Reliable Distance Vector routing protocol to handle Blackhole and Selfish ($RDVBS$) nodes in Ad hoc Networks

SANDHYA KHURANA
and
NEELIMA GUPTA
Department of Computer Science, University of Delhi, Delhi, India

Properties of Mobile Ad hoc Networks ($MANETs$) present major vulnerabilities in security. The threats considered in $MANETs$ may be due to maliciousness of a node that intentionally disrupts the network by using variety of attacks and/or due to selfishness of the node which does not perform certain operations due to a wish to save power. None of the existing algorithms to mitigate black hole attacks handles selfish nodes. We present a first such algorithm which handles blacknode attack as well as selfish nodes. A co-operative security scheme called Reliable Ad hoc On-demand Distance Vector routing protocol has been proposed to solve the problem of attack by Blackhole node as well as Selfish behavior ($RDVBS$) . $RDVBS$ behaves like $AODV$ in the absence of attack with only a slight increase in the routing overhead and, detects and isolates black nodes and selfish nodes in the presence of attack. It also recovers from the attack when a black node leaves the network or a selfish node becomes good. Our protocol is also superior to the previous known algorithms in terms of routing overhead. The protocol also handles multiple and cooperative black nodes.

Keywords: Ad hoc Network; Security; Attack; Blackhole; Selfish node

1. INTRODUCTION

Ad-hoc networks [Ramanathan and Redi 2002] have been proposed to support scenarios where no wired infrastructure exists. They can be set up quickly where the existing infrastructure does not meet application requirements for reasons such as security, cost, or quality. Examples of applications for ad hoc networks range from military operations, emergency disaster relief to community networking and interaction between attendees at a meeting or students during a lecture. In Mobile Ad hoc Networks ($MANETs$) each node has limited wireless transmission range, so the routing in $MANETs$ depends on the cooperation of intermediate nodes. Two types of routing protocols have been defined for ad hoc networks: Table-driven protocol and On-demand routing protocol. Table driven protocols are proactive in nature and consume excessive network bandwidth. On the other hand, on demand routing protocol exchange routing information only when needed. Most ad hoc routing protocols rely on implicit trust-your-neighbor relationship to route packets among participating nodes. This naive trust model allows malicious nodes and selfish nodes to paralyze the network. Selfish nodes do not directly damage other nodes but their effect should not be underestimated.

We propose a co-operative security scheme which we call 'Reliable Distance Vector routing protocol to handle Blackhole and Selfish nodes ($RDVBS$)' to mitigate attacks by blackhole and selfish nodes. $RDVBS$ provides a foundation for secure operations with little impact on existing protocols and can be used in bandwidth constrained nodes. The existing approaches to assuage the impact of blackhole nodes do not handle selfish nodes and the ones to mitigate selfish nodes do not alleviate the blackhole nodes. In [Khurana et al. 2006], we present a solution to detect blackhole nodes and selfish nodes. However, the paper does not talk of establishing an alternate secure path in case a malicious node is detected. In this work, we present first such approach that detect and isolate the both black hole attack as well as selfish nodes in a single scheme and

Author's address: S. Khurana, Department of Computer Science, University of Delhi, Delhi, India 110007.

establishing an alternate secure path.

The proposed solution mitigates multiple black holes also. We use cross-verification but do not flood the verification packets. The protocol is based on Ad hoc On Demand Vector (*AODV*) routing protocol with the assumption that nodes cannot impersonate and all other network conditions are good. It behaves like *AODV* in the absence of attack and, detects and isolates misbehaving nodes in presence of attack. The scheme allows the network to recover from the attack when a misbehaving node leaves the network or becomes good. It does not incur too much overhead as we do not flood the cross-verification packets nor does it require the nodes to listen in promiscuous mode. As the approach is deterministic attacks are detected with 100% success. The protocol does not require any fixed infrastructure as is required to implement virtual banks in incentive based schemes. We compare our algorithm with Deng et al.'s algorithm as it also uses cross-verification and show that our algorithm outperforms theirs in terms of routing overhead without affecting other parameters like end-to-end delay and packet delivery ratio.

For the ease of understanding we present our proposed solution in three phases. Phase I is a slight modification of path-discovery phase of *AODV*. Instead of keeping one route reply, we keep all the replies so that an alternate secure path can be discovered in case a blackhole node is present on a shortest path. When the source receives a route reply the reliability of the path is checked by sending verification packets in phase-II, reply to which can be generated only by the destination node. If there is a black node on the path, the destination will not receive the control packet (as there is no path from the black node to the destination) and hence no reply would be generated along that path. A secure path is established along the path through which a reply to this packet is received (of course, existence of a secure path is assumed here). Here we point out that the verification packets are not flooded but are multicast to a selected group of nodes (through which route replies were received). Once a path free of black node is discovered, in phase-III, control packets are sent periodically to maintain the reliability of the path, i.e. to detect if any selfish node has crept into the path.

2. PROBLEM STATEMENT

Blackhole attack is an active attack in which a node responds positively to a request for a shortest route even though it does not have a valid route to the destination node. The node is called **black node or blackhole node**. Since a blackhole node does not have to check its routing table it is the first one to respond to route discovery request in most cases. When data packets reach the black node it drops the packets rather than forwarding them to the destination creating a blackhole there. We call these nodes as blackhole nodes of **type 1**. Blackhole attack can be co-operative involving multiple nodes acting in coordination with each other.

Bharat Bhargava [Bhargava 2002; Wang et al. 2003] defined **blackhole** attack as *false destination sequence* attack also. In this, the blackhole node exploits the fact that the *AODV* protocol relies on the sequence number of the destination for the freshness of the route; it announces a very high sequence number thus ensuring that its route is considered as freshest and thus gets inserted in the established route. Source considers this path as freshest path and starts sending data packet through it. Again blackhole node drops the packets rather than forwarding them to the destination. We call these nodes as blackhole nodes of **type 2**.

Sometimes attacks like reducing the amount of routing information available to other nodes, failing to advertise certain routes or discarding routing packets or parts of routing packets are due to selfish behavior of a node. As the supply of power is limited, sometimes a node may wish to use its power supply for its own purposes and hence does not participate in routing operations. Such nodes are called **selfish nodes**. Selfish nodes were first discussed in [Hollick et al. 2004]. Here we define two types of selfish nodes depending upon their extent of non-cooperation in network operations.

(1) Selfish node of Type 1 uses energy only for its communication and it forwards neither control packets nor data packets.

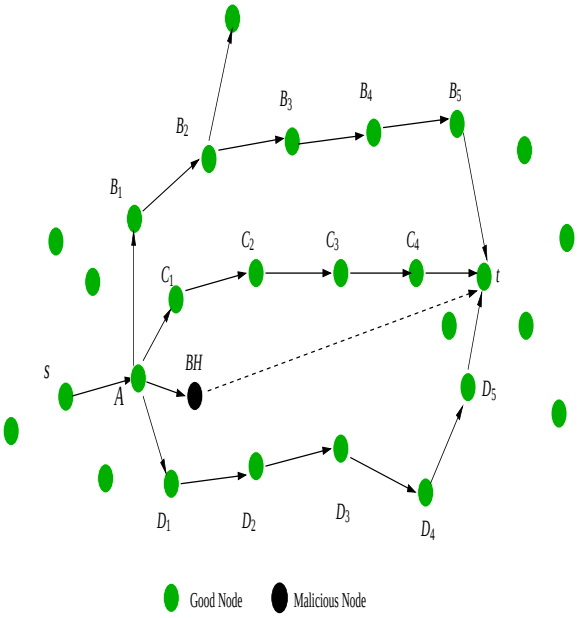

Figure. 1. Blackhole node of type 1

(2) Selfish node of Type 2 forwards control packets but does not forward data packets. Here we make an assumption that once a node stops forwarding data packets, it does not involve itself into route establishment also. Let E be the initial maximum energy of a node. When the energy of the node falls within $(T1, E]$ the node behaves properly and executes both routing functions and packet forwarding. When energy falls in $(T2, T1]$ the node forwards control packets but disables data packet forwarding. Since now the node no longer wants to participate in data packet forwarding and its intention is not to disrupt the normal functioning of the network it is legitimate to assume that it will no longer participate in route establishment until its energy is restored. With in a limited time interval the node is recharged and its energy level is set back to the initial value.

3. IMPACT OF BLACKHOLE AND SELFISH NODES

Once a route is established through a blackhole node, it drops the data packets as it does not have a valid path to the destination. As a result the network throughput degrades considerably. Parsons et al. [Parsons and Ebinger 2009] showed the impact of various attacks on $AODV$. In particular they showed that Packet Loss Ratio (PLR) increases from .13 (in the absence of an attack) to more than .5 when a blackhole attacker is present. They also showed that the routing overhead increases significantly as the number of attackers increase. We also observe in our work that the packet delivery ratio (PDR) of $AODV$ falls from .97 (in the absence of attack) to about .24 when an attacker (blackhole node) is present. A blackhole attacker can also drop received routing messages instead of relaying them, as the protocol requires, thereby making the destination unreachable. The attacker can also store the data and perform traffic analysis.

Several authors [Michiardi and Molva 2002b; Kargl et al. 2004] studied the impact of selfish nodes on dynamic source routing (DSR) algorithm. Michiardi et al. [Michiardi and Molva 2002b] showed that the PDR of the algorithm drops by 60% when 50% of the nodes of the network are selfish. Further they pointed out that the PDR degrades by $10\% - 15\%$ every time the percentage of selfish nodes increases by 10%. They also showed that end-to-end delay increases linearly with the percentage of selfish nodes in the network. Kargl et al. [Kargl et al. 2004] also showed that the performance (delivery ratio) of DSR degrades significantly as the number of selfish nodes

increase in the network. We studied the impact of presence of selfish nodes on $AODV$(Figure 2). We show that the packet delivery ratio of $AODV$ drops by about 55% when 50% of nodes are selfish and it degrades by 10%-20% with increase of 10% in the number of selfish nodes.

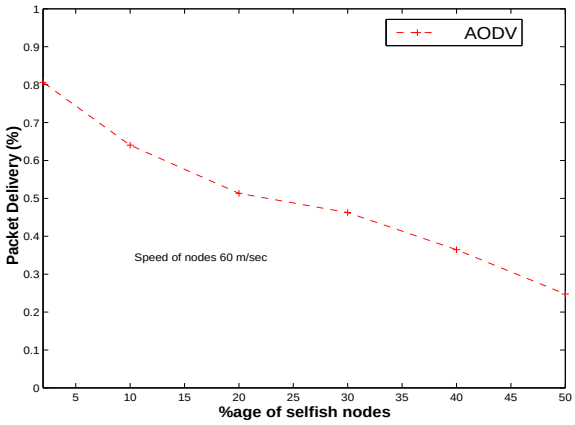

Figure. 2. Packet Delivery Ratio of $AODV$ with varying percentage of selfish nodes

4. RELATED WORK

Approaches to assuage impact of blackhole attacks either use cross-verification [Deng et al. 2002; Yin and Madria 2006; Ramaswamy et al. 2005; Banerjee 2008; Agrawal et al. 2008] or are based on watchdog mechanism [Marti and Mishra 2000; Patcha and Mishra 2003]. In [Deng et al. 2002] the source node verifies the authenticity of the intermediate node (IN) sending the *RouteReply* from its nexthop node (NHN). It does so by broadcasting a *FurtherRequest* packet to the NHN to verify if it has a route to the destination. In [Yin and Madria 2006] instead of the source node, the previous hop node broadcasts a verification packet to the NHN. Most of the approaches using cross-verification flood the verification packets and hence incur a lot of communication overhead. Watchdog mechanisms require the nodes to listen to their neighbor nodes in promiscuous mode. Switching the mode from promiscuous mode to transmit/receive mode is not easy and is error prone [Kargl et al. 2004]. In some approaches [Tamilselvan and Sankaranarayanan 2007; Shurman et al. 2004] the source node waits for some time, collects some paths and selects the one that shares at least one node with at least one more path. It is based on the hypothesis that if two paths share a node, it is unlikely that it is under attack. The approach suffers with the delay in establishing the route besides the fact that the probability of a blackhole node on the path is non-zero. Some researchers have also proposed Intrusion Detection System (IDS) and learning theory approaches to mitigate blackhole attack [Kurosawa et al. 2007; Huang and Lee 2004; Huang et al. 2003; Ruiz et al. 2008]. These approaches are compute-intensive and incur large storage and communication overhead as they collect and analyze large amount of data for anomaly detection. Sun et al. [Sun et al. 2003] proposed a mechanism to mitigate impersonation where an attacker impersonate as the destination to launch a blackhole attack. Some approaches have been proposed to handle blackhole attack launched by specifying false sequence numbers [Bhargava 2002; Kurosawa et al. 2007]. They handle blackhole attack of type 2.

Most of the work to diminish the effect of selfish nodes either propose a reputation based trust system [Wang et al. 2005; Yan et al. 2003] or are based on providing economic incentives. Reputation based systems either rely on first hand information to build reputation or use second-hand information gathered by other nodes. Though using second-hand information results in building the reputation quickly, it suffers with the drawback of spreading rumors. To handle this CORE

(a COllaborative REputation Mechanism) [Michiardi and Molva 2002a] allows sharing of only positive behavior which makes it vulnerable to positive ratings by malicious nodes. By sharing only negative reputation CONFIDANT (Co-operation of Nodes Fairness In Dynamic Ad-hoc Networks) [Buchegger and Boudec 2002a] reduces the false praise but makes the system vulnerable to false accusations. Context-aware detection [Paul and Westhoff 2002] accepts negative advertisement provided it is claimed by some threshold number of nodes else it is considered as misbehavior. It checks false accusations but at the same time also discourages legitimate reporting of misbehavior especially in sparse networks. DRBTS(Distributed Reputation-based Beacon Trust System) [Srinivasan et al. 2006], CONFIDANT [Buchegger and Boudec 2002b], SORI (Secure and Objective Reputation-based Incentive scheme) [He et al. 2004], RPA (Reputation Propagation and Agreement) [Liu and Yang 2002] and RFSN (Reputation-based Framework for High Integrity Sensor Networks) [Ganeriwal and Srivastava 2004] use both positive and negative information but use different weight functions to different type of information. OCEAN (Observation-based Cooperation Enforcement in Ad Hoc Networks) [Bansal and Baker 2003] and Pathrater [Marti and Mishra 2000] use only first hand information to check the rumors but it takes long for the reputation to fall. Whatever be the strategy, reputation based mechanisms either suffer with the danger of spreading rumors or positive ratings by co-operating malicious nodes or gaining high-reputation and trust by a malicious node and staying in the system. Moreover, most of these approaches require neighborhood monitoring in promiscuous mode. Refaei et al. [Refaei et al. 2005] proposed a reputation based trust mechanism which does not depend upon the reputation information exchange but rather takes the feedback from the destination (e.g. by TCP acknowledgement) to raise the reputation index of its next hop neighbor on successful delivery of the packets. The approach suffers with the drawback that the presence of a selfish node down the path may lead to penalizing a good neighbor.

Incentive based schemes treat packet forwarding as a service that can be priced and introduce some form of virtual currency to encourage packet forwarding [Buttyan and Hubaux 2000; 2001; 2003; Jakobsson et al. 2003; Zhong et al. 2003]. [Buttyan and Hubaux 2001] introduces 'Incentives to co-operate' scheme which uses a virtual currency called $Nuglets$ in every communication. $Nuglets$ serve as a per-hop payment for every packet forwarding. They are incremented when a node forwards for others and decremented when it sends packets for themselves. Thus a node exhibiting selfish behavior is penalized appropriately. Authors propose two conceptual models for charging the packet forwarding service. In the first one, called Packet Purse Model (PPM) the source of the packet is charged, whereas in the second one, called Packet Trade Model (PTM), the destination is charged. A hybrid solution is the one in which both source and destination are charged according to the requirement. In PPM, the source node loads the packet with a number of nuglets sufficient to reach the destination. Each forwarding node acquires some nuglets from the packet that covers its forwarding costs. The exact number of nuglets charged by the forwarding nodes may depend upon many things including the amount of energy used for the forwarding operation, the current battery status of the forwarding node, and its current number of nuglets. If a packet does not have enough nuglets to be forwarded then it is discarded. In Packet Trade Model, the packet does not carry nuglets, but it is traded for nuglets by intermediate nodes. Each intermediary buys it from the previous one for some nuglets and sells it to the next one for more nuglets. These schemes (incentive based) require tamper-proof hardware so that the correct amount of credit is added or deducted from a node [Buttyan and Hubaux 2000] or require virtual banks [Buttyan and Hubaux 2003; Jakobsson et al. 2003]. There are arguments that tamper-resistant devices in general might be next to impossible to be realized [Anderson and Kuhn 1996; 1997]. Approaches requiring virtual banks need a fixed communication infrastructure to implement the incentive schemes which is not applicable for a pure ad hoc network. Zhong et al. [Zhong et al. 2003] propose a Simple, Cheat-Proof, Credit based (*Sprite*) mechanism which does not require tamper-proof hardware but requires an infrastructure (Credit Clearance System) to implement credits.

Most of approaches to alleviate blacknhole/selfish nodes do not mitigate collaborative/multiple attacks. The existing solutions to handle multiple/collaborative attacks are either based on Intrusion Detection System (IDS) [Bhargava et al. 2009; Kurosawa et al. 2007; Huang and Lee 2004; Huang et al. 2003; Ruiz et al. 2008] or are recursive application [Ramaswamy et al. 2005] of the approach proposed for a single attack. These approaches handle only a single type of attack. To the best of our knowledge, no algorithm handles more than one type of attack in a single scheme. Handling more than one type of attack in a single scheme is a major challenge for researchers. Bhargava [Bhargava et al. 2009] in their work have suggested a scheme to classify the attacks on the basis of observed behavior and then take corrective measures accordingly.

5. $RDVB$: RELIABLE DISTANCE VECTOR ROUTING PROTOCOL TO HANDLE BLACKHOLE ATTACK

In this section we present an algorithm that assuages only blackhole node. Reliable distance vector routing protocol to handle blackhole ($RDVB$) is based on $AODV$ routing protocol. After a path has been discovered in $AODV$, instead of immediately sending out data packets, we check the reliability of the path by sending a verification packet on the discovered path, the reply to which can be generated only by the destination. If there is a blackhole on the discovered path the verification packet will not reach the destination as the blackhole node does not have a path to the destination. When the source node does not receive a reply within a fixed amount of time, it discards the route. Path discovery in $RDVB$ can be thought of as consisting of two phases. Phase I is a slight modification of path-discovery in $AODV$. In phase-II, we use two control packets called Reliable Route Discovery Unit ($RRDU$) and $RRDU$ reply ($RRDU\_REP$) to check the reliability of path.

5.1 Phase-I of Algorithm

When a node wishes to communicate with another node it looks for a route from its table. If a valid entry is found for the destination it uses that path to send data packets else it broadcasts a *RouteRequest* packet ($RREQ$) to its neighbors with hopcount set to 1. Neighbors check their routing tables for a fresh entry to the destination. If it is found, it replies with a *RouteReply* ($RREP$) packet else forwards $RREQ$ to its neighbors with hopcount incremented by 1. The process continues until either the destination or an intermediate node with a fresh route to the destination is located. At each intermediate node a reverse path is created for the source. When the $RREQ$ packet reaches the destination it also replies with an $RREP$ packet. Processing of $RREQ$ at an intermediate node and destination are explained in Figures 16 and 17 respectively.

Since intermediate nodes as well as destination send $RREPs$ in response to $RREQ$ packets, source node as well as intermediate nodes may receive multiple $RREPs$ in the process. In $AODV$ source or intermediate node receiving multiple $RREPs$ selects the one that arrives first and others are discarded. Hence, one unique path is established between the source and the destination. However, in $RDVB$, a node receiving multiple $RREPs$ maintains a list of next hops in its routing table. When an intermediate node receives an $RREP$ it appends the nexthop (NH) node to the next hop list (NHL). NHL is used to discover a new path free from a malicious node in phase-II. In $AODV$, when the source node receives $RREP$ packet, route is established whereas $RDVB$ enters phase-II. Processing of $RREP$ at an intermediate node and at the source are explained in Figures 18 and 19 respectively. Algorithm 1 summarizes phase-I of algorithm.

The format of Routing Table entry in $RDVB$ is almost the same as that of $AODV$ except for the NH entry . Next Hop entry in the table is now a list of next hops. Formats of $AODV$ routing table entry, $RDVB$ routing table entry, $RREQ$ and $RREP$ are shown below:

$AODV$ **Routing Table Entry Format**							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop	Precursors	Life Time

$RDVB$ **Routing Table Entry Format**							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop List	Precursors	Life Time

$RREQ$ **Message Format**						
Type	RREQ id	Dest id	Dest Seq Number	Src id	Src Seq Number	Hopcount

$RREP$ **Message Format**					
Type	Dest id	Dest Seq Number	Src id	Hopcount	Life Time

1.1 Source broadcasts an $RREQ$ packet.
1.2 When an intermediate node (IN) receives the $RREQ$ packet, it checks its routing table:
  if *(the node has a fresh route to destination)* **then**
    it replies with an $RREP$ packet.
  else
    it broadcasts $RREQ$ further to its neighbors.
  end
1.3 When the destination receives an $RREQ$, it also replies with an $RREP$ packet.
1.4 When an intermediate node receives the $RREP$, it appends the next hop to its NHL and forwards the $RREP$ packet on the reverse route.
1.5 When the source node receives the $RREP$ packet it enters phase -II.

Algorithm 1: Phase-I of $RDVB$

5.2 Phase-II of Algorithm

$AODV$ has been extended to $RDVB$ by adding two types of control packets: Reliable Route Discovery Unit ($RRDU$) and $RRDU$ reply ($RRDU\_REP$). $RRDU$ messages are control packets sent by the source node and $RRDU\_REP$ message is the response of $RRDU$ by the destination to the source node. $RRDU\_REP$ can only be generated by the destination. We assume that there is no impersonation i.e. no node other than the destination can generate $RRDU\_REP$ on behalf of the destination. In phase-II when the source node receives an $RREP$, it sends an $RRDU$ packet with hopcount set to 1 on the path to check its reliability. If the source node receives multiple $RREPs$, it sends out an $RRDU$ packet to each of the node from which it receives the $RREP$ packet. The path from which it receives the reply to $RRDU$ is finally established as a reliable path.

When an intermediate node receives an $RRDU$ packet, it forwards $RRDU$ to all the nodes in its NHL with hopcount incremented by one. It also keeps a copy of $RRDU$ for the future $RREPs$. It keeps on sending $RRDUs$ to the nodes from which it receives $RREPs$ until it receives an $RRDU\_REP$ packet. For example, in Figure 1, suppose A receives the first $RREP$ from BH and forwards to s. After this, it receives $RREP$ from B_1 and adds it to NHL. When it receives $RRDU$ from s, it sends $RRDU$ to nodes in NHL i.e to BH and B_1. It also keeps a copy of $RRDU$ packet. Later when it receives $RREP$ from C_1, it adds this to NHL and forwards the copy of $RRDU$ to it. Destination may also receive multiple $RRDUs$; it responds with an $RRDU\_REP$ packet (with hopcount set to 1) to the one that arrives first and discards future $RRDU$ packets as duplicates. Thus each node including the source node receives a unique $RRDU\_REP$ and a secure path is established. Each intermediate node keeps only the node from which it receives the $RRDU\_REP$ in the NHL and discards all other entries from the list. It copies the hopcount from the $RRDU\_REP$ packet in the routing table entry for the destination, increments the hopcount in the packet by 1 and forwards the packet on the reverse route. Algorithm 2 summarizes phase-II of $RDVB$.

2.1 When the source node receives an $RREP$ packet it sends out an $RRDU$ packet with hopcount set to 1 to check the reliability of the path.
2.2 When an intermediate node receives the $RRDU$ packet it increments the hopcount in the packet by one, forwards it to all the nodes in its NHL and keep a copy for future $RREPs$.
`/* Notice here that if we did not keep this list and forwarded the` $RRDU$ `packet only to the node from which it received the first` $RREP$ `and that path had a blackhole node, we had no way to discover a path free from the malicious node. We would have known that the path discovered was under attack but would not have been able to discover an alternate reliable path. */`
2.3 When the destination receives the $RRDU$ packet it replies with an $RRDU\_REP$ packet, hopcount set 1, to the first $RRDU$ it receives. It discards the $RRDU$ packets it receives in future as duplicates.
2.4 $RRDU\_REP$ travels a reliable path back to the source node and the path is established.

Algorithm 2: Phase-II of $RDVB$

Processing of $RRDU$ and $RRDU\_REP$ are explained in Figures 20, 21 and 22. Formats of $RRDU$ and $RRDU\_REP$ messages are shown below:

$RRDU$ **Message Format**							
Type	RRDU id	Dest id	Dest Seq Number	Src id	Src Seq Number	Hopcount	Life Time

$RRDU\_REP$ **Message Format**					
Type	Dest id	Dest Seq Number	Src id	Hopcount	Life Time

As in $AODV$, $RDVB$ uses $RERR$ and $HELLO$ messages for route maintenance.

5.3 Security Analysis: handling blackhole attack

In this section we will show that our scheme discovers a path free from blackhole node. See Figure 1. If BH is a malicious node then it may send $RREP$ without having a route to the destination declaring that it has a fresh route to the destination. In case of $AODV$, if A receives the first $RREP$ from BH, it keeps the path through BH and discards others if the hopcount of others is more. Hence a path through BH is set up between the source and the destination. In $RDVB$, we send $RRDU$ on this path to check the reliability of the path. Suppose A receives $RREPs$ first from BH, then subsequently from B_1 (an intermediate node with path), and then from C_1 ($RREP$ received through path $C_1 - C_2 - C_3 - C_4 - t$). It forwards the first $RREP$ to the source. Later when A receives $RREPs$ from B_1 and C_1, it stores their *id*s in NHL and discards the $RREP$ packets. When the source receives the $RREP$ packet, it sends $RRDU$ to A. A forwards $RRDU$ to BH, B_1 and C_1. However, since no node other than the destination can generate a reply to $RRDU$, A does not receive $RRDU\_REP$ from BH. Suppose t receives $RRDU$ first from C_4 as it is on a shorter route and then from B_5. It sends $RRDU\_REP$ to C_4 and discards the $RRDU$ from B_5. Thus A receives $RRDU\_REP$ from C_1 via $C_1 - C_2 - C_3 - C_4 - t$, it sets C_1 as next hop on the path to t and forwards $RRDU\_REP$ to s. When s receives $RRDU\_REP$ a secure reliable path, free from blackhole, is established.

6. $RDVBS$: RELIABLE DISTANCE VECTOR ROUTING PROTOCOL TO HANDLE ATTACKS DUE TO BLACKHOLE AND SELFISH NODES

Consider a selfish node of type 1 i.e. a selfish node that forwards neither the control packets nor the data packets. Such a node will be isolated in $RDVB$ as it will not forward $RRDU$ packet and hence $RRDU\_REP$ will not be received through it. However, if there is a selfish node of type 2 i.e. a node forwards all control packets including $RRDU$ and $RRDU\_REP$ but does not cooperate in forwarding data packets, $RDVB$ will not be able to avoid it and hence the above algorithm in its current form will not be able to isolate such a node. Also, a node on the discovered path may co-operate in forwarding data packets for some time and may become

selfish after some time due to its reduced energy levels. In order to identify such a behavior we modify $RDVB$ and call it as Reliable Distance Vector routing algorithm for Handling Blackhole and Selfish nodes ($RDVBS$). Path discovery in $RDVBS$ is same as that in $RDVB$. However once a path free from black node has been discovered, $RRDUs$ are sent periodically to maintain the reliability of the path, i.e. to detect if any misbehaving selfish node has crept into the path. We call this as phase-III of the algorithm.

6.1 Phase-III of Algorithm

To maintain the reliability of the path we introduce a field called Forward Data Packet Count ($FDPC$) in the routing table (RT) entry as well as in the $RRDU\_REP$ packet and a field called Reliability Flag (RF) in the $RRDU\_REP$ packet. Initially RF is set to 1 by the destination and it is cleared when a selfish node is detected on the path. $FDPC$ in the routing table entry keeps a count of the number of data packets forwarded by the node. For every data packet received and forwarded by a node, $FDPC$ in RT entry of the node is incremented. This $FDPC$ is copied by the node, on return, in the $RRDU\_REP$ packet to tell its previous neighbor as to how many data packets it has forwarded. The neighbor uses this count to detect whether the node has forwarded all the packets or not. If a node discovers that its next hop neighbor has not forwarded all the packets, it informs the sender by clearing the reliability flag in the $RRDU\_REP$ packet. Since the selfish node of type 2 participate in forwarding all control packets exept the ones used for discovery of path ($RREQs$ and $RREPs$) it forwards the $RRDU\_REP$ packet and since it does not intend to disrupt the normal functioning of the system, it does not lie. In case a selfish node is detected on the discovered path, a fresh route discovery is initiated by the source. Since we assume that once a node starts dropping the data packets, it does not participate in route establishment until its energy is restored, the selfish node is isolated when fresh route discovery is initiated.

A node keeps an entry for each destination in its routing table. In $RDVB$, the routing table entry for destination t does not depend upon which source is trying to communicate with t. When two source nodes say s_1 and s_2 try to communicate with t, the only thing an intermediate node remembers is the next hop required to reach t. But now the node needs to remember how many data packets it has forwarded for each communication. Thus, a list called Reliability List (RL) is added in the routing table entry of each node. An entry in the RL has source address, Forwarded Data Packet Count ($FDPC$) and $RRDU\ id$, i.e. the triplet ($Sourceaddress, FDPC, RRDU\ id$). The triplet entry keeps a count of the number of data packets forwarded by the node from source s to the destination t since the last $RRDU$. $RRDU\ id$ is incremented every time a new $RRDU$ packet is sent by the source. The triplet is initialized when the first $RRDU$ is processed in phase-II and, it is used and updated when periodic $RRDU$s are processed in phase-III. Assuming that not too many nodes will be communicating with a given node at a given time, the size of RL is not expected to be big. Algorithm 3 summarizes phase-III of the algorithm.

Processing of $RRDU$ and $RRDU\_REP$ in phase-II and phase-III of $RDVBS$ is shown in Figures 23, 24, 25 and 26. The format of $RDVBS$ routing table entry is same as that of $RDVB$ except for the additional RL field and the format of $RRDU\_REP$ is modified to include the reliability flag field. The format of $RDVBS$ routing table entry and modified $RRDU\_REP$ are shown below:

$RDVBS$ **Routing Table Entry Format**								
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop List	Precursors	RL	Life Time

$RRDU\_REP$ **Message Format modified for** $RDVBS$							
Type	Dest id	Dest Seq Number	Src id	Hopcount	FDPC	Reliability Flag	Life Time

Initialization :$RF \leftarrow 1$, $FDPC \leftarrow 0$
3.1 $RRDUs$ are sent periodically to maintain the reliability of the path.
3.2 When an intermediate node receives a periodic $RRDU\_REP$ from its next hop neighbor it checks its routing table:
if *(next hop neighbor has not forwarded all the packets)* **then**
it clears the reliability flag in $RRDU\_REP$ packet and forwards the packet on reverse route.
else
it copies $FDPC$ from the RL entry of routing table in the $RRDU\_REP$ packet and forwards the packet on reverse route.
end
3.3 When source receives $RRDU\_REP$ it checks RF in the $RRDU\_REP$ packet:
if *(RF is set to* 1*)* **then**
path is considered to be reliable; it sends more data packets, if any.
else
it initiates route discovery process again.
end

Algorithm 3: Phase-III of $RDVBS$

Note: We can improve the performance of the algorithm slightly by checking if the $RREP$ packet is from the destination itself. If so, it need not send the $RRDU$ packet and it can start sending the data packets on this path without waiting for $RRDU\_REP$ from the destination.

6.2 Security Analysis: handling selfish nodes

Suppose that the path discovered in Figure 1, is $s - A - C_1 - C_2 - C_3 - C_4 - t$ and let that a node (say C_2) on this path becomes selfish. Let C_2 be a selfish node of type 2 i.e. it forwards control packets but does not forward the data packets. Suppose s sends n data packets to t before sending next $RRDU$. Then A and C_1 forward all the n data packets to the successor. Let C_2 forwards only p out of the n packets. Then C_3 and C_4 also forward p packets and, the destination receives only p packets. After some time, s sends another $RRDU$ and t sends back the count of the received packets in $RRDU\_REP$. $FDPC$ field in the $RRDU\_REP$ is set to p by the destination. At every node x on the reverse path from the destination to the source, $FDPC$ in $RRDU\_REP$ is set to the number of data packets forwarded by x on the forward path. Hence, C_4, C_3, and C_2 set $FDPC$ in $RRDU\_REP$ to p whereas C_1 and A set it to n. When C_1 sees that it had forwarded n packets to C_2 but C_2 forwarded only p ($< n$) out of them it comes to know that C_2 is selfish and it clears the Reliability Flag in the $RRDU\_REP$ packet to 0. When the source receives this $RRDU\_REP$ packet with RF set to zero it knows that something is wrong on this path and it initiates a fresh route discovery. Here we make an assumption that once a node stops forwarding data packets, it does not involve itself into route establishment also. Thus, C_2 discards any $RREQ$ packet received from C_1 and a path ignoring C_2 is established. The previous set of data packets are sent again.

7. SIMULATION STUDY

We simulated our protocol using Network Simulator [NS2]. To study the performance of $RDVBS$, packet delivery ratio, average end-to end delay and routing overhead were studied.

7.1 Simulation Design

Simulation results were obtained for 50 nodes located over $1000m$ by $1000m$ region. The traffic sources are CBR (constant bit rate), 512-byte as data packet, sending rate is 1 pkt/sec and with maximum load of 300 packets for one transaction. The node movement speed is varied from 0 to 80 which will be closer to real applications. The mobility are done with pause time 100 second. Script was executed for 300 seconds.

7.2 Simulation Results

We compared the performance of our protocol with that of $AODV$ and $DENG$ in presence of blackhole attack.

Comparison with DENG: Both $DENG$ and $RDVBS$ are able to detect and isolate blackhole node. Average End to End Delay ($AEED$) and Packet Delivery Ratio (PDR) of $RDVBS$ and $DENG$ are comparable as shown in Figure 3 and 4. Our protocol out performs $DENG$ in terms of Routing Overhead (RO) (see Figure 5). $DENG$'s route discovery phase comprises of broadcasting route request twice, once for destination and another for intermediate node (for feedback). This leads to the generation of a much more number of $RREQ$ packets than the number of control packets generated in $RDVBS$.

Comparison with AODV: As expected, Packet Delivery Ratio (PDR) of $AODV$ drops significantly as compared (Figures 4) to $RDVBS$ and $DENG$ in presence of a blackhole node. In the absence of mobility, PDR of $AODV$ is zero whereas that of $RDVBS$ is 1. As the nodes start moving sometimes the blacknode falls on the path and sometimes not. Since $RDVBS$ isolates the blackhole node and $AODV$ does not, PDR of $RDVBS$ remains better than that of $AODV$ whereas the RO of $RDVBS$ is only slightly more than that of $AODV$ which is natural to expect.

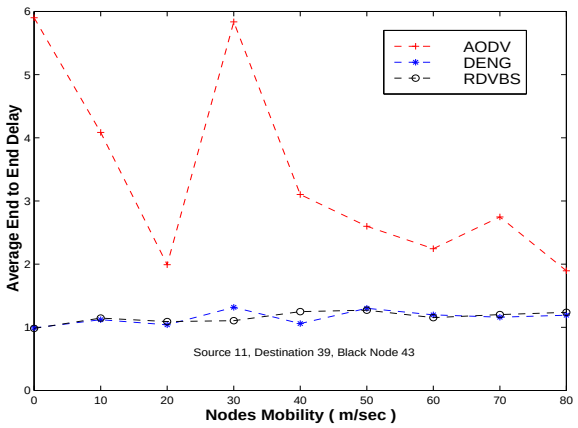

Figure. 3. Comparison of Average End to End Delay in presence of blackhole node

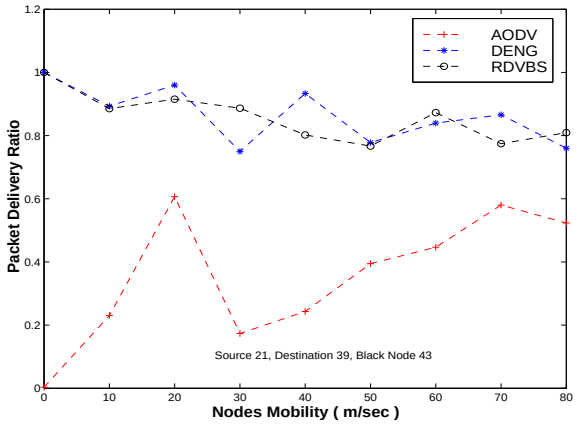

Figure. 4. Comparison of Packet Delivery Ratio in presence of blackhole node

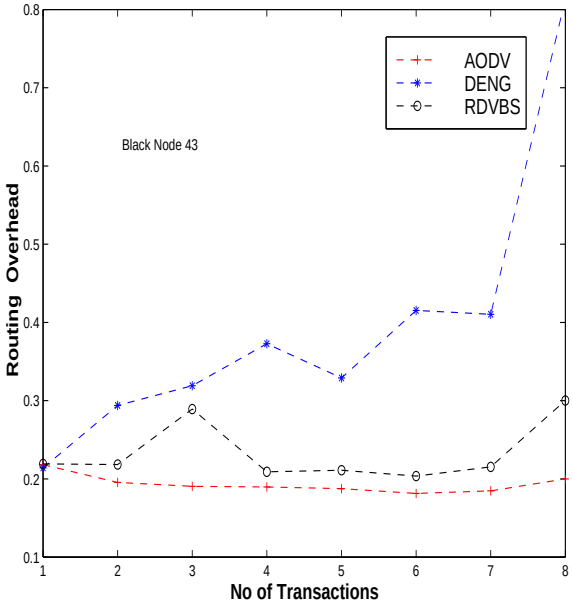

Figure. 5. Comparison of Routing Overhead in presence of blackhole node

We also compared the performance of our protocol with that of *AODV* in the presence of selfish nodes.

Comparison with AODV in presence of selfish nodes: As shown in Figure 2, PDR of *AODV* decrements by 10% - 20% with every 10% increase in the percentage of selfish nodes in network. The figure also shows that when 50% of the nodes of the network are selfish PDR degrades by more than 55%. In $RDVBS$, as shown in Figure 6, PDR degrades just by $1\% - 3\%$ every time the percentage of selfish nodes increases by 10%. On the other hand, $AEED$ and RO (Figures 7 and 8) do not increase much with increase in the number of selfish nodes. Figure 9 shows the impact of mobility on PDR in the presence of selfish nodes. For $RDVBS$, it is observed that this effect diminishes significantly.

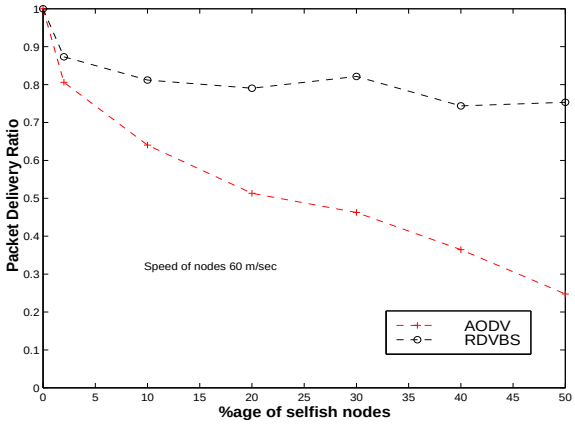

Figure. 6. Comparison of Packet Delivery Ratio in presence of selfish nodes

Comparison with AODV and DENG in the absence of attack: We also compared our protocol with *AODV* and *DENG* in the absence of black node and selfish nodes. When there are no

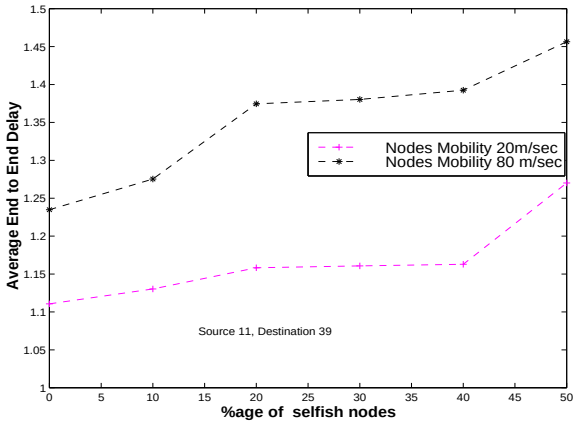

Figure. 7. Average End to End Delay of $RDVBS$ in presence of selfish nodes

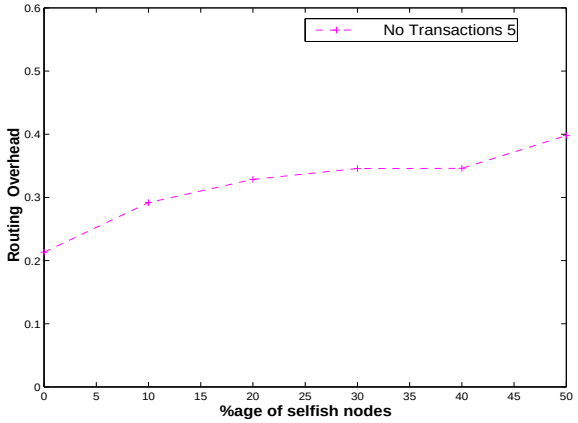

Figure. 8. Routing Overhead of $RDVBS$ in presence of selfish nodes

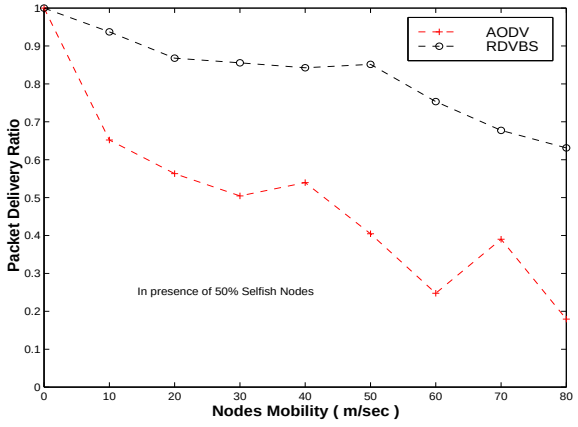

Figure. 9. Comparison of Impact of mobility on Packet Delivery Ratio in presence of selfish nodes

black nodes, verification step of the route discovery phase of $RDVBS$ and $DENG$ lead to some routing overhead (Figure 10). Here also $RDVBS$ outperforms $DENG$ whereas RO of $RDVBS$ is only slightly more than that of $AODV$. Packet delivery ratio and average end-to-end delay of

all the three protocols are comparable, see Figure 11 and 12.

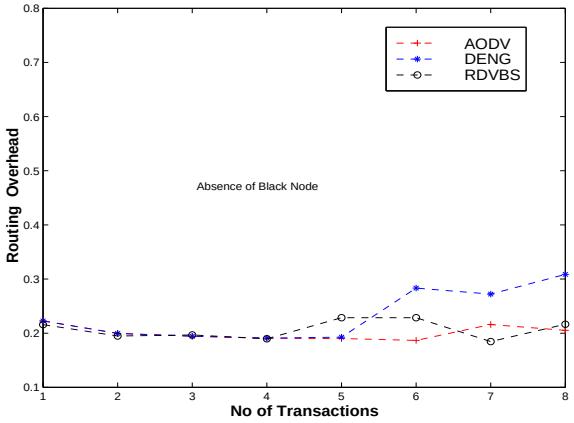

Figure. 10. Comparison of Routing Overhead in absence of blackhole node

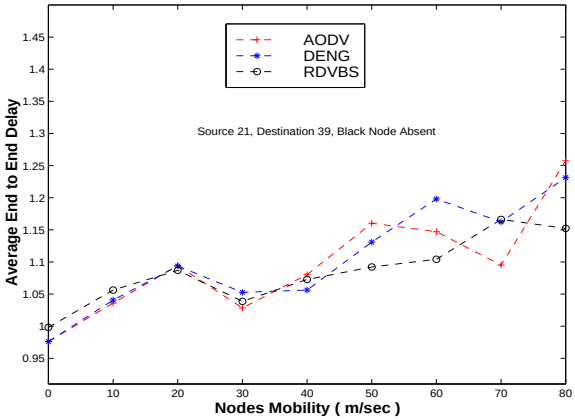

Figure. 11. Comparison of Average End to End Delay in absence of blackhole node

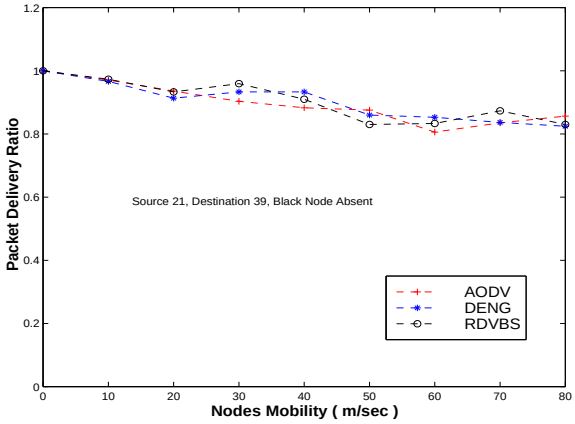

Figure. 12. Comparison of Packet Delivery Ratio in absence of blackhole node

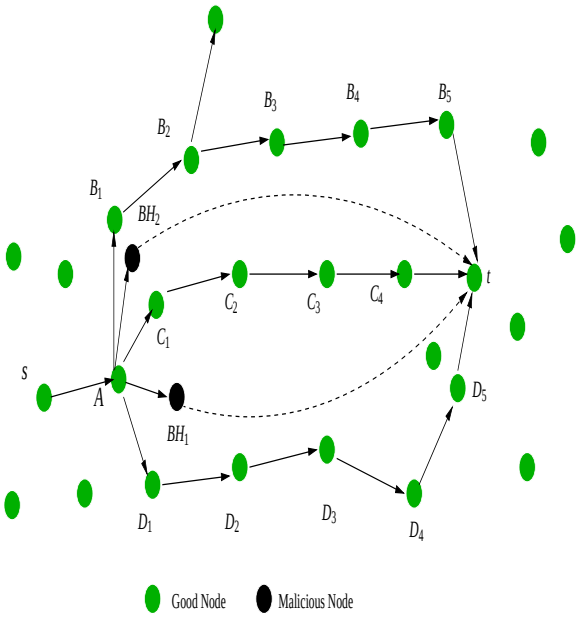

Figure. 13. Example of multiple blackhole nodes

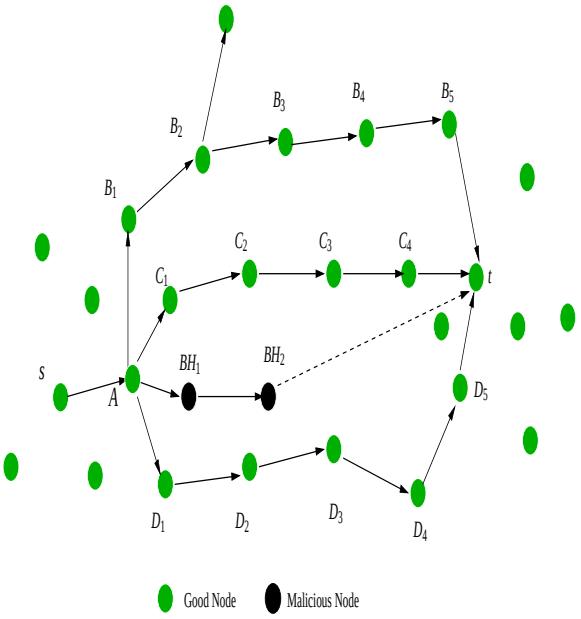

Figure. 14. Example of co-operative blackhole nodes

8. HANDLING MULTIPLE AND CO-OPERATIVE BLACKHOLE NODES

Our protocol also handles multiple black hole attacks. Consider the scenario in Figure 13. If several $RREPs$ are received from all the black nodes present in the network, our protocol sends an $RRDU$ packet to all of them. However, no $RRDU\_REP$ will be received from any black node and hence all the black nodes will be isolated. Also consider the scenario of Figure 14 where

the black nodes co-operate with each other to launch the attack. Such attacks are also detected and isolated by our protocol in a similar way as no $RRDU\_REP$ will be received from such a path. $DENG$ will be able to detect multiple attacks of Figure 13 by recursive application thereby incurring a lot of overhead. Also, it will not be able to detect the co-operative attack of Figure 14. In [Tamilselvan and Sankaranarayanan 2007], as the number of co-operating black nodes sending $RREPs$ with the same NHN increases, the chances of establishing a path through a black node increases.

Comparison with $AODV$ in presence of multiple black nodes: Figure 15 shows that the decrease in PDR with the increase in the number of black nodes is much less in case of $RDVBS$ as compared to that for $AODV$.

9. CONCLUSION

we have presented a first such protocol that handles both black nodes as well as selfish nodes. The protocol is simple to implement, without any special hardware requirement. The protocol outperforms the earlier protocols to handle black nodes in terms of secure path discovery and routing overhead without incrasing delays. The protocol handles multiple and cooperative black nodes also.

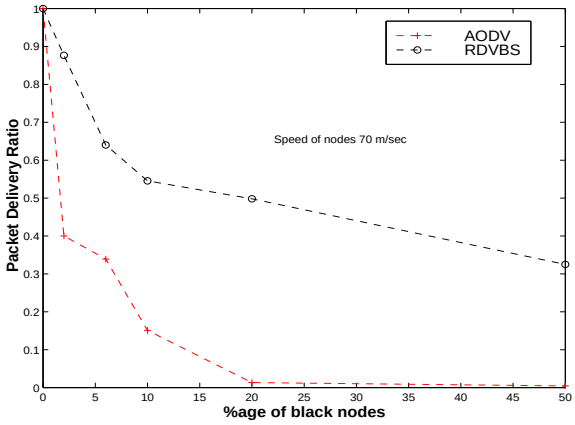

Figure. 15. Impact of multiple blackhole nodes on Packet Delivery Ratio of $RDVBS$ and $AODV$

REFERENCES

The network simulator - ns2. `http://nsnam.isi.edu/nsnam/index.php/User/_Information`.

Agrawal, P., Ghosh, R. K., and Das, S. K. 2008. Cooperative black and gray hole attacks in mobile ad hoc networks. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. Korea, 310–314.

Anderson, R. and Kuhn, M. 1996. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*. 1–11.

Anderson, R. and Kuhn, M. 1997. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols*. Paris, France, 125–136.

Banerjee, S. 2008. Detection/removal of cooperative black and gray hole attack in mobile ad-hoc networks. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS)*. San Francisco, USA.

Bansal, S. and Baker, M. 2003. Observation-based cooperation enforcement in ad hoc networks. In *Research Report cs.NI/0307012 Stanford University*.

Bhargava, B. 2002. Intruder identification in ad hoc networks. In *CERIAS Security Center and Department of Computer Sciencies*. Purdue University, research proposal.

Bhargava, B., de Oliveira, R., Zhang, Y., and Idika, N. C. 2009. Addressing collaborative attacks and defense in ad hoc wireless networks. In *Second Workshop on Specialized Ad Hoc Network Systems (SAHNS), in conjunction with the ICDCS*.

Buchegger, S. and Boudec, J.-Y. L. 2002a. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*. Lausanne, Switzerland, 403 410.

Buchegger, S. and Boudec, J.-Y. L. 2002b. Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*. Lausanne, Switzerland, 226–236.

Buttyan, L. and Hubaux, J. P. 2000. Enforcing service availability in mobile ad-hoc wans. In *First ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*. Boston, MA.

Buttyan, L. and Hubaux, J. P. 2001. Nuglets: a virtual currency to stimulate cooperation in selforganized ad hoc networks. In *Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne*.

Buttyan, L. and Hubaux, J. P. 2003. Stimulating cooperation in self-organizing mobile ad hoc networks. *Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks ACM*.

Deng, H., Li, W., and Agrawal, D. P. 2002. Routing security in wireless ad hoc networks. In *IEEE Communications Magazine*. Vol. 40. 70–75.

Ganeriwal, S. and Srivastava, M. 2004. Reputation-based framework for high integrity sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SAN' 04)*. 66–77.

He, Q., Wu, D., and Khosla, P. 2004. SORI: A secure and objective reputation-based incentive scheme for ad hoc networks. In *Proceeding of IEEE Wireless Communications and Networking Conference*. Atlanta, GA, USA.

Hollick, M., Schmitt, J., Seipl, C., and Steinmetz, R. 2004. On the effect of node misbehavior in ad hoc networks. In *Proceedings of IEEE International Conference on Communications*. Vol. 6. Paris, France, 3759–3763.

Huang, Y. A., Fan, W., Lee, W., and Yu, P. S. 2003. Crossfeature analysis for detecting ad-hoc routing anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS' 03)*. 478–487.

Huang, Y. A. and Lee, W. 2004. Attack analysis and detection for ad hoc routing protocols. In *The 7th International Symposium on Recent Advances in Intrusion Detection (RAID' 04)*. 125–145.

Jakobsson, M., Hubaux, J., and Buttyan, L. 2003. A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Crypto*. Gosier, Guadeloupe.

Kargl, F., Klenk, A., Schlott, S., and Weber, M. 2004. Advanced detection of selfish or malicious nodes in ad hoc networks. In *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks*. Germany.

Khurana, S., Gupta, N., and Aneja, N. 2006. Reliable ad-hoc on-demand distance vector routing protocol. In *Proceeding of the IEEE International Conference on Networking(ICN)*. 98–103.

Kurosawa, S., Nakayama, H., Kato, N., Jamalipour, A., and Nemoto, Y. 2007. Crossfeature analysis for detecting ad-hoc routing anomalies. *International Journal of Network Security 5*, 338–346.

Liu, Y. and Yang, Y. R. 2002. Reputation propagation and agreement in mobile ad-hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*.

Marti, S. and Mishra, A. 2000. Mitigating routing misbehavior in mobile ad hoc networks. In *6th Int'l.Conference Mobile Comp. Network*. 255–265.

Michiardi, P. and Molva, R. 2002a. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the Sixth IFIP conference on security communications and multimedia*. Portoroz, Slovenia.

Michiardi, P. and Molva, R. 2002b. Simulation-based analysis of security exposures in mobile ad hoc networks. In *Proceedings of European Wireless Conference*.

Parsons, M. and Ebinger, P. 2009. Performance evaluation of the impact of attacks on mobile ad hoc networks. In *Proceedings of Field Failure Data Analysis Workshop*. USA.

Patcha, A. and Mishra, A. 2003. Collaborative security architecture for black hole attack prevention in mobile ad hoc networks. In *Proceedings Radio and Wireless Conference RAWCON*.

Paul, K. and Westhoff, D. 2002. Context aware inferencing to rate a selfish node in dsr-based ad hoc networks. In *Proceedings of the IEEE Globecom Conference*. Taipeh,, Taiwan.

Ramanathan, R. and Redi, J. 2002. A brief overview of ad hoc networks: Challenges and directions. In *IEEE Communications Magazine*. 20–22.

Ramaswamy, S., Fu, H., and Nygard, K. 2005. Effect of cooperative black hole attack on mobile ad hoc networks. In *Proceedings of ICWN*.

Refaei, M. T., Srivastava, V., DaSilva, L., and Eltoweissy, M. 2005. A reputation-based mechanism for isolating selfish nodes in ad hoc networks. In *Proceedings of Second IEEE Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*. San Diego, CA, 3–11.

Ruiz, J. C., Friginal, J., de Andrs, D., and Gil, P. 2008. Black hole attack injection in ad hoc networks. `http://www.zdnetasia.com/whitepaper/black-hole-attack-injection-in-ad-hoc-networks\_wp-2378353.htm`.

Shurman, M. A., Yoo, S. M., , and Park, S. 2004. Black hole attack in wireless ad hoc networks. In *Proceedings of 42nd ACM Southeast Conference (ACMSE' 04)*. 96–97.

Srinivasan, A., Teitelbaum, J., , and Wu, J. 2006. DRBTS: Distributed reputation-based beacon trust system. In *Proceedings of the 2nd IEEE International Symposium on Dependable*. Indianapolis, USA.

Sun, B., Guan, Y., Chen, J., and W.Pooch, U. 2003. Detecting black-hole attack in mobile ad hoc networks. In *Proceedings of the 5th European Personal Mobile Communications Conference*. 490–495.

Tamilselvan, L. and Sankaranarayanan, V. 2007. Prevention of blackhole attack in manet. In *Proceedings of the 2nd IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*.

Wang, B., Soltani, S., Shapiro, J. K., and Tan, P.-N. 2005. Local detection of selfish routing behavior in ad hoc networks. In *International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*. Las Vegas.

Wang, W., Lu, Y., and Bhargava, B. K. 2003. On vulnerability and protection of ad hoc on-demand distance vector protocol. In *Proceedings of the 10th International Conference on Telecommunications (ICT' 03)*. Vol. 1. 375–382.

Yan, Z., Zhang, P., and Virtanen, T. 2003. Trust evaluation based security solution in ad hoc networks. In *Technical Report, Nokia Research Center*. Helsinki, Finland.

Yin, J. and Madria, S. 2006. A hierarchical secure routing protocol against black hole attacks in sensor networks. In *Proceedings of IEEE SUTC*.

Zhong, S., Chen, J., and Yang, Y. 2003. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE INFOCOM*.

When an intermediate node receives an $RREQ$ packet from source s, it does the following steps:

(1) if it has a fresh route to the destination, it replies to the source with $RREP$ else it broadcasts (forwards) the $RREQ$ packet to its neighbors with hopcount incremented by 1. If additional copies of the same $RREQ$ are later received, they are discarded as duplicates.
(2) it sets up a reverse path for the reply message.
  (a) if it has an entry in its routing table for s but it is not fresh, it refreshes it.
  (b) if there is no entry for s in its routing table it creates an entry for s by copying the hopcount and the source id from the $RREQ$ packet and, setting the NH field to the address of the neighbor from which the first copy of the broadcast packet is received.

Figure. 16. Processing of $RREQ$ at an intermediate node in $RDVB$

When the destination receives an $RREQ$ packet from source s, it does the following steps:

(1) if it has an entry in its routing table for s but it is not fresh, it refreshes it.
(2) if there is no entry for s in its routing table it creates an entry for s by copying the hopcount and the source id from the $RREQ$ packet and, setting the NH field to the address of the neighbor from which the first copy of the broadcast packet is received. It creates an $RREP$ packet and unicasts $RREP$ to the next hop on the reverse path.
(3) if additional copies of the same $RREQ$ are later received, they are discarded as duplicates.

Figure. 17. Processing of $RREQ$ at the destination in $RDVB$

When an intermediate node receives an $RREP$ message, it does the following steps:

(1) if it has an entry in its routing table for the destination but it is not fresh, it refreshes it.
(2) if it does not have an entry for the destination, it creates an entry for it and sets the NH field to the address of the neighbor from which the packet is received. It forwards it to the next hop on the reverse path.
(3) if it already has an entry for the destination (in case of multiple $RREPs$) in its routing table, it appends the next hop from which it received the $RREP$ in the NHL entry of the routing table and discards the $RREP$ packet. This is required to establish a secure path from blacknode in phase-II of the algorithm.

Figure. 18. Processing of $RREP$ at an intermediate node in $RDVB$

When the source node receives an $RREP$ packet, it does the following:

(1) if it has an entry in its routing table for the destination but it is not fresh, it refreshes it.
(2) if there is no entry for the destination in its routing table it creates an entry for it and sets the NH field to the address of the neighbor from which the packet is received, as the next hop.
(3) if it already has a fresh entry for the destination in its routing table (in case of multiple $RREPs$), it appends the next hop from which it received $RREP$ in NHL entry of the routing table.
(4) the node sends an $RRDU$ packet with hopcount set to 1 to the node from which it received the $RREP$ packet and phase-II of the algorithm starts.

Figure. 19. Processing of $RREP$ at the source in $RDVB$

When an intermediate node receives an $RRDU$ packet, it does the following:

(1) if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it is done on seeing $RREQ$ (this case may arise when an intermediate node n_1 replies to $RREQ$ with an $RREP$ packet and n_2 is a node on the path from n_1 to t).
(2) each node on the path of $RRDU$ must be having a table entry for the destination. It increments the hopcount in the $RRDU$ packet by 1 and forwards it to all the nodes in NHL.
(3) it keeps a copy of $RRDU$ packet for subsequent $RREPs$.

Figure. 20. Processing of $RRDU$ at an intermediate node in $RDVB$

When the destination receives the $RRDU$ packet, it does the following steps:

(1) if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it is done on seeing $RREQ$.
(2) it creates an $RRDU\_REP$ packet with hopcount set to 1 and replies to the $RRDU$ which arrives first. It discards the copies of $RRDU$ it receives in future, as duplicates.

Figure. 21. Processing of $RRDU$ at the destination in $RDVB$

When an intermediate node receives an $RRDU\_REP$ packet, it does the following:

(1) the node must be having a table entry for the source. It finds next hop on the path from the table entry, and forwards $RRDU\_REP$ to it with hopcount incremented by 1.

(2) in a table entry for the destination, it keeps only one entry in the NHL, the one from which it received the $RRDU\_REP$ and deletes others. It copies hop count from the packet in the routing table entry for the destination.

Since no intermediate node can generate $RRDU\_REP$, source node receives a unique $RRDU\_REP$ and a secure path is established. Source starts sending data packets on this path.

Figure. 22. Processing of $RRDU\_REP$ in $RDVB$

Phase-II of $RDVBS$
When an intermediate node receives the fist $RRDU$ packet it does the following:

(1) if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as is done on seeing $RREQ$.

(2) for the first $RRDU$ packet it receives from source s, it appends an entry $(s, 0, 0)$ in RL (i.e. id of the source is copied from the originator field of the $RRDU$ packet and, $FDPC$ and $RRDU$ id are set to zero). Each node on the path of $RRDU$ must be having a table entry for the destination. It forwards $RRDU$ with hopcount incremented by 1 to all the nodes in NHL.

(3) it keeps a copy of $RRDU$ packet for subsequent $RREPs$ as in $RDVB$.

Phase-III of $RDVBS$
If an intermediate node receives a periodic $RRDU$ packet it does the following:

(1) the node must be having a table entry for the destination. It updates the (source, $FDPC$, $RRDU$ id) triplet i.e. the $RRDU$ id is copied from the $RRDU$ packet and the $FDPC$ count is reset to 0. It finds the next hop on the path from the table entry and forwards $RRDU$ to it (at this time NHL contains a unique NH as a unique path had already been established).

Figure. 23. Processing of $RRDU$ at an intermediate node in $RDVBS$

Phase-II of $RDVBS$
When the destination receives the first $RRDU$ packet it does the following:

(1) if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it does on seeing $RREQ$.

(2) for the first $RRDU$ packet it receives from source s, it appends an entry $(s, 0, 0)$ in RL (i.e. id of source is copied from the originator field of the $RRDU$ packet and $FDPC$, $RRDU$ id are set to zero) and it discards the $RRDU$ packet.

(3) it creates an $RRDU\_REP$ packet with hopcount set to 1 and replies to the node from which the first $RRDU$ is received.

Phase-III of $RDVBS$
When the destination receives a periodic $RRDU$ packet it does the following:

(1) it creates an $RRDU\_REP$ packet and copies $FDPC$ from the RL list entry to $RRDU\_REP$. It finds the next hop for s on the path from table entry and sends $RRDU\_REP$ packet to it.

Figure. 24. Processing of $RRDU$ at the destination in $RDVBS$

Phase-II of $RDVBS$

When an intermediate node receives the fist $RRDU\_REP$ packet, it finds the next hop for the source from the table entry (on the reverse path) and forwards $RRDU\_REP$ with hopcount incremented by 1 to it.

Phase-III of $RDVBS$

When an intermediate node receives a periodic $RRDU\_REP$ packet, it compares $FDPC$ stored in the routing table entry with $FDPC$ in $RRDU\_REP$ packet; if they are same, it forwards $RRDU\_REP$ to the next hop on the reverse path else it copies the $FDPC$ value stored in its routing table entry in the $RRDU\_REP$ packet, clears the reliability flag in the $RRDU\_REP$ packet and forwards the $RRDU\_REP$ packet to the next hop on the reverse path.

Figure. 25. Processing of $RRDU\_REP$ at an intermediate node in $RDVBS$

Phase-II of $RDVBS$

When the source node receives the fist $RRDU\_REP$ packet, a secure path is discovered and the path is established; it starts sending the data packets on the path.

Phase-III of $RDVBS$

When the source node receives a periodic $RRDU\_REP$ packet, it checks the RF flag in the packet. If it is set, path is considered to be reliable and it continues sending data packet on that path else it realizes that there is a selfish node on the path and it initiates the route discovery process again.

Figure. 26. Processing of $RRDU\_REP$ at the source in $RDVBS$

Sandhya Khurana is Assistant Professor at institute of Informatics and Communication, University of Delhi South Campus. She completed her M.Tech. (Computer Science) from IIT Delhi and PhD (Computer Science) from University of Delhi. Her areas of interest are Network Security and Routing in Mobile Ad hoc Networks and Delay Tolerant Networks.

Neelima Gupta is Associate Professor at Department of Computer Science, University of Delhi. She completed her M.Tech (CS) and Ph.D. (CS) from IIT Delhi. She is interested in Routing Algorithms in Ad hoc Networks, Network Security, Algorithms, Data Mining and Bio-informatics.