SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. 2009; **2**:325–340
Published online 29 September 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.60

SEAR: a secure efficient *ad hoc* on demand routing protocol for wireless networks

Qing Li[1*,†], Meiyuan Zhao[2], Jesse Walker[3], Yih-Chun Hu[4], Adrian Perrig[5] and Wade Trappe[1]
[1]*WinLab, Rutgers University, 671 Route South, North Brunswick, NJ 08902, U.S.A.*
[2]*Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95054*
[3]*Intel Corporation, 2111 N.E. 25th Avenue, Hillsboro, OR 97214*
[4]*Department of Electrical and Computing Engineering, University of Illinois at Urbana Champaign, 1308 West Main Street, Urbana, Illinois 61801*
[5]*Departments of Electrical and Computing Engineering, Carnegie, Mellon University, CIC 2107, 4720 Forbes Avenue, Pittsburgh, PA 15213*

Summary

Multi-hop routing is essential to the operation of wireless *ad hoc* networks. Unfortunately, it is very easy for an adversary to forge or modify routing messages to inflict severe damage on the underlying routing protocol. In this paper, we present SEAR, a secure efficient *ad hoc* routing (SEAR) protocol for *ad hoc* networks that is mainly based on efficient symmetric cryptography, with asymmetric cryptography used only for the distribution of initial key commitments. SEAR uses one-way hash functions to protect the propagation of the routing messages. Intermediate nodes verify the routing messages by applying one-way functions, while malicious nodes cannot construct beneficial false routing messages when forwarding them. Route error (RERR) messages are protected through a variation of the TESLA broadcast authentication scheme. The SEAR protocol does not require any additional routing packet formats, and thus follows the same basic design as *ad hoc* on-demand distance vector (AODV). We show, through both theoretical examination and simulations, that SEAR provides better security with significantly less overhead than other existing secure AODV (SAODV) protocols. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: *ad hoc* networks; authenticator; one-way hash function; SEAR; secure routing

1. Introduction

New paradigms for wireless networking are gaining popularity as a low-cost solution to providing extended wireless connectivity. Mobile *ad hoc* networks (MANETs) and mesh-style networks can be built using affordable wireless hardware (e.g., 802.11) and, due to their promise of high-bandwidth at low cost, are an attractive alternative to traditional cellular networks. In fact, recently, such networks have started to make their way out of the laboratory and into field tests, as has been evidenced by recent deployments where mesh networks are being deployed to support municipal (such as police officers and city services) and public communication needs [1,2].

In *ad hoc* networks, each node performs multihop routing to deliver data across a geographically broad area. Assuring the reliability of routing protocols is

*Correspondence to: Qing Li, WinLab, Rutgers University, 671 Route South, North Brunswick, NJ 08902, U.S.A.
†E-mail: qingli@winlab.rutgers.edu

made significantly more challenging when security issues are considered as the wireless medium makes it easy for an adversary to cause severe damage to the underlying routing protocol. Routing disruption attacks, which aim to prevent packets from being routed through appropriate paths, and resource consumption attacks that exhaust available resources (e.g., bandwidth and storage), are easily performed by forging or modifying routing packets [3]. In fact, by clever forging or modification of routing packets, an attacker can prevent data from reaching its destination, and cause data packets to be routed through target nodes specified by the attacker for the purpose of eavesdropping, alteration, or packet dropping.

Although there are numerous routing protocols that exist for MANETs and mesh networks, ranging from dynamic source routing (DSR) [4] to destination sequenced distance vector (DSDV) [5], the ad hoc on-demand distance vector routing protocol (AODV) [6] is one of the more popular routing algorithms for MANETs and mesh networks. Due to its scalability, rapid convergence, and ability to quickly respond to link dynamics, AODV is being considered by various standards efforts, such as the IETF [6], and the 802.11*s* standard for ESS mesh networking [7]. Unfortunately, providing a secure and trustworthy version of AODV has been elusive. Current secure AODV (SAODV) protocols, such as SAODV [8] and authenticated routing for *ad hoc* network (ARAN) [9], have significant security weaknesses. For instance, both SAODV and ARAN provide message authenticity only when all intermediate nodes are trustworthy, which is an overly restrictive assumption.

Unlike source routing schemes, such as DSR, that can be secured by authenticating the intermediate nodes appended fields in the routing messages (*cf.* Ariadne [10]), the nature of the AODV protocol has made it challenging to verify routing messages without knowledge of other nodes on the path. In AODV, the optimum routes are selected by choosing the path with the largest sequence number and smallest hop counts. Therefore, the adversaries can easily achieve the routing disruption goal by manipulating the sequence number and hop count fields in the routing messages in such a way that these false routing messages appear to be a better path to reach the destination.

In this paper, we present a SAODV protocol, which we call the secure efficient *ad hoc* routing (SEAR) protocol, that uses one-way hash functions to construct authenticators associated with each node. Intermediate nodes verify the authenticators by the simple application of efficient one-way hash functions, yet are not able to increase sequence numbers or decrease hop counts when forwarding messages. Route error (RERR) messages are protected through a variation of the broadcast authentication scheme TESLA [11]. SEAR is the first SAODV protocol to provide comprehensive solutions to securing both sequence numbering and hop counts simultaneously. Further, SEAR mainly involves highly efficient symmetric cryptography and requires asymmetric cryptography only in the initial bootstrap phase. Compared to existing SAODV routing protocols, which use intensive public key cryptographic operations, SEAR provides better security with significantly less overhead. Additionally, while designing SEAR, we have sought to introduce minimal changes to the AODV specification that is being considered by the standards community [6,7].

The rest of the paper is organized as follows. In Section 2, we place SEAR in context by summarizing other efforts that have focused on securing ad hoc routing. In Section 3, we briefly overview AODV and identify major threats we faced when trying to SAODV. We then introduce the SEAR protocol in Section 4. In Section 5, we provide a performance analysis of the SEAR protocol and compare its security with several other existing SAODV protocols. Finally, Section 6 concludes the paper.

2. Related Work

Secure routing in *ad hoc* networks has become an increasingly important topic, and many routing protocols have been proposed to secure ad hoc networks under different attack models [10,12–18]. A brief summary of several notable works follows. Hu *et al.* proposed the secure efficient ad hoc distance vector routing protocol (SEAD) [12], which is based on the DSDV [5] routing protocol. Acs *et al.* developed a mathematical framework for securing source routing in wireless *ad hoc* networks [19]. They also presented endairA routing protocol based on the proposed mathematical framework. Papadimitratos *et al.* proposed the secure routing protocol (SRP) [13] which can be used with DSR. An on-demand SRP, known as Ariadne [10], was also proposed to secure DSR. Yi *et al.* proposed security-aware *ad hoc* routing (SAR) that incorporates security attributes as parameters into ad hoc route discovery [20]. Awerbuch *et al.* proposed an on demand SRP which is resilient to byzantine failures [21]. Zhou *et al.* made use of threshold cryptography and the redundancy multiple routes between nodes to defend routing against denial of service attacks [22]. Other protocols employed intrusion detection systems or reputation systems to secure routing [17,18]. Marti *et al.* used a watchdog that identifies misbehaving nodes and avoids routing through these nodes to improve throughput [15].

There are two SRPs proposed to address the security vulnerabilities in AODV algorithms, SAODV [8], and ARAN [9]. Zapata *et al.* proposed secure AODV (SAODV) [8] to protect routing messages exchanged in AODV. Two mechanisms were incorporated into AODV to secure routing messages: digital signatures to authenticate non-mutable fields and hash chains to secure mutable fields. SAODV assumes that each node has a pair of public/private keys, and has a way to obtain public keys from other nodes in the network. Two versions of the signature schemes, referred to as the single signature scheme and the double signature scheme, were proposed to protect non-mutable fields in routing messages. Each node that receives routing messages will first check the validity of the signature before further processing the messages. SAODV uses hash chains to secure the hop count in route request (RREQ) and route reply (RREP) messages in such a way that it allows any node who receives the message to verify that the hop count has not been decremented by any attackers.

Sanzgiri *et al.* designed another SAODV algorithm, ARAN [9]. Similar to SAODV, each node has a certificate signed by a trusted certificate authority. ARAN changed some of the basic design features of AODV in order to achieve security. In particular, in ARAN the optimal path is selected based on first received RREQ, rather than based on the sequence number and hop count as is specified in AODV. Further, ARAN uses a nonce plus a time stamp to assure the freshness of a route, rather than employing sequence numbers for this purpose as in AODV. These nonce and time stamps stay unchanged throughout the propagation of the routing messages. ARAN achieves security through the usage of signatures on a hop-by-hop basis. When a node initiates a RREQ or RREP, it includes its certificate in the message and signs the packet. Intermediate nodes first check the validity of the signature of the previous hop, and then replace the signature of the previous hop with its own signature.

3. Threat Analysis for AODV

As a basis for the discussion in the paper, we begin with an overview of the AODV routing protocol, and highlight several challenges that we faced when trying to secure AODV.

3.1. Overview of AODV

AODV is an on-demand routing algorithm for mobile *ad hoc* wireless networks that only maintains routes between communicating nodes [6]. Unlike DSR, the routing messages do not include the whole routing path. AODV relies on intermediate nodes to construct and maintain routing information for communicating nodes. Thus, the size of each routing message remains constant as the size of the network increases and the number of hops increases. In order to prevent the formation of routing loops, AODV uses an approach similar to that used in DSDV [5], where each node maintains its own sequence number. In its routing table, every node maintains its best knowledge of the latest destination sequence number of each destination. Nodes update their routing entries for a destination only if they receive a routing message with a higher sequence number or the same sequence number with smaller hop count for that destination.

When an originator needs to communicate with a destination, it will first check whether it has an active route to the destination. If an active route exists, then the node will use this route for communication. Otherwise, the node will initiate a route discovery process by broadcasting a RREQ message, which contains the originator's best knowledge of the destination's sequence number, to its neighbors. All nodes that receive the RREQ will set up a reverse path to the originator. When the destination (or intermediate node who has an active route to the destination with a higher or equal destination sequence number) receives the RREQ, it sends a unicast reply, known as the RREP, to the RREQ originator. The intermediate nodes set the forward path to the destination as the RREP travels back to the originator. Once a node detects a link failure for the next hop of an active route while transmitting data, or receives a data packet for an inactive route or unknown destination, or receives a RERR from a neighbor for one or more active routes, it will broadcast a RERR to all affected neighbors.

3.2. Threat Analysis for AODV

In general, attacks on wireless networks are either passive or active. Passive attacks involve attackers monitoring and listening to the traffic transmitted in wireless networks. Passive attacks do not inflict significant damage, however, passive attackers could gather useful information needed to launch powerful active attacks. In contrast, in an active attack, the attacker can actively inject faked packets, modify traffic or drop received data in the network. As such, active attacks are targeted at disrupting the normal operation of the network. In this paper, we focus on active attacks targeted at the disruption of the normal operation of the AODV routing protocol.

AODV does not define any security mechanisms. Instead, it assumes that all nodes are trustworthy and faithfully execute the protocol. This is a dubious assumption, as there is no assurance that all of the nodes are controlled by an honest service provider with incentives to correctly provide AODV. This makes a network based upon the protocol vulnerable to numerous threats [23]. These threats include AODV message forgery and modification attacks. AODV also makes it difficult to assign any accountability for the information conveyed by its routing messages. Other vulnerabilities that are not specific to AODV and can be combated through generic techniques. In this section, we first discuss vulnerabilities that are specific to AODV, then identify other threats that are generic to wireless *ad hoc* routing protocols.

Outright forgeries are the most obvious threat against AODV. Since AODV defines no message authenticity mechanisms, an adversary can forge AODV RREQs, RREPs, or RERRs on behalf of other nodes. This is a serious flaw. It is possible to define end-to-end authentication schemes based on message authentication codes (MACs) to detect such forgeries, but these schemes would not allow intermediate nodes to verify message authenticity prior to forwarding, so a forgery can be used to configure invalid routes to legitimate destinations, even though the route would be rejected by the message's ultimate target. Furthermore, to be useful, a MAC would interfere with functionality the protocol requires of intermediate nodes, namely incrementing the hop count.

A related problem is that an intermediate node can alter a legitimate AODV message. For instance, attackers can affect the routing by changing the AODV hop count and sequence numbers since, in AODV, the optimal routes are selected based on the largest sequence numbers and smallest hop counts.

Since nodes select the shortest path based on the hop count in the RREQ and RREP messages, an adversary can fail to increment the hop count to increase its chance of being selected as the shortest path. Figure 1 depicts an example of such an attack. In Figure 1, two paths exist from S to D, S → A → B → D and S → M → C → E → D. If all nodes are benign, the

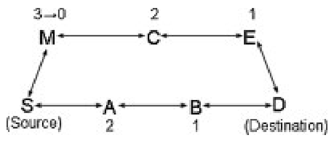

Fig. 1. An attacker can launch a decrease hop count attack against the AODV routing protocol. Source (S) has two neighbors, A and M. S should choose A as its next hop to destination (D) in normal AODV operation. If M is malicious and decreases hop count to 0 as it forward RREP, S will choose M as its next hop to D.

path S → A → B → D will be selected. But if M is malicious, and changes the hop count to 0 as it forward the RREP to S, S will select M as its next hop to D and thus the path S → M → C → E → D will be picked instead.

Similarly, an adversary can increase the sequence numbers in RREQ, RREP and RERR messages, as the protocol requires a node to consider a route with a higher sequence number as newer. This analysis implies that intermediate nodes should not be able to increase the sequence number or decrease the hop count as they forward routing messages, and AODV's failure to prevent these behaviors is a serious vulnerability.

Another form of attack is the change of address attack, whereby the adversary can change the addresses in RREQs and RREPs. The change of address attack may result in the suppression of valid routing messages. As a result, nodes could fail to receive the correct routing messages destined for them.

The proposed SEAR protocol addresses all above attacks against wireless *ad hoc* networks. The traditional philosophy to protect the modification of the routing messages is to apply authentication to prevent the alternation of all fields included in the routing messages. In contrast, SEAR allows for the alteration of certain fields in routing messages, but it guarantees that such an alteration will not result in any benefit to an adversary.

Finally, it is worth noting that there are some threats that are not specific to AODV, but apply more generally to any *ad hoc* routing protocol. *Ad hoc* wireless networks often consist of resource-constrained nodes. Attackers can inject large amounts of bogus traffic into the network in an attempt to consume one or more nodes' valuable resources, such as bandwidth, computational capacity, and storage. Further, conspiring adversaries at opposite ends of a network can launch a wormhole attack by advertising a single hop route between themselves and then tunneling traffic between them. For these types of attacks, we note that defense mechanisms complementary to secure routing methods are needed.

For example, we will not consider wormhole attacks further in this paper, but note that techniques like packet leashes [16] can be used in conjunction with SEAR to address this specific problem.

4. SEAR: The Secure Efficient *Ad hoc* Routing Protocol

In this section, we propose SEAR protocol which is based on efficient symmetric cryptography, where public key cryptography is only used in an initial bootstrapping procedure. SEAR provides routing message authenticity via the use of one-way hash functions to construct a set of hash values, called authenticators, associated with each node. Intermediate nodes can verify the authenticators by simply applying efficient one-way functions, and cannot increase the sequence number or decrease the hop count in the messages when forwarding them. RERR messages are protected through a variation of the TESLA [11] broadcast authentication scheme. We begin by first presenting our assumptions and the initial bootstrap procedure, then we move onto discussing the details behind route discovery and route maintenance in SEAR.

4.1. Assumptions

We assume that each node A has a pair of public/private keys K_{A}, K_{A}^{-1}. The public key K_{A} is certified in some way, such as *via* a certificate signed by an authority trusted by all nodes. We also assume that the originator trusts the destination to make correct routing decision, that is, both the source and destination are benign.

Conventionally, there is no discrimination between even and odd sequence number usage in AODV. However, in order to protect RERR messages, we have adopted a strategy from DSDV, whereby even sequence numbers are used for messages that originate from the destination, while odd sequence numbers are used for those messages that originate from other nodes [5]. Therefore, in SEAR, RERR messages only contain odd sequence numbers.

We will employ a variation of TESLA to provide authenticity verification of RERR messages, and thus note that loose time synchronization is required by TESLA. Hence, we assume that all nodes are loosely time synchronized with their neighbors. This assumption can be realized by having nodes periodically perform a two-way time synchronization procedure, as is described in TESLA [11]. Since RERR messages are only broadcast to upstream neighbors, we only require that nodes are loosely time synchronized with their immediate neighbors, as opposed to network-wide synchronization.

We also assume that attackers do not control a set of nodes that partition the network. Therefore, we are assuming that there is at least one route from each source to each destination that consists of only benign nodes. Finally, for the purpose of providing route freshness and authenticity, we assume that caching mechanisms are disabled on all nodes.

4.2. Initial Bootstrapping

We require that each node maintains two hash chains in SEAR. Firstly, each node constructs an authenticator hash chain to protect the sequence numbers and hop counts for routing packets associated with routes to that node. Secondly, each node creates a TESLA key chain for authenticating RERR messages. These two chains need to be created, and their initial key value commitments need to be delivered securely during the bootstrap procedure. Additionally, the hash functions used in these hash chains must have the cryptographic one-way property.

We begin by providing a high-level overview of the usage of the authenticator hash chain. For discussion, let us denote m to be the maximum hop count that exists between any two network nodes, and $n+1$ to be the length of the hash chain. The authentication hash chain $\{h_0, h_1, \cdots, h_n\}$ is generated using a one-way hash function, H, *via* the recursion $h_{j+1} = H(h_j)$, with h_n corresponding to the initial commitment for this hash chain. The values of the hash chain, which are called authenticators, are partitioned and used according to the sequence numbers and hop counts associated with the routing messages they protect. A consequence of the one-way function and the partitioning of the authenticators is the fact that other nodes with smaller even sequence numbers will not be able to generate hashes for higher even sequence numbers. For authenticators associated with each even sequence number, other nodes with higher hop counts will not be able to generate hashes for lower hop counts. In SEAR, each individual hop for each even sequence number has a corresponding hash value. In contrast, for RERR messages, we use odd sequence numbers and, since they do not contain a hop count field, we only associate one hash value for each odd sequence number. In order to allow nodes to transmit RERR messages, any node other than the destination that has an even sequence number should also have the corresponding next higher odd sequence number so that it may transmit

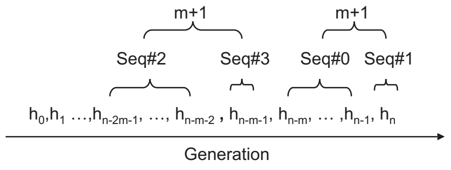

Fig. 2. Structure of the SEAR authenticator hash chain.

a RERR message. Consequently, authenticators are broken down into groups of $m+1$ consecutive hash values. These groups of $m+1$ hash values will be associated with an even and odd sequence number in a manner that is the reverse of the hash chain generation procedure. For a specific group of $m+1$ values, one hash value will be assigned to each hop associated with an even sequence number. As noted, the remaining one hash value is associated with the odd sequence number and is used in authenticating RERR messages.

In order to follow the above discussion, we present a diagram that depicts the typical structure of the hash chain in Figure 2. There are a total of $n+1$ hash values, and we assume that $n+1$ is a multiple of $m+1$, where m is the maximum hop count that exists between any two network nodes. For discussion, let us look at the group of hash values associated with sequence numbers 0 and 1 as an example. The m hash values from h_{n-m} to h_{n-1} are used to protect hop counts associated with sequence number 0 and the hash value h_n is used to protect sequence number 1 (which is only used in a RERR message). Inside each group, hash values are used in the same direction as the generation process to prevent the decrease of the hop count field by intermediate nodes. While using sequence number 0, the node itself will attach the hash value h_{n-m} to its routing messages. All neighbors of the node will apply the hash function to the received hash value and will replace h_{n-m} with $h_{n-m+1} = H(h_{n-m})$ in their routing messages. Thus, intermediate nodes cannot decrease the hop count as they forward the routing messages. All nodes that receive a routing message with sequence number 0, will receive a copy of the hash value h_n contained in this message, and thus any of them can create RERR messages should they detect a link failure in the future. The procedure follows a similar methodology for protecting routing messages with any other even sequence number.

Attackers cannot decrease the hop count field, but they can still attempt to commit another type of fraud where they transmit/forward routing messages they receive directly, without incrementing the hop count field. In order to prevent such 'same hop count fraud', the node identity should be encoded into the hash values

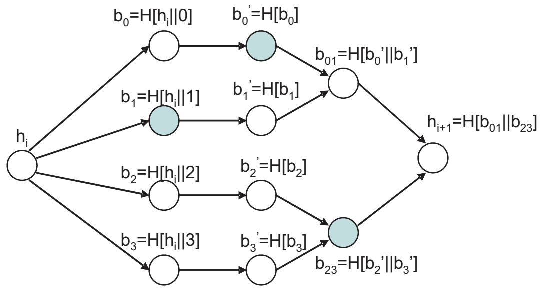

Fig. 3. Structure of authenticators with node identity encoded where node 1 would send all shaded hash values to its neighbors.

to form an authenticator hash tree, as was used in SEAD [12]. Consequently, each node cannot forward routing messages with authenticators encoded with another node's identity, and they must increase the hop count of RREQs and RREPs. As an example, in Figure 3, we present a hash tree scenario corresponding to a network of four nodes, following the example of Reference [12]. In this tree, the hash value h_i is used to generate intermediate hash values *en route* to calculating h_{i+1}. Specifically, the identity of each node is encoded to form a Merkle tree [24]. Rather than use h_i as the authenticator for a routing message, instead a selection of intermediate hash values is used. Each node sends its own hash values from the Merkle tree and the minimum set of intermediate hash values needed to generate h_{i+1}. For example, let us consider a situation where a node with identity 1 wishes to create an authenticator from h_i that has its identity encoded in the authenticator. Node 1 uses h_i to calculate the $b_i = H[h_i \| \text{id}]$, where id ranges from 0 to 3, as depicted in Figure 3. The subsequent values of the Merkle tree are formed and appropriately merged to yield h_{i+1}. Node 1 will use the shaded hash values $[b_1, b'_0, b_{23}]$ as its authenticator. These shaded hash values correspond to the minimal set of values needed to reconstruct h_{i+1}. Further, we note that if another node should want to forge a message with the same hopcount, it will be impossible for that node to alter the identity encoding since it would have to invert the one-way hash function.

We note that for small networks, each node can encode its identity into the hash tree directly, and no adversary can derive its value from neighbors' hash values that correspond to the same hop count. However, this procedure for generating and verifying the authenticator hash tree can become prohibitive as the size of the network increases. To overcome this problem for larger networks, we use a technique similar to in SEAD [12], where γ-tuple values are used to encode node identities into hash values. Now instead of having a unique leaf hash value for each node, each node will have γ leaf hashes. Each node encodes its identity along with the current sequence number to determine which γ leaves it will advertise. The authenticator will have a length of $m\gamma$. As an example, for a hash tree consisting of 8 leaves, if each node has a unique leaf hash value, it can serve a network up to 8 nodes. On the other hand, if each node could use $\gamma = 2$ hash values, the hash tree can be used for a network of up to $\binom{8}{2} = 28$ nodes. As derived in SEAD, attackers will have a sufficiently small probability of a successful attack.

In order to simplify the description of SEAR in the remaining part of this section, we will only include a discussion involving the one-way hash chain authenticators and will not use the hash tree chains. Extensions of the discussions to incorporate hash tree chains are straightforward.

Finally, each node broadcasts its authenticator commitment in an authenticated manner to all other nodes before the communication starts. For instance, we might use public key cryptography and have each node sign its authenticator commitments with its private key. Other nodes would verify the commitments through the corresponding public key. In order to prevent replay attacks under authenticator wrapping, a timestamp or a monotonically increasing sequence number can be delivered along with the authenticator commitments. In addition to initiating the authenticator hash chain, each node must also initiate a TESLA key chain. It does this by creating a one-way hash chain for keys used in TESLA, which will be used to authenticate RERRs. TESLA key commitments can be distributed to neighboring nodes in a manner similar to the authenticator commitments.

4.3. Securing Route Discovery

Authenticators can be incorporated into RREQ messages to protect the originator sequence number, destination sequence number, and hop count. In AODV, whenever the source needs a route, it broadcasts a RREQ:

$$\text{S} \rightarrow \text{All nodes, \{S, D, ID, SrcNum, DstNum, hop\}}$$

where S is the source address, D the destination address, ID the RREQ id, SrcNum the source sequence number, DstNum the last destination sequence number known to the source, and hop is the number of hops to the source traversed so far.

The source increments its sequence number each time it starts a new RREQ and thus the ID may be redundant since each RREQ can be uniquely defined by {S, SrcNum}. Nonetheless, for the sake of following the AODV standard as closely as possible, we have chosen to keep this field in SEAR. Let $v_{s,i,c}$ denote the hash authenticator for node S, with sequence number i and hop count c. Assume that the next even sequence number for the source is $2i$, and the most recent destination sequence number known by the source is j with hop count c. The source thus sends out

$$\text{S} \rightarrow \text{All nodes,}\{\text{S, D,ID}, 2i, v_{s,2i,0}, j, v_{d,j,c}, 0, \text{HERR}\}$$

where HERR is used in authenticating RERRs, which will be further discussed in Section 4.6. Each neighbor A will first check the authenticity of the authenticators $v_{s,2i,0}$ and $v_{d,j,c}$ with the authenticator commitments. If the verification fails, the routing message is ignored. Otherwise, A applies the hash function on the authenticator $v_{s,2i,0}$ to obtain the corresponding one hop authenticator $v_{s,2i,1}$. If A has a larger destination sequence number j' with hop count c', it will replace $v_{d,j,c}$ with $v_{d,j',c'}$. Additionally, A replaces HERR with its own HERR$'$. A thus forward

$$\text{A} \rightarrow \text{All nodes, }\{\text{S, D, ID}, 2i, v_{s,2i,1}, j', v_{d,j',c'}, 1, \text{HERR}'\} \quad \text{A has a larger DstNum}$$

$$\text{A} \rightarrow \text{All nodes, }\{\text{S, D, ID}, 2i, v_{s,2i,1}, j, v_{d,j,c}, 1, \text{HERR}'\} \quad \text{Otherwise}$$

The procedure repeats until the RREQ reaches the destination or the TTL expires.

Authenticators for even, higher sequence numbers can only be released by the node with hop count 0, therefore other nodes are not able to release a new, even sequence number or same even sequence number with smaller hop count. S and D addresses can be authenticated by association with the source/destination authenticators. Because each RREQ is uniquely defined by {S, SrcNum}, attackers cannot fake any new RREQs since they cannot invert the one-way function to obtain a valid new, even source sequence number. On the other hand, attackers can still launch replay attacks by putting an old source sequence number into the RREQ message. However, these old stale RREQs will be suppressed by newer RREQs eventually. The attacker can put a different source address with a corresponding source sequence number in the RREQs. This may result in setting up paths between nodes who do not want to communicate with each other. However, there will be at least one RREQ that reaches the correct destination with the correct source field as long as there exists at least one path entirely comprised of benign nodes. Attackers can also change the destination address and destination sequence number fields. The RREQ with the correct destination address field may get suppressed if intermediate nodes have seen the source address and source sequence number before. In Section 4.5, we propose to use routing suppression extension on RREQs to address this issue.

4.4. Securing Route Establishment

The sequence number and hop count in RREP messages can be secured in a manner similar to the approach taken to secure these fields in RREQ messages. Lifetime is usually a system-wide configuration parameter, and hence we do not need to explicitly secure it. The destination or intermediate nodes that have valid routes with equal or higher sequence numbers unicast

$$\mathrm{D} \rightarrow \mathrm{S}, \{\mathrm{S}, \mathrm{D}, \mathrm{DstNum}, \mathrm{hop}, \mathrm{lifetime}\}$$

to the originator of the RREQ. For security purposes and to provide freshness of routes, we assume that caching mechanisms are disabled on all nodes. Therefore, only the destination can initiate RREPs in SEAR. Assume the next even sequence number of the destination is $2j$, then the destination unicasts the following RREP to the originator

$$\mathrm{D} \rightarrow \mathrm{S}, \{\mathrm{S}, \mathrm{D}, 2j, v_{d,2j,0}, 0, \mathrm{lifetime}, \mathrm{HERR}\}$$

where HERR is used to authenticate RERR messages, which will be further discussed in Section 4.6. Neighbor A will first check the authenticity of the authenticator, then it will apply the one-way hash function to the authenticator $v_{d,2j,0}$ to get $v_{d,2j,1}$. It then replaces $v_{d,2j,0}$ with $v_{d,2j,1}$. Additionally, A replaces HERR with its own HERR$'$. A forward

$$\mathrm{A} \rightarrow \mathrm{S}, \{\mathrm{S}, \mathrm{D}, j, v_{d,2j,1}, 1, \mathrm{lifetime}, \mathrm{HERR}'\}$$

The procedure repeats until RREP reaches the originator.

Attackers can still fake RREPs with old destination sequence numbers, but cannot create any new RREPs with valid higher even destination sequence numbers. These stale RREPs will not cause update to the routing table and will be suppressed quickly by new RREPs. The attacker can change the source address field and block RREPs from reaching the correct originator and, as noted earlier, the routing suppression extension on RREPs will be described next to address this issue.

4.5. Routing Suppression

In this section, we propose to use a different routing message suppression approach than that is used in AODV in order to address the change of address attack that might be used against *ad hoc* routing. We refer to our technique as the routing suppression extension, and apply this method to address the change of address in RREQs and the change of the source address in RREP messages.

In AODV, intermediate nodes keep track of {S, SrcNum} pairs and forward only those RREQs they have never seen. In order to address the change of destination address problem, we add a destination address entry into the routing table. Intermediate nodes will base the decision of whether to forward the RREQ or not on the triple {S, SrcNum, D}. If more than one route to the same destination exists, the node picks the one with the fewest hops. Because there exists at least one path from the source to the destination that consists entirely of benign nodes and attackers cannot increase the source sequence numbers, at least one RREQ will reach the correct destination.

Intermediate nodes keep track of the {D, DstNum, hop} triplet and forward only those RREPs with higher destination sequence number or the same destination sequence number with smaller hop count in AODV. In order to address the problem of the change of source address in forwarding RREPs, we add a source address entry into the routing table. Intermediate nodes will make the decision of whether to forward the RREP based on the set of {D, DstNum, hop, S}. If there exists more than one route to the same destination, the node picks the one with the fewest hops. Since there exists at least one path from the source to the destination that consists entirely of benign nodes and attackers cannot increase the destination sequence numbers, at least one RREP will reach the correct originator.

When a node receives a RREQ, it has no clue whether this RREQ is destined for itself or not. Similarly, a node cannot assure that the RREP is actually intended for itself. The problem can be addressed by adding an end-to-end MAC to both RREQ and RREP messages. Then the destination will only reply to RREQs which have valid MACs and the originator will only accept RREPs which have valid MACs. There are several tradeoffs in adding the end-to-end MACs. Adding MACs can reduce the number of RREPs sent by 'wrong' destinations, but it can only prevent the destination from replying to the wrong RREQs, and intermediate nodes still cannot authenticate the RREQs. Similarly, adding MACs can prevent sources from accepting the wrong RREPs, but intermediate nodes cannot authenticate the RREPs and will accept them anyway. Further, adding end-to-end MACs requires the set up of a shared key between each pair of communicating nodes, which involves the existence of reliable key management/maintenance. This is not always possible for some network scenarios. Further, adding MACs increases the communication overhead associated with transmitting RREQs and RREPs, especially the broadcast of RREQs. Overall, we took the overhead of setting and managing the shared key between communicating nodes into account during the design of SEAR, and decided not to use MACs in RREQs and RREPs for SEAR.

4.6. Securing Route Maintenance

Once a node detects a link failure for the next hop of an active route while transmitting data, or receives a data packet for an inactive route or unknown destination, or receives a RERR from a neighbor for one or more active routes, a RERR message is generated and broadcasted to all upstream neighbors. All unreachable destinations with the same next hop, together with their corresponding sequence numbers, should be placed in the same RERR message.

The original format of a RERR created by node A is

$$\text{A} \rightarrow \text{upstream neighbors}, \{\text{D} - \text{list}, \text{DstNum} - \text{list}\}$$

with D − list describing the list of unreachable destination addresses and DstNum − list describing the corresponding destination sequence number list with each sequence number one larger than the newest even destination sequence number known by A.

First let us look at how the HERR fields attached in the RREQ and RREP messages are created. Assume that the last even source sequence number before forward RREQs for originator node D, or last even destination sequence number before forwards RREPs for destination node D known by A is $2j$, A can derive the next odd sequence number authenticator $v_{d,2j+1,0}$ *via* applying the one-way hash function. A will form the RERR message for this sequence number

$$\text{RERR}' = \{\text{Nonce}, \text{D}, 2\text{j} + 1, \text{v}_{\text{d},2\text{j}+1,0}\}$$

let $k_{\text{A},t}$ denote the TESLA key for node A used in time interval t. Then A attaches

$$\text{HERR} = \{\text{H}(\text{RERR}'), \text{MAC}_{k_{\text{A},t}}(H(\text{RERR}'))\}$$

to the RREQ or RREP messages and saves the nonce and TESLA key in the error messages for later use. When nodes receive the RREQs and RREPs, they buffer the HERR contained in the messages. Further, they replace the HERR in the messages with their own HERR′ and save the corresponding information while forwarding RREQ and RREP messages.

Later, when A detects a link failure, or receives a data packet for an inactive route or unknown destination, or receives a RERR from a neighbor for one or more active routes, it will send a RERR for all unreachable destinations sharing the same next hop with the saved nonce list, corresponding TESLA release key list and destination sequence number list in the same RERR message. If any of the TESLA keys are not ready to be released yet, A will hold the RERR until it can release all the keys. Since nodes need to try a certain number of retransmissions or wait for several losses of HELLO messages before declaring a routing error, this scenario should rarely occur. The new format of RERR is

$$\begin{aligned}\text{RERR} = \{&\text{Nonce} - \text{list}, \text{TESLAkey} - \text{list},\\ &\text{D} - \text{list}, (2j + 1) - \text{list}, v_{d,2j+1,0} - \text{list}\}\end{aligned}$$

The upstream nodes extract the information needed from the various lists to reconstruct the HERR and to verify the RERR instantaneously. Since the list of the destinations affected by the broken link is changing over time, the hash values and MACs for the RERR are sent along with the RREQ and RREP separately for each destination. It's desirable to have minimum changes to the AODV protocol and send one slightly longer message versus sending multiple short messages.

In order to limit the number of RERRs messages sent by each node, a node is not allowed to generate more than $\text{RERR}_{\text{RATELIMIT}}$ RERR messages per second in ADOV. Further we can incorporate route flap dampening for nodes that send too many RERRs [25].

Table I. Performance comparison of different secure AODV routing protocols.

	ARAN	SAODV	SEAR
Computation efficiency	Low	Low	High
Communication overhead	High	Medium	Low
Standard compliant	Low	Medium	High

5. Performance and Security Analysis

In this section, we examine the performance of SEAR and provide a comparison of SEAR's performance and security *versus* other existing secure AODV routing protocols in Sections 5.1 and 5.2. Following this discussion, we provide a concrete comparison through *ns*2 simulations.

5.1. Performance Evaluation

We will compare SEAR with two other secure AODV protocols (ARAN [9] and SAODV [8]) in terms of their protocol efficiency. We summarize the comparison of these three protocols in Table I, where we have identified several different features for comparison.

5.1.1. Computation efficiency

ARAN requires the originator to sign each packet it sends, while intermediate nodes are required to verify and sign the signature for each routing packet it processes. SAODV also requires the originator to sign each routing packet, and intermediate nodes to verify those routing packets. In both cases, the signatures are based on public key operations. On the other hand, SEAR is mainly based on symmetric cryptography. It only requires the originator to apply a computationally efficient one-way hash function to generate authenticators and likewise intermediate nodes need only apply the one-way hash function to verify and generate next hop authenticators. We provide an efficiency comparison for these protocols in Section 5.3.

5.1.2. Communication overhead

In ARAN, a certificate and a signature are added to all routing messages at creation. Another certificate and signature are added to these routing messages as intermediate nodes process them. The size of the certificate depends on various parameters, such as the certificate type, the signature scheme that is employed, and the key size. In SAODV, there are several hashes and one signature introduced in RREQ and RREP single signature extension. There is another copy of the signature and hash in the RREQ and RREP double signature extension for route caching. Further, there is a signature added in the RERR messages. In contrast, in SEAR, we only require several hashes in RREQ and RREP routing messages, and for a RERR we only need a nonce, a key and a hash. Typically, the size of the hash fields and message authentication codes are much smaller than the size of a signature, leading to SEAR requiring less communication overhead than either ARAN or SAODV. A more detailed comparison of routing overhead for SEAR and SAODV can be found in Section 5.3.

5.1.3. Standard compliance

ARAN changes some of the basic design features of AODV in order to achieve security. First, it does not use hop count to select the optimal path, but instead it uses the first received RREQ. Secondly, it uses a nonce plus time stamp to provide the freshness of a route rather than the sequence number. These nonces and time stamps stay unchanged throughout the propagation of the messages. It should be noted that there has not been any evidence provided in ARAN to demonstrate that such modifications would be as effective as using sequence numbers to prevent the formation of routing loops. In AODV, when a node receives a RERR along its active path, a RERR message is created and broadcast to its upstream neighbors. In ARAN, however, the RERR is forwarded along the path toward the source without modification. This change facilitates error spoofing attacks, which will be described later in this section. Further, the caching mechanism is disabled in ARAN. There are several adaptations made by SAODV in order to incorporate security mechanisms into AODV. First of all, a node should never update any destination sequence number in its routing table based on RERR messages, but they may use them to decide whether they should invalidate a route. Secondly, an intermediate node must not replace the destination sequence number in the RREQ with its own sequence number, even when it has a newer copy of sequence number. Consequently, the only mutable field in RREQ and RREP messages is the hop count, and there are no mutable fields in RERR messages. SAODV introduces a double signature extension for caching, which complicates the protocol design and increases both the communication and computational requirements. While designing SEAR, we only require

Table II. Security comparison of different secure AODV routing protocols.

	ARAN	SAODV	SEAR
Message forgery	No	No	No
Increase sequence number	N/A	No	No
Decrease hop Count	N/A	No	No
Same hop count fraud	Yes	Yes	No
Signature DoS	Yes	Yes	No
Forming routing loop	No	Yes	No
Error spoofing attack	Yes	No	No

the removal of one of the optimizations of the AODV protocol, that is, the caching mechanism. Finally, each of these protocols requires the addition of extra fields in order to provide security.

Overall, SEAR requires significantly less computation and communication overhead compared to SAODV and ARAN.

5.2. Security Evaluation

We now compare the security characteristics of SEAR with ARAN and SAODV. A comparison summary is provided in Table II.

5.2.1. Message forgery

Both SAODV and ARAN append digital signatures to each routing message to prevent message forgery. Further, SAODV uses hashing to prevent the decrement of the hop count field. SEAR, on the other hand, uses the inherent properties of the AODV routing algorithm to assure that, even if an adversary attempts to forge old or less optimal routes, these routes will be suppressed and not introduce any benefit to an adversary.

5.2.2. Increase sequence number

SAODV stops the adversaries from increasing the sequence numbers by adding digital signatures to each routing message. While SEAR achieves the same goal by using efficient one-way functions.

5.2.3. Decrease hop count

Both of SAODV and SEAR prevents the decrease of the hop count by using one-way functions.

5.2.4. Same hop count fraud

Both SAODV and SEAR use one-way hashes to protect the hop count field against a malicious decrement during transmission. SAODV only prevents the decrease of the hop count, while attackers can still transmit routing messages with the same hop count as the messages they receive. In SEAR, we encode node identity into sequence numbers and hop counts, hence attacks would have to increase the hop count as they forward the messages. In ARAN, each intermediate node replaces the signature of the previous hop with its own signature. Malicious nodes can take advantage of this weakness and forward the routing packets directly, without replacing the signature. Because the signature appears to be a valid signature, other nodes would fail to observe the malicious behavior. Malicious nodes can strip the outer signature and replay these messages as if it were the source. Under both cases, malicious nodes do not need to verify and generate the signatures. These routing packets are likely to arrive at the originator or destination earlier than the regular routing packets, and thus would be selected as the route that would be used for future data communication. A simulation analysis for same hop count fraud will be illustrated in Section 5.4.

5.2.5. Signature DoS

Both ARAN and SAODV are based on expensive public key operations, therefore both of them are susceptible to denial of service attacks in which an attacker may flood a large amount of bogus routing packets, requiring the node to verify these faked signatures, and as a result can exhaust a node's computational capacity. Since SEAR is based on efficient one-way hash functions during the main operation of the protocol, and only requires public key operations during initial key setup, it is not vulnerable to signature denial of service attacks after the initial bootstrap phase.

5.2.6. Formation of the routing loops

In SAODV, intermediate nodes do not authenticate the previous hop, and hence an attacker can exploit this vulnerability and create routing loops. Specifically, a malicious node can create a routing loop by setting the previous hop to be another node it knows to be upstream. For example, suppose there exists a path from S to D, $S \to A \to B \to M \to D$, where M is malicious. M knows that A is its upstream node from

S to D. When M forward the RREP, it puts A as the previous hop. When B receives the RREP, it will set A as its next hop toward D and forward the RREP to S. Similarly, A sets B as its next hop to D. As a result, the packets will loop between A and B. However, in SEAR, node identity is encoded with the sequence numbers and hop counts, malicious node could not change the previous hop and try to formate the routing loops.

5.2.7. Error spoofing attack

In AODV, when a node receives a RERR along its active path, a RERR message is created and broadcast to its upstream neighbors. In ARAN, however, the RERR is forwarded along the path toward the source without modification. ARAN only authenticates the original source of the RERR. Each node that generates a RERR would sign the message and send it to its upstream neighbors. All nodes on the path toward the originator would simply rebroadcast the RERR, without including its own signature. The adversary can take advantage of this weakness and launch error spoofing attacks. For example, there exists a path from S to D, S $\rightarrow$ A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D. Hence if M is malicious and knows that A uses B as a next-hop to D, then M can send a RERR forged from B and sign it with its own private key. Since there is no signature from B included, A will believe that B's route to D is broken, and will rebroadcast the RERR.

In summary, SEAR's design and use of symmetric cryptographic operations gives better security properties than SAODV or ARAN.

5.3. Performance Analysis

We used *ns*2 to study the performance efficiency of SEAR and SAODV when there is no attacker. Specifically, we increased the size of the routing messages to include the authenticators, hashes, MACs, and keys we used in SEAR. We assume that all nodes were loosely time synchronized with each other with fixed synchronization errors. Further, we assume that each node had securely distributed the authenticator commitments to every other node in the network and that TESLA key commitments were established with neighbors.

In the simulations, nodes moved following the random waypoint mobility model. Each node was initially randomly placed within a rectangular ragion, where it paused for a short time. Then it started moving toward a new location with a speed uniformly distributed between 0 and a maximum speed. It repeated the process of pausing and moving to new locations. The simulation space was a rectangular region with a size of $1500 \times 300\,\text{m}^2$, which was chosen to ensure that there existed multiple hops within the network. The communication range for each node was set to 250 m. There were a total 15 pairs of communicating nodes, with each source sending out constant bit rate (CBR) traffic with packet sizes of 64 bytes at a rate of 4 packets/s. Since the time taken to compute a hash function is on the order to 1 μs, we omitted the time taken to compute hashes in the simulations. The summary of the parameters used in the simulation is shown in Table III.

Table III. Simulation parameters.

Number of nodes	50
Dimension of space	$1500 \times 300\,\text{m}^2$
Communication range	250 m
Node maximum movement speed	20 m/s
Total number of connections	15
Traffic pattern CBR packet size	64 bytes
Traffic rate	4 packes/s
Link bandwidth	1 Mbps
TESLA time interval	1 s
Maximum end to end network delay	0.1 s
Synchronization error	0.1 s
Hash size	80 bits
MAC size	80 bits
Key size	80 bits
Signature size	1024 bits
Generate a signature	10 ms
Verify a signature	1 ms

We studied the performance of SEAR for both the versions where there is no identity encoded and the version in which identity is encoded. Since the scope of our simulation corresponds to a medium-scale scenario, each node had a unique hash which corresponded to its identity. Further, we compared the performance of SEAR with SAODV under the same network topology and simulation parameters. In order to maximize the advantages of SAODV, we performed the simulations for both cache-enabled and cache-disabled versions of SAODV. We evaluated the performance of SEAR by comparing the following metrics to those of AODV and SAODV:

(1) *Packet delivery ratio* is defined as the total number of packets received at the receivers divided by the total number of packets sent by the sources.
(2) *Routing overhead in terms of the number of packets transmitted* is defined as the total number of routing packets transmitted or forwarded.
(3) *Routing overhead in terms of the number of bytes transmitted* is defined as the total number of bytes of the routing packets transmitted or forwarded.

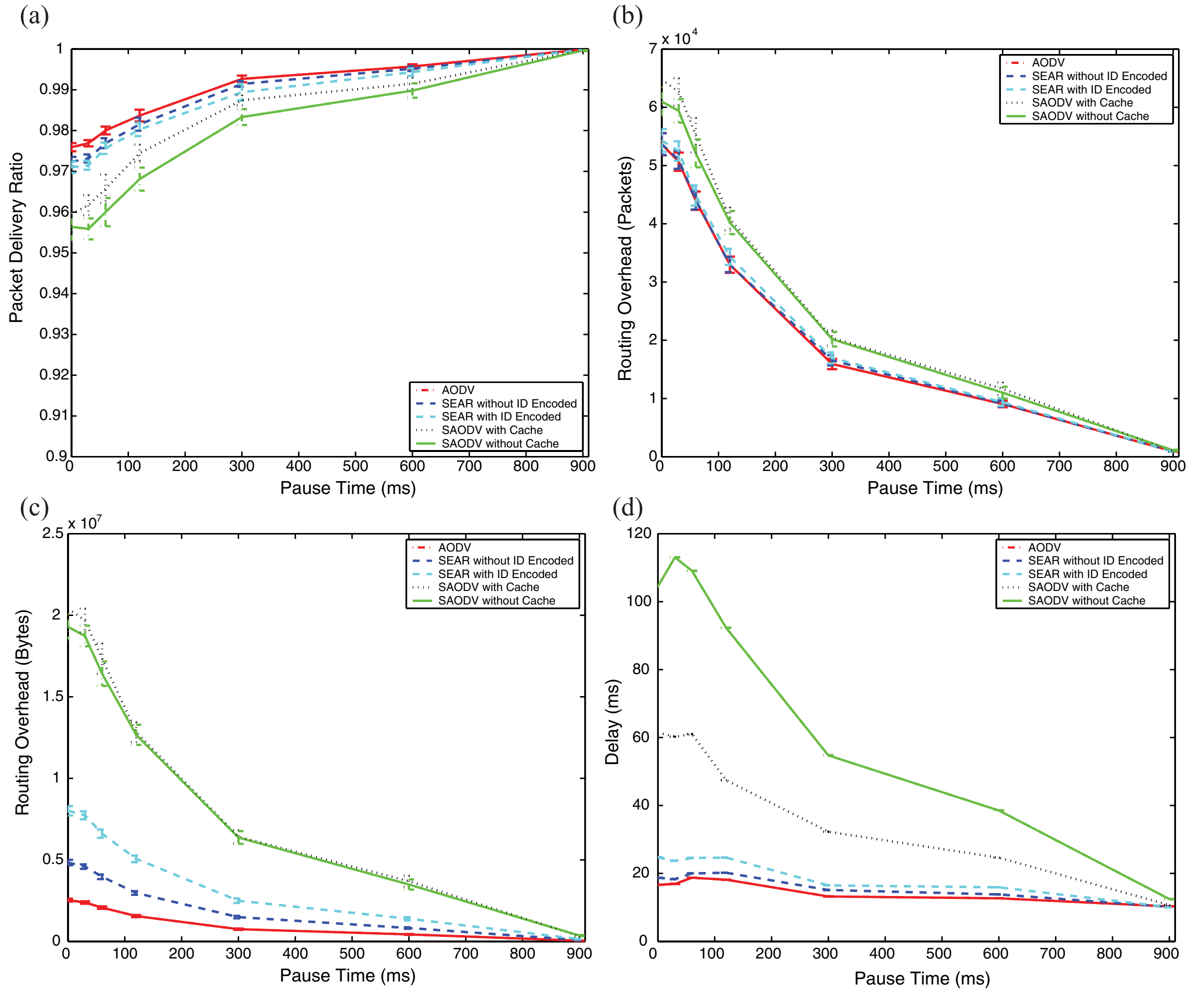

Fig. 4. (a) Packet delivery ratio for AODV, SEAR and SAODV. (b) Routing overhead in terms of number of packets for AODV, SEAR, and SAODV. (c) Routing overhead in terms of number of bytes for AODV, SEAR, and SAODV. (d) Packet delivery delay for AODV, SEAR, and SAODV.

(4) *Packet delivery delay* is defined as the average delay between the moment when the packet is sent by the sender application and the moment when the packet is received at the receiver application.

The simulation results are shown in Figure 4, which is based on an average over 60 runs of different movement files for each pause time. The 95% confidence intervals for the metrics are plotted as error bars. The packet delivery ratio of SEAR with either identity encoded or without identity encoded degrades at most 1% for all pause times, which illustrates that SEAR performs better than both versions of SAODV. The packet delivery ratio for SAODV with cache-enabled is better than that for SAODV without cache. The routing overhead in terms of the number of packets for SEAR is at the same level as AODV. Again, SEAR outperforms both SAODV versions. Both SEAR and SAODV introduce extra fields in the routing messages for authentication. For SEAR, the routing overhead in terms of bytes transmitted when identity is not encoded is about twice the amount of baseline AODV. Because each node needs to send auxiliary hashes to allow next hop neighbors construct the authenticators when identity is encoded, the routing overhead in terms of number of bytes when identity is encoded is higher than without identity encoded. In contrast, the routing overhead in terms of bytes transmitted is about four times greater for SAODV than for SEAR without identity encoded. This large amount of communication overhead for SAODV is primarily a result of its use of lengthy signatures. For SEAR, the average packet delivery delay is slightly increased due to the increased communication overhead, while the increase in delay for SAODV is roughly three times with cache enabled and five times without cache support. Overall, SEAR outperforms both SAODV versions in all of the performance metrics that we examined.

In the above simulations, we only compared the performance of SEAR and SAODV to AODV without caching enabled since we have inherently disabled caching in the design of SEAR. We now briefly comment on the performance of SEAR and SAODV relative to AODV with caching. AODV with caching achieves a slightly better packet delivery ration than AODV without caching. However, the routing overhead of AODV with caching is higher than that of AODV without caching, but is smaller than all versions of SEAR and SAODV. The average packet delivery delay of AODV without caching is slightly increased compared to AODV with caching. Consequently, AODV with caching provides a better packet delivery rate and smaller delivery delay than all SEAR and AODV versions when operating in benign settings. However, we emphasize that, both caching-enabled and caching-disabled versions of AODV do not have security mechanisms designed into their operation, and thus both experience significant performance degradation in adversarial settings.

5.4. Effects of Same Hop Count Fraud

The performance and robustness of SEAR in the presence of malicious attackers was also studied *via ns*2 simulations. Specifically, we studied the performance of SEAR against same hop count fraud attacks. We then compared the performance of SEAR with the performance of AODV without any security extensions

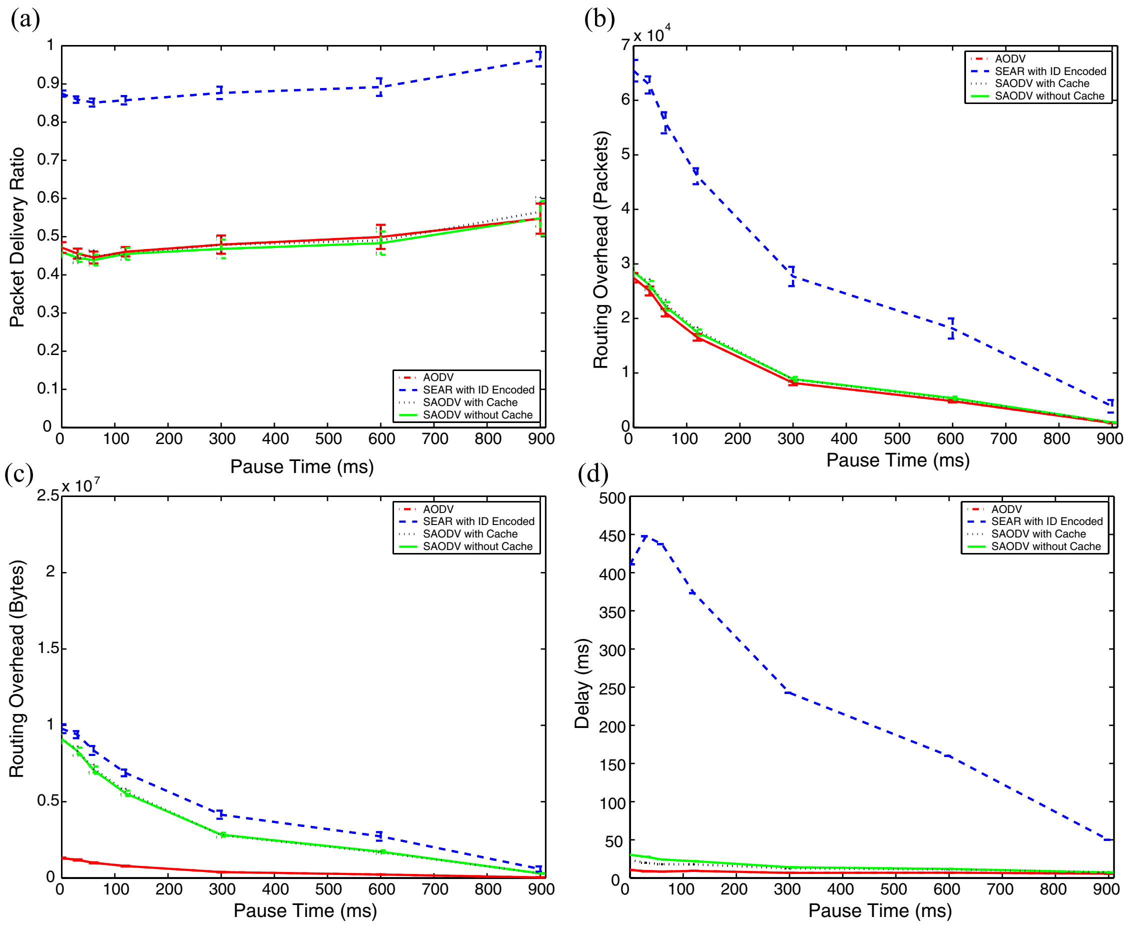

Fig. 5. (a) Packet delivery ratio for AODV, SEAR, and SAODV when there were 30% malicious nodes. (b) Routing overhead in terms of number of packets for AODV, SEAR, and SAODV when there were 30% malicious nodes. (c) Routing overhead in terms of number of bytes for AODV, SEAR, and SAODV when there were 30% malicious nodes. (d) Packet delivery delay for AODV, SEAR, and SAODV when there were 30% malicious nodes.

and with the performance of SAODV with and without cache support. To make the comparison as fair as possible, the network topology, and parameters were the same for all three protocols. These parameters can be found in Table I. In the simulations, a fraction of the nodes were set as malicious nodes, which would forward the routing messages they received directly without appropriately increasing the hop count. Further, these malicious nodes would drop the data packets later if they were selected on any of the routing paths. These malicious nodes were picked randomly with the restriction that they would not be either the source or the destination of any data flows.

We examined the same performance metrics as in Section 5.3. The simulation results are based on an average of 60 runs of different movement files for each pause time, as shown in Figure 5, where the 95% confidence intervals are plotted as error bars. We studied several scenarios by varying the number of attackers, and observed that all cases exhibited comparable results and trends. Therefore, in this paper, we only present the simulation results from the case where we had a total of 15 malicious nodes (i.e., 30% of the network) performing same hop count fraud and packet dropping attacks. Because malicious nodes did not increase hop count as they forwarded routing messages, these routes would appear to be shorter paths and would be selected as the desired paths for data transmission. For the SEAR protocol with node identity encoded, the packet delivery ratio experiences a drop of about 10% when there are a total of 30% attackers. In contrast, packet delivery takes a performance hit of roughly 50% for both AODV and SAODV in this adversarial setting. These 'shorter' paths may be selected as the optimum paths and thus reduce the number of routing messages transmitted between nodes in AODV and SAODV. The average routing overhead and delay decreased for AODV and SAODV in the presence of same hop count fraud attackers. On the other hand, the routing overhead and delay increased for SEAR, because sources ended up choosing longer (and more secure) paths to the destinations in order to avoid including any malicious nodes in the paths. In some cases, with 30% of the network consisting of malicious nodes, it was observed that in SEAR the sources would fail to find a route to the destination that consisted entirely of benign nodes, while SAODV and AODV would choose paths with malicious nodes. From these results, we can conclude that SEAR performs much better than either AODV or SAODV in the presence of same hop count fraud attacks.

6. Conclusion

In this paper, we have proposed SEAR protocol for wireless *ad hoc* networks. SEAR is mainly based on efficient symmetric cryptography, while asymmetric cryptography is only used for the initial key commitments distribution. The format of routing packets used in SEAR follows the same basic specification as the AODV protocol. The performance and security of SEAR were compared with other existing secure AODV protocols and analyzed *via* *ns*2 simulations. SEAR provides the same level of packet delivery delay and the total number of routing packets needed as AODV. Packet delivery ratio is only slightly degraded due to the increased size of the routing packets. We also have shown through both theoretical examination and simulations that SEAR is less vulnerable to routing attacks when compared to existing secure AODV protocols.

References

1. Seattle wireless. http://www.seattlewireless.net/
2. Roofnet. http://pdos.csail.mit.edu/roofnet/doku.php
3. Hu Y, Perrig A. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy* 2004; **2**(3): 28–39.
4. Johnson DB, Maltz DA. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Vol. 353. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996; 153–181.
5. Perkins C, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244.
6. Perkins C, Belding-Royer E, Das S. Ad hoc on-demand distance vector (AODV) routing, 2003, IETF RFC 3561.
7. IEEE standard 802.11s: ess mesh networking standard. *INTERNET-DRAFT*.
8. Zapata MG, Asokan N. Securing ad hoc routing protocols. In *Proceedings of ACM Workshop on Wireless Security (WiSe)*, 2002, pp. 1–10.
9. Sanzgiri K, Dahill B, Levine BN, Shields C, Belding-Royer EM. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, 2002, pp. 78–87.
10. Hu Y, Perrig A, Johnson DB. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, 2002, pp. 12–23, New York, NY, U.S.A., ACM Press.
11. Perrig A, Canetti R, Tygar D, Song D. The TESLA broadcast authentication protocol. *Cryptobytes* 2002; **5**: 2–13.
12. Hu Y, Johnson D, Perrig A. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks* 2003; **I**: 175–192.
13. Papadimitratos P, Haas ZJ. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, 2002.
14. Buchegger S, Boudec JL. Performance analysis of the CONFIDANT protocol: Cooperation of nodes—fairness in

dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002, pp. 226–236.
15. Lai K, Marti S, Giuli TJ, Baker M. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, 2000, pp. 255–265.
16. Hu Y, Perrig A, Johnson DB. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of IEEE INFOCOM*, 2003, pp. 1976–1986.
17. Yu W, Sun Y, Han Z, KJR Liu. A trust evaluation framework in distributed networks: vulnerability analysis and defense against attacks. In *Proceedings of the IEEE INFOCOM06*, 2006.
18. Sun Y, Yu W, Liu KJR. HADOF: defense against routing disruptions in mobile ad hoc networks. In *Proceedings of the IEEE INFOCOM05*, 2005.
19. Acs G, Buttyan L, Vajda I. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing* 2006; **5**(11): 1533–1546.
20. Yi S, Naldurg P, Kravets R. Security-aware ad hoc routing for wireless networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, October 2001 Long Beach, CA.
21. Nita-Rotaru C, Awerbuch B, Holmer D, Rubens H. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe)*, September 2002.
22. Zhou L, Haas ZJ. Securing ad hoc networks. *IEEE Network* 1999; **13**(6): 24–30.
23. Ning P, Sun K. How to misuse AODV: a case study of insider attacks against mobile adhoc routing protocols. In *Proceedings of the 4th Annual IEEE Information Assurance Workshop*, 2003, pp. 60–67.
24. Merkle RC. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1980, pp. 122–134.
25. Varghese G, Mao Z, Govindan R, Katz R. Route flap dampening exacerbates internet routing convergence. In *Proceedings of ACM SIGCOMM*, 2002.