Securing the Destination-Sequenced Distance Vector Routing Protocol (S-DSDV)

Conference Paper · October 2004

DOI: 10.1007/978-3-540-30191-2_28 · Source: DBLP

Tao Wan
Carleton University
17 PUBLICATIONS **547** CITATIONS

Evangelos Kranakis
Carleton University
381 PUBLICATIONS **6,710** CITATIONS

Securing the Destination Sequenced Distance Vector Routing Protocol (S-DSDV) ⋆

Tao Wan Evangelos Kranakis P.C. van Oorschot

{twan, kranakis, paulv}@scs.carleton.ca
School of Computer Science, Carleton University, Ottawa, Canada

Abstract. *A mobile ad hoc network (MANET) is formed by a group of mobile wireless nodes, each of which functions as a router and agrees to forward packets for others. Many routing protocols (e.g., AODV, DSDV, DSR, etc) have been proposed for MANETs. However, most assume that nodes are trustworthy and cooperative. Thus, they are vulnerable to a variety of attacks. We propose a secure routing protocol based on DSDV, namely S-DSDV, in which, a well-behaved node can successfully detect a malicious routing update with any sequence number fraud (larger or smaller) and any distance fraud (shorter, same, or longer) provided no two nodes are in collusion. We compare security properties and efficiency of S-DSDV with superSEAD. Our efficiency analysis shows that S-DSDV generates high network overhead, however, which can be reduced by configurable parameters. We believe that the S-DSDV overhead is justified by the enhanced security.*

Keywords: MANET, Routing Security and Analysis, Distance Vector

1 Introduction

A MANET is formed by a group of wireless nodes, each of which performs routing functions and forwards packets for others. No fixed infrastructure (i.e., access point) is required, and wireless nodes are free to move around. A fixed infrastructure can be expensive, time consuming, or impractical. Another advantage of wireless ad hoc networks is the expansion of communication distance. In an infrastructure wireless network, nodes are restricted to move within the transmission range of access points. Ad hoc networks relax this restriction by cooperative routing protocols where every node forwards packets for the rest of the nodes in the network. Potential applications of wireless ad hoc networks include military battle field, emergency rescue, campus networking, etc.

Wireless ad hoc networks face all the security threats of wireline network routing infrastructures, as well as new threats due to the fact that mobile nodes have constrained resources (e.g., CPU, memory, network bandwidth, etc), and lack physical protection. One critical threat faced by most routing protocols is that a single misbehaving router may completely disrupt routing operations by spreading fraudulent routing information since a trustworthy and cooperative environment is often assumed. Consequences include, but are not limited to: 1) packets may not be able to reach their ultimate destinations; 2) packets may be routed to their ultimate destinations over non-optimal routes; 3) packets may be routed over a route in the control of an adversary.

Many mechanisms [20, 19, 1, 7, 18] have been proposed for securing routing protocols by providing security services, e.g., entity authentication and data integrity, or by detecting forwarding level misbehaviors [12, 10]. However, most do not validate the factual correctness of routing updates. One notable protocol is superSEAD proposed by Hu, et al [8, 9]. SuperSEAD is based on the Distance Sequenced Distance Vector (DSDV) routing protocol [14], and uses efficient cryptographic mechanisms, including one-way hash chains and authentication trees, for authenticating sequence numbers and distances of advertised routes. SuperSEAD can prevent a misbehaving node from advertising a route with 1) a sequence number larger than

[1] Draft version: May 30, 2004.

the one it received most recently (*larger sequence number fraud*); and 2) a distance shorter than the one it received most recently (*shorter distance fraud*) or the same as the one it received most recently (*same distance fraud*). However, superSEAD does not prevent a misbehaving node from advertising a route with 1) a sequence number smaller than any one it has received (*smaller sequence number fraud*); or 2) a distance longer than any one it has received (*longer distance fraud*). Another disadvantage is that SuperSEAD assumes the cost of a network link is one hop, which may limit its applicability. For example, it may not applicable to a DV which uses network bandwidth as a parameter for computing cost metrics.

1.1 Problems and Results

Smaller sequence number and longer distance frauds clearly violate the routing protocol specifications, and can be used for non-benevolent purposes (e.g., selfishness). Although the damage they can cause has been thought less serious than those of larger sequence number fraud or shorter distance fraud, we believe they still need to be addressed for many reasons. Two of them are as follows: 1) they can be used by selfish nodes to avoid forwarding traffic, thus detecting these frauds would significantly reduce the means of being selfish; 2) it is always desirable to detect any violation of protocol specifications even though its damage may remain unclear or the probability of such violation seems low. Past experience has shown that today's naive security vulnerabilities can often be exploited to launch serious attacks and to cause dramatic damages in the future. For example, a vulnerability of TCP sequence number prediction was discussed as early as 1989 [3], but was widely thought to be very difficult to exploit given the extremely low probability (2^{-32}) of guessing a correct sequence number. It did not attract much attention until April 2004 when a technique was discovered which takes less time to predict a correct TCP sequence number.

In this paper, we propose the use of *consistency checks* to detect sequence number frauds and distance frauds in DSDV. Our protocol, namely S-DSDV, has the following security properties, provided that no two nodes are in collusion: 1) detection of any distance fraud (longer, same, or shorter); 2) detection of both larger and smaller sequence number fraud. One notable feature of S-DSDV is that a misbehaving node surrounded by well-behaved nodes can be contained. Thus, misinformation can be stopped in the first place before it spreads into a network. Our efficiency analysis shows that S-DSDV produces higher network overhead than superSEAD. However, S-DSDV overhead can be controlled by adjusting configurable parameters, e.g., interval of consistency checks. We believe that network overhead caused by S-DSDV can be justified by its enhanced security.

The sequel is organized as follows. Section 2 provides background information of distance vector routing protocol and DSDV. Section 3 presents overview and security analysis of SEAD. A threat model is discussed in Section 4. S-DSDV is presented and analyzed in Section 5. Efficiency of S-DSDV is compared with superSEAD by analysis and simulation in Section 6. We conclude the paper in the last section.

2 Background

In this section, we provide background information for simple distance vector routing protocols and DSDV [14]. Readers familiar with these topics can skip this section.

2.1 Notations

We use $G = (V, E)$ to represent a network where V is a set of nodes and E is a set of links. A distance vector route may consist of some of the following fields: seq - a sequence number; dst - a destination node; cst - a cost metric or distance; nhp - a next hop node; aut - an authentication value.

2.2 Distance Vector

In a traditional DV algorithm, each node $v_i \in V$ maintains a cost metric or a distance for each destination node v_j in a network. Let $d^t(v_i, v_j)$ be the distance from v_i to v_j at time t. Initially or at time 0,

$$d^0(v_i, v_j) = \begin{cases} 0 & \text{if } v_i = v_j \\ \infty & \text{if } v_i \neq v_j \end{cases}$$

Suppose at time 1, each node v_i learns all of its direct neighbors (denoted by $N(v_i)$) by some mechanisms, e.g., receiving a special message from v_j may confirm v_j as a direct neighbor. Suppose each node v_i also knows the distance to each of its direct neighbors $v_j \in N(v_i)$, which can be the cost of the edge linking v_i and v_j, $c(v_i, v_j)$. At time 1, node v_i's routing table may look like:

$$d^1(v_i, v_j) = \begin{cases} 0 & \text{if } v_i = v_j \\ c(v_i, v_j) & \text{if } v_j \in N(v_i) \\ \infty & \text{if } v_i \neq v_j \text{ and } v_i \notin N(v_i) \end{cases}$$

Each node broadcasts its routing table to each of its direct neighbors periodically or when a distance changes. Let $v_j \in N(v_i)$. The broadcast of routing tables at time t from v_i to v_j can be specified as:

$$v_i \rightarrow v_j : \{(v_k, d^t(v_i, v_k)) | v_k \in V\}$$

v_i also receives routing updates from each of its direct neighbors at time t. v_i updates its distance to v_k using the minimum of all known distances to v_k. Thus, at time $t+1$,

$$d^{t+1}(v_i, v_k) = \min_{v_j \in N(v_i)} \{d^t(v_j, v_k) + c(v_i, v_j)\}$$

The advantages of DV routing protocols include: simplicity, low storage requirement, and ease of implementation. However, they are subject to short or long alive routing loops. Routing loops are primarily caused by the fact that selection of next hops is made in a distributed fashion based on partial and possibly stale information. Routing loops can also be manifested in the propagation of routing updates by the problem of count-to-infinity [11].

To mitigate the problem of count-to-infinity, several mechanisms may be used: 1) setting the maximum network diameter to k (*limited network boundary*). As a result, the problem of count-to-infinity becomes count-to-k; 2) not advertising a route back to the node this route is learned from (*split-horizon*); 3) advertising a infinite route back to the node this route is learned from (*split-horizon with poisoned reverse*).

2.3 DSDV

DSDV [14] is a routing protocol based on a DV approach, specifically designed for wireless ad hoc networks. DSDV solves the problem of routing loops and count-to-infinity by associating each route entry with a sequence number indicating its freshness. The split-horizon mechanism cannot be applied to a wireless ad hoc network due to its broadcast nature. In wireline network, a node can decide over which link (or to which node) a routing update will be sent. However, in a wireless ad hoc network, a routing update is transmitted by broadcast and is received by every wireless node within a transmission range. Thus, it is impossible to selectively decide which nodes to receive a routing update.

In DSDV, a sequence number is linked to a destination node, and usually is originated by that node (the owner). The only case that a non-owner node updates a sequence number of a route is when it detects a link break on that route. An owner node always uses even-numbers as sequence numbers, and a non-owner node always uses odd-numbers. With the addition of sequence numbers, routes for the same destination are selected based on the following rules: 1) a route with a newer sequence number is preferred; 2) in the case that two routes have a same sequence number, the one with a better cost metric is chosen.

2.4 Security Threats to DSDV

DSDV guarantees all routes are loop free. However, it assumes that all nodes are trustworthy and cooperative. Thus, a single misbehaving node may be able to completely disrupt the routing operation of a whole network. We focus on two serious threats - the manipulation of sequence numbers and the manipulation of cost metrics. Specifically, a misbehaving node can poison other nodes' routing tables or affect routing operations by advertising routes with fraudulent sequence numbers or cost metrics.

To protect a routing update message against malicious modification, public key based digital signatures may be helpful. For example, v_i sends to v_j a routing update signed with v_i's private key. v_j can verify the authenticity of the routing update using v_i's public key. However, digital signatures cannot prevent a malicious entity (v_i itself or any entity with the knowledge of v_i's private key) from advertising false information (e.g., false sequence number or distance). In other words, message authentication cannot guarantee the factual correctness of a routing update. For example, when v_i advertises to v_j a route for v_d with a distance of 2, v_j is supposed to re-advertise that route with a distance of 3 provided that it is the best known route to v_i. However, v_j can advertise that route with any other distance value without being detected by a message authentication mechanism.

3 SEAD Review

Hu, et al [8, 9] made a first attempt to authenticate the factual correctness of routing updates using one-way hash chains. Their proposal, based on DSDV and called SEAD [8], can prevent a malicious node from increasing a sequence number or decreasing a distance of an advertised route. In the above example, v_j cannot successfully re-advertise the route with a distance shorter than 2. However, SEAD cannot prevent v_j from advertising a distance of 2 or longer (e.g., 4). In SuperSEAD [9], they proposed to use combinations of one-way hash chains and authentication trees to force a node to increase the distance of an advertised route when it re-advertises that routing update. In the above example, v_j cannot advertise a distance of 2. However, v_j is free to advertise a distance longer than 3.

We describe SEAD in the remainder of this section. Due to space limitation, we omit description of SuperSEAD since it involves complex usage of authentication trees. We give a brief introduction of one-way hash chains, then provide an overview of SEAD, including its assumptions, protocol details, security properties, and some limitations.

3.1 One-Way Hash Chains

A one way hash function, $h()$, is a function such that for each input x it is easy to compute $y = h(x)$, but given y and $h()$ it is computationally infeasible to compute x such that $y = h(x)$ [13].

A one way hash chain of a length n, denoted by $hc(x, n)$, can be constructed by applying $h()$ on a seed value x iteratively n times, i.e., $h^i(x) = h(h^{i-1}(x))$ for $i \geq 2$. Thus, $hc(x, n) = (h(x), h^2(x), \ldots, h^n(x))$.

It follows from the definition of a one way hash chain that given $h^i(x), h^j(x) \in hc(x, n)$ and $i < j$, it is easy to compute $h^j(x)$ from $h^i(x)$, i.e., $h^j(x) = h^{j-i}(h^i(x))$, but it is computationally infeasible to compute $h^i(x)$ from $h^j(x)$.

3.2 Assumptions

As any other secure routing protocol, SEAD requires cryptographic secrets for entity and message authentication. Public key infrastructure or pair-wise shared keys can meet such requirement. Other key establishment mechanisms can also be used. For simplicity, we assume that each node (v_i) has a pair of public key (V_{v_i}) and private key (S_{v_i}). Each node's public key is certified by an central authority trusted by every node in the network. To minimize computational overhead, every node also establishes a different secret key shared with every other node in the network. A secret key shared between v_i and v_j is denoted $K_{v_i v_j}$.

A network *diameter*, k, is defined as the maximum distance between any two nodes in the network. Given a network $G = (V, E)$, $k = max\{d(u, v)|u, v \in V\}$. It would be ideal if a routing protocol can scale to a large network without a limitation on its boundary. However, a distance vector routing protocol is usually used for a small or medium size network. Thus, it is realistic to assume that the diameter of a network is limited to k_m. Other distance vector approaches may also have such assumption. For example, RIP [11] assumes $k_m = 15$. Any node located 15 hops away is treated as unreachable.

3.3 Review of SEAD Protocol Details

SEAD authenticates the sequence number and the distance of a route with an authentication value which is an element from a hash chain. Specifically, to advertise a route $r_{v_i}(v_d, seq, d(v_i, v_d))$, v_i needs to include an authentication value $aut(r_{v_i})$ to allow a recipient to verify the correctness of r_{v_i}.

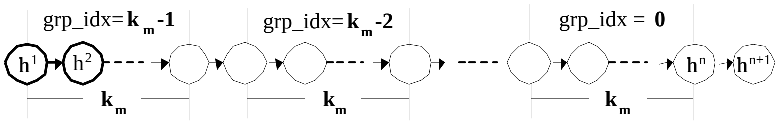

Fig. 1. A hash chain is arranged into groups of k_m elements.

The following process illustrates how SEAD works:

1. $\forall v_i \in V$, v_i constructs a hash chain from a secret x_i, $hc_{v_i}(x_i, n+1) = (h^1(x_i), h^2(x_i), \ldots, h^{n+1}(x_i))$. Let s_m be the maximum sequence number. We assume $n = s_m \cdot k_m$ for convenience of presentation. Arrange $hc_{v_i}(x_i, n+1)$, or simply hc_{v_i}, into s_m groups of k_m elements. The last element $h^{n+1}(x_i)$ is not in any group and is referred as the *anchor* of hc_{v_i}. Each group is assigned an integer in the range $[0, k_m - 1]$ as its index. We number the groups from right to left (Figure 1). The hash elements within a group are numbered from left to right starting from 0 to $k_m - 1$. This way, each hash element $h^j(x_i)$ can be uniquely located within hc_{v_i} by two numbers a, b, where a is the index of the group which $h^j(x_i)$ is in and b is the index of the element within the group. We use $hc_{v_i}[a, b]$ to represent $h^j(x_i)$, where $j = (s_m - a) \cdot k_m + b + 1$.
2. $\forall v_i \in V$, v_i makes $h^{n+1}(x_i)$ accessible to every other node in the network. Many methods can be used. For example, v_i can publish $h^{n+1}(x_i)$ in a central directory, signing it with v_i's private key. Another method is to broadcast to the whole network $h^{n+1}(x_i)$ along with v_i's digital signature. The result is that every node in the network has a copy of $h^{n+1}(x_i)$ and can trust that it is the anchor value of a hash chain constructed by v_i.
3. $\forall v_i \in V$, v_i advertises a route r_{v_i} for v_k with a distance of d and a sequence number of s, $r_{v_i} = (v_k, s, d)$. To support r_{v_i}, v_i includes an authentication value $aut = hc_{v_k}[s, d]$ with r_{v_i}.

$$v_i \to N(v_i) : \ r_{v_i}(v_k, s, d, aut), \ aut = \begin{cases} hc_{v_i}[s, 0] & \text{if } v_k = v_i \\ hc_{v_k}[s, d] & \text{if } v_k \neq v_i \end{cases}$$

4. Upon receiving an advertised route $r_{v_i}(v_k, s, d, aut)$, v_j validates d and s using the one-way hash chain. We know that aut should be $hc_{v_k}[s, d]$, or $h_{v_k}^{(s_m - s)\cdot k_m + d + 1)}(x_k)$. Given the anchor of $hc_{v_k} = h_{v_k}^{n+1}(x_k) = h_{v_k}^{s_m \cdot k_m + 1}$, it is easy to confirm if $aut = hc_{v_k}[s, d]$ by applying $h()$ on aut for x times, where $x = (s_m \cdot k_m + 1) - [(s_m - s) \cdot k_m + d + 1] = s \cdot k_m - d$. If $aut = hc_{v_k}[s, d]$, then $r_{v_i}(v_k, s, d, aut)$

is treated valid. Otherwise, invalid. In the former case, r_{v_i} is used to update the existing route in v_j's routing table for v_k, let's say $r_{v_j}(v_k, s', d', aut')$ if 1) $s > s'$ or 2) $s = s'$ and $d < d'$. In either case, d', s' and aut' are replaced with $d + 1, s$ and $h(aut)$ respectively.

3.4 Security Analysis

SEAD has a number of desirable security properties (Table 1):

1. Message and identity authentication.
2. Sequence number authentication. Provided there are no two nodes in collusion, a bad node cannot corrupt another node's routing table by advertising a route with a sequence number greater than the latest one originated by the destination of that route.
3. Cost metric authentication. Provided there are no two nodes in collusion, a bad node cannot corrupt another node's routing table by advertising a route with a distance shorter than the one it learns from one of its neighbors.
4. Partially Resilient to collusion. Given a group of colluding nodes, the shortest distance they can claim to a destination without being detected is the shortest distance from any node in the colluding group to that destination. For example, if u, v are in collusion, and u, v are 3 and 5 hops away from x respectively. The shortest distance to x which u and v can claim is 3-hop. Thus, we say that SEAD is partially resisting to collusion given that colluding nodes cannot claim an arbitrary distance to a destination as they will.

Security Property		SEAD	superSEAD	S-DSDV
Data Integrity		$\checkmark$	$\checkmark$	$\checkmark$
Data Origin Authentication		$\checkmark$	$\checkmark$	$\checkmark$
Destination Authentication		$\checkmark$	$\checkmark$	$\checkmark$
Sequence Number Authentication	larger	$\checkmark$	$\checkmark$	$\checkmark$
	smaller	$\times$	$\times$	$\checkmark$
Cost Metric Authentication	longer	$\times$	$\times$	$\checkmark$
	same	$\times$	$\checkmark$	$\checkmark$
	shorter	$\checkmark$	$\checkmark$	$\checkmark$
Resisting to 2-node collusion		$\diamond$	$\diamond$	$\times$

Table 1. Comparison of Security Properties: $\times$ - not supported; $\diamond$ - partially supported; $\checkmark$ - fully supported;

Despite its distinguishable security properties, SEAD has some limitations.

1. *Vulnerable to longer distance fraud fraud.* A misbehaving node can advertise a route with a distance longer than the actual distance of that route without being detected. For example, a node i located k hops away from j can successfully advertise a route for j with a distance $d > k$. This is possible because i which receives a hash $h^{k-1}()$ can compute it forward as many times as it likes and use a computed hash value (i.e., $h^d()$) to support its claim of distance d.
2. *Vulnerable to lower sequence number fraud.* A misbehaving node i can advertise a sequence number lower than the one it receives. Thus, i may be able to advertise a route with a shorter distance by lowering its sequence number.
3. *A risk window.* SEAD has a risk window of p_1, where p_1 is the interval of periodic routing update. For example, a node i which had been k hops away from j can still claim that distance when it actually has moved further away from j since i has the authentication value $h^k()$ to support its claim. Such claim would continue to be valid until a victim receives an advertised route for j from other nodes with a newer sequence number. Although such risk window is usually short (e.g., 15 seconds in SEAD), it is still desirable to minimize it.

4 A Threat Model

There are many threats against a routing protocol. In this section, we discuss these threats and identify those of our interest. We first clarify the difference between a routing protocol and a routing algorithm, which are often not clearly addressed in literature.

4.1 Threat Targets

The primary objective of a network layer is to provide routing functionality to allow non-directly connected nodes to communicate with each other. Thus, two fundamental functions are required for a router:

1. Establishing valid routes (usually stored in a routing table) to each destination in a network. Although routing tables for a small and relatively static network can be manually created and maintained, it is desirable to automate the task. Automatic mechanisms for building and updating routing tables are often referred to as route propagation mechanisms or routing protocols.
2. Routing datagrams to a next hop(s) leading to their ultimate destinations. Such function is often referred to as routing algorithms, which selects next hops for a datagram given a set of established routes and other factors (e.g., routing policies). Example routing strategies include, but not limited to: 1) routing datagrams to a default gateway; 2) routing datagrams over shortest paths; 3) routing datagrams equally over multiple paths; 4) stochastic routing.

Although these two functions are equally important and both deserve attentions, this paper only considers threats against automatic route propagation mechanisms, specifically, DSDV. A routing protocol is usually built upon other protocols (e.g., IP, TCP, or UDP). Thus, it is vulnerable to all threats against its underlying protocols (e.g., IP spoofing). In this paper, we do not consider the threats against the underneath protocols. However, some of the inherited threats can be mitigated by proposed cryptographic mechanisms.

4.2 Threat Sources

In a wireline network, threats can be from a network node or a network link (i.e., an attacker is in control of that link). Attacks from a controlled link include modification, deletion, insertion, or replay of routing update messages. In MANET, attacks from network links are less interesting due to the broadcast nature of wireless networks. It appears difficult, if not impossible, for an attack to modify or delete a message (m), i.e., to stop the neighbors of the originator of m from receiving untampered m. However, insertion and replay are still possible. For simplicity, we model a compromised network link as an adversary node. Thus, a network $G = (V, E)$ becomes $G = (V_g, V_b, E)$, where V_g is a set of well-behaved nodes, V_b is a set of misbehaving nodes, and E is a set of edges. $V = V_g \cup V_b$. A misbehaving node can be a compromised legitimate node (i.e., with legitimate cryptographic credentials), namely an *insider*, or an illegitimate node brought to the network by an attacker (i.e., without any legitimate cryptographic credentials), namely an *outsider*.

4.3 Individual Threats

Barbir, Murphy and Yang [2] identified a number of generic threats to routing protocols, including *Deliberate Exposure, Sniffing, Traffic Analysis, Interference, Overload, Spoofing, Falsification, Byzantine Failures* (Table 2). We consider falsification as one of the most serious threats to DSDV due to the fact that each node builds its own routing table based on other nodes' routing tables. This implies that a single misbehaving node may be able to compromise the whole network by spreading falsified routing updates. Our proposed S-DSDV can defeat this serious threat by containing a misbehaving node (i.e., by detecting and stopping misinformation from further spreading).

5 S-DSDV

In this section, we present the details of S-DSDV, which can prevent any distance fraud, including longer, same, or shorter, provided that there are no two nodes in collusion.

Generic Threats	Addressed by S-DSDV?
Deliberate Exposure	$\times$
Sniffing	$\times$
Traffic Analysis	$\times$
Byzantine Failures	$\diamond$
Interference	$\diamond$
Overload	$\checkmark$
Falsification by Originators	$\checkmark$
Falsification by Forwarders	$\checkmark$

Table 2. Routing Threats: $\times$ - no; $\diamond$ - partially; $\checkmark$ - fully;

5.1 Cryptographic Assumptions

As any other secure routing protocol, S-DSDV requires cryptographic mechanisms for entity and message authentication. Any security mechanisms providing such security services can meet our requirements, e.g., pair-wise shared secret keys, public key infrastructure (PKI), etc. Thus, S-DSDV has similar cryptographic assumptions as SEAD (see §3.2) and S-AODV (requiring PKI). For convenience, we assume that every node ($v_i \in V$) shares with every other node ($v_j \in V, i \neq j$) a different pair-wised secret key (k_{ij}). Combined with message authentication algorithms (e.g., MD5), pair-wise shared keys provide entity and message authentication. Thus, all messages in S-DSDV are cryptographically protected. For example, when i sends a message m to j, i also sends to j the Message Authentication Code (MAC) of m generated using k_{ij}.

5.2 Notations

We use $r_u(w) = (w, seq(u,w), cst(u,w), nhp(u,w))$ to denote the route from u to w, where $seq(u,w)$ denotes the sequence number of $r_u(w)$, $cst(u,w)$ denotes the cost of $r_u(w)$, and $nhp(u,w)$ denotes the next hop of $r_u(w)$. With no ambiguity, we also use (w, seq, cst), (w, seq, cst, nhp), or (w, seq_u, cst_u, nhp_u) to denote $r_u(w)$.

5.3 Route Classification

We classify routes $R_u = \{r_u\}$ advertised by node u into two categories: 1) those that u is authoritative of, denoted by R_u^{auth}; and 2) those that u is unauthoritative of, denoted by R_u^{naut}. $R_u = R_u^{auth} \cup R_u^{naut}$.

Definition 1 *(***Authoritative Routes***). Given a route $r_u = (w, seq, cst)$, $r_u \in R_u^{auth}$ if 1) $w = u$ and $cst = 0$; or 2) $cst = \infty$.*

It is obvious that u is authoritative of r_u if r_u is a route for u itself with a distance of zero. We also say that u is authoritative of r_u if r_u is an unreachable route. This is because u has the authority to assert the unavailability of a route from u to any other node w even there factually exists such a path between u and w. This is equivalent to the case that u implements a local route selection policy which filters out traffic to and from w. We believe that a routing protocol should provide such flexibility for improving security since u may have its own reasons to distrust w. BGP [15] is a good example which allows for local routing policies. However, this feature should not be considered the same as malicious packet dropping [12, 10]. In the latter case, a node promises to forward packets to another node (i.e., announcing reachable routes to that node) but fails to do so.

Definition 2 *(***Non-Authoritative Routes***). Given a route $r_u = (w, seq, cst)$, $r_u \in R_u^{naut}$ if $w \neq u$ and $0 < cst < \infty$.*

If u advertises a reachable route r_u for another node w, we say that u is not authoritative of r_u since u must learn r_u from another node, i.e., the next hop from u to w along the route r_u.

5.4 Route Validation

When a node v receives a route r_u from u, v validates r_u based on the following rules.

Rule 1 *(***Validating Authoritative Routes***). If u is authoritative of r_u, a recipient node v validates the message authentication code (MAC) of r_u. If it succeeds, v accepts r_u. Otherwise, v drops r_u.*

Since u is authoritative of r_u, v only needs to validate the data integrity of r_u, which includes data origin authentication [13]. If it succeeds, v accepts r_u since it in fact originates from u and is not tampered. Otherwise, r_u may have originated from a node impersonating u or it may have been modified by unauthorized parties. Thus, r_u should be ignored.

Rule 2 *(***Validating Non-Authoritative Routes***). If u is unauthoritative of r_u, a recipient node v validates the data integrity of r_u. If it succeeds, v additionally validates the consistency (defined by Definition 3) of r_u. If it succeeds, v accepts r_u. Otherwise, v drops r_u.*

Since u is unauthoritative of r_u, we require that v should not accept r_u right away even if the validation of data integrity succeeds. Instead, v should check the consistency with the node which r_u is learned from. Ideally, v should consult with the authority of r_u if it exists. Such authority should have perfect knowledge of network topology and connectivity (i.e., it knows the every route and its associated cost from every node to every other node in a network). Such authority may exist for a small static network. However, it does not exist in a dynamic wireless ad hoc network where nodes may move frequently.

Since there does not exist an authority that can tell the correctness of r_u, we propose that v should consult with the node which r_u is learned from. Such node should have partial authority of r_u. This method is analogous to the way human beings acquire their trust. Let x be a student, and y be x's supervisor. Suppose x receives an email from z, saying that y has called for a research meeting tomorrow. x must trust the email if z is y since y is authoritative of arranging such a meeting. However, if z is a student, x may need to confirm the message with another person, ideally with y, or possibly with another student.

Definition 3 *(***Consistency***) Given a network $G = (V, E)$, let $u, v, w \in V$ and link $e(u, v) \in E$. For two routes $r_u(w) = (w, seq(u, w), cst(u, w))$, $r_v(w) = (w, seq(v, w), cst(v, w))$, and $r_u(w)$ is directly computed from $r_v(w)$. We say that $r_u(w)$ and $r_v(w)$ are* **consistent** *if 1) $seq(u, w) = seq(v, w)$; and 2) $cst(u, w) = cst(v, w) + cst(u, v)$.*

From the definition, we know that r_u and r_v are consistent if r_u is directly computed from r_v following DSDV specifications: 1) the sequence number should not be changed during route propagation; 2) the cost metric of r_u should be the sum of the cost metrics of r_v and $e(u, v)$. To complete a consistency check, a node needs to talk to another node in 2-hop away by route requests and route responses. To obtain such information, we require that the next hop of a route should be advertised along with that route. For example, if u learns a route $r_u(w)$ from v, u should advertise $r_u(w) = (w, seq(u, w), cst(u, w), nhp(u, w))$, where $nhp(u, w) = v$.

To check the consistency of $r_u(w) = (w, seq(u, w), cst(u, w), v)$, a node x sends a route request to v, asking for v's route entry for w, which is $r_v(w) = (w, seq(v, w), cst(v, w), nhp(v, w))$. In addition, x also asks v's route entry for u, which is $r_v(u) = (u, seq(v, u), cst(v, u), nhp(u, v))$. Assuming $cst(v, u) = cst(u, v)$, $cst(v, u)$ allows x to check the consistency of $cst(u, w)$ and $cst(v, w)$. $nhp(v, u)$ allows x to check if u is directly connected with v, i.e., if $nhp(v, u) = u$.

5.5 Protocol Summary

The following process illustrates how S-DSDV works:

1. $\forall u, w \in V$, u advertises a route $r_u = (w, seq, cst, nhp)$ for w. Note r_u is protected by MAC.
2. Upon receiving from u a route r_u, $x \in V$ validates the MAC of the message carrying r_u. If it fails, the message is dropped. Otherwise, x further determines if u is authoritative of r_u (Definition 1). If yes, x accepts r_u. Otherwise, x checks the consistency of r_u with its next hop node (nhp), let's say v (see Step 3). If it succeeds, r_u is accepted. Otherwise, it is dropped.
3. x sends a route request to v (likely via u), asking $r_v(w)$ and $r_v(u)$. v should send back a route response containing its route entries for w and u. Upon receiving $r_v(w)$ and $r_v(u)$, x can perform consistency check of $r_u(w)$ and $r_v(w)$ according to Definition 3. Note u may manipulate x's route request and/or v's route response. However, such misbehavior will not go unnoticed since all message are MAC-protected.

5.6 Security Analysis

In this section, we analyze security properties of S-DSDV. We hope that our security analysis methodology can lead to a common framework for analyzing and comparing different securing routing proposals.

Theorem 1 (Data Integrity) *In S-DSDV, data integrity is protected.*

Justification. S-DSDV uses pair-wise shared keys with Message Authentication Code (MAC) to protect integrity of routing updates. A routing update message with a invalid MAC can be detected.

Remarks. Data integrity can prevent unauthorized modification and insertion of routing updates. However, it cannot prevent deletion or replay attacks. Thus it partially counters the threat of interference [2].

Theorem 2 (Data Origin Authentication) *In S-DSDV, data origin is authenticated.*

Justification. S-DSDV uses pair-wise shared keys with Message Authentication Code (MAC) to protect integrity of routing updates. Since every node shares a different key with every other node, a correct MAC of a message also indicates that the message is originated from the only other party the recipient shares a secret key is with. Thus, data origin is authenticated.

Remarks. Data origin authentication can prevent node impersonation since any node without holding the key materials of x cannot originate messages using x as the source without being detected. It can also thwart the threat of falsification by originators [2].

Given a route update $r = (cst, seq, cst, nhp)$ in S-DSDV, the threat of falsification by forwarders can be instantiated as follows: 1) falsifying the destination cst; 2) falsifying the sequence number seq; 3) falsifying the cost metric, or the distance, cst; 4) falsifying the next hop nhp.

Lemma 1 (Destination Authentication) *In S-DSDV, a route with a falsified destination can be detected.*

Justification. Since S-DSDV assumes a pair-wised shared secret keys, we know that $\forall u, v \in V$ and $u \neq v$, u shares a secret key with v. If a destination node (x) in r is falsified or illegitimate, then $\forall u \in V$, u does not share a secret with x. Thus, x is detected as an illegitimate node.

Lemma 2 (Sequence Number Authentication) *In S-DSDV, an advertised route r with a falsified sequence number can be detected provided there is at most one bad node in the network.*

Justification. Suppose b is the only bad node in the network. b advertises $r_b = (x, seq_b, cst_b, nhp)$ to all of its direct neighbors $N(b)$, where seq is falsified (i.e., it is different from the value b learns from nhp). Since there is at most one bad node (b) in the network, $\forall u \in V, u \neq b$, u is a good node. Obviously, every of b's direct neighbors is good, including nhp. Thus, $\forall v \in N(b), v \neq nhp$, v will check the consistency of seq with nhp. Since nhp is a good node, it will provide a correct sequence number which will be inconsistent with seq_b if seq_b is faked. Therefore, the statement is proved.

Lemma 3 (Cost Metric Authentication) *In S-DSDV, an advertised route r with a falsified cost metric (or distance) can be detected if there is at most one bad node in the network.*

Justification. Since a good node can uncover misinformation from a bad node by cross checking its consistency with a good node, a falsified cost metric always causes inconsistency, thus can be detected (see justification for Lemma 2).

Lemma 4 (Next Hop Auth) *In S-DSDV, an advertised route r with a falsified next hop can be detected if there is at most one bad node in the network.*

Justification. Let b is the only bad node in the network, which advertises $r = (x, seq, cst, nhp)$. We say nhp is falsified if: 1) $nhp \notin V$; or 2) $nhp \notin N(b)$; or 3) $nhp \in N(b)$ but r is not learned from nhp. If $nhp \notin V$, it can be detected since a legitimate node does not share a secret key with nhp. If $nhp \notin N(b)$, nhp will report a node $a \neq b$ as its next hop to b. If r is not learned from nhp, nhp will report a route to x with a distance inconsistent with cst. Therefore, Lemma 4 is proved.

Theorem 3 (Routing Update Authentication) *In S-DSDV, a routing update with misinformation can be detected provided there is at most one bad node in a network.*

Justification. A routing update R consists of a number of routes with one for each destination. Based on Lemma 1, 2, 3, and 4, we know $\forall r \in R$, any misinformation in r can be detected if there is at most one bad node in the network. Therefore, we conclude that misinformation in a routing update, which is a collection of r, can be detected.

Theorem 4 (Multiple Bad Nodes) *In S-DSDV, a routing update with misinformation can be detected with a high probability provided that no two nodes are in collusion.*

Justification. Let x be the router advertising a route r_x. Let y be the next hop router of r_x, and r_y be the route provided by y when it is consulted. If there are multiple bad nodes in the network, it is possible that both x and y are bad. We say x and y are in collusion if y intentionally covers x's misbehavior, i.e., y intentionally provides a falsified r_y so that it is consistent with r_x. In this case, consistency check with only one node will fail. $n + 1$ nodes need to be consulted to detect a collusion involving n nodes.

If y is not colluding with x, it is also possible that y provides falsified r_y which happens to be consistent with r_x if y is bad. We look at the probability of such event. For r_x and r_y to be consistent, both of their sequence numbers and cost metrics must be consistent, i.e., $seq_x = seq_y$ and $cst_x = cst_y + 1$. Suppose the maximum sequence number is s_m and the network diameter is k_m. The probability that r_x and r_y are consistent, denoted by $p(r_x \doteq r_y)$, is equivalent to the probability of drawing two same pairs of (s, d) from a space of $s_m \cdot k_m$, which is $\frac{1}{s_m \cdot k_m}$. If $s_m = 2^{32}$ and $k_m = 32$, we know that $p(r_x \doteq r_y)$ is extremely low. In practical, a misbehaving node may not use a sequence number too far away from a correct one (denoted by s_c) since it is easy to get caught. Suppose the space of a faked sequence number s is 6 ($s_c - 3 \leq s \leq s_c + 3$), then $p(r_x \doteq r_y) = \frac{1}{6 \times 32} = 0.53\%$, which is still low. Thus, the probability that r_x and r_y are detected as inconsistent is high.

6 Efficiency Analysis

We analyze routing overhead caused by S-DSDV (S-DSDV overhead) and compare it with those caused by DSDV, SEAD, and superSEAD. We use the notations in Table 3 for efficiency analysis.

Notation	Description	Value
n	total number of nodes in a network	50
T	simulation time	900 seconds
φ	routing update interval	15 seconds
L_{udp_hdr}	length of a UDP header	8 bytes
L_{ip_hdr}	length of an IP header	20 bytes
L_{hash}	length of a hash from a hash function	10 bytes
L_{dsdv_rt}	length of a DSDV route entry	10 bytes
L_{sead_rt}	length of a SEAD route entry	20 bytes
L_{ssead_rt}	length of a SuperSEAD route entry	80 bytes
L_{sdsdv_rt}	length of an S-DSDV route entry	14 bytes
O_{dsdv_ppu}	DSDV overhead per periodic routing update	528 bytes
O_{dsdv_ptu}	DSDV overhead per triggered routing update	38 bytes
O_{sead_ppu}	SEAD overhead per periodic routing update	1028 bytes
O_{sead_ptu}	SEAD overhead per triggered routing update	48 bytes
O_{ssead_ppu}	superSEAD overhead per periodic routing update	4612 bytes
O_{ssead_ptu}	superSEAD overhead per triggered routing update	118 bytes
O_{sdsdv_ppu}	S-DSDV overhead per periodic routing update	728 bytes
O_{sdsdv_ptu}	S-DSDV overhead per triggered routing update	42 bytes
O_{sdsdv_pcc}	S-DSDV overhead per consistency check	168 bytes
U_{pd}	total number of periodic routing updates	$*$
U_{tg}	total number of triggered routing updates	$*$
U_{pc}	total number of S-DSDV periodic consistency checks	$*$
U_{tc}	total number of S-DSDV consistency checks	$*$
O_{dsdv}	total DSDV overhead	†
O_{sead}	total SEAD overhead	†
O_{ssead}	total superSEAD overhead	†
O_{sdsdv_r}	total S-DSDV-R overhead	†
O_{sdsdv}	total S-DSDV overhead	†

Table 3. Notations for Efficiency Analysis ($*$ - obtained by simulation; † - dependent on $*$ values)

6.1 Analysis Methodology

We adopt a method of using both analysis and simulation for comparing routing overhead. Analysis has the advantage that it is easy for others to verify our results. Simulation has the advantage of dealing with the implications of random events which are difficult to obtain by analysis.

To analyze routing overhead, we need to obtain the total number of routing updates generated by all nodes in a network during a time period of T. In DSDV, there are two types of routing updates: 1) periodic routing updates; and 2) triggered routing updates (e.g., by broken links). In theory, the total number of periodic routing updates (U_{pd}) can be calculated. However, the total number of triggered updates (U_{tg}) cannot be easily calculated since they are related to random events, i.e., broken links caused by node movement. In the absence of an analytic method for computing the number of broken links resulting from a node mobility pattern, we use simulation to obtain U_{tg}. We also use simulation to obtain U_{pd} since it is affected by U_{tg} in the DSDV implementation in NS-2 [4]. For simplicity, we use the following assumptions and notations:

1. DSDV, SEAD, and S-DSDV run over UDP and IP. 802.11 MTU (Maximum Transfer Unit) is 1500 bytes. A routing update message including IP and UDP headers larger than 1500 bytes is split into multiple messages. There is no network congestion, packet dropping, or retransmission.

2. Each triggered routing update consists of a single entry for a route involved in the triggering event. If there are multiple routes affected by that event, multiple triggered routing updates are generated.
3. A DSDV route entry consists of a destination (4-byte), a sequence number (4-byte), and a cost metric (2-byte). Thus, $L_{dsdv\_rt} = 10$ bytes.
4. A SEAD route entry consists of a DSDV route entry plus a field of length L_{hash} for holding an authentication value. In this paper, we assume $L_{hash} = 80$ bits (10 bytes). Thus, $L_{sead\_rt} = 20$ bytes.
5. A superSEAD route entry consists of a DSDV route entry plus (k+1) fields of length L_{hash} holding authentication values, where $k = lg(n)$ $(lg \equiv log_2)$. In this paper, $k = lg(64) = 6$. Thus, $L_{ssead\_rt} = L_{dsdv\_rt} + (k+1) \times L_{hash} = 10 + (6+1) \times 10 = 80$ bytes.
6. An S-DSDV route entry consists of a DSDV route entry plus a 4-byte length field holding the identity of a next hop node. Thus, $L_{sdsdv\_rt} = L_{dsdv\_rt} + 4 = 14$ bytes.
7. An S-DSDV consistency check involves a route request and a response message; each message has an S-DSDV route entry (plus IP and UDP headers), and traverses two hops. Thus, routing overhead generated per consistency check is $O_{sdsdv\_pcc} = (L_{sdsdv\_rt} + L_{ip\_hdr} + L_{udp\_hdr}) \times 4 = 168$ bytes.

We expected and observed that S-DSDV produces high network overhead since it checks the consistency of a route whenever it is updated for sequence number, distance, or the next hop. Since the sequence number changes persistently, a large number of consistency checks are triggered. To reduce S-DSDV overhead, we introduce a variation of S-DSDV, namely, S-DSDV-R. S-DSDV-R checks the consistency of a route when it is first installed in a routing table. A timer is set for that route when a consistency check is performed for that route. In our simulation, the timer interval is same as the routing update interval. A new consistency check is only performed for a route when its consistency check timer expires. One security vulnerability of S-DSDV-R is that a falsified route may be accepted during the interval of two consistency checks. This is similar to the risk window of SEAD and superSEAD (§3.4). We use the following equations to calculate network overhead of each protocol:

$$O_{dsdv} = O_{dsdv\_ppu} \cdot U_{pd} + O_{dsdv\_ptu} \cdot U_{tg} \tag{1}$$

$$O_{sead} = O_{sead\_ppu} \cdot U_{pd} + O_{sead\_ptu} \cdot U_{tg} \tag{2}$$

$$O_{ssead} = O_{ssead\_ppu} \cdot U_{pd} + O_{ssead\_ptu} \cdot U_{tg} \tag{3}$$

$$O_{sdsdv\_r} = O_{sdsdv\_ppu} \cdot U_{pd} + O_{sdsdv\_ptu} \cdot U_{tg} + O_{sdsdv\_pcc} \cdot U_{pc} \tag{4}$$

$$O_{sdsdv} = O_{sdsdv\_ppu} \cdot U_{pd} + O_{sdsdv\_ptu} \cdot U_{tg} + O_{sdsdv\_pcc} \cdot U_{tc} \tag{5}$$

6.2 Simulation Results

We use simulation to obtain U_{pd}, U_{tg}, U_{pc}, and U_{tc}. We simulate a network with $n = 50$ mobile nodes for T = 900 seconds. Different pause times represent different dynamics of a network topology. A pause time of 0 seconds represents a constantly changing network, while a pause time of 900 seconds represents a static network. Simulation results for parameters as given in Table 3 and 4 are illustrated in Figure 2.

6.3 Discussions

S-DSDV produces higher network overhead than superSEAD due to the overhead of significant number of consistency checks, which we see as the price paid for improved security. S-DSDV-R significantly reduces the network overhead, but introduces a risk window similar to the one of SEAD and superSEAD. However, such risk window can be managed by adjusting the value of the consistency check timer. Overall, we think S-DSDV-R provides a desirable balance between security and efficiency.

Pause	Periodic Updates	Triggered Updates	S-DSDV Consistency	S-DSDV-R Consistency
Time (T)	(U_{pd})	(U_{tg})	Checks (U_{tc})	Checks (U_{pc})
0	12 863	46 914	450 396	69 863
100	7 640	69 976	268 038	65 894
200	5 764	72 021	205 282	63 407
300	5 588	51 090	197 470	62 079
400	5 382	51 088	191 729	62 154
500	5 279	45 368	189 338	61 644
600	4 788	44 673	168 427	59 977
700	4 659	41 566	167 387	59 683
800	4 844	44 322	172 441	59 866
900	3 708	39 334	135 706	59 250

Table 4. Averaged values obtained from 6 simulation runs for the total number of periodic and triggered routing updates, total number of consistency checks of S-DSDV and S-DSDV-R.

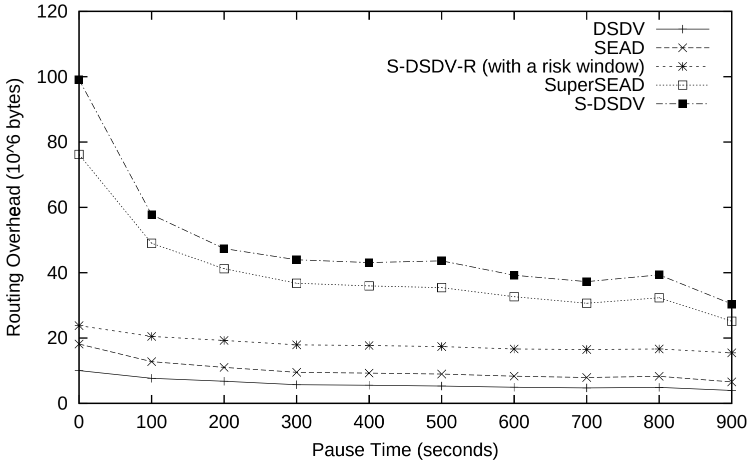

Fig. 2. Overhead of DSDV, SEAD, S-DSDV-R, SuperSEAD, and S-DSDV. Note that S-DSDV has higher network overhead than superSEAD due to significant number of consistency checks but offers better security. S-DSDV-R significantly reduces network overhead, but introduces a risk window similar to those of SEAD and superSEAD.

7 Concluding Remarks

We propose to use consistency checks for validating DSDV routing updates. Information required for consistency checks can be obtained out-of-band (i.e., by route requests and responses), or in-band (i.e., included within a routing update). Let u,v,w be three nodes. When u advertises a route r_u to v, it digitally signs r_u. When v computes a route r_v based on r_u and sends r_v to w, v digitally signs r_v. In addition, v also forwards r_u along with u's signature to w. In this way, w can perform consistency check of r_v and r_u. Since in-band mechanism involves generation and verification of digital signatures, it increases computational overhead and may subject to denial of service attacks. We plan to generalize the mechanisms for validating routing updates, and apply to other routing protocols.

Acknowledgment

The first author is supported in part by Alcatel Canada, MITACS (Mathematics of Information Technology and Complex Systems), and NCIT (National Capital Institute of Telecommunications). The second author is supported in part by MITACS and NSERC (Natural Sciences and Engineering Research Council of Canada). The third author is Canada Research Chair in Network and Software Security, and is supported in part by NCIT, a NSERC Discovery Grant, and the Canada Research Chairs Program.

References

[1] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security* (WiSe), September 2002.

[2] A. Barbir, S. Murphy, and Y. Yang. Generic Threats to Routing Protocols. Internet Draft, April 13, 2004.

[3] S. Bellovin. Security Problems in the TCP/IP Protocol Suite. *ACM Computer Communications Review*, 19(2): 32-48, Apr 1989.

[4] K. Fall and K. Varadhan, editors. The *ns* Manual (formerly *ns* Notes and Documentation). April 14, 2002. http://www.isi.edu/nsnam/ns/doc/index.html

[5] J.J. Garcia-Luna-Aceves and S. Murthy. A Loop-Free Algorithm Based on Predecessor Information. In *Proceedings of IEEE INFOCOM* , Boston, MA, USA. April 1995.

[6] M.T. Goodrich. Efficient and Secure Network Routing Algorithms. *Provisional Patent Filing*, USA. 2001.

[7] Y.C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (MobiCom 2002)*, September 23-28, 2002.

[8] Y.C. Hu, D.B. Johnson, and A. Perrig. Secure Efficient Distance Vector Routing Protocol in Mobile wireless Ad Hoc Networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, June 2002.

[9] Y.C. Hu, D.B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *Ad Hoc Networks Journal*, 1 (2003):175-192.

[10] M. Just, E. Kranakis, and T. Wan. Resisting Malicious Packet Dropping in Wireless Ad Hoc Networks. In *Proceedings of 2nd Annual Conference on Adhoc Networks and Wireless (ADHOCNOW'03)*, Montreal, Canada, Oct 09-10, 2003.

[11] G. Malkin. RIP Version 2. RFC 2453 (standard). November 1998.

[12] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, August 2000.

[13] A.J. Menezes, P.C. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[14] C.E. Perkins and P.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*, August 1994.

[15] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4), RFC 1771, March 1995.

[16] R. Rivest. The MD5 Message-Digest Algorithm, RFC 1321, April 1992.

[17] B.R. Smith, S. Murphy, and J.J. Garcia-Luna-Aceves. Securing Distance-Vector Routing Protocols. In *Proceedings of 1997 Internet Society Symposium on Network and Distributed System Security (NDSS'97)*, San Diego, USA. February 1997.

[18] M.G. Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe 2002)*, September 2002.

[19] Y.G. Zhang, W. Lee and Y.A. Huang. Intrusion Detection in Wireless Ad-Hoc Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, August 2000.

[20] L. Zhou and Z.J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6), Nov/Dec 1999.

[21] L. Eschenauer and V. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conferences on Computer and Communication Security (CCS'02)*, Washington, D.C., USA. November 18-22 2002.