BSMR: Byzantine-Resilient Secure Multicast Routing in Multi-hop Wireless Networks

Reza Curtmola
Department of Computer Science
Johns Hopkins University
crix@cs.jhu.edu

Cristina Nita-Rotaru
Deptartment of Computer Science
Purdue University
crisn@cs.purdue.edu

March 2007
CSD-TR-07-005

Abstract

Multi-hop wireless networks rely on node cooperation to provide multicast services. The multi-hop communication offers increased coverage for such services, but also makes them more vulnerable to insider (or Byzantine) attacks coming from compromised nodes that behave arbitrarily to disrupt the network. In this work we identify vulnerabilities of on-demand multicast routing protocols for multi-hop wireless networks and discuss the challenges encountered in designing mechanisms to defend against them. We propose BSMR, a novel secure multicast routing protocol designed to withstand insider attacks from colluding adversaries. Our protocol is a software-based solution and does not require additional or specialized hardware. We present simulation results which demonstrate that BSMR effectively mitigates the identified attacks.

I. Introduction

Multicast routing protocols deliver data from a source to multiple destinations organized in a multicast group. In the last few years, several protocols [1]–[7] were proposed to provide multicast services for multi-hop wireless networks.

A major challenge in designing protocols for wireless networks is ensuring robustness to failures and resilience to attacks. Wireless networks provide a less robust communication than wired networks due to frequent broken links and a higher error rate. Security is also more challenging in multi-hop wireless networks because the open medium is more susceptible to outside attacks and the multi-hop communication makes services more vulnerable to insider attacks coming from compromised nodes. Although an effective mechanism against outside attacks, authentication is not sufficient to protect against insider attacks because an adversary that compromised a node also gained access to the cryptographic keys stored on it. Insider attacks are also known as Byzantine [8] attacks and protocols able to provide service in their presence are referred to as Byzantine resilient protocols.

Previous work focused mainly on the security of unicast services. Several routing protocols [9]–[12] were proposed to cope with outsider attacks or insider attacks [11], [13]–[16]. Methods proposed to address insider threats in unicast routing include monitoring [13], multi-path routing [11], [14] and acknowledgment-based feedback [15], [16]. The problem of secure multicast in wireless networks was less studied and only outside attacks were addressed [17].

Security aspects in multicast protocols relate to either routing specific security, such as the management of the routing structure and data forwarding, or application specific security such as data confidentiality and authenticity. In this work we are concerned with multicast routing specific security.

Several differences make the multicast communication model more challenging than its unicast counterpart. Designing secure multicast protocols for wireless networks requires a more complex trust model, as nodes which are members of the multicast group cannot simply organize themselves in a dissemination structure without the help of other non-member nodes acting as routers. Unlike unicast protocols which establish and maintain routes between two nodes, multicast protocols may establish and maintain more complex structures, such as trees or meshes. For example, protocols relying on trees require additional operations such as route activation, tree pruning and tree merging. These actions do not have a counterpart in the unicast case and may expose the routing protocol to new vulnerabilities. Last but not least, multicast protocols deliver data from one sender to multiple receivers making scalability a major problem when designing attack-resilient protocols. In particular, solutions that offer resiliency against Byzantine attacks for unicast are not scalable in a multicast setting. For example, multi-path routing affects significantly the data dissemination efficiency, while strategies based on acknowledgments have high overhead.

We study vulnerabilities of multicast routing protocols in multi-hop wireless networks and propose a new protocol that provides resilience against Byzantine attacks. Specifically:

- We identify several aspects that make the design of attack-resilient multicast routing protocols more challenging than their unicast counterpart, such as a more complex trust model and underlying routing structure, and scalability. We focus on tree-based multicast routing protocols and discuss attacks against such protocols.
- We propose BSMR, an on-demand Byzantine-resilient multicast protocol for multi-hop wireless networks. BSMR uses a selective data forwarding detection mechanism that relies on a reliability metric capturing adversarial behavior. Nodes determine the reliability of links by comparing the perceived data rate with the one advertised by the source. Adversarial links are avoided during the route discovery phase. BSMR also prevents attacks that try to prevent or arbitrarily influence route establishment.
- We show through simulations that the impact of several Byzantine attacks (flood rushing, black hole and wormhole) on a previously proposed secure multicast routing protocol is considerable and cannot be ignored. We also demonstrate through simulations that our protocol BSMR mitigates the attacks, while incurring a small overhead.
- Through simulations, we confirm some results observed also in unicast settings [16]. Without adequate protection, the wormhole attack is more effective than the black hole attack (causing more damage per adversary) and strategic adversarial positioning can significantly amplify the strength of an attack. We also reveal new results that are multicast-specific. We show that the impact of flood rushing is related to the group membership status of adversarial nodes, and it varies considerably depending on whether or not adversaries explicitly join the group and depending on their join order.

The remainder of the paper is organized as follows. Section II overviews related work. Section III presents our network and system models. We discuss the attacks against multicast in IV-B and present BSMR in Section V. We present experimental results in Section VI and conclude in Section VII.

II. Related Work

Significant work focused on the security of unicast wireless routing protocols. Several secure routing protocols resilient to outside attacks were proposed in the last few years such as Ariadne [11], SEAD [10], ARAN [12], and the work in [9].

Wireless specific attacks such as flood rushing and wormhole were recently identified and studied. RAP [18] prevents the rushing attack by waiting for several flood requests and then randomly selecting one to forward, rather than always forwarding only the first one. Techniques to defend against wormhole attacks include *Packet Leashes* [19] which restricts the maximum transmission distance by using time or location information, Truelink [20] which uses MAC level acknowledgments to infer if a link exists or not between two nodes, and the work in [21], which relies on directional antennas.

The problem of insider threats in unicast routing was studied in [11], [13]–[16]. Watchdog [13] relies on a node monitoring its neighbors if they forward packets to other destinations. If a node does not overhear a neighbor forwarding more than a threshold number of packets, it concludes that the neighbor is adversarial. The scheme does not require any explicit cryptography and is effective against the basic black hole attack in single rate fixed transmission power networks. SDT [14] and Ariadne [11] use multi-path routing to prevent a malicious node from selectively dropping data; the disadvantage is that in a sparsely connected network, where the number of available disjoint paths is small, all of the discovered paths may contain an attacker and thus, the schemes will be less effective. ODSBR [15], [16] provides resilience to Byzantine attacks caused by individual or colluding nodes by detecting malicious links based on an acknowledgement-based feedback technique.

Most of the work addressing application security issues related to multicast in wireless networks focused on the problem of group key management in order to ensure data confidentiality and authenticity [22]–[26]. Work studying multicast routing specific security problems in wireless networks is scarce with the notable exception of the authentication framework by Roy *et al.* [17]. The framework allows MAODV to withstand several external attacks targeted against the creation and maintenance of the multicast tree. However, it does not provide resilience against Byzantine attacks.

Multicast routing specific security was also studied in overlay networks [27]–[29]. Solutions proposed exploit overlay specific properties such as: Existence of network connectivity between each pair of nodes which allows

nodes to directly probe non-neighboring nodes, and highly redundant connectivity which guarantees that many disjoint paths exist. None of these properties hold in multi-hop wireless networks.

III. Network and System Model

Network Model. We consider a multi-hop wireless network where nodes participate in the data forwarding process for other nodes. We assume that the wireless channel is symmetric. All nodes have the same transmitting power and consequently the same transmission range. The receiving range of a node is identical to its transmission range.

Nodes are not required to be equipped with additional hardware such as GPS receivers or tightly synchronized clocks. Also, nodes are not required to be tamper resistant: If an attacker compromises a node, it can extract all key material, data or code stored on that node.

We do not address attacks against lower layers such as the MAC or the physical layer. We assume that the physical layer uses jamming-resilient techniques such as direct sequence spread spectrum (DSSS) or frequency hopping spread spectrum (FHSS) (as in the case of 802.11).

Multicast Protocol. We assume a tree-based on-demand multicast protocol [6], which maintains bi-directional multicast trees connecting multicast sources and receivers. Each tree defines a multicast group. The multicast source is a special node, the group leader, whose role is to eliminate stale routes and coordinate group merges. Route freshness is indicated by a group sequence number updated by the group leader and broadcast periodically as a message in the entire network. For convenience, we refer to this message as a *GroupHello* message. Higher group sequence numbers denote fresher routes.

The main operations of the protocol are route discovery, route activation and tree maintenance. During route discovery a node discovers a path to a node that is part of the multicast tree. A requester first broadcasts a route request message that includes the latest known group sequence number. The route request message is flooded in the network using a basic flood suppression mechanism and establishes reverse routes to the source of the request. Upon receiving the route request, a node that is part of the multicast tree and has a group sequence number at least as large as the one in the route request, generates a route reply message and unicasts it on the reverse route. The route reply message includes the last known group sequence number and the number of hops to the node that originated the route reply.

During route activation, the requester selects the freshest and shortest route (i.e., with the smallest number of hops to the multicast tree) from the routes returned by route discovery and activates that route by unicasting an activation message.

Three main operations ensure the tree maintenance: Tree pruning, broken link repair and tree merging. Tree pruning occurs when a group member that is a leaf in the multicast tree decides to leave the group. To prune itself from the tree, the node sends a message to indicate this to its parent. The pruning message travels up the tree causing leaf nodes that are not members of the multicast group to prune themselves from the tree, until it reaches either a non-leaf node or a group member. A non-leaf group member must continue to act as a router and cannot prune itself from the multicast tree.

A node initiates a link repair when the upstream link in the multicast tree breaks. If the node cannot reconnect to the tree, it means the tree is partitioned. In this case the node runs a special procedure to prune non-member leaf nodes and elect a group leader for the partition. When two partitions of the same tree reconnect, the leader of one of the partitions coordinates the merge of the partitions, suppressing the other leader.

IV. Attacks Against Multicast Routing

A. Adversarial Model

Nodes may exhibit Byzantine behavior, either alone or colluding with other nodes. Examples of such behavior include: Not forwarding packets, injecting, modifying or replaying packets, rushing packets or creating wormholes. We refer to any arbitrary action by authenticated nodes resulting in disruption of the routing service as Byzantine behavior, and to such an adversary as a Byzantine adversary. Adversaries do not have control on the physical and MAC layers.

We consider a three-level trust model that captures the interactions between nodes in a wireless multicast setting and defines a node's privileges: A first level consists of the source which must be continually available and assumed

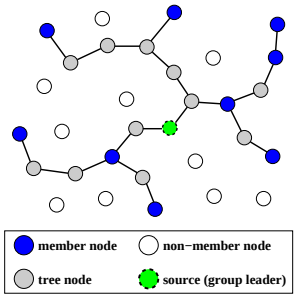


Fig. 1: Types of nodes in a multicast setting for ad hoc wireless networks.

not to be compromised; a second level consists of the multicast group member nodes, which are allowed to initiate requests for joining multicast groups; and a third level consists of non-member nodes which participate in the routing but can not initiate group join requests. In order to cope with Byzantine attacks, even group members are not fully trusted.

Fig. 1 gives an example with the types of nodes that can appear in a multicast setting for wireless networks. We refer to the nodes in the multicast tree as *tree nodes*. Tree nodes can be either *member nodes* or *non-member nodes*.

A common attack in open networks is the Sybil attack [30], in which a single adversary can control a significant fraction of a network by claiming multiple identities. Our adversarial model does not include this attack, under the assumption that identities (i.e., certified public/private key pairs) cannot be easily obtained. Nonetheless, independently of the routing protocol, Sybil attackers can be detected by employing several mechanisms proposed for sensor networks [31] and MANETs [32].

This work only considers attacks targeted against the network level. We do not focus on preventing traffic analysis.

B. Attacks in Multicast in Multi-Hop Wireless Networks

An adversary can attack the control messages corresponding to the route discovery, route activation and tree management operations, or can attack data messages. The route discovery phase can be disrupted by outside attackers creating undesired results by injecting, replaying, or modifying control packets. Nodes that are not in the tree can mislead other nodes into believing that they found and are connected to the tree. Nodes can flood the network with bogus requests for joining multicast groups. A Byzantine adversary can prevent a route from being established by dropping the request and/or response, or can influence the route selection by using wireless specific attacks such as wormhole and flood rushing. A Byzantine adversary can also modify the packets carrying the route selection metric such as hop count or node identifiers.

An attacker can prevent a path from being activated by injecting bogus route activation messages, or by dropping correct route activation messages. A node authorized to join a multicast group can initiate route activation packets to more than one tree node, which may result in unnecessary branches being grafted to the multicast tree.

Nodes can maliciously report that other links are broken or generate incorrect pruning messages resulting in correct nodes being disconnected from the network or tree partitioning. In the absence of authentication, any node can pretend to be the group leader. Although many routing protocols do not describe how to select a new group leader when needed, we note that the leader election protocol can also be influenced by attackers.

Attacks against data messages consist of eavesdropping, modifying, replaying, injecting data, or selectively forwarding data after being selected on a route. A special form of packet delivery disruption is a denial of service attack, in which the attacker overwhelms the computational, sending or receiving capabilities of a node. In general, data source authentication, integrity and encryption can solve the first attacks and are usually considered application specific security. Defending against selective data forwarding and denial of service cannot be done exclusively by using cryptographic mechanisms.

V. Secure Multicast Routing Protocol

A. BSMR Overview

Our protocol, BSMR, ensures that multicast data is delivered from the source to the members of the multicast group, even in the presence of Byzantine attackers, as long as the group members are reachable through non-adversarial paths and a non-adversarial path exists between a new member and a node in the multicast tree. To

achieve this strong guarantee, BSMR builds on the basic operation of the tree-based on-demand protocol presented in Sec. III.

To eliminate a large class of outside attacks we use an authentication framework that ensures only authorized nodes can perform certain operations (e.g., only tree nodes can perform tree operations and only nodes that possess valid group certificates can connect to the group multicast tree). For example, only member nodes can send route request and route activation messages, and only tree nodes can reply to route activation messages.

BSMR mitigates inside attacks that try to prevent a node from establishing a route to the multicast tree by flooding both route request and route reply, and by using a timeout based mechanism which ensures a path is established even if route activation messages are dropped. If an adversarial-free route exists, BSMR guarantees that a route is established.

BSMR provides resilience to selective data forwarding attacks by using a reliability metric that captures adversarial behavior. The metric consists of a list of link weights where high weights correspond to low reliability. Each node maintains its own *weight list* and includes it in each route request to ensure that a new route to the tree avoids adversarial links.

A link's reliability is determined based on the number of packets successfully delivered on that link. Tree nodes monitor the rate of receiving data packets and compare it with the transmission rate indicated by the source in the form of a MRATE message. If the perceived transmission rate falls below the rate indicated in the MRATE message by more than a threshold, an honest node that is a direct descendant of an adversarial node updates its weight list by penalizing the link to its parent and then tries to discover a new route to the tree. Only weights corresponding to penalized links are included in route requests, all non-faulty links have a default weight of 1.

A node can be in one of two states, *Disconnected* or *Connected*, and has two timers: A *waiting to connect timer* (*WTC_Timer*) associated with its tree connection status, and a timer associated with receipt of MRATE messages (*MRATE_Timer*). A node's state and the status of its timers determine the node's reaction to various events in the network.

BSMR's approach to defend against selective data forwarding attacks is generic and can protect against potentially unknown attacks as long as drops in delivery ratio are detected and correctly attributed to malicious entities.

Strategies based on end-to-end acknowledgments, although shown effective in unicast [14], [16], are not scalable in multicast. As the size of the multicast group increases, ACK implosion occurs at the source, which may cause a drastic decrease in data delivery and a significant increase in packet delivery latency [33]. Mechanisms such as feedback aggregation at intermediate routers, or combination of ACK/NACK messages proposed to address congestion control for multicast protocols in wired [34] and wireless [35] networks can not be applied in networks with Byzantine adversaries. In such networks, feedback aggregation becomes challenging if nodes cannot be relied upon to perform it correctly, while NACKs can lead to blacklisting of innocent nodes and a node may have a difficult time "proving" it did not receive a packet.

Without loss of generality, we limit our description to one multicast group. Below we describe the previously mentioned authentication framework, the *route discovery*, the *route activation*, *multicast tree maintenance* and the *selective data forwarding detection* mechanisms.

B. Authentication Framework

We assume that nodes have a method to determine the source authenticity of the received data (e.g., TESLA [36]). This allows a node to correctly determine the rate at which it receives multicast data.

In order to protect from external attacks against the creation and maintenance of the multicast tree BSMR uses a framework similar with the one in [17]. The framework prevents unauthorized nodes to be part of the network, of a multicast group, or of a multicast tree. These forms of authentication correspond to the trust model described in Sec. IV-A. Each node authorized to join the network has a pair of public/private keys and a *node certificate* that binds its public key to its IP address. Each node authorized to join a multicast group has an additional *group certificate* that binds its public key and IP address to the IP address of the multicast group.

Secure Tree Token Dissemination. Nodes in the multicast tree are authenticated using a *tree token*, which is periodically refreshed and disseminated in the tree by the group leader with the help of *pairwise shared keys* established between tree neighbors. Thus, only nodes that are currently on the tree will have a valid tree token. To allow any node in the network to check that a tree node possesses a valid tree token, the group leader periodically

broadcasts in the entire network a tree token authenticator $f(tree\ token)$, where f is a collision resistant one-way function. Nodes can check the validity of a given tree token by applying the function f to it and comparing the result with the latest received tree token authenticator.

Hop Count Authentication. To prevent tree nodes from claiming to be at a smaller hop distance from the group leader than they actually are, we use a technique based on a hash chain, similar with the technique proposed by Zapata [37]. Let h be a one-way collision-resistant hash function and let $\mathtt{MAX}$ be the maximum hop-length of a path from a tree node to the group leader (e.g., $\mathtt{MAX}$ is the network diameter). We use the notation h^x to denote the function h applied x times.

The group leader chooses a random number s and computes the value $hop\ count\ anchor = h^{\mathtt{MAX}}(s)$. The group leader includes the following information in messages sent in the multicast tree:

$$(s, 0, \mathtt{MAX}, h^{\mathtt{MAX}}(s)),$$

where the fields $\mathtt{MAX}, h^{\mathtt{MAX}}(s)$ are digitally signed by the group leader. In general, a tree node receives the following information from its tree parent:

$$(hop\ count\ authenticator, \mathsf{d}, \mathtt{MAX}, h^{\mathtt{MAX}}(s)),$$

where d is the parent's hop distance to the group leader and $h^{\mathtt{MAX}}(s)$ is the hop count anchor. The hop count authenticator is set to a value that allows a node to prove its hop distance to the group leader. The tree node verifies the signature on $(\mathtt{MAX}, h^{\mathtt{MAX}}(s))$ and checks if $h^{\mathtt{MAX}-\mathsf{d}}(hop\ count\ authenticator) = h^{\mathtt{MAX}}(s)$. If this holds, the node sends to its tree children a message in which the first two fields are updated, while the last two fields (signed by the group leader) are preserved:

$$(h(hop\ count\ authenticator), \mathsf{d}+1, \mathtt{MAX}, h^{\mathtt{MAX}}(s)).$$

For example, a direct tree child of the group leader receives $(s, 0, \mathtt{MAX}, h^{\mathtt{MAX}}(s))$, checks if $h^{\mathtt{MAX}-0}(s) = h^{\mathtt{MAX}}(s)$ and sends to its tree children $(h(s), 1, \mathtt{MAX}, h^{\mathtt{MAX}}(s))$.

The one-way hash function h, prevents a node whose parent is d hops away from the group leader to claim to be at a distance smaller than $\mathsf{d}+1$ from the group leader.

This hop count authentication mechanism is used by the group leader when sending tree token and MRATE messages. It is also used during route discovery to allow nodes that forward a route reply message to prove their hop distance from the tree node that initiated the route reply message.

The hop count anchor is also included by the group leader in *GroupHello* messages, which are broadcast periodically in the entire network. This allows a tree node to prove its hop distance from the group leader to any node in the network.

C. Route Discovery

BSMR's route discovery allows a node that wants to join a group to find a route to the multicast tree. The protocol follows the route request/route reply procedure used by on-demand routing protocols, with several differences. To prevent outsiders from interfering, all route discovery messages are authenticated using the public key corresponding to the network certificate. Only group authenticated nodes can initiate route requests and the group certificate is required in each request. Tree nodes use the tree token to prove their tree status.

Several mechanisms are used to address internal attackers: (a) both route request and reply are flooded in order to ensure that, if an adversarial-free path exists, it will be found; (b) the path selection relies on the weights list carried in the response flood and allows the requester to select a non-adversarial path; (c) the propagation of weights and path accumulation is performed using an onion-like signing to prevent forwarding nodes from modifying the path carried in the response.

As seen in Fig. 2, the requesting node broadcasts a route request (RREQ) message that includes the node identifier and its weight list, the multicast group identifier, the last known group sequence number, a request sequence number and its group certificate (lines 1-2). The RREQ message is flooded in the network (line 10) and only new valid requests are processed by intermediate nodes (lines 3-4).

When a tree node receives for the first time a RREQ from a requester and the node's group sequence number is at least as great as that contained in the RREQ, it initiates a response (lines 3-5). The tree node verifies the included group certificate and broadcasts a route reply (RREP) message that includes that node's identifier, the group identifier, its recorded group sequence number, a response sequence number, the requester's identifier, and

Procedure list:
CreateSign(item1, item2, ...) - creates a message out of the concatenated item list and signs it using this node's private key
Broadcast(message) - broadcasts a message
Send(message, node) - unicasts a message to a node
VerifySig(node, signature) - verifies the signature using the a node's public key and exits the procedure if the signature is invalid
Find(list, key1, key2, ...) - returns a list item determined by a list of keys, or NULL if the item does not exist
UpdateList(list, item) - if the item exists in the list then replaces the item; if the item does not exist in the list then inserts the item
LinkWeight(weight_list, nodeA, nodeB) - returns the listed weight of the link between nodeA and nodeB, or one if the link is not listed
Encrypt(node, data) - encrypts data using a node's public key
Decrypt(node, encrypted_data) - returns a decryption of encrypted_data using a node's private key
VerifyHopCount(auth_data) - verifies if the hop count authentication data auth_data is valid and exits the procedure if it is not valid
UpdateHopCount(node, auth_data) - updates the hop count authentication data auth_data according to a node's position in the tree
VerifyTreeToken(token, token_authenticator) - verifies the validity of a token according to token_authenticator and exits if the token is invalid

Executed at node requester **to initiate a new request message:**
1. req = CreateSign(RREQ, requester_id, group_id, group_seq, req_seq, weight_list, GC)
2. Broadcast(req)

Executed at node this_node **when a request message** req **is received:**
3. **if** (Find(requests_list, req.requester_id, req.group_id, req.req_seq) = NULL) **then**
4. VerifySig(req.requester_id, req.signature)
5. **if** ((this_node.group_id = req.group_id) AND (this_node.group_seq $\geq$ req.group_seq)) **then**
6. VerifySig(req.requester_id, req.GC)
7. res = CreateSign(RREP, this_node.id, this_node.group_id, this_node.group_seq, req.req_seq, req.requester_id, req.weight_list, Encrypt(req.requester_id, this_node.tree_token), hop_count_auth_info)
8. Broadcast(res)
9. **else**
10. Broadcast(req)
11. UpdateList(requests_list, req)

Executed at node this_node **when a response message** res **is received:**
12. update = **false**; prev_node = res.responder_id; total_weight = 0
13. **if** ((this_node.group_id = res.group_id) AND (this_node.group_seq $\geq$ res.group_seq)) **then**
14. **exit**
15. **for** ($i = 0$; $i <$ res.no_hops; i++) **do**
16. total_weight += LinkWeight(res.weight_list, prev_node, res.hops[i].node)
17. prev_node = res.hops[i].node
18. res.total_weight = total_weight + LinkWeight(res.weight_list, prev_node, this_node)
19. prev_response = Find(responses_list, res.requester_id, res.group_id, res.res_seq)
20. **if** (prev_response $\neq$ NULL) **then**
21. **if** (res.total_weight $<$ prev_response.total_weight) **then**
22. update = **true**
23. **else**
24. update = **true**
25. **if** (update) **then**
26. VerifyHopCount(res.hop_count_auth_info)
27. VerifySig(res.responder_id, res.signature)
28. **for** ($i = 0$; $i <$ res.no_hops; i++) **do**
29. VerifySig(res.hops[i].node, res.hops[i].signature)
30. **if** (this_node = res.requester_id) **then**
31. tree_token = Decrypt(this_node, res.encrypted_tree_token)
32. VerifyTreeToken(tree_token, this_node.tree_token_authenticator)
33. **else**
34. UpdateHopCount(this_node, res.hop_count_auth_info)
35. CreateSignSend(res, this_node)
36. forward_route_entry.prev_node = prev_node
37. forward_route_entry.req_seq = res.requester_id
38. UpdateList(forward_routes_list, forward_route_entry)
39. UpdateList(responses_list, res)

Fig. 2: Route Discovery algorithm

the weight list from the request message (lines 6-8). To prove its current tree node status, the node also includes in the response the current tree token, encrypted with the requester's public key. It also includes the hop count authentication information (see Sec. V-B) to allow nodes that forward RREP to prove their hop distance from this

```
Executed at node requester when an activation message act is sent:
 1. forward_route_entry = Find(forward_routes_list, group_id, req_seq)
 2. act = CreateSign(MACT, requester_id, responder_id, req_seq, h(requester_id, tree_token))
 3. Send(act, forward_route_entry.prev_node)
 4. start WTC_Timer

Executed at node this_node when an activation message act is received from node neighbor:
 5. VerifySig(act.requester_id, act.signature)
 6. forward_route_entry = Find(forward_routes_list, act.group_id, act.req_seq)
 7. if (forward_route_entry ≠ NULL) then
 8.     if (this_node = act.responder_id) then
 9.         VerifyTreeToken(tree_token, act.tree_token_authenticator)
10.         next_hop.node = forward_route_entry.prev_node
11.         next_hop.direction = upstream
12.         UpdateList(tree_neighbors_list, next_hop)
13.     else
14.         start WTC_Timer
15.         next_hop.node = neighbor; next_hop.direction = downstream
16.         UpdateList(tree_neighbors_list, next_hop)
17.         next_hop.node = forward_route_entry.prev_node
18.         next_hop.direction = upstream
19.         UpdateList(tree_neighbors_list, next_hop)
20.         this_node.group_id = act.group_id
21.         Send(act, forward_route_entry.prev_node)
```

Fig. 3: Multicast Route Activation algorithm

tree node.

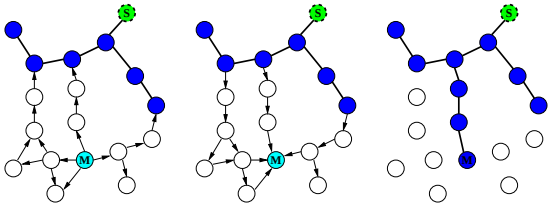
The RREP message is flooded in the network until it reaches the requester, using the following *weighted flood suppression* mechanism. Tree nodes with a group sequence number at least as great as that in the RREP ignore RREP messages (lines 13-14). Otherwise, a node computes the total path weight by summing the weight of all the links on the specified path from the multicast tree to itself (lines 15-18). If the total weight is less than any previously forwarded matching response (same requester, multicast group and response sequence number), the hop count authentication is valid and all the signatures accumulated on the response are valid, then the node appends its identifier to the end of the message, updates the hop count authentication information, signs the entire message and rebroadcasts it. As the RREP message propagates across the network, nodes establish the *forward route* by setting pointers to the node from which the RREP was received. Although several tree nodes may initiate the response flood, the weighted flood suppression mechanism insures the communication overhead is equivalent to only one flood (lines 25-39).

When the requester receives a response, it performs the same computation as an intermediate node during the response propagation and also checks the validity of the tree token included in the RREP message (lines 30-32). The requester updates its information upon receipt of a valid response that contains a better path according to our reliability metric.

Fig. 4(a) and 4(b) show the propagation of RREQ and RREP messages when node M wants to join the multicast group.

D. Multicast Route Activation

As shown in Fig. 3, the requester signs and unicasts on the selected route a multicast activation (MACT) message that includes its identifier, the group identifier, the sequence number used in the route request phase (lines 1-3). The MACT message also includes a one-way function applied on the tree token extracted from RREP, $f(requester, tree\ token)$, which will be checked by the tree node that sent the RREP message to verify that the node that activated the route is the same as the initial requester (line 9). An intermediate node on the route checks if the signature on MACT is valid and if MACT contains the same sequence number as the one in the original RREQ (lines 5-7). The node then adds to its list of tree neighbors the previous node and the next node on the route as downstream and upstream neighbors, respectively, and sends MACT along the forward route (lines 15-21). During the propagation of the MACT message tree neighbors use their public keys to establish pairwise shared keys, which will be used to securely exchange messages between tree neighbors.



(a) RREQ propagation (b) RREP propagation (c) Route activation

Fig. 4: Multicast Join Operation.

The requester and the nodes that received the MACT message could be prevented from being grafted to the tree by an adversarial node, selected on the forward route, which drops the MACT message. To mitigate the attack, these nodes will start a *waiting to connect timer* (*WTC_Timer*) (lines 4, 14) upon whose expiration nodes isolate a faulty link and initiate Route Discovery (Fig. 6 of Sec. V-F). The timers are set to expire after a value proportional to a node's hop distance to the tree, in the hope that the nodes closer to the tree will succeed in avoiding the adversarial node and will manage to connect to the tree. After a node becomes aware of its expected receiving data rate, it cancels its *WTC_Timer* and behaves as described in Sec. V-F.

Fig. 4 shows that a joining node may receive several RREP messages corresponding to several different routes to the tree, out of which only one route will be selected and activated. Malicious nodes that try to activate more than one route to the tree can be prevented from doing so by using a mechanism similar with Watchdog [13]: When a node's neighbors overhear that the node sends twice a MACT message for the same multicast group over a short time interval, they can choose to isolate the malicious node. We note that route activation is a legitimate action which is not forbidden by the protocol semantics.

E. Multicast Tree Maintenance

Routing messages exchanged by tree neighbors (e.g., pruning messages) are authenticated using the pairwise keys shared between tree neighbors. If a malicious node prunes itself even if it has a subtree below it, the honest nodes in this subtree will reconnect to the tree following the procedure described in Sec. V-F. The link repair procedure is initiated by nodes that detect a broken link and is similar with Route Discovery.

The group leader periodically broadcast in the entire network a signed *GroupHello* message that contains the current group sequence number, the tree token authenticator and the hop count anchor (described in Sec. V-B).

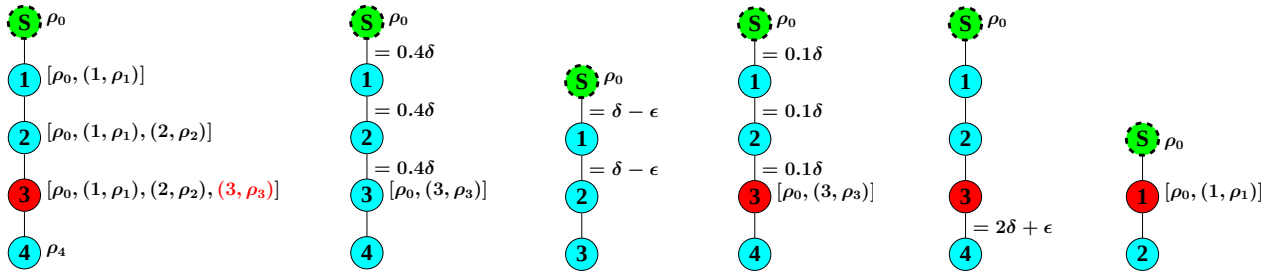
F. Selective Data Forwarding Detection

The source periodically signs and sends in the tree a multicast rate (MRATE) message that contains its data transmission rate ρ_0. As this message propagates in the multicast tree, nodes may add their perceived transmission rate to it. Each tree node keeps a copy of the last heard MRATE packet. The information in the MRATE message allows nodes to detect if tree ancestors perform selective data forwarding attacks. Depending on whether their perceived rate is within acceptable limits of the rate in the MRATE message, nodes alternate between two states. The initial state of a node is *Disconnected*; After it joins the multicast group and becomes aware of its expected receiving data rate, the node switches to the *Connected* state. Upon detecting a selective data forwarding attack, the node switches back to the *Disconnected* state.

A network operating normally exhibits some amount of natural "loss", which may cause the rate perceived by a node to be smaller than the rate perceived by its tree parent. This natural rate decrease is cumulative as data travels further away from the source. We define a threshold δ as the upper bound for the tolerable loss rate on a single link. Depending on certain conditions, nodes may add their own perceived rates to the MRATE message. In addition to its rate, a node also adds its identifier, its hop distance to the group leader and the hop count authentication information (described in Sec. V-B). Thus, an MRATE message has the format described below, where ρ_0 is the source's rate and $\rho_i, 1 \leq i \leq k$, are the rates added by nodes on the path to the source:

$$\text{MRATE} = (\rho_0, (\mathsf{id}_1, \mathsf{d}_1, \rho_1), (\mathsf{id}_2, \mathsf{d}_2, \rho_2), \ldots, (\mathsf{id}_k, \mathsf{d}_k, \rho_k))$$

Note that when a node adds its rate to MRATE, it also signs the entire MRATE message. When a tree node receives an MRATE message, it checks all the accumulated signatures before forwarding it. The node also checks



(a) MRATE propagation (b) MRATE propagation (c) $\Delta = 1.5\delta$ (d) $\Delta > 2\delta$ (e) $\Delta = 2.5\delta$ (f) single link with loss $\geq \delta$

Fig. 5: Strategies for selective data forwarding attack detection: (a) basic strategy; (b)-(f) advanced strategy.

the validity of the hop count authentication information. Because the entries in the MRATE message contain the hop distance from the group leader, a node can compute the hop distance between two nodes that added an entry to MRATE, or the hop distance between itself and a node that added an entry to MRATE (as the difference between two distances).

Basic Strategy. We first present a basic strategy that allows nodes to detect selective data forwarding attacks. When a tree node receives an MRATE message, it always adds its perceived rate and forwards the MRATE message down the tree. Whenever a tree node notices a difference between the last rate in MRATE and its perceived rate greater than δ, it will incriminate the link to its parent and will initiate the Route Discovery procedure in order to find a new route to the tree.

For example, in Fig. 5(a) nodes 1, 2 and 4 are honest, while node 3 is malicious and performs a selective data forwarding attack. There are two cases, depending on whether or not node 3 adds its rate:

- If node 3 does not add any rate to MRATE, then node 4 will receive MRATE $= (\rho_0, (1, \rho_1), (2, \rho_2))$, will notice that $\rho_4 - \rho_2 > \delta$, will penalize the link $(3, 4)$, and will initiate the Route Discovery procedure.
- If node 3 adds its rate to MRATE, then node 4 will receive MRATE $= (\rho_0, (1, \rho_1), (2, \rho_2), (3, \rho_3))$, will notice that $\rho_4 - \rho_3 > \delta$, will penalize the link $(3, 4)$, and will initiate the Route Discovery procedure.

Although effective in preventing the attack, this strategy requires all tree nodes to add their rates to the MRATE message. After a node adds its rate, it also needs to digitally sign the entire MRATE message before forwarding it. This computational overhead may be rather significant and it is quite unnecessary under ideal conditions (i.e., small natural losses and no adversaries present). We present next a more efficient strategy, which can reduce the computational load on tree nodes, especially under ideal conditions.

Advanced Strategy. A more efficient strategy is that a node concatenates its identifier and rate to MRATE only if the node's perceived rate is smaller than the last recorded rate in MRATE by more than δ. The node signs the entire message before forwarding it. These added rates serve as proofs that nodes which previously forwarded the MRATE message did not perceive losses much larger than natural losses.

In Fig. 5(b), there is a natural loss of 0.4δ on each of the links (S,1), (1,2) and (2,3). As a result, nodes 1 and 2 do not add their rates and only node 3 adds its rate ρ_3 to MRATE, because it notices that $\rho_0 - \rho_3 = 1.2\delta$, which is larger than δ.

Notice in Fig. 5(d) that a malicious node 3 could introduce a large artificial decrease in the rate by simply adding a low rate to MRATE. Even if the rate perceived by node 3 is only 0.3δ smaller than ρ_0, node 3 could simply add a rate ρ_3 such that the difference $\rho_0 - \rho_3$ is arbitrarily large.

In order to prevent a malicious node from introducing a rate decrease significantly larger than δ, we use another threshold $\Delta > \delta$. Upon receiving an MRATE message, each node first checks if the difference between the last rate in MRATE and the node's perceived rate is greater than Δ. If so, this indicates that there exists at least an adversarial node in between this node and the node that added the last rate to MRATE. The first honest node that notices a difference larger than Δ incriminates the link to its tree parent as faulty (by using a multiplicative weight increase scheme) and assumes responsibility for finding a new route to the tree.

The question then arises: What should be the value of Δ relative to the value of δ? We argue next that $\Delta = 2\delta$.

We first show that we need $\Delta \geq 2\delta$. Assume that $\Delta < 2\delta$, for example $\Delta = 1.5\delta$. In Fig. 5(c), nodes 1, 2 and 3 are honest nodes, but on the links (S,1) and (1,2) there are natural losses just below the allowed threshold δ (i.e., the loss on each of these links is $\delta - \epsilon$, where ϵ is an arbitrarily small value). Node 2 will receive MRATE $= (\rho_0)$

and will notice that the difference $\rho_0 - \rho_2 = 2\delta - 2\epsilon$ is larger than Δ. As a result, node 2 will incriminate the link (1,2) and will initiate Route Discovery, even if none of the nodes are malicious. To avoid cases like this, the value of Δ needs to be set to at least 2δ.

We now show that we need $\Delta \leq 2\delta$. First, it is clear that we cannot allow Δ to be much larger than δ, because this will allow an adversarial node to artificially introduce a large rate decrease. In Fig. 5(d) we see that if Δ was set to a value much larger than 2δ (e.g., $\Delta = 10\delta$), then a malicious node 3 could artificially decrease the rate in MRATE by adding a false rate ρ_3 such that $\rho_0 - \rho_3 = \Delta - \epsilon$, without triggering an alarm at node 4. Second, assume that $\Delta > 2\delta$, for example $\Delta = 2.5\delta$. In Fig. 5(e), there are no losses on links (S,1), (1,2) and (2,3) and node 3 acts adversarially by selectively forwarding data packets. As a result, node 3 artificially introduces a loss of $2\delta + \epsilon$ on the link (3,4). Node 4 receives MRATE $= (\rho_0)$ and notices that $\rho_0 - \rho_4 = 2\delta + \epsilon$. Because node 3 did not add any rate to MRATE, there are only two possible cases:

- if the loss on the link (3,4) is less than δ, then $\rho_0 - \rho_3 > \delta$ and node 3 did not add its rate to MRATE, as it should have done; or
- if $\rho_0 - \rho_3 \leq \delta$, then the loss on the link (3,4) must be larger than δ.

In both cases node 4 will incriminate the link (3,4) to its parent and will initiate a route discovery. Thus, the value for Δ must be at most 2δ, otherwise a decrease larger than 2δ must immediately trigger a route discovery.

The advantage of this strategy is that under ideal conditions (small natural losses and no adversaries), the computational overhead of tree nodes required to sign MRATE messages is minimized. The disadvantage is that adversarial nodes can artificially decrease the rate in MRATE by up to $\Delta = 2\delta$. This is an inherent limitation of our general strategy which allows an upper bound of δ for losses registered on a single link.

The ability of an adversarial node to introduce an artificial rate decrease can be limited by noticing that a decrease larger than δ must involve at least two hops; that is, a malicious node cannot decrease the rate by more than δ such that the decrease corresponds to a single hop, without incriminating one of its own links. For example, in Fig. 5(f), node 1 is malicious, adds its rate to MRATE and selectively forwards data packets such that node 2 perceives a rate ρ_2, with $\rho_1 - \rho_2 > \delta$. Node 2 notices a loss larger than δ for the a single hop, will incriminate the link (1,2) and will initiate a Route Discovery.

Nodes in the subtree below the node that initiated a Route Discovery will notice a gap greater than Δ between the rates included in MRATE, or a gap greater than δ that corresponds to a single hop; they will defer taking any action to isolate the faulty link for an amount of time proportional to the distance from the node that already started the repair procedure, in the hope that the nodes closer to the faulty link will succeed in isolating it. Upon detecting that the expected data packet rate has been restored, nodes cancel the repair procedure.

Fig. 6 describes how a *Connected* node reacts to the following events: (1) Receipt of an MRATE message, (2) Timeout of the *MRATE_Timer*, and (3) Timeout of the *WTC_Timer*.

We now describe in detail a node's behavior when each of these events occurs. We start with the description of a node's actions upon receipt of an MRATE message (Event 1, Fig. 6). If a node has just been grafted to the multicast tree, but has not received an MRATE message yet, it is in the *Disconnected* state. Upon receipt of the first MRATE message, it switches to the *Connected* state and cancels its *WTC_Timer* (lines 1-3). Lines 5-9 capture the case of a *Connected* node which notices that MRATE contains a gap larger than Δ, or a gap larger than δ that corresponds to a single hop; this means that one of its ancestors already initiated a route discovery and the node stars its *WTC_Timer*. On the other hand, if the node's *WTC_Timer* is pending and the node notices a gap larger than Δ in MRATE, then it simply adds its perceived rate and forwards the MRATE message (lines 10-14). If MRATE contains no such large gap, the node can hope that it has been reconnected to the tree; the node cancels its *WTC_Timer* and switches into *Connected* state (lines 16-17). If the difference between the node's perceived rate and the last rate recorded in MRATE is larger than Δ, or if this difference is larger than δ and it corresponds to a single hop, then the node switches into *Disconnected* state, increases the weight of the link to its parent and initiates a route discovery with its updated weight list (lines 18-24). The node also adds its perceived rate to MRATE and signs it, so that the nodes in the subtree below it are aware that one of their ancestors has already taken steps to find a different route to the multicast tree (line 19). In case the node detects an acceptable decrease in the transmission rate, it just adds its perceived rate to MRATE and signs it (lines 25-26). The node then forwards the MRATE message (line 27). If the transmission rate decrease is less than δ, then the MRATE message is forwarded unmodified.

We use the following additional notation:
– ρ_{node} is the rate at which this node receives packets from its tree parent.
– hop_distance(node$_k$,node) is the hop distance between the node that added the k-th rate to MRATE and this node.

Event 1: Receipt of MRATE $= (\rho_0, (\mathsf{id}_1, \mathsf{d}_1, \rho_1), \ldots, (\mathsf{id}_k, \mathsf{d}_k, \rho_k))$

1. **if** (this is the first MRATE message received) **then**
2. switch to *Connected* state
3. cancel *WTC_Timer*
4. store MRATE message and cancel *MRATE_Timer*
5. **if** ((state = *Connected*) AND ($WTC\_Timer \neq$ PENDING)) **then**
6. **if** ((MRATE contains a "gap" larger than Δ) OR (MRATE contains a "gap" larger than δ that corresponds to a single hop)) **then**
7. start *WTC_Timer* timer
8. forward MRATE
9. **return**
10. **else if** (*WTC_Timer* = PENDING) **then**
11. **if** ((MRATE contains a "gap" larger than Δ) OR (MRATE contains a "gap" larger than δ that corresponds to a single hop)) **then**
12. MRATE = cat_and_sign(MRATE, $(\mathsf{id}_{\text{node}}, \mathsf{d}_{\text{node}}, \rho_{\text{node}})$)
13. forward MRATE
14. **return**
15. **else**
16. cancel *WTC_Timer*
17. switch to *Connected* state
18. **if** $((\rho_k - \rho_{\text{node}} > \Delta)$ OR $((\rho_k - \rho_{\text{node}} > \delta)$ AND (hop_distance(node$_k$, node) = 1)) **then**
19. MRATE = cat_and_sign(MRATE, $(\mathsf{id}_{\text{node}}, \mathsf{d}_{\text{node}}, \rho_{\text{node}})$)
20. **if** (*WTC_Timer* = PENDING) **then**
21. cancel *WTC_timer*
22. switch to *Disconnected* state
23. increase weight of the link to the parent
24. initiate Route Discovery
25. **else if** $(\rho_k - \rho_{\text{node}} > \delta)$ **then**
26. MRATE = cat_and_sign(MRATE, $(\mathsf{id}_{\text{node}}, \mathsf{d}_{\text{node}}, \rho_{\text{node}})$)
27. forward MRATE message
28. start *MRATE_Timer*

Event 2: Timeout of *MRATE_Timer*:

1. **if** ((state = *Connected*) AND ($WTC\_Timer \neq$ PENDING)) **then**
2. retrieve stored MRATE = $(\rho_0, (\mathsf{id}_1, \mathsf{d}_1, \rho_1), \ldots, (\mathsf{id}_k, \mathsf{d}_k, \rho_k))$
3. **if** $((\rho_k - \rho_{\text{node}} > \Delta)$ OR $((\rho_k - \rho_{\text{node}} > \delta)$ AND (hop_distance(node$_k$, node) = 1)) **then**
4. MRATE = cat_and_sign(MRATE, $(\mathsf{id}_{\text{node}}, \mathsf{d}_{\text{node}}, \rho_{\text{node}})$)
5. switch to *Disconnected* state
6. increase weight of the link to the parent
7. initiate Route Discovery
8. **else if** $(\rho_k - \rho_{\text{node}} > \delta)$ **then**
9. MRATE = cat_and_sign(MRATE, $(\mathsf{id}_{\text{node}}, \mathsf{d}_{\text{node}}, \rho_{\text{node}})$)
10. forward MRATE message

Event 3: Timeout of *WTC_Timer*:

1. switch to *Disconnected* state
2. increase weight of the link to the parent
3. initiate Route Discovery

Fig. 6: Selective Data Forwarding Detection algorithm

Tree nodes expect to periodically receive MRATE messages, otherwise the *MRATE_Timer* will expire. In this case, a node proceeds similarly as when it receives an MRATE message, but it considers the latest received MRATE message, which has been previously stored at the node (Event 2, Fig. 6). If the *WTC_Timer* expires, then a node simply initiates the route repair procedure, which involves incriminating the link to its tree parent and starting Route Discovery (Event 3, Fig. 6).

Note that each tree node stores the latest received MRATE message and uses it to re-initiate the propagation of MRATE if *MRATE_Timer* expires. Also, when *MRATE_Timer* expires a node compares its perceived rate with the expected rate from the stored MRATE message.

VI. Experimental Results

In this section, we study the effectiveness of BSMR in mitigating the identified Byzantine attacks. We compare our protocol with A-MAODV which is, to the best of our knowledge, the only security mechanism proposed to address routing specific security of on-demand multicast protocols in multi-hop wireless networks [17]. Although A-MAODV withstands several external attacks targeted against the creation and maintenance of the multicast tree, it does not provide additional resilience against Byzantine attacks.

Implementation. We implemented BSMR using the ns2 simulator [38], starting from an MAODV implementation [39]. We assumed the protocol uses RSA [40], [41] with 1024-bit keys for public key operations, AES [42] with 128-bit keys for symmetric encryptions and HMAC [43] with SHA1 as the message authentication code. The impact of these cryptographic operations is represented by adjusting the packet size and by introducing packet delay accordingly, as if the packet actually contained authenticating data (e.g., digital signatures or MACs), and as if CPU time was spent performing cryptographic operations.

The values used for δ and Δ were 10% and 20% of the source's rate, respectively. We developed a protocol-independent Byzantine attack simulation module for ns2.

A. Experimental Methodology

To capture a protocol's effectiveness in delivering data to the multicast group, we used as a performance metric the average packet delivery ratio (PDR), defined as:

$$PDR = \frac{P_r}{P_s \cdot N}$$

where P_r is the number of data packets received by all multicast group members, P_s is the number of data packets sent by the source and N is the size of the group.

Because external attacks can be prevented using the authentication framework described in Sec. V-B, we focus on the following three Byzantine attacks:

– **black hole attack:** One or several adversaries forward only routing control packets, while dropping all data packets.

– **wormhole attack:** Two colluding adversaries tunnel packets between each other in order to create a shortcut in the network. The adversaries use the low cost appearance of the wormhole to increase the probability of being selected on paths; once selected on a path, they attempt to disrupt data delivery by executing a black hole attack.

– **flood rushing attack:** One or several adversaries rush an authenticated flood through the network before the flood traveling through a legitimate route, exploiting the flood duplicate suppression technique used by many wireless routing protocols. This allows the adversaries to control many paths. Flood rushing can be used to increase the effectiveness of a black hole or wormhole attack. The attack can be implemented by simply ignoring the small randomized delays which are normally required to reduce the number of collisions.

In order to quantify the impact of adversarial positioning, we consider the following scenarios:

– **random placement:** adversaries are placed randomly in the simulation area;

– **strategic placement:** adversarial placement is as follows:

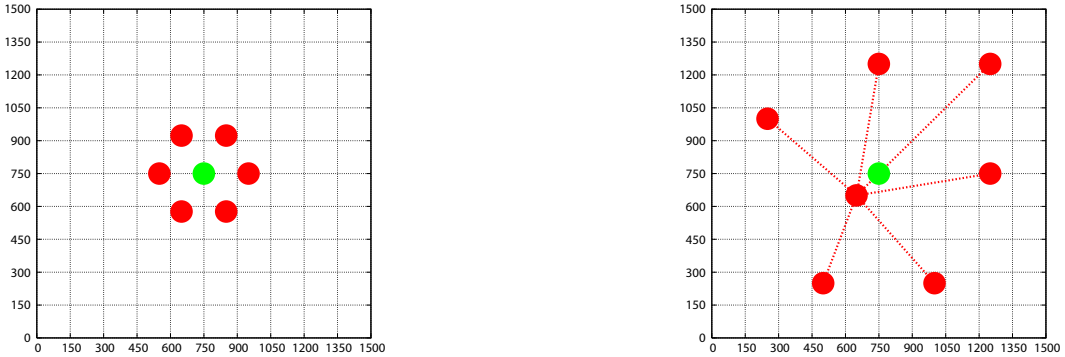
- **black hole attack:** adversaries are placed strategically around the multicast source, equidistant on a circle with radius of 200 meters.
- **wormhole attack:** Given k adversaries, one adversary is placed near the source, at coordinates (650,650). The other $k-1$ adversaries are placed throughout the simulation area so that the areas covered by their transmission range overlap as little as possible. Each one of the $k-1$ adversaries is connected via a wormhole tunnel to the adversary placed near the source, creating $k-1$ wormholes.

In Fig. 7, we illustrate the strategic adversarial placement for the black hole attack (in the presence of 6 adversaries) and for the wormhole attack (in the presence of 7 adversaries).

To study the influence of adversaries joining the multicast group and the order of joining, we consider two scenarios:

– **NJOIN:** adversarial nodes do not join the group;

– **JOIN:** adversarial nodes explicitly join the group before any of the honest members join. The adversaries are considered group members in the formula for PDR.



(a) Black hole: 6 adversaries (b) Wormhole: 7 adversaries

Fig. 7: Examples of strategic adversarial placement for the black hole and wormhole attacks

We chose these scenarios in order to study the impact of the attacks under a light set of conditions (adversaries are placed randomly, or they do not explicitly join the multicast group) and a more extreme set of conditions (adversaries are placed strategically, or they join the group before honest nodes do).

B. Simulation Setup

We performed simulations using the ns2 network simulator [38]. Nodes were set to use 802.11 radios with 2 Mbps bandwidth and 250 meters nominal range. The simulated time was 600 seconds. We randomly placed 100 nodes within a 1500 by 1500 meter area and the multicast source in the center of the area at coordinates (750,750). We experimented with different values for group size (10, 30 and 50), for number of adversaries (between 16% and 60% of the group size) and for maximum speed (0, 2 and 5 m/s).

Group members join the group sequentially in the beginning of the simulation, each one at an interval of 3 seconds. Then the source transmits multicast data for 600 seconds at a rate of 5 packets per second, each packet of 256 bytes (resulting in loads between 100-500 Kbps across all receivers). The members stay in the group until the end of the simulation. Adversaries added to the network replace honest nodes, thus modeling the capture of honest nodes.

We used a random way-point mobility model, modified to address concerns raised in [44]. Nodes select a speed uniformly between 10% and 90% of the given maximum speed to achieve a more steady mobility pattern and ensure that the average speed does not drop drastically over the course of the simulation. 300 seconds of mobility are generated before the start of the simulation so that nodes are already in motion. This allows the average speed and node distribution to stabilize before the simulation starts.

Each data point is averaged over 30 different random environments and over all group members. A-MAODV and BSMR are simulated in the same set of random environments in order to generate paired statistics (a standard method of statistical variance reduction). A paired T-test analysis of all our data shows that, in the presence of adversaries, the largest p-value for any particular attack configuration is 0.0053. Therefore, the observed differences between A-MAODV and BSMR are statistically significant with an interval confidence of over 99.4%. We evaluate the PDR as a function of the number of adversaries, for different group sizes and levels of mobility. The graphs in the next sections illustrates the effect of the attacks both with and without flood rushing.

C. Attack Resilience: Black Hole Attack

Impact of Adversarial Placement. Figures 8 and 9 show the results for random and strategic adversarial placement, respectively. For random placement we see that, for the same group size, the PDR of A-MAODV decreases as the number of adversaries increases. For the same number of adversaries, it also decreases as we increase the group size. However, random adversarial placement causes the number of group members in the subtree below an adversary to be low; thus a relatively large number of adversaries is needed to cause a significant disruption (e.g., 30 adversaries for a group of size 50 can cause the PDR to drop below 50%). In the presence of flood rushing, the PDR decreases further because adversaries actively try to get selected as part of the tree. The impact of flood rushing decreases as the group size and the nodal speed increase.

We notice that, for A-MAODV, increasing the speed does not have a negative effect on the PDR; On the contrary, at higher speeds we even see a slight increase in PDR. The effect of link breaks due to mobility is compensated by the fact that group members get a chance to reconnect to the multicast tree through a different path, possibly connected to the source through an adversarial-free path. For the same reason, the effect of flood rushing is diminished as the speed increases.

BSMR is effective in mitigating the attack (see Fig. 8). The PDR drops by less than 10% even in the presence of 20 adversaries. The influence of flood rushing is unnoticeable. This shows the effectiveness of BSMR's strategy against flood rushing, which includes the processing of all response flood duplicates and the metric capturing past behavior of adversarial nodes. Mobility causes a slight PDR decrease, which is natural because higher speeds will cause more link breaks.

Fig. 9 shows the results when adversaries are strategically positioned as described in Section VI-A. For A-MAODV we notice a drastic drop in the PDR. For example, at 0 m/s, when the group size is 30, only 5 adversaries (representing 16% of the group size) are able to reduce the PDR to 25% by executing the black hole attack with flood rushing. This is a direct consequence of the fact that an adversary is now selected in the tree closer to the root and the subtree below it may potentially contain many group members. For the same reason, the negative effect of the flood rushing attack is now emphasized when compared to the random placement case. We conclude that strategic adversarial positioning has a crippling effect on the performance of A-MAODV.

On the contrary, the effect of strategic adversarial positioning on BSMR is minor (Fig. 9). Like for random placement, the PDR drops by less than 10% even in the presence of 20 adversaries, at low nodal speeds. When more adversaries are present, we see a slightly larger PDR decrease because there are less available honest nodes left in the network to serve as intermediaries for the group members. The resilience of BSMR to attacks that otherwise have a devastating effect on A-MAODV validates the effectiveness of BSMR's approach.

Impact of Explicit Join and Join Order. To analyze the impact of adversaries joining the multicast group (**JOIN**), when compared to the **NJOIN** case, we examine the cases where adversaries are randomly and strategically placed (Figures 10 and 11, respectively). Each figure also shows the ideal PDR (labeled **ideal**), which would be obtained if every one of the honest group members would receive all the packets sent by the source. The effectiveness of an attack should be interpreted as the *difference* between the **ideal** line and a protocol's PDR line. An attack is effective if this difference increases as the number of adversarial nodes is increased; on the other hand, a protocol resilient to attack appears as a line that remains parallel to the ideal line.

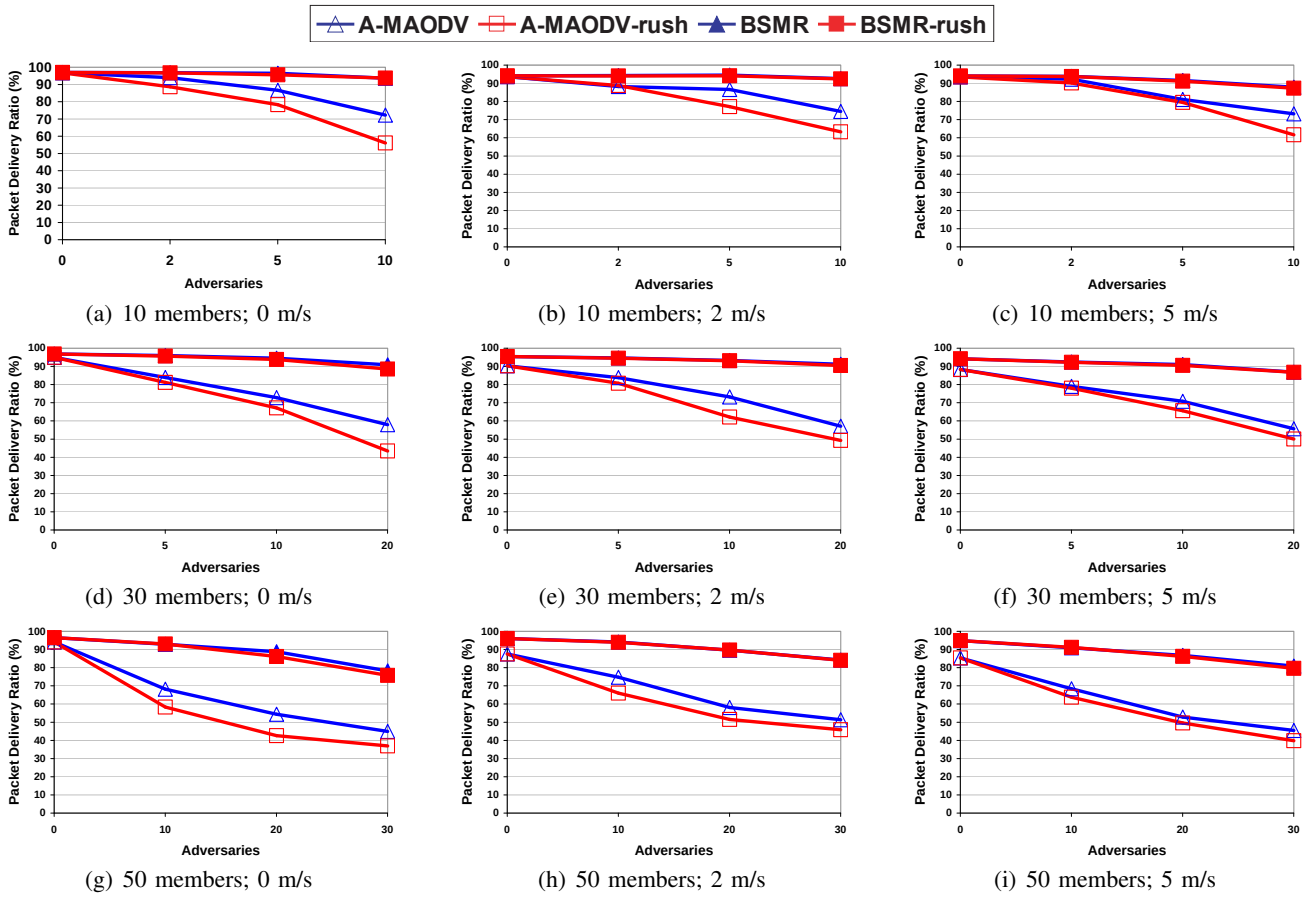
For random adversarial placement in Fig. 10, just like in the NJOIN case, the PDR decreases as the number of adversaries increases. However, we see a major difference from the NJOIN case: When the adversaries explicitly join the group before the honest nodes, the impact of flood rushing is minimal because the adversaries are already part of the group and rushing control packets does not provide any additional benefit. On the contrary, in this case flood rushing may actually improve the PDR because, by rushing control packets, adversaries may help legitimate nodes to find routes faster.

A drastic drop in the PDR is observed for A-MAODV when adversaries are placed strategically (see Fig. 11). Strategic positioning has a crippling effect on the performance of A-MAODV even more when adversaries explicitly join the multicast group. For both random and strategic adversarial placement, BSMR is barely affected by the attacks: The PDR line remains almost parallel to the ideal line, which shows little degradation occurs as the number of adversaries increases. The impact of the attacks on BSMR increases slightly when a large number of adversaries have joined the group, because there are less available honest nodes left in the network to serve as intermediaries for honest group members. We conclude that BSMR's strategy is effective in the **JOIN** case as well.

D. Attack Resilience: Wormhole Attack

Impact of Adversarial Placement. Figures 12 and 13 show the results for random and strategic adversarial placement, respectively. While in the experiments for the black hole attack the maximum number of adversaries was varied proportionally with the group size, for the wormhole attack a fixed number of adversaries was used, up to 12 adversaries (forming 6 wormholes) for random placement and up to 7 adversaries (forming 6 wormholes) for strategic placement.

For random placement, A-MAODV's performance decreases as the number of adversaries increases. The wormhole attack causes more damage per adversary than the black hole attack because less adversarial nodes are needed

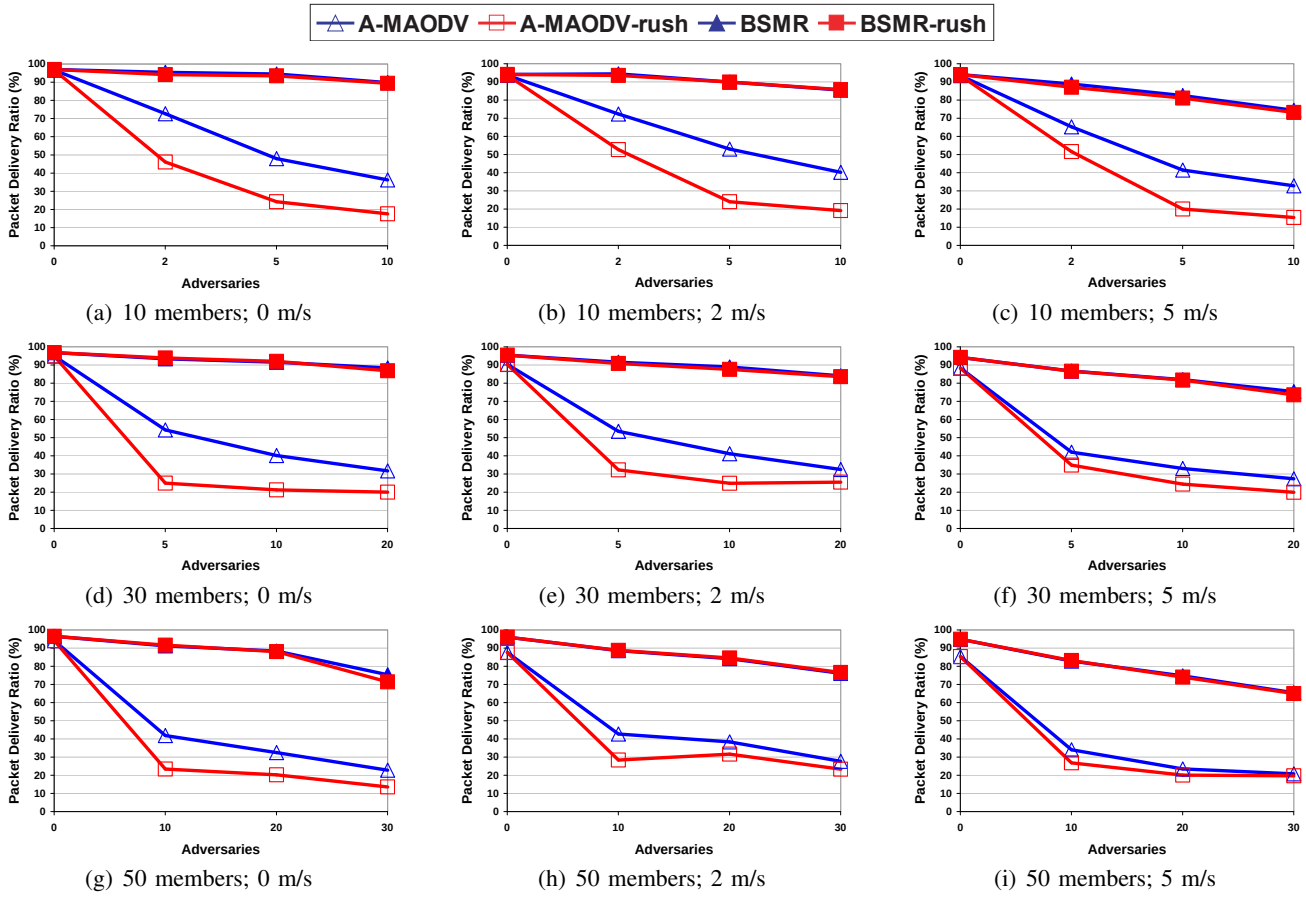


(a) 10 members; 0 m/s (b) 10 members; 2 m/s (c) 10 members; 5 m/s

(d) 30 members; 0 m/s (e) 30 members; 2 m/s (f) 30 members; 5 m/s

(g) 50 members; 0 m/s (h) 50 members; 2 m/s (i) 50 members; 5 m/s

Fig. 8: Black hole attack and flood rushing combined with black hole: **Random placement (NJOIN)**

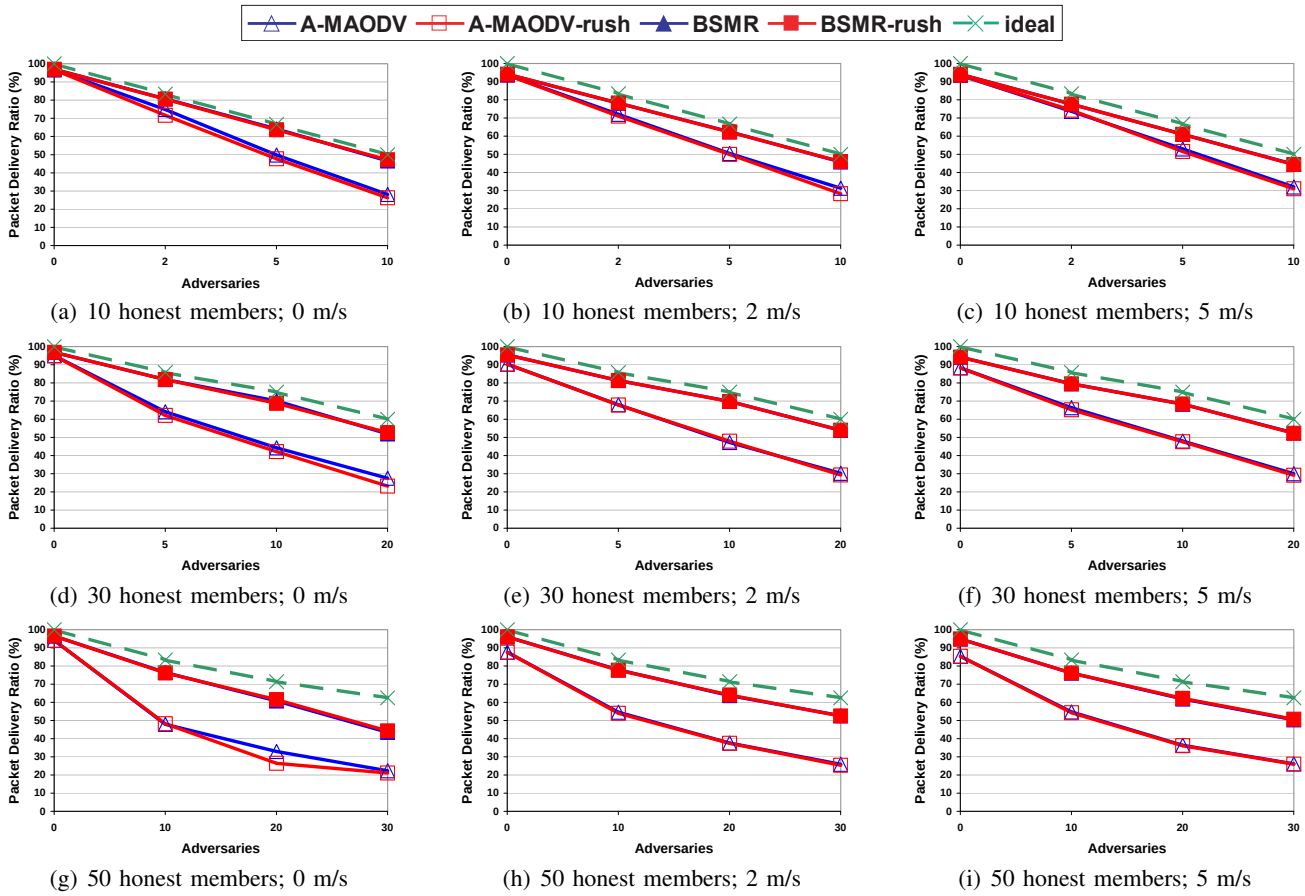


(a) 10 members; 0 m/s (b) 10 members; 2 m/s (c) 10 members; 5 m/s

(d) 30 members; 0 m/s (e) 30 members; 2 m/s (f) 30 members; 5 m/s

(g) 50 members; 0 m/s (h) 50 members; 2 m/s (i) 50 members; 5 m/s

Fig. 9: Black hole attack and flood rushing combined with black hole: **Strategic placement (NJOIN)**

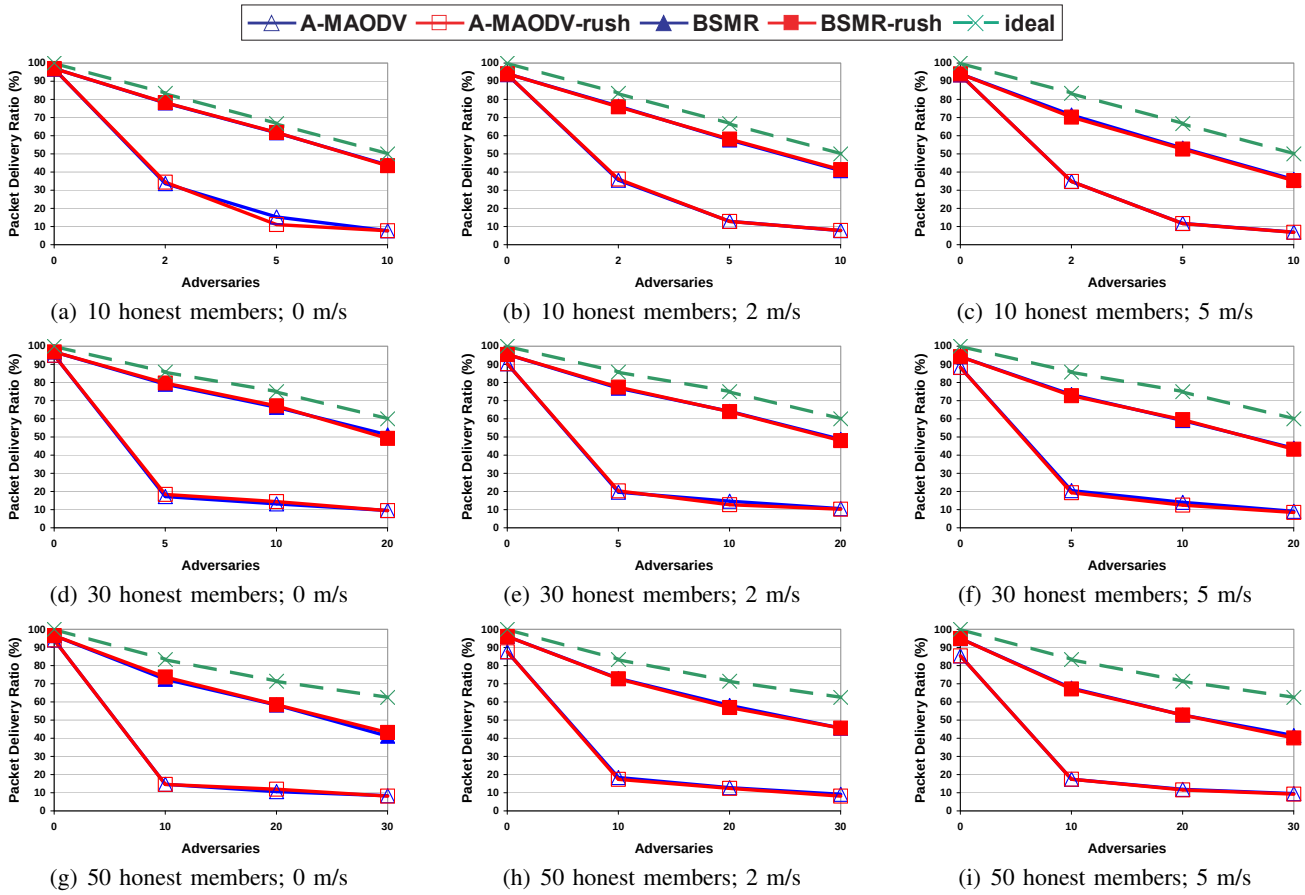


(a) 10 honest members; 0 m/s (b) 10 honest members; 2 m/s (c) 10 honest members; 5 m/s

(d) 30 honest members; 0 m/s (e) 30 honest members; 2 m/s (f) 30 honest members; 5 m/s

(g) 50 honest members; 0 m/s (h) 50 honest members; 2 m/s (i) 50 honest members; 5 m/s

Fig. 10: Black hole attack and flood rushing combined with black hole: **Random placement (JOIN)**



(a) 10 honest members; 0 m/s (b) 10 honest members; 2 m/s (c) 10 honest members; 5 m/s

(d) 30 honest members; 0 m/s (e) 30 honest members; 2 m/s (f) 30 honest members; 5 m/s

(g) 50 honest members; 0 m/s (h) 50 honest members; 2 m/s (i) 50 honest members; 5 m/s

Fig. 11: Black hole attack and flood rushing combined with black hole: **Strategic placement (JOIN)**

to decrease the delivery ratio by the same amount. As an example, for speed of 0 m/s and group size of 30 members, only 10 nodes forming wormholes can reduce the PDR to 60%; on the other hand, 20 adversaries executing the black hole attack are needed to reduce the PDR by the same amount. Flood rushing has a noticeable impact especially for small group sizes and for low mobility levels. Increasing the nodal speed reduces slightly the impact of the wormhole and flood rushing attacks: Mobility causes link breaks and honest nodes reattach themselves in different positions in the multicast tree, possibly connected to the source through non-adversarial paths.

The wormhole attack has a minor effect on BSMR and the addition of flood rushing has no effect. The PDR drops by less than 10% for all simulated scenarios, and remains above 90% for most simulated scenarios. This confirms the effectiveness of BSMR's strategy against the wormhole attack.

Fig. 13 shows the results when adversaries are strategically placed and form wormholes as described in Section VI-A. Because one end of each wormhole is near the source, the other end of the wormhole will present attractive, short routes to the nodes that are in its transmission range. This has a serious impact on the delivery ratio of A-MAODV: When 7 adversaries are present, forming 6 wormholes, they cover almost completely the entire simulation area; thus, every honest node that is normally more than three hops away from the source will be tricked into connecting to the multicast tree through one of the adversarial nodes. When flood rushing is employed, even nodes that are normally less than three hops away from the source are also tricked into connecting through adversarial nodes. For example, the PDR of A-MAODV drops as low as 23% (a drop of over 70%) when flood rushing is present for 0 m/s and group size is 30. We conclude that a small number of strategically placed adversaries can cause considerable damage for A-MAODV.

For BSMR, strategic placement results in a slim increase in the delivery ratio drop, when compared to the random placement case. For most simulated scenarios, the PDR drops by less than 10%. The solid performance of BSMR serves as proof for the robustness of its approach.

Impact of Explicit Join and Join Order. To analyze the impact of explicit join of adversaries to the multicast group (**JOIN**), when compared to the **NJOIN** case, we study the random and strategic placement of adversaries (Figures 14 and 15). Each figure also shows the ideal PDR (labeled **ideal**), which would be obtained if every one of the honest group members would receive all the packets sent by the source. Similarly with the black hole attack, the effectiveness of the wormhole attack should be interpreted as the *difference* between the **ideal** line and a protocol's PDR line.

In general, for both random and strategic adversarial placement, the effect of flood rushing on both A-MAODV and BSMR is unnoticeable. However, the reasons behind the immunity to flood rushing are different: BSMR prevents flood rushing attacks by using a reliability metric and by processing all flood duplicates, forwarding the ones with a better metric. As already explained in Sec. VI-C, A-MAODV's immunity to flood rushing is simply an artefact that the adversarial nodes join the multicast tree before the honest nodes do.

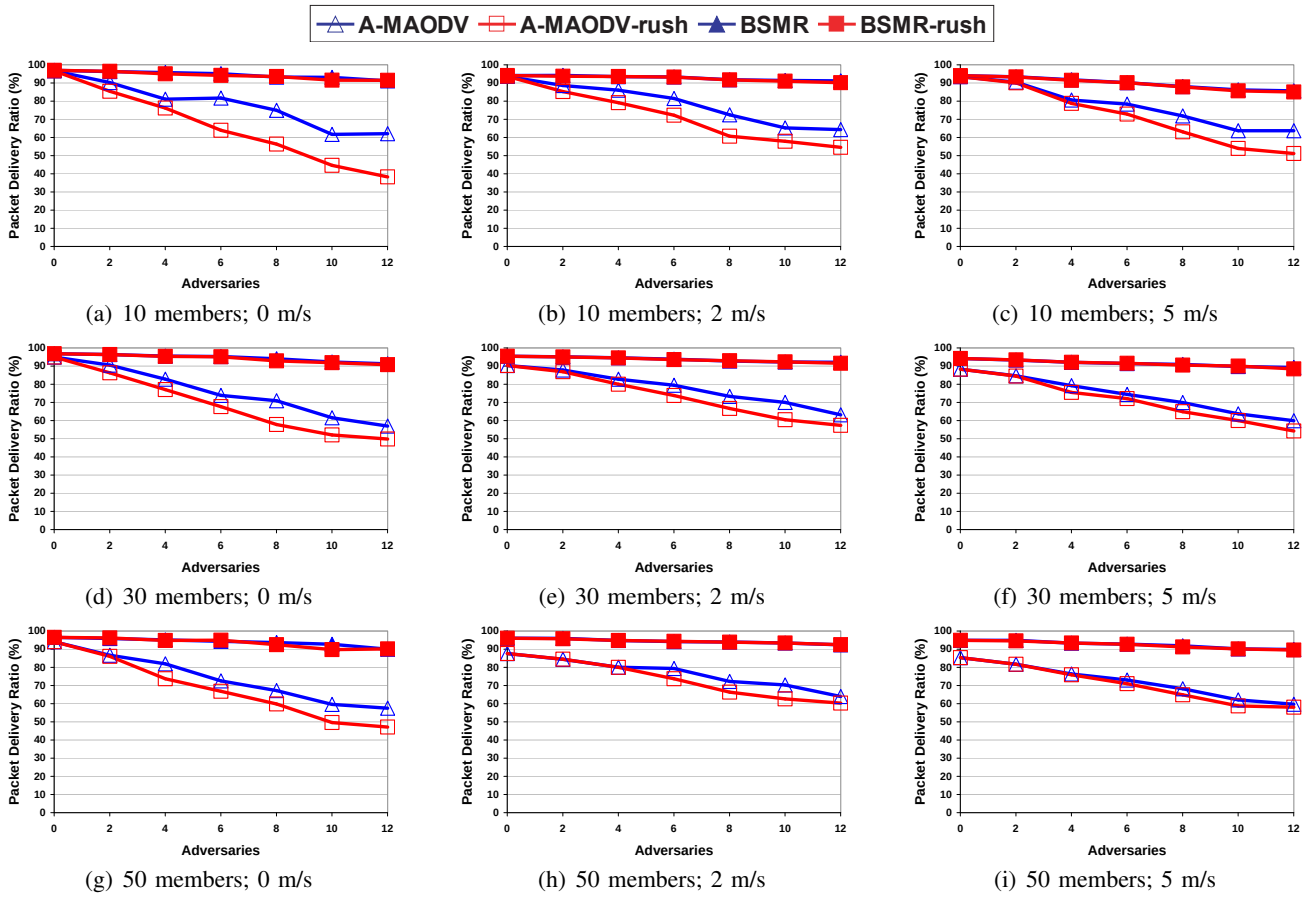
When adversaries are placed randomly (Fig. 14), the impact of the wormhole attack on A-MAODV is significant and increases as the number of wormholes increases. BSMR's delivery ratio line remains almost parallel with the ideal line, which means that little degradation occurs as more adversaries join the multicast group.

The devastating impact of the wormhole attack on A-MAODV becomes obvious when adversaries are strategically placed (Fig. 15). When 7 adversaries are present, the difference between the ideal line and A-MAODV's line increases by more than 60% at 0 m/s, for both group sizes of 30 and 50 (causing the PDR to drop to 15-20%). BSMR is affected slightly more (per adversary) than when adversaries are randomly placed, but in all simulated scenarios the difference between the ideal line and BSMR's line does not grow more than 8%, even in the presence of 7 adversaries. This validates the effectiveness of BSMR's strategy in the **JOIN** case a well.

E. Protocol Overhead

We first compare the overhead incurred by A-MAODV and BSMR in a non-adversarial scenario (Fig. 16(a)), for group sizes of 10 and 30, under different levels of mobility. BSMR has higher overhead because both route request and route reply are broadcast, and because of the extra MRATE packets broadcast periodically. BSMR's overhead due to double flooding in the route discovery phase becomes more noticeable especially at higher levels of mobility, when link breaks occur more often.

We then compare in Fig. 16(b) the overhead under adversarial conditions (the black hole attack with strategic adversarial placement). For the NJOIN case, BSMR's additional overhead compared to A-MAODV grows slowly

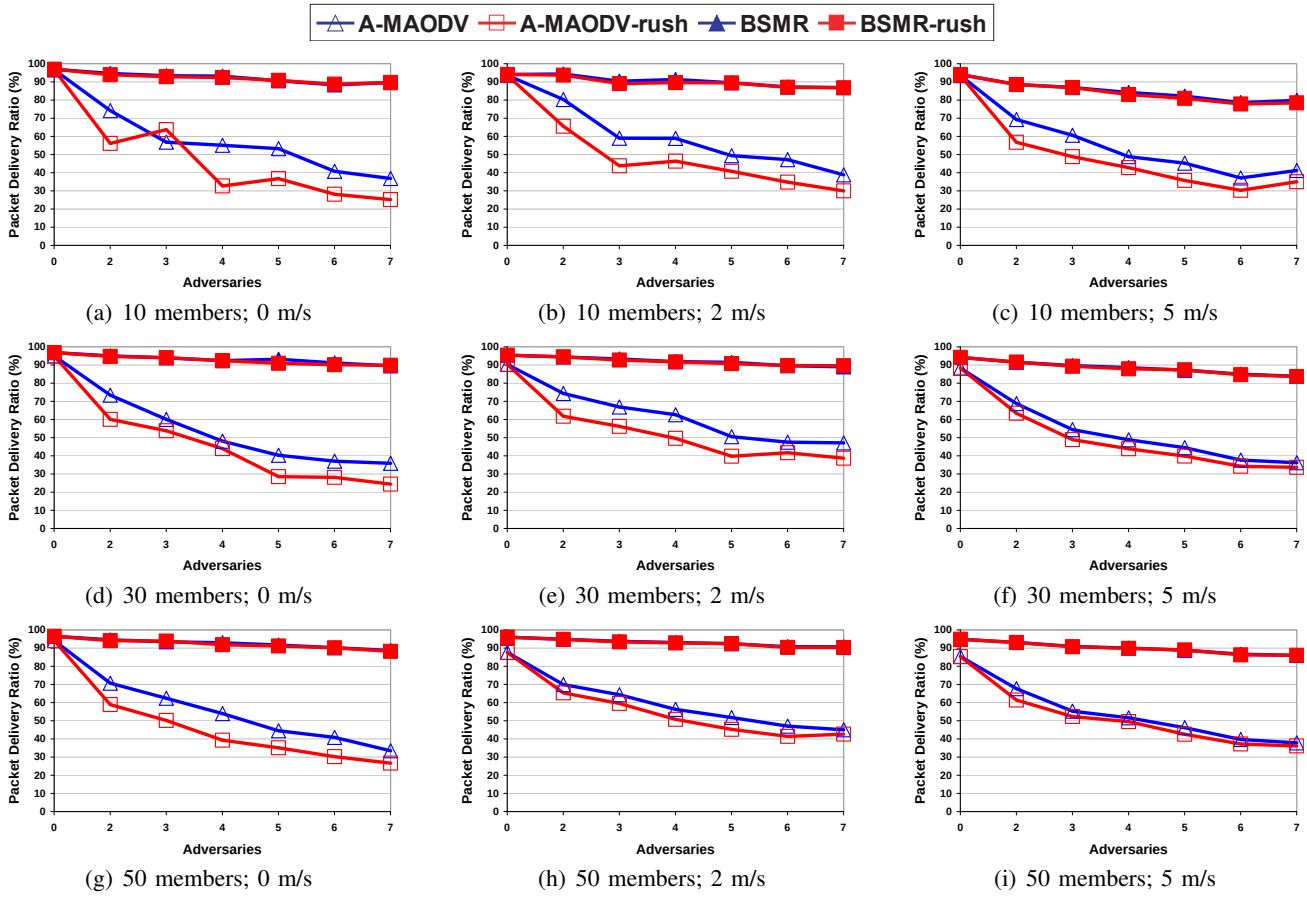


(a) 10 members; 0 m/s (b) 10 members; 2 m/s (c) 10 members; 5 m/s

(d) 30 members; 0 m/s (e) 30 members; 2 m/s (f) 30 members; 5 m/s

(g) 50 members; 0 m/s (h) 50 members; 2 m/s (i) 50 members; 5 m/s

Fig. 12: Wormhole attack and flood rushing combined with wormhole: **Random placement (NJOIN)**

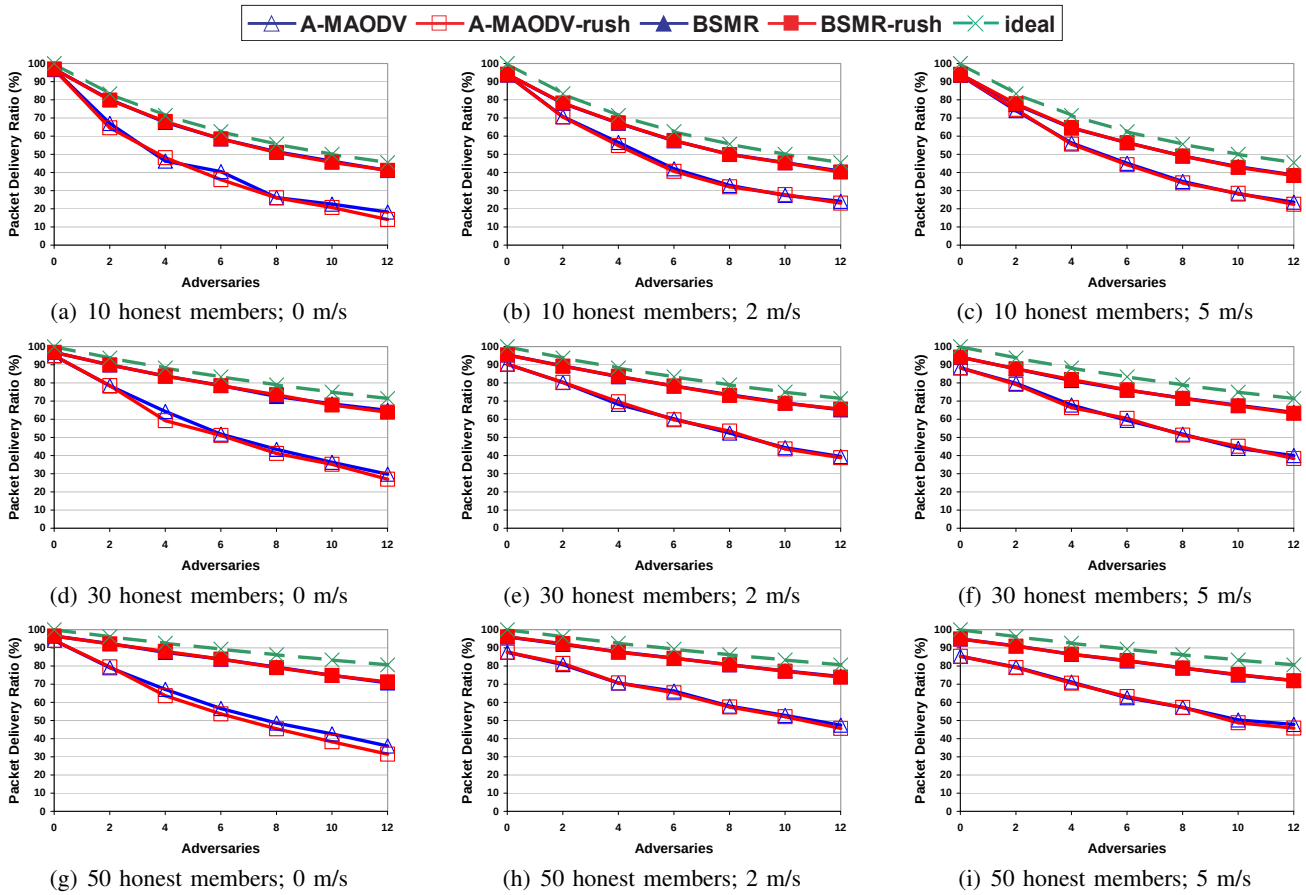


(a) 10 members; 0 m/s (b) 10 members; 2 m/s (c) 10 members; 5 m/s

(d) 30 members; 0 m/s (e) 30 members; 2 m/s (f) 30 members; 5 m/s

(g) 50 members; 0 m/s (h) 50 members; 2 m/s (i) 50 members; 5 m/s

Fig. 13: Wormhole attack and flood rushing combined with wormhole: **Strategic placement (NJOIN)**

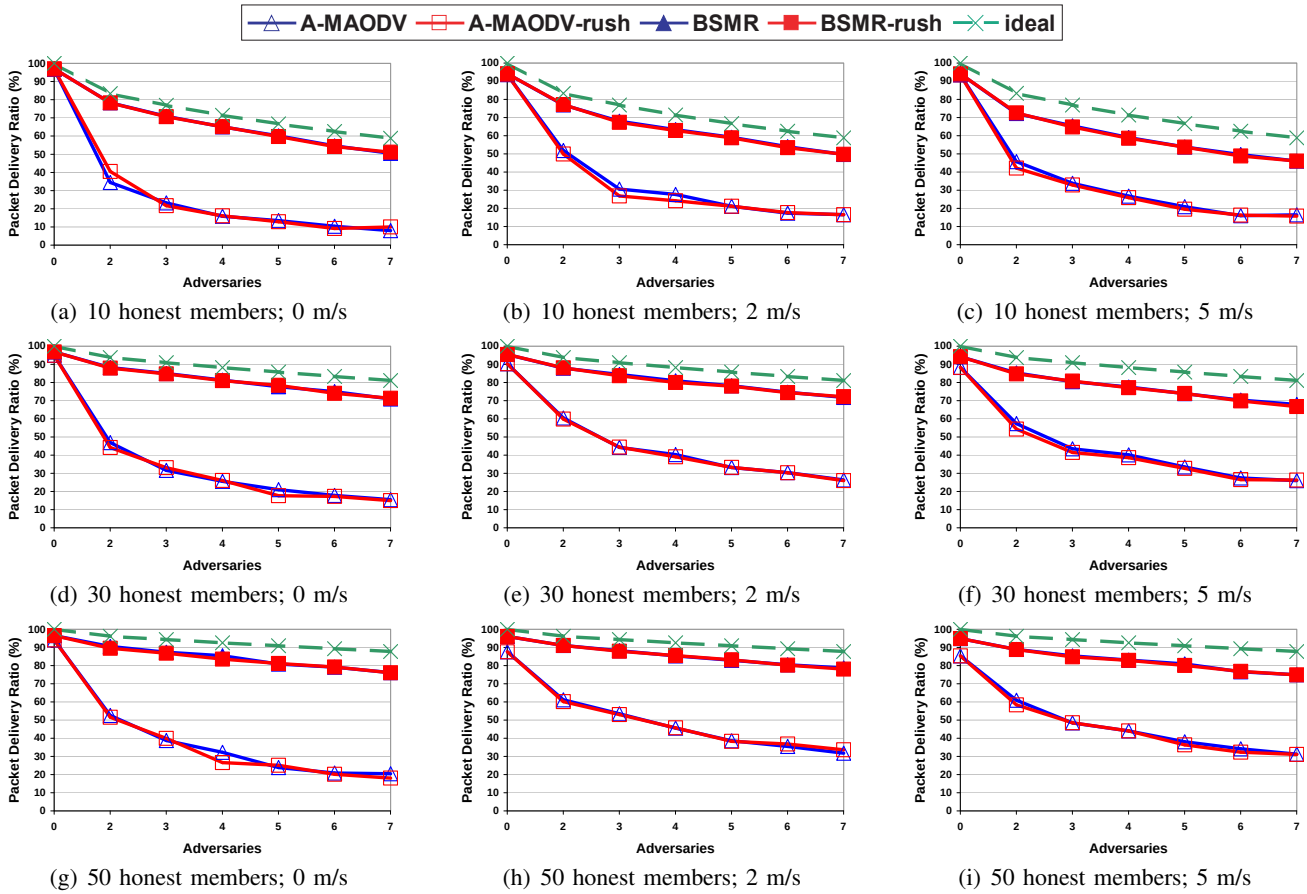


(a) 10 honest members; 0 m/s (b) 10 honest members; 2 m/s (c) 10 honest members; 5 m/s

(d) 30 honest members; 0 m/s (e) 30 honest members; 2 m/s (f) 30 honest members; 5 m/s

(g) 50 honest members; 0 m/s (h) 50 honest members; 2 m/s (i) 50 honest members; 5 m/s

Fig. 14: Wormhole attack and flood rushing combined with wormhole: **Random placement (JOIN)**

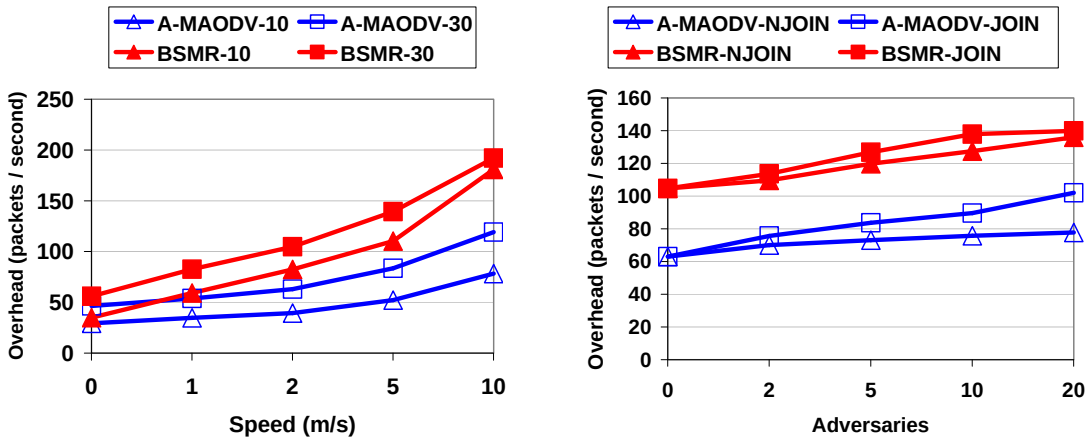


(a) 10 honest members; 0 m/s (b) 10 honest members; 2 m/s (c) 10 honest members; 5 m/s

(d) 30 honest members; 0 m/s (e) 30 honest members; 2 m/s (f) 30 honest members; 5 m/s

(g) 50 honest members; 0 m/s (h) 50 honest members; 2 m/s (i) 50 honest members; 5 m/s

Fig. 15: Wormhole attack and flood rushing combined with wormhole: **Strategic placement (JOIN)**



(a) Non-adversarial scenario (group size $\in \{10, 30\}$)

(b) Attack scenario (group size = 30, 2m/s, strategic placement)

Fig. 16: BSMR overhead

as the number of adversaries increases (from 40 pkt/sec for 0 adversaries to 55 pkt/sec for 20 adversaries). For the JOIN case, BSMR incurs little extra overhead over the non-adversarial case. This is not surprising: The bulk of the additional overhead is caused by the initial route discovery phase which leads to the creation of the multicast tree with avoidance of adversaries; afterwards, BSMR's additional overhead consists only of periodical MRATE packets and of occasional route discovery in case a link breaks due to mobility.

VII. Conclusion

In this paper we have discussed several aspects that make designing attack-resilient multicast routing protocols for multi-hop wireless networks more challenging than their unicast counterpart. A more complex trust model and underlying structure for the routing protocol make solutions tailored for unicast settings not applicable for multicast protocols. In the absence of adequate defense mechanisms, Byzantine attacks can prevent multicast protocols to achieve their design goals. As in the unicast setting, strategic adversarial positioning and flood rushing can increase the effectiveness of an attack. However, the multicast group membership status introduces an extra dimension which can alter the attack effectiveness: For example, the impact of flood rushing is minimized when adversaries join the group before the honest nodes.

We have proposed BSMR, a routing protocol which relies on novel general mechanisms to mitigate Byzantine attacks. BSMR identifies and avoids adversarial links based on a reliability metric associated with each link and capturing adversarial behavior. Our experimental results show that BSMR's strategy is effective against strong insider attacks such as wormholes, black holes and flood rushing.

Acknowledgements

The first author would like to thank Răzvan Musăloiu-E. for fruitful discussions in the early stages of this work. This work is supported by National Science Foundation CAREER Award No. 0545949. The views expressed in this research are not endorsed by the National Science Foundation.

References

[1] Y. B. Ko and N. H. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks," *Mob. Netw. Appl.*, vol. 7, no. 6, pp. 471–480, 2002.

[2] R. Chandra, V. Ramasubramanian, and K. Birman, "Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks," in *Proc. of ICDCS*, 2001.

[3] Y.-B. Ko and N. H. Vaidya, "GeoTORA: a protocol for geocasting in mobile ad hoc networks," in *Proc. of ICNP*. IEEE, 2000, p. 240.

[4] E. L. Madruga and J. J. Garcia-Luna-Aceves, "Scalable multicasting: the core-assisted mesh protocol," *Mob. Netw. Appl.*, vol. 6, no. 2, pp. 151–165, 2001.

[5] S. J. Lee, W. Su, and M. Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," *Mob. Netw. Appl.*, vol. 7, no. 6, pp. 441–453, 2002.

[6] E. Royer and C. Perkins, "Multicast ad-hoc on-demand distance vector (MAODV) routing," in *Internet Draft*, July 2000.

[7] J. G. Jetcheva and D. B. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks." in *Proc. of MobiHoc*, 2001, pp. 33–44.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," in *Advances in Ultra-Dependable Distributed Systems*. IEEE Computer Society Press, 1995.

[9] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proc. of CNDS*, January 2002, pp. 27–31.

[10] Y.-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," in *Proc. of WMCSA*, June 2002.

[11] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proc. of MOBICOM*, September 2002.

[12] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Proc. of ICNP*, November 2002.

[13] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. of MOBICOM*, August 2000.

[14] P. Papadimitratos and Z. Haas, "Secure data transmission in mobile ad hoc networks," in *Proc. of WiSe*, 2003, pp. 41–50.

[15] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proc. of WiSe'02*. ACM Press, 2002.

[16] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "On the survivability of routing protocols in ad hoc wireless networks," in *Proc. of SecureComm'05*. IEEE, 2005.

[17] S. Roy, V. G. Addada, S. Setia, and S. Jajodia, "Securing MAODV: Attacks and countermeasures," in *Proc. 2nd IEEE Int'l. Conf. SECON*. IEEE, 2005.

[18] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proc. of WiSe*, 2003.

[19] ——, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proc. of INFOCOM*, 2003.

[20] J. Eriksson, S. Krishnamurthy, and M. Faloutsos, "Truelink: A practical countermeasure to the wormhole attack in wireless networks," in *Proc. of ICNP'06*. IEEE, 2006.

[21] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," in *Proc. of NDSS*, 2004.

[22] D. Bruschi and E. Rosti, "Secure multicast in wireless networks of mobile hosts: protocols and issues," *Mobile Networks and Applications*, vol. 7, no. 6, pp. 503–511, 2002.

[23] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, "Secure multicast groups on ad hoc networks," in *Proc. of SASN'03*. ACM Press, 2003, pp. 94–102.

[24] S. Zhu, S. Setia, S. Xu, and S. Jajodia, "Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks," in *Proc. of Mobiquitos'04*. IEEE, 2004, pp. 42–51.

[25] L. Lazos and R. Poovendran, "Power proximity based key management for secure multicast in ad hoc networks," 2005, ACM Journal on Wireless Networks (WINET).

[26] R. Balachandran, B. Ramamurthy, X. Zou, and N. Vinodchandran, "CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks," in *Proc. of ICC 2005*, vol. 2, 2005, pp. 1123– 1127.

[27] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 102–113, 2003.

[28] V. Pappas, B. Zhang, A. Terzis, and L. Zhang, "Fault-tolerant data delivery for multicast overlay networks," in *Proc. of ICDCS '04*, 2004.

[29] L. Xie and S. Zhu, "Message dropping attacks in overlay networks: Attack detection and attacker identification," in *Proc. SecureComm '06*. IEEE and Create-NET, 2006.

[30] J. R. Douceur, "The Sybil attack," in *Proc. of IPTPS '01*, ser. LNCS, vol. 2429. Springer-Verlag, 2002, pp. 251–260.

[31] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: analysis & defenses," in *Proc. of IPSN '04*. New York, NY, USA: ACM Press, 2004, pp. 259–268.

[32] C. Piro, C. Shields, and B. N. Levine, "Detecting the Sybil attack in mobile ad hoc networks," in *Proc. SecureComm*, 2006.

[33] R. Curtmola and C. Nita-Rotaru, "Secure multicast routing in wireless networks," ACM Mobicom 2006 poster session, To be published in ACM MC^{2}R, 2006.

[34] R. Ghosh and G. Varghese, "Congestion control in multicast transport protocols," Tech. Rep., June 29 1998.

[35] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, "A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks," in *Proc. of IEEE WPMC'02*, 2002.

[36] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. of NDSS'01*, 2001.

[37] M. G. Zapata, "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing," IETF - Mobile Ad Hoc Networking Working Group, Internet Draft 06, September 2006, draft-guerrero-manet-saodv-06.

[38] "The network simulator - ns2," http://www.isi.edu/nsnam/ns/. [Online]. Available: http://www.isi.edu/nsnam/ns/

[39] Y. Zhu and T. Kunz, "MAODV implementation for NS-2.26," Carleton University, Technical Report SCE-04-01.

[40] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[41] *Digital Signature Standard (DSS)*. National Institute for Standards and Technology (NIST), 2006, no. FIPS 186-3, http://csrc.nist.gov/publications/drafts/fips_186-3/Draft-FIPS-186-3_March2006.pdf.

[42] *Advanced Encryption Standard (AES)*. National Institute for Standards and Technology (NIST), 2001, no. FIPS 197.

[43] *The Keyed-Hash Message Authentication Code (HMAC)*. NIST, 2002, no. FIPS 198.

[44] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *Proc. of INFOCOM '03*, 2003.