which are all against traffic key chains. We assume TESLA is secure, given loose time synchronization in the network.

3.1. Outsider Attacks

Outsider attacks are attacks launched by nodes that do not possess a valid certificate. We identify three types of outsider attacks here.

Single Outsider Attack In Figure 1, we 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 authentic commitments of node A's key chains. Now we consider a scenario in Figure 2 where node E has moved out of node A's transmission range for a time period (say for several TESLA intervals). During this time, node A has disclosed many of its TESLA keys and TRAFFIC keys. An outside attacker, node P_2, may eavesdrop and use these keys to impersonate node A. For instance, suppose node A has broadcast a packet with content M and

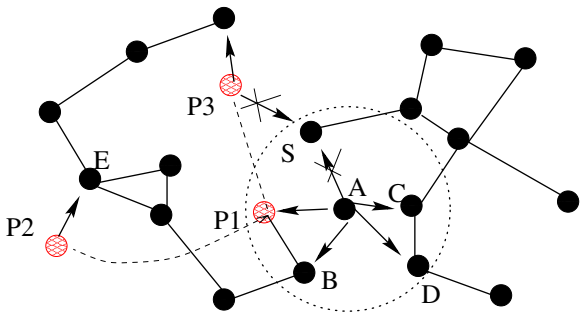

Figure 2. Various attacks on LHAP. P1, P2 and P3 are malicious nodes, and the dashed lines between them are private channels

$K_A^F(i)$. Suppose that node P_2 eavesdropped on the message and then moved within range of node E. It may modify M to M' with the same TRAFFIC key, and send it to node E. Because TRAFFIC keys are not periodically disclosed, node E cannot determine which TRAFFIC key node A is using.

To thwart this attack, we designed the *trust termination* phase as in Section 2.3. Since node E has not heard from node A for a time period of more than one TESLA interval, it will not forward any traffic from node P_2 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_2 cannot use the TESLA keys that node A previously disclosed to forge an KEYUPDATE message. On the other hand, node P_2 cannot forge a valid KEYUPDATE message using any TESLA keys that have not been released yet by node A due to the one-wayness of hash functions.

Collaborative Outsider Attack A *collaborative outsider attack* (also called a *wormhole* attack [4]) is launched by multiple colluding outside attackers. In Fig. 2, the attackers P_1 and P_2 have a private channel that allows them to communicate directly. P_1 forwards every message it eavesdropped from node A, including KEYUPDATE messages and traffic packets, to P_2 through the *wormhole*. P_2 then rebroadcasts the KEYUPDATE messages and modify the traffic packets to deceive node E. Due to time synchronization errors, node E may still accept the replayed KEYUPDATE messages, and will forward the modified traffic packets from node P_2.

This attack can be detected if the mobile nodes carry devices such as Global Positioning Systems (GPS). A node can put its GPS coordinates in its KEYUPDATE messages to allow a receiving node to determine if they should be able to hear each other. For example, in Figure 2 node E and A

should not be able to hear each other based on their coordinates; therefore, node E will detect the inconsistency of node A's position when it receives the replayed KEYUPDATE messages from P_2, and drop the messages received later from P_2. We note that our protocol does not address this attack completely. Due to the one-TESLA-interval delay for verifying an KEYUPDATE message, node E will still forward the (probably modified) traffic packets from P_2 in this interval if the packets carry TRAFFIC keys that are verified to be correct and later than the one in the KEYUPDATE message in node A's TRAFFIC key chain. However, there is an upper bound on the number of forged packets, which is the number of packets node A actually sends in this interval, because an attacker cannot compute the TRAFFIC keys which node A has not disclosed yet.

Hidden Terminal Attack A *hidden terminal attack* on TRAFFIC keys is more subtle, and it is motivated by the hidden-terminal problem in mobile networking. IEEE 802.11 solves the problem using CSMA/CA with ACKs and optional RTS/CTS control packets. For this scheme to work, however, an assumption that the contending nodes will cooperate has to be made. In Figure 2 we show an attack that tries to disrupt this cooperation, which we call a *hidden-terminal attack*.

Suppose node A broadcasts a traffic packet that includes a TRAFFIC key $K_A^F(j)$ for packet authentication. To attack node S, a malicious node P_3 transmits a packet to node S at the same time, which causes node S to drop both packets. Meanwhile, node P_1 can send $K_A^F(j)$ to node P_3 through a *wormhole*. Now node P_3 can send an erroneous packet to node S impersonating node A using $K_A^F(j)$ before node A does a retransmission; consequently, node S will drop the retransmitted authentic packet from node A.

However, since the retransmission interval is usually very small (tens of microseconds), the attackers may have to run continuous attacks on S to prevent the packets that node A retransmitted from successfully being received. This can be easily detected because the hidden-terminal problem does not happen that frequently in a network where the RTS/CTS control packets are deployed. In addition, impersonating a node within a range of two hops is very likely to be detected by other nodes.

3.2. Insider Attacks

Insider attacks are attacks launched by one or more compromised nodes that possess valid certificates. We identify three possible insider attacks.

Single Insider Attack A compromised node might attempt to flood the network with many traffic packets. For schemes which provide source authentication, having an upper bound on traffic rate could limit this attack. However, our hop-by-hop authentication scheme does not provide strong source authentication, because every node only authenticates its neighbors instead of the original traffic sources for the purpose of scalability. Thus, a compromised node might broadcast malicious traffic while pretending to be a forwarding node.

Insider Clone Attack A *clone attack* occurs when a compromised node shares its private key (hence its identity) with its outside conspirators. Due to their having the same identity, these nodes are less likely to launch collaborative attacks without being detected. The cloned nodes are therefore more likely to be distributed in different locations of the network. Indeed, this clone attack can be considered as multiple independent *single insider attack*s.

Multiple-insider Attack This attack is launched by multiple compromised insiders, each of which holding a legitimate certificate. Coalition of these insiders could result in very sophisticated attacks.

Generally, it is more difficult to detect the attacks launched by insider nodes, especially by multiple collaborative nodes. LHAP alone does not have complete solutions for addressing these attacks, although the use some techniques might mitigate the severity. For instance, in the *single insider attack*, the neighbors could be aware of the attack if a node pretends to be a forwarder for the packets originated from itself. We note a better solution is that every node is installed with an intrusion detection system (IDS) which collects trace data imported from all the network layers, because compromised nodes could launch attacks against multiple layers, such as routing layer and application layer. Moreover, multiple nodes could also perform cooperative detection. For example, in the *insider clone attack*, after exchanging its trace data with another node E, a node A might detect the attack if they both have met a third node P at about the same time but at quite different locations (if GPS is equipped). Zhang and Lee [12], Marti et al. [6] have studied the intrusion and misbehavior detection issue in mobile networks.

Finally, after detecting the attacks and identify the compromised nodes, all the remaining nodes add the compromised nodes into their local revoked node lists (RNL). As a result, they drop all the traffic from these compromised nodes in the future.

4. Performance Analysis

We mainly consider the following performance metrics in LHAP.

- **Computational Overhead** The main computational expense of LHAP is one RSA digital signature that each node creates before joining the network, which can be done off-line. Other computational overhead arises from signature verifications and hash computations which are affordable even for devices with very constrained computational capability.

- **Latency** In LHAP, a node verifies a traffic packet it receives by computing one or more hashes. Thus, the additional latency LHAP introduces is usually negligible in comparison to the end-to-end transmission latency of a packet.

- **Traffic Byte Overhead** We define traffic byte overhead as the number of all the non-traffic bytes a node transmits per time unit. There are four sources of traffic byte overhead in LHAP. First, a node adds a traffic key to every traffic packet it sends, so the overhead is one key per traffic packet and the overall overhead is mainly determined by the number of data sources and their traffic models. Second, a node sends a JOIN message at the time it joins the network, and the overhead for a JOIN message is determined by the size of a public key certificate and the size of a digital signature. Third, a node sends an ACK packet to every new neighbor, and an ACK packet is one key size and one MAC size larger than a JOIN message. The overall overhead is determined by network density and node mobility. Fourth, a node periodically sends a KEYUPDATE message (that includes two keys and one MAC), and its overhead depends on the TESLA interval. The greater the TESLA interval, the smaller the overhead. We note the byte overhead in the periodic KEYUPDATE messages and the traffic packets accounts for a major fraction of the overall byte overhead, because other types of overhead are amortized during the lifetime of a node in the network.

  Example Let us assume we use 10 bytes for a key (including a 2-byte key id), 10 bytes for a MAC, 500 bytes on average for a traffic packet, 256 bytes for a public key certificate, 128 bytes (1024 bits RSA) for a digital signature. Suppose a node joins the network for one hour, during which it sends (or forwards) 1000 packets and encounters 100 nodes. If the TESLA interval is 1 second, the traffic byte overhead is 44 bytes/s; If we use 2 second for the TESLA interval, the traffic byte overhead is 29 bytes/s. We believe this overhead is reasonable for a security service.

- **Traffic Delivery Ratio** We define this metric as the ratio of the number of traffic packets that a node accepts to the total number of packets that it receives from its neighbors. In LHAP, a node might drop traffic packets that are from legitimate neighbors in two scenarios. The first scenario arises when it encounters a neighbor for the first time, and the second scenario arises when it re-encounters a neighbor after more than one TESLA interval has elapsed since their last encounter. In both cases, it will drop packets from this neighbor if the neighbor is broadcasting[1] packets until it receives a valid ACK message or KEYUPDATE message from this neighbor within at most one TESLA interval. Therefore, the traffic delivery ratio is mainly affected by the TESLA interval, traffic rate and node mobility model. Our detailed simulation using the Network Simulator (ns2) [7] and similar parameter settings as in [3], shows that in most cases the traffic delivery ratio in LHAP is greater than 99.9%. Moreover, we note that one or several nodes occasionally dropping broadcast packets has very little effect in the delivery ratio of the application data due to the flooding nature of the broadcast packets.

The above performance analysis shows LHAP is a lightweight security protocol in terms of both computation and communication.

5. Issues In Deployment

5.1. Interaction With Routing Protocols

LHAP is independent of and resides under the network layer protocols. In practice, it could take advantage of the deployed network routing protocol to achieve better efficiency. Some of the ad hoc routing protocols in literature, e.g., AODV, TORA, require nodes periodically exchange routing information or beacon messages with their neighbors. Thus LHAP can piggyback its KEYUPDATE messages in these messages to avoid transmitting separate KEYUPDATE packets, although both the bandwidth and the energy for transmission and receiving are not much reduced. Note that this does not affect the transparency of LHAP with respect to the routing protocol, because the LHAP agent in a receiving node removes the piggybacked KEYUPDATE message prior to submitting the message to the network layer.

Since none of the secure routing protocols proposed in the literature deal with the authenticity of data packets, running LHAP beneath these protocols will further enhance the security of an ad hoc network. Furthermore, by deploying LHAP below the (insecure) multicast routing protocols that have been proposed for ad hoc networks, we believe that

[1] A node usually does not send a unicast packet to another node that has just become its neighbor in most routing protocols

many of the attacks on such protocols that are made possible by the lack of support for packet authentication can be prevented. Thus, the design of secure multicast routing protocols could be more focused on the functionality which LHAP cannot provide, e.g., source authentication. We expect that this will lead to more lightweight secure routing protocols.

5.2. 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 length of $5 * 3600 = 18,000$ keys. On the other hand, the TRAFFIC keys are usually consumed at a much higher rate, depending on the application. As a result, very long key chains are required in LHAP. LHAP adopts the multilevel key chain scheme proposed by Liu and Ning [5] to generate long key chains.

6. Related Work

Our work is related to previous work on secure routing in ad hoc networks. Dahill et al [1] identify several security vulnerabilities in AODV and DSR, and proposed to use asymmetric cryptography for securing ad hoc routing protocols. Although their approach could provide strong security, performing a digital signature on every routing control packet could lead to performance bottleneck on both bandwidth and computation. Perrig et al. [10] use symmetric primitives for securing routes between nodes and a trusted base station in a resource extremely constrained sensor network. Papadimitratos and Hass [9] propose a 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 [2] which uses one-way hash chains for securing DSDV, and Ariadne [3] which uses TESLA and HMAC for securing DSR. In LHAP, we also utilize these techniques due to their efficiency. The main difference between LHAP and their protocols is in the design goals. Their protocols are designed for securing *specific routing* protocols, while we design LHAP as a *general network access control* protocol which provides authentication for every packet and is independent of the routing protocols.

7. 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 our security and performance analysis, we show LHAP is beneficial and also practical.

References

[1] 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.

[2] Y. Hu, D. Johnson, A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In Proc. of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), IEEE, Calicoon, NY, June 2002.

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

[4] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In Proc. of INFOCOM 2003, IEEE, San Francisco, CA, April 2003, to appear.

[5] D. Liu and P. Ning. Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks. In Proc. of NDSS'03, Feb. 2003

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

[7] NS-2. http://www.isi.edu/nsnam/ns/index.html.

[8] 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.

[9] 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.

[10] A. Perrig, R. Szewczyk, V. Wen, D. culler, and J. Tygar. SPINS: Security Protocols for Sensor Networks. In Seventh Annual ACM International Conference on Mobile Computing and Networks(Mobicom 2001). Rome Italy, July 2001.

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

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