

# SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks

Yih-Chun Hu  
Carnegie Mellon University  
yihchun@cs.cmu.edu

David B. Johnson  
Rice University  
dbj@cs.rice.edu

Adrian Perrig  
Carnegie Mellon University  
perrig@cmu.edu

## Abstract

*An ad hoc network is a collection of wireless computers (nodes), communicating among themselves over possibly multihop paths, without the help of any infrastructure such as base stations or access points. Although many previous ad hoc network routing protocols have been based in part on distance vector approaches, they have generally assumed a trusted environment. In this paper, we design and evaluate the Secure Efficient Ad hoc Distance vector routing protocol (SEAD), a secure ad hoc network routing protocol based on the design of the Destination-Sequenced Distance-Vector routing protocol (DSDV). In order to support use with nodes of limited CPU processing capability, and to guard against Denial-of-Service (DoS) attacks in which an attacker attempts to cause other nodes to consume excess network bandwidth or processing time, we use efficient one-way hash functions and do not use asymmetric cryptographic operations in the protocol. SEAD performs well over the range of scenarios we tested, and is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, even in spite of any active attackers or compromised nodes in the network.*

## 1. Introduction

In a mobile wireless ad hoc network, computers (nodes) in the network cooperate to forward packets for each other, due to the limited wireless transmission range of each individual node. The network route from some sender node to a destination node may require a number of intermediate nodes to forward packets to create a “multihop” path from this sender to this destination. The role of the routing protocol in an ad hoc network is to allow nodes to learn such multihop paths. Since the nodes in the network may move at any time, or may even move continuously, and since sources of wireless interference and wireless transmission propagation conditions may change frequently, the routing protocol must also be able to react to these changes and to learn new routes to maintain connectivity.

Ad hoc networks require no centralized administration or fixed network infrastructure such as base stations or access points, and can be quickly and inexpensively set up as

needed. They can thus be used in scenarios where no infrastructure exists, or where the existing infrastructure does not meet application requirements for reasons such as security, cost, or quality. Examples of applications for ad hoc networks range from military operations and emergency disaster relief, to community networking and interaction between attendees at a meeting or students during a lecture. In these and other applications of ad hoc networking, security in the routing protocol is necessary in order to guard against attacks such as malicious routing misdirection, but relatively little previous work has been done in securing ad hoc network routing protocols.

Secure ad hoc network routing protocols are difficult to design, due to the generally highly dynamic nature of an ad hoc network and due to the need to operate efficiently with limited resources, including network bandwidth and the CPU processing capacity, memory, and battery power (energy) of each individual node in the network. Existing *insecure* ad hoc network routing protocols are often highly optimized to spread new routing information quickly as conditions change, requiring more rapid and often more frequent routing protocol interaction between nodes than is typical in a traditional (e.g., wired and stationary) network. Expensive and cumbersome security mechanisms can delay or prevent such exchanges of routing information, leading to reduced routing effectiveness, and may consume excessive network or node resources, leading to many new opportunities for possible Denial-of-Service (DoS) attacks through the routing protocol.

Routing protocols for ad hoc networks generally can be divided into two main categories: *periodic* protocols and *on-demand* protocols. In a periodic (or proactive) routing protocol, nodes periodically exchange routing information with other nodes in an attempt to have each node always know a current route to all destinations (e.g., [4, 5, 8, 10, 23, 31, 34]). In an on-demand (or reactive) protocol, on the other hand, nodes exchange routing information only when needed, with a node attempting to discover a route to some destination only when it has a packet to send to that destination (e.g., [22, 33, 35]). In addition, some ad hoc network routing protocols are hybrids of periodic and on-demand mechanisms (e.g., [12]).

Each style of ad hoc network routing protocol has advantages and disadvantages. In this paper, we focus on securing ad hoc network routing using periodic (or proactive) protocols, and in particular, using *distance vector* routing protocols. Distance vector routing protocols are easy to implement, require relatively little memory or CPU processing capacity compared to other types of routing protocols, and are widely used in networks of moderate size within the (wired) Internet [14, 27, 28]. A number of proposed periodic ad hoc network routing protocols are based on adapting the basic distance vector routing protocol design for use in mobile wireless ad hoc networks, including PRNET [23], DSDV [34], WRP [31], WIRP [10], and ADV [5]. Distance vector routing has also been used for routing within a zone in the ZRP hybrid ad hoc network routing protocol [12].

We present the design and evaluation of a new secure ad hoc network routing protocol using distance vector routing. Our protocol, which we call the *Secure Efficient Ad hoc Distance vector* routing protocol (SEAD), is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, even in spite of active attackers or compromised nodes in the network. We base the design of SEAD in part on the *Destination-Sequenced Distance-Vector* ad hoc network routing protocol (DSDV) [34], which was designed for trusted environments. In order to support use of SEAD with nodes of limited CPU processing capability, and to guard against Denial-of-Service attacks in which an attacker attempts to cause other nodes to consume excess network bandwidth or processing time, we use efficient *one-way hash functions* and do not use asymmetric cryptographic operations in the protocol.

In Section 2 of this paper, we summarize the basic operation of distance vector routing, and we describe the DSDV ad hoc network routing protocol on which we base our work. Section 3 presents our assumptions about the network and nodes involved in the ad hoc network. In Section 4, we describe possible attacks on distance vector routing protocols and specifically on DSDV routing, and in Section 5, we present the design of SEAD, our ad hoc network distance vector routing protocol that protects against those attacks. Section 6 presents the results of a simulation-based study of the performance of SEAD in ad hoc networks of 50 mobile nodes, comparing its performance to that of the original (insecure) DSDV protocol; we show the overhead created by the security mechanisms and the impact of these mechanisms on the protocol's ability to successfully route packets. In Section 7, we discuss related work, and finally, in Section 8, we present conclusions.

## 2. Distance vector routing and DSDV

A distance vector routing protocol finds shortest paths between nodes in the network through a distributed implementation of the classical Bellman-Ford algorithm. As noted

in Section 1, distance vector protocols are easy to implement and are efficient in terms of memory and CPU processing capacity required at each node. A popular example of a distance vector routing protocol is RIP [14, 28], which is widely used in IP networks of moderate size. Distance vector routing can be used for routing within an ad hoc network by having each node in the network act as a router and participate in the routing protocol.

In distance vector routing, each router maintains a routing table listing all possible destinations within the network. Each entry in a node's routing table contains the address (identity) of some destination, this node's shortest known distance (usually in number of hops) to that destination, and the address of this node's neighbor router that is the first hop on this shortest route to that destination; the distance to the destination is known as the *metric* in that table entry. When routing a packet to some destination, the node transmits the packet to the indicated neighbor router, and each router in turn uses its own routing table to forward the packet along its next hop toward the destination.

To maintain the routing tables, each node periodically transmits a routing update to each of its neighbor routers, containing the information from its own routing table. Each node uses this information advertised by its neighbors to update its own table, so that its route for each destination uses as a next hop the neighbor that advertised the smallest metric in its update for that destination; the node sets the metric in its table entry for that destination to 1 (hop) more than the metric in that neighbor's update. A common optimization to this basic procedure to spread changed routing information through the network more quickly is the use of *triggered updates*, in which a node transmits a new update about some destination as soon as the metric in its table entry for that destination changes, rather than waiting for its next scheduled periodic update to be sent.

Distance vector routing protocols are simple, but they cannot guarantee not to produce routing loops between different nodes for some destination. Such loops are eventually resolved by the protocol through many rounds of routing table updates in what is known as "counting to infinity" in the metric for this destination; to reduce time needed for this resolution, the maximum metric value allowed by the protocol is typically defined to be relatively small, such as 15 as is used in RIP [14, 28]. To further reduce these problems, a number of extensions, such as *split horizon* and *split horizon with poisoned reverse* [14, 28], are widely used. These extensions, however, can still allow some loops, and the possible problems that can create routing loops are more common in wireless and mobile networks such as ad hoc networks, due to the motion of the nodes and the possible changes in wireless propagation conditions.

The primary improvement for ad hoc networks made in DSDV over standard distance vector routing is the addition

of a *sequence number* in each routing table entry. The use of this sequence number prevents routing loops caused by updates being applied out of order; this problem may be common over multihop wireless transmission, since the routing information may spread along many different paths through the network. Each node maintains an *even* sequence number that it includes in each routing update that it sends, and each entry in a node’s routing table is tagged with the most recent sequence number it knows for that destination. When a node detects a broken link to a neighbor, the node creates a new routing update for that neighbor as a destination, with an “infinite” metric and the next *odd* sequence number after the even sequence number in its corresponding routing table entry. When a node receives a routing update, for each destination in the update, the node prefers this newly advertised route if the sequence number is greater than in the corresponding entry currently in the node’s routing table, or if the sequence numbers are equal and the new metric is lower than in the node’s current table entry for that destination; if the sequence number in the update is less than the current sequence number in the table entry, the new update for that destination is ignored.

DSDV sends both periodic routing updates and triggered updates. These updates may be either a “full dump,” listing all destinations, or an “incremental” update, listing only destinations for which the route has changed since the last full dump sent by that node. A node in DSDV chooses to send a triggered update when important routing changes should be communicated as soon as possible, although there are multiple interpretations suggested in the published description of DSDV as to which changes should cause a triggered update. One interpretation suggests that the receipt of a new metric for some destination should cause a triggered update, while the alternative interpretation suggests that the receipt of a new sequence number also should cause a triggered update. The latter interpretation has been shown to outperform the former in detailed ad hoc network simulations [6, 21] and is referred to as DSDV-SQ (for sequence number) to distinguish it from the interpretation based only on metrics.

### 3. Assumptions

As a matter of terminology in this paper, we use the acronym “MAC” to refer to the network Medium Access Control protocol at the link layer, and not to a Message Authentication Code used for authentication.

We assume that all wireless links in the network are bidirectional, since this is necessary for the distributed Bellman-Ford algorithm of distance vector routing to function correctly. Specifically, if a node  $A$ ’s wireless transmissions reach  $B$ , then  $B$ ’s transmissions would reach  $A$ . Wireless links are often bidirectional, and many MAC layers require bidirectional frame exchange to avoid collisions [20].

Network physical layer and MAC layer attacks are beyond the scope of this paper. Use of spread spectrum has been studied for securing the physical layer against jamming [40]. MAC protocols that do not employ some form of carrier sense, such as ALOHA and Slotted ALOHA [1], are less vulnerable to Denial-of-Service attacks, although they generally use the channel less efficiently.

We assume that the wireless network may drop, corrupt, duplicate, or reorder packets. We also assume that the MAC layer contains some level of redundancy to detect randomly corrupted packets; however, this mechanism is not designed to replace cryptographic authentication mechanisms.

The *network diameter* of an ad hoc network is the maximum, across all pairs of nodes in the network, of the length of the optimal route between that pair of nodes. As noted in Section 2, standard distance vector routing protocols limit the maximum metric value (and thus the maximum network diameter supported by the protocol). We also limit the maximum network diameter, and we use  $m - 1$  to denote this upper bound, such that all routes that can be used by the routing protocol are of length less than  $m$  hops. Internal to a node’s routing table, the value  $m$  can be used to denote the infinity metric in distance vector routing, although in SEAD, entries in the routing table with an infinite metric are not included in routing update messages sent by a node.

We assume that nodes in the ad hoc network may be resource constrained. Thus, in securing our distance vector ad hoc network routing protocol SEAD, we use efficient *one-way hash chains* [26] and *Merkle hash trees* [30] rather than relying on expensive asymmetric cryptographic operations. Especially on CPU-limited devices, symmetric cryptographic operations (such as block ciphers and hash functions) are three to four orders of magnitude faster than asymmetric operations (such as digital signatures).

#### 3.1. One-Way Hash Chains

A one-way hash chain is built on a one-way hash function. Like a normal hash function, a one-way hash function,  $H$ , maps an input of any length to a fixed-length bit string. Thus,  $H: \{0, 1\}^* \rightarrow \{0, 1\}^p$ , where  $p$  is the length in bits of the output of the hash function. The function  $H$  should be simple to compute yet must be computationally infeasible in general to invert. A more formal definition of one-way hash functions is provided by Goldwasser and Bellare [11], and a number of such functions have been proposed, including MD5 [44] and SHA-1 [32].

To create a one-way hash chain, a node chooses a random initial value  $x \in \{0, 1\}^p$  and computes the list of values

$$h_0, h_1, h_2, h_3, \dots, h_n$$

where  $h_0 = x$ , and  $h_i = H(h_{i-1})$  for  $0 < i \leq n$ , for some  $n$ . The node at initialization generates the elements of its hash chain as shown above, from “left to right” (in order

of increasing subscript  $i$ ) and then over time uses certain elements of the chain to secure its routing updates; in using these values, the node progresses from “right to left” (in order of decreasing subscript  $i$ ) within the generated chain.

Given an existing authenticated element of a one-way hash chain, it is possible to verify elements later in the sequence of use within the chain (further to the “left,” or in order of decreasing subscript). For example, given an authenticated  $h_i$  value, a node can authenticate  $h_{i-3}$  by computing  $H(H(H(h_{i-3})))$  and verifying that the resulting value equals  $h_i$ .

To use one-way hash chains for authentication, we assume some mechanism for a node to distribute an authentic element such as  $h_n$  from its generated hash chain. A traditional approach for this key distribution is for a trusted entity to sign public-key certificates for each node; each node can then use its public-key to sign new a hash chain element for itself. Hubaux, Buttyán, and Čapkun bootstrap trust relationships from PGP-like certificates without relying on a trusted public key infrastructure [19]. Alternatively, a trusted node can securely distribute an authenticated hash chain element using only symmetric-key cryptography [17, 39] or non-cryptographic approaches [46].

Since in SEAD, a node uses elements from its one-way hash chain in groups of  $m$  (Section 5.2), we assume that a node generates its hash chain so that  $n$  is divisible by  $m$ . When a node first enters the network, or after a node has used most of its available hash chain elements, it can pick a new random  $x$ , generate a new hash chain from this  $x$ , and send the new generated  $h_n$  value to a trusted entity or an alternative authentication and distribution service, as described above.

### 3.2. Tree-Authenticated Values

The mechanism of *tree-authenticated values* is an efficient hash tree authentication mechanism, first presented by Merkle and also known as Merkle hash trees [30]. To authenticate values  $v_0, v_1, \dots, v_{w-1}$ , we place these values at the leaf nodes of a binary tree. (For simplicity we assume a balanced binary tree, so  $w$  is a power of two.) We first blind all the  $v_i$  values with a one-way hash function  $H$  to prevent disclosing neighboring values in the authentication information (as we describe below), so  $v'_i = H[v_i]$ . We then use the Merkle hash tree construction to commit to the values  $v'_0, \dots, v'_{w-1}$ . Each internal node of the binary tree is derived from its two child nodes. Consider the derivation of some parent node  $m_p$  from its left and right child nodes  $m_l$  and  $m_r$ :  $m_p = H[m_l || m_r]$ , where  $||$  denotes concatenation. We compute the levels of the tree recursively from the leaf nodes to the root node. Figure 1 shows this construction over the eight values  $v_0, v_1, \dots, v_7$ , e.g.,  $m_{01} = H(v'_0 || v'_1)$ ,  $m_{03} = H[m_{01} || m_{23}]$ .

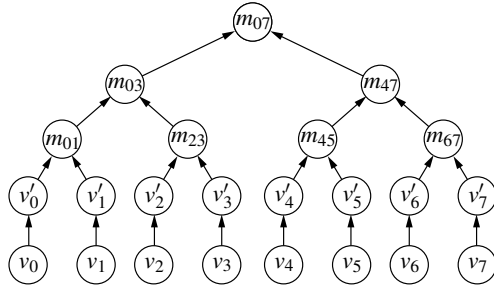


Figure 1: Tree authenticated values

The root value of the tree is used to commit to the entire tree, and in conjunction with additional information, it can be used to authenticate any leaf value. To authenticate a value  $v_i$  the sender discloses  $i$ ,  $v_i$ , and all the sibling nodes of the nodes on the path from  $v_i$  to the root node. The receiver can then use these nodes to verify the path up to the root, which authenticates the value  $v_i$ . For example, if a sender wants to authenticate key  $v_2$  in Figure 1, it includes the values  $v'_3, m_{01}, m_{47}$  in the packet. A receiver with an authentic root value  $m_{07}$  can then verify that

$$H \left[ H \left[ m_{01} || H \left[ H \left[ v_2 \right] || v'_3 \right] \right] || m_{47} \right]$$

equals the stored root value  $m_{07}$ . If the verification is successful, the receiver knows that  $v_2$  is authentic.

The extra  $v'_0, v'_1, \dots, v'_7$  in Figure 1 are added to the tree to avoid disclosing (in this example) the value  $v_3$  for the authentication of  $v_2$ .

## 4. Attacks

Kumar [25] and Smith et al [45] discuss attacks against distance vector routing protocols. In addition, in prior work we presented some attacks against ad hoc network routing protocols [17]. In this section, we summarize relevant attacks.

An attacker can attempt to reduce the amount of routing information available to other nodes, by failing to advertise certain routes or by destroying or discarding routing packets or parts of routing packets. A node failing to advertise a route indicates its unwillingness to forward packets for those destinations. We do not attempt to defend against this attack, since the attacker could also otherwise drop data packets sent to those destinations. A node can drop routing packets it receives, in which case it becomes ignorant of links available to it and fails to pass potentially improved knowledge to its neighbors. This *ignorance attack* has even more limited impact than failing to advertise routes that the node itself knows. Finally, an intruder can jam routing packets; we will disregard such attacks in this paper, since prevention of such attacks begins at the physical layer.

An attacker can modify an advertisement by changing the destination, metric, or source address (and hence next-hop). For example, an attacker advertising a zero metric for all destinations can cause all nodes around it to route packets for all destinations toward it rather than toward each actual destination. Alternatively, an attacker can modify the source address of the advertisement, thus spreading inaccurate next-hop information.

An attacker can mount a *replay attack* by sending an old advertisement to some node, in an attempt to get that node to update its routing table with stale routes.

A more subtle type of attack is the creation of a *wormhole* in the network, using a pair of attacker nodes *A* and *B* linked via a private network connection. In a wormhole, every packet that *A* receives from the ad hoc network, *A* forwards through the wormhole to *B*, to then be forwarded normally by *B*; similarly, *B* may send all ad hoc network packets to *A*. Such an attack potentially disrupts routing by short-circuiting the normal flow of routing packets, and the attackers may also create a virtual vertex cut of nodes in the network that they control. We describe the wormhole attack and solutions [38] and we give more details on the vertex cut and other attackers [17] elsewhere.

An attacker may be a compromised node. If so, it will have access to all cryptographic keys of that compromised node, and it may cooperate with other attackers or compromised nodes.

## 5. Securing distance vector routing

### 5.1. Basic design of SEAD

We base the design of our secure routing protocol SEAD on the DSDV-SQ version [6] of the insecure DSDV ad hoc network routing protocol, as described in Section 2. In particular, to avoid long-lived routing loops in SEAD, we use destination sequence numbers, as in DSDV; we also use these destination sequence numbers to provide replay protection of routing update messages in SEAD.

We differ from DSDV in that we do not use an average weighted settling time in sending triggered updates. To reduce the number of redundant triggered updates, each node in DSDV tracks, for each destination, the average time between when the node receives the *first* update for some new sequence number for that destination, and when it receives the *best* update for that sequence number for it (with the minimum metric among those received with that sequence number); when deciding to send a triggered update, each DSDV node delays any triggered update for a destination for this average weighted settling time, in the hope of only needing to send one triggered update, with the best metric, for that sequence number.

SEAD does not use such a delay, in order to prevent attacks from nodes that might maliciously not use the delay. Since a node selects the first route it receives with highest

sequence number and lowest metric, an attacker could otherwise attempt to cause more traffic to be routed through itself, by avoiding the delay in its own triggered updates. Such an attack could otherwise put the attacker in a position to eavesdrop on, modify, or discard other nodes' packets.

In addition, unlike DSDV, when a node detects that its next-hop link to some destination is broken, the node does not increment the sequence number for that destination in its routing table when it sets the metric in that entry to infinity. Since higher sequence numbers take priority, this node's routing update with this new sequence number must be authenticated, but we did not include a mechanism for authenticating these larger sequence numbers. Instead, the node flags its routing table entry for this destination to not accept any new updates for this same sequence number, effectively preventing the possible routing loop and traditional distance vector "counting to infinity" problem [14, 28] that could otherwise occur in this case.

### 5.2. Metric and sequence number authenticators

In addition to the differences between our SEAD protocol and DSDV-SQ described in Section 5.1, the lower bound on each metric in a routing update in SEAD is secured through authentication; in addition, the receiver of SEAD routing information also authenticates the sender (ensures that the routing information originates from the correct sender). We describe the authentication of the lower bound on the distance metric in this section and the neighbor authentication in the following section. Whereas DSDV-SQ (and DSDV) are subject to all of the attacks in Section 4, SEAD thus resists those attacks. SEAD is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, even in spite of active attackers or compromised nodes in the network. A description of the detailed security properties provided by the complete SEAD protocol is provided in Section 6.1.

One possible approach that could be used for authenticating routing updates in a distance vector routing protocol is for each node to sign each of its routing updates using asymmetric cryptography. However, this approach raises three distinct problems for use in an ad hoc network.

First, an attacker could send a large number of arbitrary forged routing updates to some victim node, such that the victim is forced to spend all of its CPU resources attempting to verify this stream of updates, creating an effective Denial-of-Service attack; this attack would be particularly easy in many ad hoc networks, since ad hoc network nodes tend to have less powerful CPUs than workstations in wired networks. Second, an attacker who has compromised a node can send updates claiming that any other node is a neighbor (metric 1), causing other nodes to incorrectly direct packets for this destination node toward the attacker. Finally, even with no attacker present, the larger signatures and longer

signature generation and verification times of asymmetric cryptography would reduce the resources that could otherwise be used for running useful applications and doing useful communication; this problem is more severe in an ad hoc network than in a traditional (i.e., wired and stationary) network due to the limited resources of nodes and links in an ad hoc network, such as available bandwidth, CPU capacity, and battery power (energy).

Instead, in securing routing in SEAD, we use efficient one-way hash chains [26]. The basic operation of a one-way hash chain was described in Section 3. Each node in SEAD uses a specific single next element from its hash chain in each routing update that it sends about itself (metric 0). Based on this initial element, the one-way hash chain conceptually provides authentication for the lower bound of the metric in other routing updates for this destination; the authentication provides only a lower bound on the metric, since it does not prevent a malicious node from claiming the *same* metric as the node from which it heard this route. In particular, the one-way hash function provides the property that another node can only increase a metric in a routing update, but cannot decrease it. Due to the properties of the one-way hash function, given any value in the hash chain, an attacker cannot generate any value in the chain that will be used by this node in a future update that it sends about itself (a value to the “left” of the given value in the chain, with smaller subscript). Similarly, for each entry in its routing update describing a route to another destination, the hash chain of that destination node allows the metric in that entry to be authenticated by nodes receiving it.

As noted in Section 3, we assume that an upper bound can be placed on the diameter of the ad hoc network, and we use  $m - 1$  to denote this bound. Thus, within the routing protocol, all metrics in any routing update are less than  $m$ . The method used by SEAD for authenticating an entry in a routing update uses the *sequence number* in that entry to determine a contiguous group of  $m$  elements from that destination node’s hash chain, one element of which must be used to authenticate that routing update. The particular element from this group of elements that must be used to authenticate the entry is determined by the *metric* value being sent in that entry. Specifically, if a node’s hash chain is the sequence of values

$$h_0, h_1, h_2, h_3, \dots, h_n$$

and  $n$  is divisible by  $m$ , then for a sequence number  $i$  in some routing update entry, let  $k = \frac{n}{m} - i$ . An element from the group of elements

$$h_{km}, h_{km+1}, \dots, h_{km+m-1}$$

from this hash chain is used to authenticate the entry; if the metric value for this entry is  $j$ ,  $0 \leq j < m$ , then the value  $h_{km+j}$  here is used to authenticate the routing update entry for that sequence number.

When a node in SEAD sends a routing update, the node includes one hash value with each entry in that update. If the node lists an entry for itself in that update, it sets the address in that entry to its own node address, the metric to 0, the sequence number to its own next sequence number, and the hash value to the first element in the group of its own hash chain elements corresponding to that sequence number. In the example given above for sequence number  $i$ , the node sets the hash value in that entry to its  $h_{km}$ . If the node lists an entry for some other destination in that update, it sets the address in that entry to that destination node’s address, the metric and sequence number to the values for that destination in its routing table, and the hash value to the hash of the hash value received in the routing update entry from which it learned that route to that destination.

This use of a hash value corresponding to the sequence number and metric in a routing update entry prevents any node from advertising a route to some destination claiming a greater sequence number than that destination’s own current sequence number, due to the one-way nature of the hash chain. Likewise, no node can advertise a route better than those for which it has received an advertisement, since the metric in an existing route cannot be decreased.

Nodes receiving any routing update can easily authenticate each entry in the update, given any earlier authentic hash element from the same hash chain, as described in Section 3. In order to guard against attacks in which a malicious update claiming a high sequence number attempts to force a receiving node to perform a large number of hash operations in order to authenticate the update, a receiving node may limit the number of hashes it is willing to perform for each such authentication, discarding updates that cannot be authenticated; since DSDV-SQ (and thus SEAD) spreads new routing information across the network, this limit assumes a bound on the number of routing updates about a destination that the receiving node may have missed before any authentic update is received. A similar solution to such an attack would be to have each node tie its own sequence number generation to a loosely synchronized clock value, thus allowing a receiving node to determine if a claimed sequence number in an update could be authentic before performing the implied hashes to confirm that fact.

When a node receives a routing update, for each entry in that update, the node checks the authentication on that entry, using the destination address, sequence number, and metric in the received entry, together with the latest prior authentic hash value received by this node from that destination’s hash chain. Based on the sequence number and metric in the received entry and the sequence number and metric of this latest prior authentic hash value for that destination, the node hashes the hash value received in this entry the correct number of times, according to the description above as to which hash value must be used for any given sequence num-

ber and metric, to confirm that the resulting value equals the prior authentic hash value. If so, the entry is authentic and the node processes it in the routing algorithm as a normal received routing update entry; otherwise, the node ignores the received entry and does not modify its routing table based on it.

It may be possible for an attacker to modify routing update messages in transit, and such an attacker would be able to prevent certain routes from being advertised; however, such an attacker would also be able to corrupt the entire routing update, which is equivalent to a jamming attack. The protocol can also be secured against modification of the source address for a routing update and against wormhole attacks, by use of other mechanisms at the MAC layer, including mechanisms that rely only on symmetric cryptography [38]. In particular, these MAC layer approaches authenticate the transmitting source of a packet and ensure that this transmitting source is within some distance of the receiver.

### 5.3. Neighbor authentication

The source of each routing update message in SEAD must also be authenticated, since otherwise, an attacker may be able to create routing loops. Any efficient broadcast authentication mechanism, such as TESLA [37], HORS [42], or TIK [38], can be used to authenticate the neighbor. The drawbacks of these approaches are that they require synchronized clocks, and that they incur either an authentication delay or a relatively high communication overhead.

An alternative approach that does not require time synchronization is to assume a shared secret key among each pair of nodes, and to use the respective key in conjunction with a Message Authentication Code. The sender would include one Message Authentication Code for each neighbor with each routing update. Since SEAD includes periodic neighbor sensing functionality, each node knows the set of neighbors for which it needs to authenticate routing updates. In particular, each node trusts any zero-metric update with a valid authenticator; if a node has received such an update from another node for a recent sequence number, it considers that node a neighbor and computes a Message Authentication Code for it in subsequent updates.

When two nodes first become neighbors, one of the two nodes will transmit a routing update first. That update will cause the receiving node to detect the new neighbor. As a result of hearing this update, the receiving node will send a triggered routing update, allowing the other node to detect the new neighbor.

### 5.4. Preventing Same-Distance Fraud

In Section 5.2, we authenticate the metric and sequence number with a one-way hash chain. This solution does not protect against *same-distance fraud*: that is, a node receiving an advertisement for sequence number  $s$  and distance

(metric)  $d$  can re-advertise the same sequence number  $s$  and distance  $d$ . To defend against same-distance fraud, we designed *hash tree chains*, which have properties similar to hash chains but allow the detection of same-distance fraud, when used in conjunction with packet leashes [18] to prevent an adversary from replaying a routing update in wireless networks.

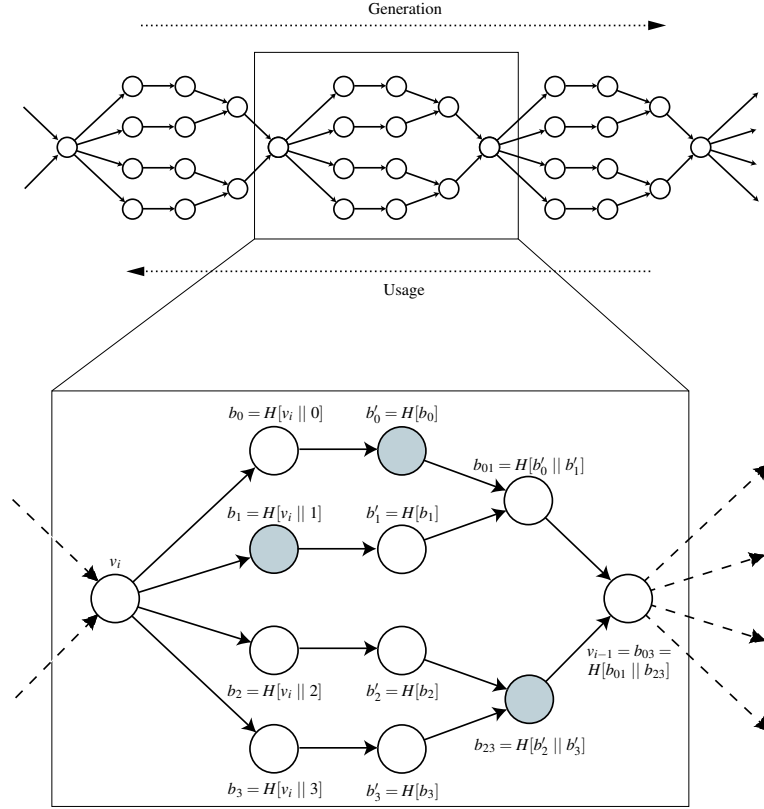
We prevent same-distance fraud by tying the authenticator to the address of the node sending a route advertisement, thus preventing an attacker from replaying an authenticator that it hears from a neighbor. We construct a special one-way chain, which we call a hash tree chain, where each element of the chain encodes the node id, thus forcing a node to increase the distance metric if it wants to encode its own id. Each step in this one-way chain contains a collection of values, one or more of which are used to authenticate any particular node. This approach is similar to that used in the HORS signature scheme [43]. These values are authenticated using a Merkle tree, and the root of that Merkle tree is used to generate the collection of values in the next step.

A hash tree chain is a hybrid between a hash tree and a one-way chain. The one-way chain property is used in the same way as in Section 5.2 (to enforce that nodes cannot decrease the distance metric), and the hash tree property is used to authenticate the node id. We construct the hash tree between each pair  $v_{i-1}, v_i$  of one-way chain values as follows. From the value  $v_i$ , we derive a set of values  $b_0, \dots, b_n$ , using a one-way hash function  $H$  as  $b_j = H[v_i || j]$ , for each  $j$ . We then build a hash tree above those values for authentication, as described in Section 3.2. The root of the tree becomes the previous value of the one-way chain  $v_{i-1} = b_{0n}$ . Figure 2 shows an example. The node with the id 1 forwards the shaded values  $b'_0, b_1$ , and  $b_{23}$  to the neighboring nodes, which can compute the one-way hash tree chain forward to verify the authenticity of values  $b'_0, b_1$ , and  $b_{23}$ , and use the value  $b_{03}$  to sign their own id when forwarding the route update, thus automatically increasing the distance metric.

We now present two examples of how the hash tree chain can be used: when a single value corresponds to a node, and when a  $\gamma$ -tuple of values corresponds to a node. For notational and analytic convenience, we describe hash tree chains for which the number of values between each hash chain value is a power of two.

In a small network, each value  $b_j$  can correspond to a single node; since no two nodes share a single value, an attacker has no way to derive its value from the advertisements of neighboring nodes, and hence it must follow the hash tree chain to the next step in order to provide a valid authenticator.

In larger networks, with  $n$  nodes, the  $O(n)$  overhead of generating each step of the chain may be too great; as a result, we authenticate each node with a  $\gamma$ -tuple of values. Although two nodes share the same  $\gamma$ -tuple of values, an



**Figure 2:** Authenticating one distance metric within a sequence of a hash tree chain. In this example, each element  $b_i$  stands for one router, so this hash tree chain supports 4 routers.

attacker could learn each of its  $\gamma$  values from different neighbors that advertise the same metric, and could then forge an advertisement without increasing the metric. We show that an attacker's probability of success may be sufficiently small. We also change the encoding of a node id for each update, so that an attacker in a static network cannot continue to forge updates once it finds an appropriate set of values from its neighbors. Consider a hash tree chain with  $2^m$  values in each step (and thus a hash tree of height  $m+1$ ). For example, if each node has a unique node id between 0 and  $\binom{2^m}{\gamma} - 1$ , then the  $\gamma$ -tuple encodes

$$x = (\text{node id} + H[\text{sequence number}]) \bmod \binom{2^m}{\gamma},$$

such that the  $\gamma$ -tuple changes for each sequence number.

### 5.5. Bounding Verification Overhead

The overhead to verify authentication values can be large if a node has missed several routing updates. In particular, an attacker can force a victim node to verify a hash chain as long as  $O(ks)$ , where  $k$  is the maximum number of hops and  $s$  is the maximum number of sequence numbers represented

by a hash chain. We can prevent this attack by using a new hash chain for each sequence number.

A node using this scheme generates a random hash chain root  $h_{0,s}$  for each sequence number  $s$ , for example by using a PRF  $\mathcal{F}$  and a secret master key  $X$  to derive  $h_{0,s} = \mathcal{F}(X, s)$ . Given the authentic anchor of this hash chain  $h_{k,s} = H^k[h_{0,s}]$  (where  $k$  is the maximum metric), any node can authenticate  $h_{m,s}$ , which is the authenticator for sequence number  $s$  and metric  $m$ .

To allow nodes to authenticate these anchors  $h_{k,s}$ , each node builds a hash tree, using the hash chain anchors as leaves (Section 3.2). When a node sends an update with a new sequence number  $s$ , it includes the root of the hash chain  $h_{0,s}$ , the anchor of the hash chain  $h_{k,s}$ , and the path to the root of the hash tree. To authenticate any update, the node verifies the anchor by following the path to the root of the hash tree. It then verifies the hash value  $h_{m,s}$  by verifying that  $h_{k,s} = H^{k-m}[h_{m,s}]$ . Since the maximum hash chain length is  $k$  and the anchor verification requires  $O(\log(s))$  effort, where  $s$  is the number of sequence numbers represented by any root, the computation required to verify any update is bounded by  $k + \log(s)$ .

**Table 1:** Parameters for SEAD performance study

Scenario Parameters	
Number of Nodes	50
Maximum Velocity ( $v_{\max}$ )	20 m/s
Dimensions of Space	1500m $\times$ 300m
Nominal Radio Range	250m
Source-Destination Pairs	20
Source Data Rate (each)	4 packets/second
Application Data Payload Size	512 bytes/packet
Total Application Data Load	327 kbps
Raw Physical Link Bandwidth	2 Mbps

SEAD Parameters	
Periodic Route Update Interval	15 seconds
Periodic Updates Missed before Link is Declared Broken	3
Maximum Packets Buffered per Node per Destination	5
Hash Length ( $p$ )	80 bits

## 6. Evaluation

### 6.1. Security analysis

Securing a distance vector protocol seems fundamentally harder than securing link-state or on-demand protocols such as DSR [22]. Since distance vector protocols compress the route information into a hop count value and a next hop, it is challenging to verify the correctness of the hop count value. In this section, we discuss some of the security properties of the SEAD protocol.

Using SEAD, any attacker cannot create a valid advertisement with larger (better) sequence number than it received. Furthermore, for advertisements sent using the largest received sequence number, attackers that do not collude cannot advertise a route shorter than the one it has heard. For example, if the best metric  $m$  received by a node at the current sequence number  $s$ , the attacker cannot advertise a better metric than  $m$ . When hash tree chains are used (as described in Section 5.4), SEAD achieves even stronger properties:

- If each node corresponds to a single hash tree chain value ( $\gamma = 1$ ), the attacker is forced to advertise metric at best  $m + 1$
- Otherwise, the attacker is forced to advertise metric at best  $m + 1$  with high probability, and otherwise cannot advertise with metric better than  $m$ .

We now explore the probability of successful metric replay using the hash tree chain scheme when each node corresponds to a set of hash tree chain values ( $\gamma \neq 1$ ). Let  $A_i$  be the set of combinations of nodes that do not include value  $b_i$  needed by the attacker. The attacker, then, has  $|\cup_{i=1}^{\gamma} A_i|$  ways to fail. We now apply the inclusion-exclusion principle:

$$\begin{aligned} \left| \bigcup_{i=1}^{\gamma} A_i \right| &= \sum_i |A_i| - \sum_{i_1, i_2} |A_{i_1} \cap A_{i_2}| + \dots + (-1)^{\gamma+1} \left| \bigcap_{i=1}^{\gamma} A_i \right| \\ &= \sum_{i=1}^{\gamma} (-1)^{i+1} \binom{\gamma}{i} \binom{2^m - i}{q} \end{aligned}$$

Then the probability of a successful defense is

$$\frac{\sum_{i=1}^{\gamma} (-1)^{i+1} \binom{\gamma}{i} \binom{2^m - i}{q}}{\binom{2^m}{q} - 1}$$

This probability can be quite high; for example, when  $m = 6$ ,  $\gamma = 3$ , and  $q = 3$  as before, an attacker has a  $1.675 \times 10^{-3}$  probability of success; when three consecutive advertisements are required for the same metric before a routing change is made, the attacker succeeds once every 6.74 years.

An attacker that has not compromised any node (and hence does not possess any cryptographic keys from a node), cannot successfully send any routing messages, since an uncompromised neighbor node will reject the messages due to the failed neighbor authentication. A repeater can function as a one-node wormhole; this is not addressed by SEAD, though TIK [38] can prevent this attack.

A collection of a number of attackers that have compromised one or more nodes can only redirect the path from a source to a destination through one or more attackers if the length of the best (minimum metric) attacker-free route for which the source receives an advertisement is at least as large as the number of nodes between the destination and the first attacker, plus the number of nodes between the last attacker and the destination.

If each node using SEAD (including attackers) keeps routing tables where the next-hop for a given destination is set to the authenticated source address of the first advertisement received by that node containing the minimum metric for the greatest sequence number, then the next-hop pointers in all nodes' routing tables will describe a route back to the destination.

With SEAD, no routing loop is possible, unless the loop contains one or more attackers. Furthermore, no loop is possible unless no non-attacker node on the loop has received a better advertisement (in terms of sequence number and metric) for this destination than the best advertisement received by some attacker on the loop.

If a collection of attackers form a vertex cut between two groups of nodes in the network [17], the attackers can arbi-

trarily control the routes between any node in one group and a node in the other group. Since in a vertex cut, any packet between such nodes must physically pass through a node on the vertex cut, no routing protocol can eliminate such attacks.

## 6.2. Simulation evaluation methodology

To evaluate the performance impact of our security approach in SEAD without attackers, we modified the DSDV-SQ implementation in our extensions to *ns-2* [6]. Specifically, we increased the size of each routing update to represent the authentication hash value in each table entry. We also removed the settling time and the sequence number changes, as described in Section 5.1. We did not simulate the mechanisms in Sections 5.4 because they provide minimal protection without the use of packet leashes, and packet leashes provide no-cost packet authentication. Because we wanted to determine the cost of SEAD without significant additional assumptions, we simulated pairwise shared key authentication. We also did not simulate the mechanisms in Section 5.5, because such precautions are not always necessary. For example, if nodes are loosely time synchronized, an upper bound on the maximum sequence number can be easily determined. Alternatively, intrusion detection techniques can be used to avoid the need to authenticate many bogus updates. In particular, a node can check the neighbor authentication very easily. If certain neighbors persist in sending updates with bogus metric authenticators, those neighbors can be ignored, or the verification of their updates can be relegated to a lower priority.

We chose the *ns-2* simulator for this study because it realistically models arbitrary node mobility as well as physical radio propagation effects such as signal strength, interference, capture effect, and wireless propagation delay. Our propagation model is based on the *two-ray ground reflection* model [41]. The simulator also includes an accurate model of the IEEE 802.11 Distributed Coordination Function (DCF) wireless MAC protocol [20].

In our simulations, nodes moved according to the *random waypoint* mobility model [22]. Each node is initially placed at a random location and pauses for a period of time called the *pause time*; it then chooses a new location at random and moves there with a velocity randomly chosen uniformly between 0 and the maximum speed  $v_{\max}$ . When it arrives, it repeats the process of pausing and then selecting a new destination to which to move. The data communication pattern in our study uses 20 source-destination pairs, each sending a Constant Bit Rate (CBR) flow of 4 data packets per second. Each data packet is 512 bytes in size. Table 1 details the parameters used in our simulations.

We evaluated SEAD by comparing it to DSDV-SQ, as described in Section 2. We measured performance along four metrics:

- *Packet Delivery Ratio*: The total over all nodes of the number of application-level packets received, divided by the total number of application-level packets originated.
- *Byte Overhead*: The total over all hops of the number of overhead bytes transmitted.
- *Packet Overhead*: The total over all hops of the number of overhead packets transmitted.
- *Median Latency*: The median packet delivery latency, where latency is calculated as the elapsed time between the application layer passing a packet to the routing layer and that packet first being received at the destination.

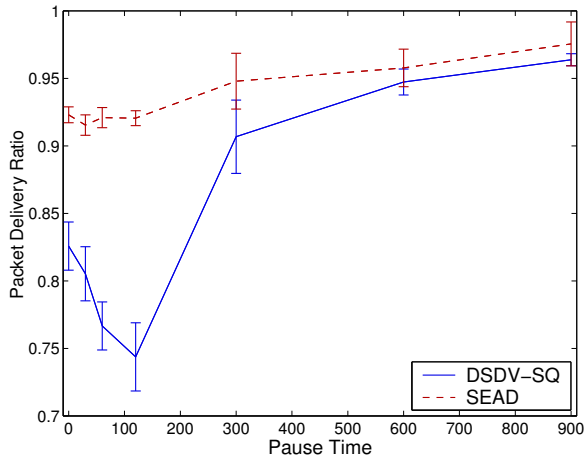
## 6.3. Simulation results

The results of our performance study of SEAD are shown in Figure 3 as a function of pause time in the random waypoint mobility model. Each figure represents the average over 65 randomly generated runs at each pause time, and the error bars show the 95% confidence intervals; the runs used for SEAD and those for DSDV-SQ were identical. On the right side of each graph (pause time 900), the nodes are stationary, and on the left side of each graph (pause time 0), the nodes are all in continuous motion.

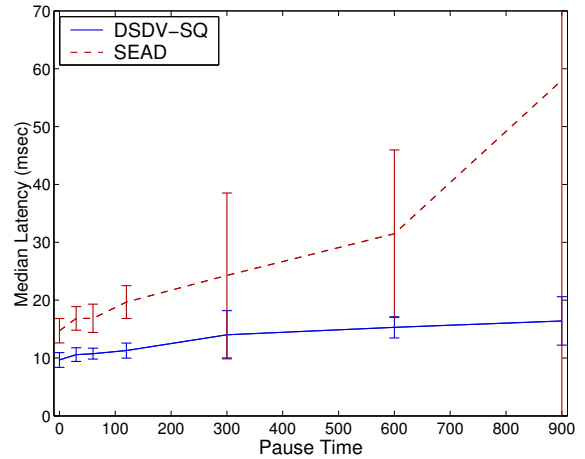
The packet delivery ratios for SEAD and DSDV-SQ are shown in Figure 3(a), and the median latency of delivered application-level packets for these simulations is shown in Figure 3(b). Surprisingly, SEAD consistently outperforms DSDV-SQ in terms of packet delivery ratio. By not using a weighted settling time delay in sending triggered updates in SEAD, the number of routing advertisements sent by SEAD generally increases relative to DSDV-SQ, allowing nodes to have more up-to-date routing tables.

However, SEAD also increases overhead, both due to this increased number of routing advertisements, and due to the increase in size of each advertisement from the addition of the hash value on each entry for authentication. This increased overhead is shown in Figures 3(c) and 3(d), which show the number of routing overhead packets and the number of routing overhead bytes, respectively, caused by the two protocols in these same simulations. The vertical scale in Figure 3(c) is magnified to show the difference between the two protocols; the vertical scale here ranges only between 40 and 46.

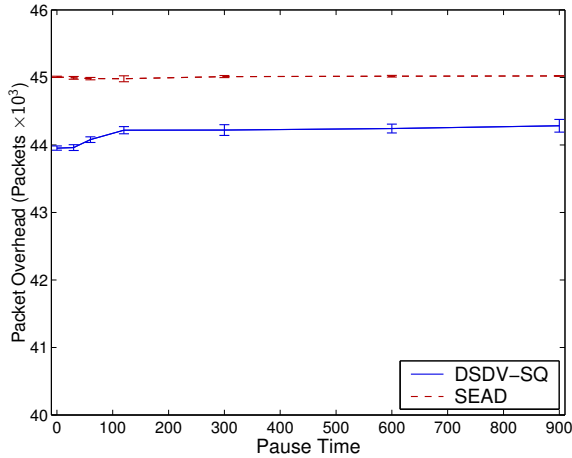
The increased overhead in SEAD causes some congestion in the network in these simulations, as shown in the latency results in Figure 3(b). At all pause times, SEAD exhibits higher latency than DSDV-SQ, due to the decreased available network capacity from the increased overhead in SEAD. The rise in latency at higher pause times is due to the nonuniform distribution of nodes in space caused by node motion in the random waypoint model. Although the initial node locations and the locations to which each node moves



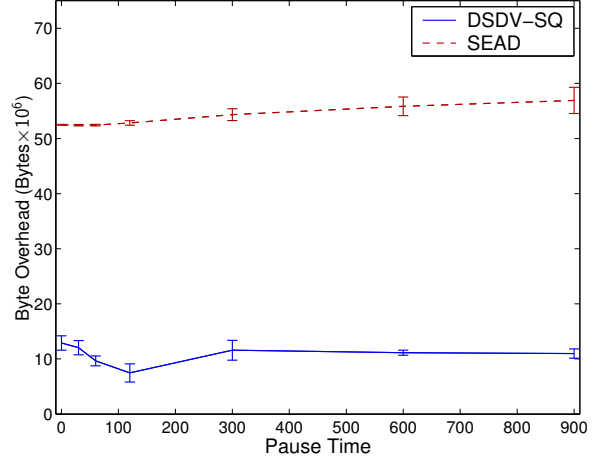
(a) Packet Delivery Ratio



(b) Median Latency



(c) Packet Overhead



(d) Byte Overhead

Figure 3: SEAD performance evaluation results (average over 65 runs)

during the run are uniformly chosen over the space, the straight line path of a node from one location to the next tends to distribute nodes on average closer to the center of the space; at higher pause times, nodes spend most (or all) of the time in their initial uniformly distributed locations. For example over the 65 simulation runs, the average route length used by SEAD at pause time 900 was about 28% longer than at pause time 0 (for DSDV-SQ, the average route length at pause time 900 was about 33% longer than at pause time 0). This increased route length, together with SEAD’s increased overhead, created additional congestion at higher pause times in the simulations.

## 7. Related work

Kumar [25] discusses attacks against distance vector routing protocols, and describes mechanisms to secure them using Message Authentication Codes. Although these

mechanisms ensure the integrity of router-to-router communications, they do not withstand node compromise. In particular, they do not secure the metric in each routing table entry, and thus a compromised router could claim routes of any length to any destination.

Smith et al [45] discuss attacks against distance vector routing protocols, and present countermeasures that provide security. However, their techniques do not apply well in an ad hoc network since they require knowledge of which links are possible, whereas in an ad hoc network, any pair of nodes could be within range and form a link.

Zapata [47] proposes SAODV, which uses a new one-way hash chain for each Route Discovery to secure the metric field in an RREQ packet. Zapata’s protocol differs in two ways. First, it uses a digital signature to authenticate the anchor of each such chain, which is significantly more expensive than the use of a single hash chain (Section 5.1) or

the use of Merkle hash trees (Section 5.5). Second, SAODV operates on-demand, which results in somewhat different assumptions in areas such as neighbor authentication.

A number of security protocols have been designed for RIPv2 [2, 28]. These protocols protect the integrity of the packet from modification, but they do not prevent a node from advertising a route that does not actually exist.

Several researchers have proposed the use of *asymmetric cryptography* to secure both wired and ad hoc network routing protocols [9, 24, 36, 47, 48]. However, when the nodes in an ad hoc network are unable to verify asymmetric signatures quickly enough, these protocols may not be suitable and may create Denial-of-Service (DoS) attacks; these protocols also generally require more network bandwidth than does SEAD with its hash values.

Cheung [7] and Hauser et al [13] describe *symmetric-key* approaches to the authentication of updates in link state protocols, but neither work discusses the mechanisms for detecting the status of these links. In wired networks, a common technique for authenticating HELLO packets is to verify that the incoming network interface is the expected interface and that the IP TTL of the packet is 255. In a wireless network, this technique cannot be used. Heffernan [15] and Basagni et al [3] use shared keys to secure routing communication, which is vulnerable to some single-node compromises. Perrig et al [39] use symmetric primitives to secure routing only between nodes and a trusted base station.

As mentioned in Section 3, some researchers have explored the establishment of trust relationships and authenticated keys in ad hoc networks [17, 19, 39, 46].

Marti et al [29] consider the problem of detecting intermediate nodes that do not forward packets. However, their scheme is limited to certain types of network Medium Access Control layers and may trigger false alarms in congested networks.

In other work, we have designed a secure *on-demand* routing protocol for ad hoc networks, called Ariadne [17]. The mechanisms we used for security in Ariadne are end-to-end in nature, whereas our approach here for SEAD operates on a hop-by-hop basis due to the basic operation of distance vector routing. Furthermore, unlike Ariadne, the techniques presented here do not rely on a Message Authentication Code to authenticate routing table entries, but instead directly use elements from a one-way hash chain to provide authentication for both the sequence number and the metric in each entry. An earlier version of SEAD appeared as [16].

## 8. Conclusions and future work

In this paper, we have presented the design and evaluation of SEAD, a new secure ad hoc network routing protocol using distance vector routing. Many previous routing protocols for ad hoc networks have been based on distance vector approaches (e.g., [5, 10, 12, 23, 31, 34]), but they have

generally assumed a trusted environment. Instead, in designing SEAD, we carefully fit inexpensive cryptographic primitives to each part of the protocol functionality to create an efficient, practical protocol that is robust against multiple uncoordinated attackers creating incorrect routing state in any other node, even in spite of active attackers or compromised nodes in the network. Together with existing approaches for securing the physical layer and MAC layer within the network protocol stack, the SEAD protocol provides a foundation for the secure operation of an ad hoc network.

We base the design of SEAD in part on the DSDV ad hoc network routing protocol [34], and in particular, on the DSDV-SQ version of the protocol, which has been shown to outperform other DSDV versions in previous detailed ad hoc network simulations [6, 21]. For security, we use efficient one-way hash functions and do not use asymmetric cryptographic primitives. Consequently, SEAD is efficient and can be used in networks of computation- and bandwidth-constrained nodes. SEAD actually outperforms DSDV-SQ in terms of packet delivery ratio, although it does create more overhead in the network, both due to an increased number of routing advertisements it sends, and due to the increase in size of each advertisement due to the addition of the hash value on each entry for authentication.

In future work, we plan to also consider mechanisms to detect and expose nodes that advertise routes but do not forward packets, and to merge this work with our other work in securing on-demand routing protocols to create a secure protocol based on ZRP [12]. We are also considering the possibility of extending DSDV to behave like a path-vector routing protocol, allowing the source address of each advertisement to be more readily authenticated.

## References

- [1] Norman Abramson. The ALOHA System—Another Alternative for Computer Communications. In *Proceedings of the Fall 1970 AFIPS Computer Conference*, pages 281–285, November 1970.
- [2] Fred Baker and Randall Atkinson. RIP-2 MD5 Authentication. RFC 2082, January 1997.
- [3] Stefano Basagni, Kris Herrin, Emilia Rosti, and Danilo Bruschi. Secure Pebblenets. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 156–163, Long Beach, California, USA, October 2001.
- [4] Bhargav Bellur and Richard G. Ogier. A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99)*, pages 178–186, March 1999.
- [5] Rajendra V. Boppana and Satyadeva Konduru. An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks. In *Proceedings of the Twentieth Annual Joint*

- Conference of the IEEE Computer and communications Societies (INFOCOM 2001)*, pages 1753–1762, 2001.
- [6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pages 85–97, October 1998.
- [7] Steven Cheung. An Efficient Message Authentication Scheme for Link State Routing. In *13th Annual Computer Security Applications Conference*, 1997.
- [8] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol. Internet-Draft, draft-ietf-manet-olsr-05.txt, October 2001. Work in progress.
- [9] Bridget Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Shields. A Secure Routing Protocol for Ad Hoc Networks. Technical Report 01-37, Department of Computer Science, University of Massachusetts, August 2001.
- [10] J.J. Garcia-Luna-Aceves, Chane L. Fullmer, Ewerton Madruga, David Beyer, and Thane Frivold. Wireless Internet Gateways (WINGS). In *Proceedings of IEEE MILCOM '97*, pages 1271–1276, November 1997.
- [11] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. Summer Course “Cryptography and Computer Security” at MIT, 1996–1999, August 1999.
- [12] Zygmunt J. Haas. A Routing Protocol for the Reconfigurable Wireless Network. In *1997 IEEE 6th International Conference on Universal Personal Communications Record: Bridging the Way to the 21st Century (ICUPC '97)*, volume 2, pages 562–566, October 1997.
- [13] Ralf Hauser, Antoni Przygienda, and Gene Tsudik. Reducing the Cost of Security in Link State Routing. In *Symposium on Network and Distributed Systems Security (NDSS'97)*, pages 93–99, February 1997.
- [14] C. Hedrick. Routing Information Protocol. RFC 1058, November 1988.
- [15] Andy Heffernan. Protection of BGP Sessions via the TCP MD5 Signature Option. RFC 2385, August 1998.
- [16] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 3–13, June 2002.
- [17] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Wireless Ad Hoc Networks. Technical Report TR01-383, Department of Computer Science, Rice University, December 2001.
- [18] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of IEEE Infocomm 2003*, April 2003.
- [19] Jean-Pierre Hubaux, Levente Buttyán, and Srdjan Čapkun. The Quest for Security in Mobile Ad Hoc Networks. In *Proceedings of the Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA, USA, October 2001.
- [20] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.
- [21] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 195–206, August 1999.
- [22] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.
- [23] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.
- [24] Stephen Kent, Charles Lynn, Joanne Mikkelsen, and Karen Seo. Secure Border Gateway Protocol (S-BGP) — Real World Performance and Deployment Issues. In *Symposium on Network and Distributed Systems Security (NDSS'00)*, pages 103–116, February 2000.
- [25] Brijesh Kumar. Integration of Security in Network Routing Protocols. *SIGSAC Review*, 11(2):18–25, 1993.
- [26] Leslie Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24(11):770–772, November 1981.
- [27] Gary Scott Malkin. RIP Version 2 Protocol Applicability Statement. RFC 1722, November 1994.
- [28] Gary Scott Malkin. RIP Version 2. RFC 2453, November 1998.
- [29] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 255–265, Boston MA, USA, August 2000.
- [30] Ralph Merkle. Protocols for Public Key Cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, 1980.
- [31] Shree Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.
- [32] National Institute of Standards and Technology (NIST). Secure Hash Standard, May 1993. Federal Information Processing Standards (FIPS) Publication 180-1.
- [33] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of INFOCOM '97*, pages 1405–1413, April 1997.
- [34] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August

1994. A revised version of the paper is available from <http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps>.
- [35] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, February 1999.
  - [36] Radia Perlman. *Interconnections: Bridges and Routers*. Addison-Wesley, 1992.
  - [37] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Network and Distributed System Security Symposium (NDSS'01)*, February 2001.
  - [38] Adrian Perrig, Yih-Chun Hu, and David B. Johnson. Wormhole Protection in Wireless Ad Hoc Networks. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.
  - [39] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, Rome, Italy, July 2001.
  - [40] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of Spread Spectrum Communications — A Tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.
  - [41] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, New Jersey, 1996.
  - [42] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short One-time Signatures with Fast Signing and Verifying. Cryptology ePrint Archive, Report 2002/014, 2002. Available at <http://eprint.iacr.org/>.
  - [43] Leonid Reyzin and Natan Reyzin. Better than Biba: Short One-Time Signatures with Fast Signing and Verifying. In *Information Security and Privacy — 7th Australasian Conference (ACSIP 2002)*, edited by Jennifer Seberry, number 2384 in Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, July 2002.
  - [44] Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.
  - [45] Bradley R. Smith, Shree Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance Vector Routing Protocols. In *Symposium on Network and Distributed Systems Security (NDSS'97)*, February 1997.
  - [46] Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop*, edited by B. Christianson, B. Crispo, and M. Roe. Springer Verlag Berlin Heidelberg, 1999.
  - [47] Manel Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector (SAODV) Routing. IETF MANET Mailing List, Message-ID 3BC17B40.BBF52E09@nokia.com, <ftp://manet.itd.navy.mil/pub/manet/2001-10.mail>, October 8, 2001.
  - [48] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6), November/December 1999.