Journal of Information Processing Vol.26 98–110 (Feb. 2018)

[DOI: 10.2197/ipsjjip.26.98]

Regular Paper

Monitoring-Based Method for Securing Link State Routing against Byzantine Attacks in Wireless Networks

Babatunde Ojetunde[1,a)] Naoki Shibata[1,b)] Juntao Gao[1,c)]

Received: May 9, 2017, Accepted: November 7, 2017

Abstract: Secure communications is essential in many areas such as disaster management and battlefield communications. To detect and prevent attacks in such applications, most existing protocols adopt a cryptography-based approach, trust-based approach (reputation of nodes), or incentive-based approach. However, such protocols still have drawbacks, such as expensive overhead, difficulty in maintaining secure key and session management, or leaving routes unsecured against Byzantine attacks. In this paper, we introduce a monitoring-based method in the link state routing protocol to secure the packets' route against Byzantine attacks. The goal of our proposed scheme is to guarantee communication among connected benign nodes in the network. Specifically, each node monitors the action of neighboring nodes and compares the optimal packet route against the packet route history. Nodes in the network create a packet history field which is used to record all activities of an intermediate node when receiving and forwarding packets. Our scheme provides mutual monitoring in which nodes in the network can validate the packet history field of other nodes and report malicious activities. Also, our scheme uses a statistical method to know if a node is dropping packets intentionally by analyzing the packet dropping behavior of each node. The proposed scheme provides protection against colluding attacks and other Byzantine attacks. The proposed monitoring-based method achieves an average of 89% to 96% packet delivery ratio when 11% to 21% active malicious links are excluded from the network.

Keywords: secure routing, routing protocol, routing attack, monitoring, Byzantine attacks, link state routing

1. Introduction

The need to ensure routing security in wired and wireless networks has kept growing over the years with the development of various secure routing protocols. Secure routing allows easy packet transmission between nodes without any fear of compromise. However, routing protocols are still vulnerable to security challenges. One of the foremost challenges to routing protocol is Byzantine attacks. In such attacks, a node can interrupt route discovery, impersonate a destination node, corrupt routing information, completely drop packets, or inject fake packets into the network. These attacks prevent timely delivery of packets from the source to the destination. These types of attacks can be carried out by a malicious node either outside the network or within the network. Even though these types of attacks can be easily detected in a wired network, ad-hoc networks are still very vulnerable to such threats.

Most work already carried out on route security has adopted one of three main approaches: the cryptography-based approach, the trust-based approach, or the incentive-based approach [1]. In the cryptography-based approach, various cryptography mechanisms such as private and public key encryption schemes, digital signatures, hash functions, and/or end-to-end authentication are adapted to secure the packet in the routing protocols. The major drawbacks of this approach are the high computation overhead, and the difficulties of maintaining secure key management and session management. In the trust-based approach, nodes participating in the routing of packets are assumed to be trustworthy as the assigned trust value is used to determine each node's reputation, so the security mechanism provided focuses more on the information being exchanged among nodes. In this approach, lost packets are often attributed to poor link quality which may not be the case when malicious nodes may drop packets. In the incentive-based approach, nodes participating in routing are given some form of incentive to report malicious nodes. However, this approach is often combined with other approach such as a trust-based approach to be successful. In addition, tamper resistant hardware is added in this approach as a security measure and this is not generally applicable in all scenarios.

In our research, we proposed the monitoring approach to secure the link state routing protocol against Byzantine attacks [2]. This paper differs from our previous paper in which we only highlight the security goals our proposed monitoring approach achieves. In this paper, we give detailed explanations about the implementation of our monitoring-based method and show the results as validated by simulation. Our monitoring-based method secures the link state routing protocol against Byzantine attacks except Denial of Services (DoS) attacks. Here, a DoS attack is an attack where one or more malicious nodes transmit an overwhelming number of packets or jamming signal to clog some links. The goal of our proposed scheme is to guarantee communication among connected benign nodes in the network. Our ap-

[1] Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a)] ojetunde.babatunde.nq3@is.naist.jp
[b)] n-sibata@is.naist.jp
[c)] jtgao@is.naist.jp

proach focuses on using the link state routing protocol to analyze and record the actions of each node within the network. Specifically, each node monitors the actions of neighboring nodes and compare the optimal packet route against the route history. This allows monitoring nodes in the network to track the past events of packets sent. Our monitoring scheme adopts three main methods:

(1) Hello message verification - where a node collects hello messages and digital signatures of neighboring nodes, which are used to verify the validity of hello messages and to identify inconsistent information when a malicious node tries to corrupt routing table information.
(2) Packet history field monitoring - here the source node calculates the optimal path and stores it in each packet (like Dynamic Source Routing), and then neighboring nodes check whether packets are forwarded correctly according to the stored optimal path. Also, the event history is recorded in each packet at each intermediate node.
(3) Statistical hypothesis testing - while some packets may be dropped due to poor link quality, we need to know if a node is intentionally dropping packets. To determine this, we adopt a statistical measure in which monitoring nodes observe the packet-dropping behavior of other nodes, and then calculate the probability (*P-value*) of an intermediate node dropping a packet.

To detect malicious nodes, the *P-value* is compared to a significance level value (reflecting the number of dropped packets that can be tolerated), while packet history field monitoring is used to identify at which node a malicious action is carried out.

The rest of this paper is organized as follows. Section 2 reviews related work on securing routing protocols. In Section 3, we present an overview of routing protocols and Byzantine attacks. Then in Section 4, we introduce our proposed monitoring scheme to secure the link state routing protocol against Byzantine attacks, and in Section 5 describe our evaluation of the proposed scheme. Section 6 concludes the entire paper.

2. Related Work

In this section, we review previous work on securing routing protocols and Byzantine attacks.

Geetha et al. [1] classified routing protocols into three distinct types: proactive, reactive, and hybrid protocols. They described proactive protocols as protocols where nodes frequently exchange network topology information and construct routing tables to send packets from the source to the destination. Examples of such protocols include the Optimized Link State Routing protocol, and the Destination-Sequenced Distance-Vector protocol. Reactive protocols are described as protocols that ensure packets are sent from the source to the destination only when needed. Ad-hoc On-Demand Vector (AODV) and Dynamic Source Routing are examples of reactive protocols. Finally, hybrid protocols are produced by combining both proactive and reactive protocols. For example, route discovery makes use of a proactive protocol scheme while a reactive protocol scheme is adopted for sending packets. The Zone Routing Protocol and Fisheye State Routing are both examples of hybrid protocols.

Harshavardhan [3] surveyed security issues in ad-hoc routing protocols and identified ways to mitigate such security threats. Harshavardhan first defined the properties of an ad-hoc routing protocol as providing distributed operation, loop free, demand-based operation, unidirectional link support, security, quality-of-service support, multiple routes, and power conservation. Then they used findings from related work to summarize different ad-hoc routing protocols before analyzing various security threats and techniques to mitigate them. Some of the security threats they included were: impersonation or spoofing, black-hole attack, sinkhole attack, and wormhole attack. They classified solutions to these attacks into categories including: trust values, wormhole detection method, intrusion detection systems, credibility management and routing test, and multi-factor authentication techniques.

Ali et al. [4] also surveyed security challenges in mobile ad-hoc networks (MANETs). They introduced three important security parameters, and further divided security aspects into two areas, which are security services and attacks. They classified security services into five important services which are used to protect the network before attacks happen, while attacks are the threats to the network. In addition, they analyzed and discussed various mitigating approaches against attacks in MANETs. Mojtaba et al. [5] also investigated routing attacks and various solutions to such attacks. They highlighted security attacks that MANET routing protocols are vulnerable to and identified mechanisms such as cryptography schemes, key management, and special hardware using GPS as some possible solutions to such attacks. Similarly, Kannhavong et al. [6] surveyed routing attacks in MANETs. They investigated various security issues in MANETs and examined routing attacks, such as flooding, black holes, wormholes, replays, link spoofing, and colluding attacks, as well as solutions to such attacks in MANETs. They identified the advantages and drawbacks of the reviewed solutions, then recommended improvement of the effectiveness of the security schemes they had surveyed.

Jhaveri et al. [7] surveyed various DoS attacks that are security concerns in MANETs and some of the proposed solutions to identify and prevent such attacks. They describe various routing protocols, and DoS attacks such as a wormhole, black hole, gray hole attacks and their operations. Zapata et al. [8] introduced a security mechanism to secure AODV routing information. First, they identified integrity, authentication, confidentiality, and non-repudiation as security goals for routing. Then they proposed two mechanisms to secure AODV packets, hash chains and digital signatures. Specifically, the hash chain is used to verify that the hop count was not decreased by a malicious node, while the digital signature is used to safeguard the integrity of other information in the packets besides the hop count.

Alajeely et al. [9] proposed a new detection scheme for malicious nodes to detect packet faking by a malicious node. In this type of attack, malicious nodes drop one or more packets and inject another packet to replace the dropped packet. They introduced a hash chain technique to detect the attack and trace the malicious nodes. They compared their approach to an acknowledgment-based mechanism and a network coding based mechanism. Baadache et al. [10] proposed a scheme to check if

packets are routed correctly in the network. They adopt the acknowledgement of packets at each intermediate node which is used to construct a Merkle tree. Packet dropping is detected if the root of the Merkle tree is not the same with a precalculated value.

Papadimitratos et al. [11] proposed a secure link state protocol (SLSP) for MANETs to secure neighbor discovery and adopted a neighbor lookup protocol to further strengthen their system against DoS attacks. In addition, the proposed SLSP restricted the forwarding of packets within a cluster, and adopted the use of public and private keys to validate that the packets are only forwarded within the cluster. Unlike our proposed monitoring scheme, their protocol only focused on securing the topology discovery and protected the link state update packets, but did not secure the routing of packets. Our proposed scheme addresses routing security using a monitoring mechanism to protect packets and also guarantees the communication of benign nodes. Another main difference found in our work is that our proposed scheme secures the routing protocol against colluding attacks where a group of nodes collaborates to carry out an attack.

To secure the packet route and provide secure message transmission in MANETs, Papadimitratos et al. [12] proposed a different mechanism from their previous work. Their mechanism is based on four main schemes: secure end-to-end transmission of packets and feedback, dispersion of a packet, multi-path routing of packets, and adaptation to topology changes. In their protocol, the source node will first select several disjointed paths that are valid, referred to as an active path set (APS). Then the node splits the packet into a number of pieces, which are transmitted simultaneously across the selected APS. After receiving a sufficient number of pieces of the divided packet, the destination node will then reconstruct the packet, even when some fraction of the pieces are dropped or invalid. Whenever a piece of the packet is not received by the destination, that route is considered broken or compromised. In addition, their mechanism also introduced path rating based on feedback from the destination node. Paths that fall below a given threshold are discarded from the network. Their secure protocol focused on detecting unsecured routes, unlike our approach in which the actual malicious nodes in a selected route are detected and discarded to prevent further relaying of packets.

Although some of the proposed schemes successfully mitigate routing attacks, they are either too expensive for resource-constrained networks or the solution provided is not applicable to mitigate colluding attacks from malicious nodes. Also, it is possible for malicious nodes to drop packets and attribute the cause to poor communication links. Therefore, we propose a mechanism to analyze the action of all nodes in the network. Specifically, our scheme focuses on mitigating Byzantine attacks in link state routing protocols.

3. Overview of Byzantine Attacks

In this section, we describe Byzantine attacks.

3.1 Byzantine Attacks

Byzantine attacks can be described as attacks in which malicious nodes take control of one or more network nodes and disrupt the network functions [1]. Malicious nodes can selectively drop packets, corrupt routing information, or send packets on non-optimal paths. When carried out by a fully authenticated node in the network, these types of attacks are difficult to detect. Some of the Byzantine attacks are described below.

3.1.1 Corruption of Routing Table Attacks

In these attacks, the goal of a malicious node is to corrupt the routing table, either by falsifying neighbor information, or by capturing and modifying the neighbors' link information broadcast by a benign node. Doing this can cause the routing protocols to maintain the wrong information in the routing tables, which now include the malicious nodes in almost all routes to destinations. **Figure 1** (a) shows an example of corruption of routing table attack.

3.1.2 Falsifying Location Information Attacks

In this attack, a malicious node forges a position in the network which is completely different from its actual position, and reports the forged position to other nodes in the network. This causes benign nodes in the network to calculate the wrong status and cost of the malicious node's links, which leads to invalid information in the routing table and packet loss. Figure 1 (b) shows an example of a falsifying location information attack.

3.1.3 Black Hole Attacks

In this form of attack, a malicious node injects fake routing information to attract all packets to itself, and then either drops all of the packets, modifies some packets, or selectively drops packets. To avoid detection, such malicious nodes sometimes actively participate in routing packets to the destination in a normal way. This makes it difficult for other nodes in the network to detect such malicious node action. Figure 1 (c) shows an example of a black hole attack.

3.1.4 Sink Hole Attacks

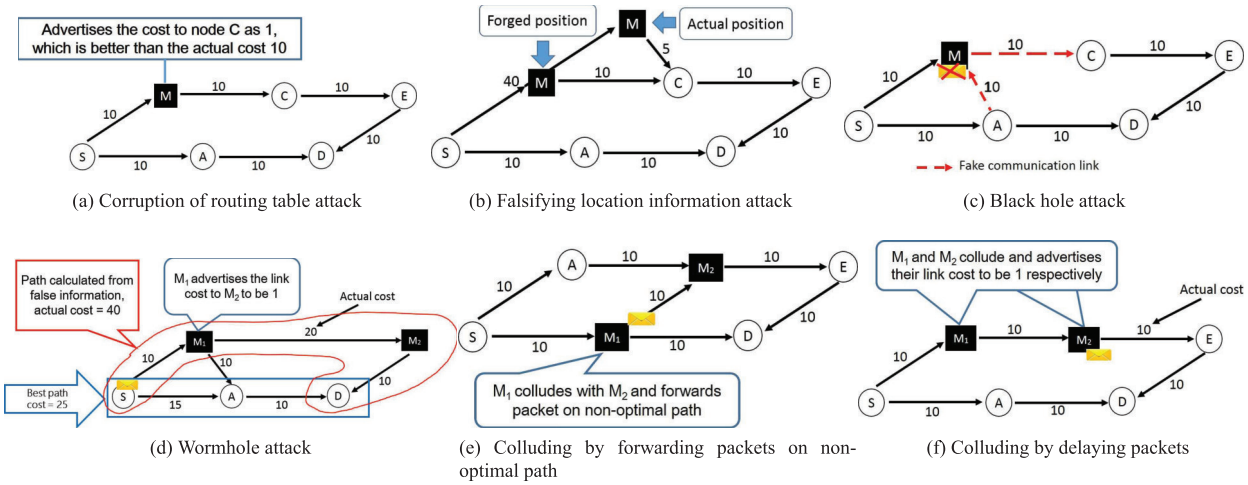
Similar to the black hole attack is a sink hole attack, in this attack a malicious node attracts all packets to itself by claiming to have shortest path to all destinations in the network. Other intermediate nodes then relay their packets through the malicious node. The malicious node can then either modify, fabricate, or eavesdrop on the packets.

3.1.5 Wormhole Attacks

In this form of attack, a malicious node advertises an artificial route as the best path to the destination node, and tunnels the packets to another malicious node, thereby causing the source node to ignore the genuine route. Such malicious nodes can either drop all packets, or selectively drop packets, preventing timely delivery of packets and causing packet loss in the network. This is also a form of colluding attack. Figure 1 (d) illustrates an example of a wormhole attack.

3.1.6 Colluding Attacks

In a colluding attack [13], a group of nodes collaborates to carry out an attack by dropping or modifying packets. One of the nodes will advertise itself as having the shortest path to the destination. The shortest path may or may not include other collaborating nodes to complete the attack. This form of attack is hard to detect, especially when the nodes align with each other as neighbors. For example, Fig. 1 illustrates two example scenarios for colluding attacks. As shown in Fig. 1 (e), the first colluding scenario is when a malicious node M_1 is part of the selected



(a) Corruption of routing table attack (b) Falsifying location information attack (c) Black hole attack

(d) Wormhole attack (e) Colluding by forwarding packets on non-optimal path (f) Colluding by delaying packets

Fig. 1 Various types of Byzantine attacks.

packet route, but decides to forward the packets to another colluding malicious node M_2 on a non-optimal path.

The second colluding scenario, shown in Fig. 1 (f), is when the source node S decides to send a packet to destination D. The best shortest path is S - A - D. However, since node M_1 is malicious and colluding with another malicious node M_2, both malicious nodes advertise the wrong link costs, e.g., 1 and 2 respectively, so the best route appears to be S - M_1 - M_2 - E - D. Then the malicious nodes M_1 and M_2 forward the packet from the source node S at the actual link cost, which causes packet delays to node E. The colluding malicious nodes can also drop packets.

These types of Byzantine attacks are difficult to detect or prevent, especially when carried out by an insider attacker. Therefore, as described in the next section, we adopt a monitoring scheme to secure routing in the LSR protocol.

4. Proposed Secure Routing Protocol with a Monitoring Scheme

In this section, we first describe the link state routing protocols and its features. Then we explain our secure routing protocol and monitoring scheme designed to protect a network against Byzantine attacks. We explain how a valid routing table is formed, then we describe the statistical method used to detect malicious nodes and the monitoring scheme used to secure the LSR protocols. Finally, we explain how our proposed scheme mitigates Byzantine attacks.

4.1 Link State Routing Protocols (LSR)

Link state routing (LSR) protocols [14] are proactive protocols in which a node exchanges Hello messages with other surrounding nodes to know the entire network information. Based on the information acquired from other nodes, the node first creates a topology of the network and positions itself at the root of the spanning tree, then uses a Shortest Path First algorithm, such as Dijkstra's Algorithm, to find the best path to a destination. Examples of LSR protocols are open shortest path first (OSPF) and intermediate system to intermediate system (IS-IS).

Each node needs to discover neighboring nodes. In order to do this, a node sends a Hello message periodically (e.g., every 10 seconds). The Hello message contains the node's unique ID. By receiving Hello messages from other nodes, a node can determine which nodes it is directly connected to. The Hello messages are only sent to directly connected neighbors, for example, as shown in **Fig. 2**. Hello message from node S is only sent to node A, while Hello message from node A is sent to both nodes (node S and node D) which are its direct neighbors. Also, Hello messages are used to detect link failures and node availability in the network. A link failure is detected if a Hello message is not received from a particular neighbor within a time interval (e.g., 30 seconds). The time interval in which a node is not able to send/receive a Hello message to/from its neighbor is also called a dead interval.

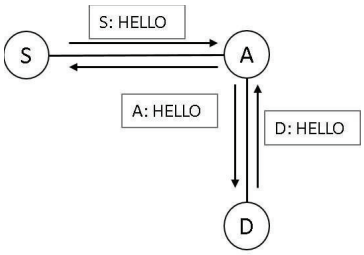


Fig. 2 Exchange of Hello messages by nodes.

After a node has learned all the information about the network topology, the node can now distribute its local link information (i.e., its view of the network) to other nodes in the network. This will assist other nodes in forming their own view of the whole network. To distribute its local view, a node needs to build a link-state packet (LSP) which contains: node's ID, LSP age, LSP sequence number, LSP links (i.e., links advertised by another neighbor node which include the neighbor's ID and link cost). A flooding algorithm is adopted for the distribution of the LSP to all nodes in the network. Also, each node maintains a link-state database (LSDB) to store the recent LSP received from other nodes. Once a new LSP is received, each node verifies it against the one that is stored in the LSDB. If the LSP did not match any of the LSP stored in the LSDB, then the node will forward it to all nodes except for the node it receives it from. **Figure 3** highlights an example of LSP flooding.

In addition, acknowledgments and retransmission are used to

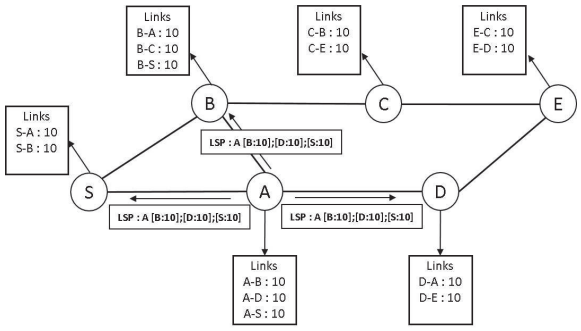


Fig. 3 An example of link-state packet flooding.

ensure that all LSPs are received by all nodes in a situation where there is a link failure. It is possible that a link failure is detected by one of the nodes directly connected to each other. In that case a new LSP is generated and the failed link is removed from the LSP. The new LSP is then flooded to all nodes in the network. Each node can then replace the previous LSP with the new one. After receiving all LSPs in the network, each node can compute the complete network topology. Then, using Dijkstra shortest path algorithm, each node computes a spanning tree and positions itself as the root of the tree. The routing table is automatically formed from the spanning tree. Packet from a particular node in the network is routed to the specified destination based on its routing table. Each node consumes energy while broadcasting and receiving Hello messages from neighbor nodes. Also, as the number of transmitting packet increases, the power consumption of a node will also increase.

4.2 Assumptions and Preliminaries

In this paper, we make the following assumptions.

- All benign nodes are connected in the network topology. i.e., there is always a route only consisting of benign nodes between any pair of benign nodes in the network.
- Each node in the network maintains low mobility.
- All nodes can generate pairs of public and private keys.
- A key pair is kept secret by a benign node.
- A benign node only generates one pair of public/private keys.
- Links are not stable, i.e., not all packets are received by neighboring nodes.
- All benign nodes know the link states of all neighboring nodes.
- Due to wireless channel fading during transmission between two nodes, a packet may be dropped with probability q. We assume a benign node can estimate the probability q of packet dropping between itself and a neighbor node.
- All packets are forwarded in First-In-First-Out order.
- There is time synchronization between benign nodes.
- Each node knows the upper limit of the time synchronization error.

Any node can join the network without pre-registration. We adopt the use of symmetric and asymmetric keys for the encryption/decryption of packets. The asymmetric key scheme is used to generate/verify a digital signature, while a session key, generated using symmetric key algorithms, is used to encrypt/decrypt the data of a packet. A unique key pair can be safely created from random numbers by any node. The public and private keys are unique to each benign node and the private key is kept secret by each node. Benign nodes create and exchange public keys beforehand. In addition, nodes authenticate each other with a digital signature. A node will sign its signature on the ID which can be verified by other nodes. Our method adopts the digital signature algorithm (DSA) described in Ref. [15].

To generate a digital signature, a node applies a one-way secure hash function to a message (e.g. ID or packet), then encrypts the hash value with its private key to form a signature (i.e., the encrypted hashed ID is the signature of the node). The calculation of encrypting and decrypting the hash value of a message by a public key cipher is faster than the calculation of encrypting and decrypting the message directly by a public key cipher. Other nodes can verify a node signature by first decrypting the signature using the signing node's public key to reveal the hash values (i.e., hashed message), then applying the same one-way secure hash function to the signing node's original message to generate a new hash value. Finally, the verifying node compares the two hash values to validate if it matches. The signature is valid if the two hash values are the same.

Moreover, to encrypt/decrypt a packet, a source node creates a session key, then encrypts the data of the packet with the session key. The session key is encrypted with the destination node's public key using the asymmetric key algorithm such as RSA. The encrypted packet and the encrypted session key are sent through the selected route to the destination node. After receiving the packet, the destination node decrypts the encrypted session key with its private key, then decrypts the data with the session key to reveal the data sent from the source node.

4.3 Routing Table Formation

A malicious node might possibly corrupt routing table information by sending inconsistent information to other nodes in the network. To prevent this and ensure that each node can verify the validity of a Hello message, in our proposed method we make a slight modification to the Hello message of standard LSR protocol by introducing security parameters in the Hello message as shown in **Fig. 4**.

In addition to the information in the standard LSR hello message, the hello message of our protocol includes the node's ID, digital signature, number of packets dropped, number of packets sent, number of packets received, number of packets forwarded, a timestamp, and a list of neighbors. Also, each node appends to their own hello message information in the collected hello messages from their neighbors which includes the neighbor's ID, neighbor's link cost, timestamp, and the neighbor's digital signature. Figure 4 shows an example of typical information in the hello message of an LSR protocol, which is 48 bytes when a node is connected to one neighbor (with a 24-bytes header and a 24-bytes hello message) and the information in our protocol which we specifically introduced to achieve routing security with additional information of 272 bytes when RSA signature is adopted. The size of the hello message of a node varies depending on the number of neighbors.

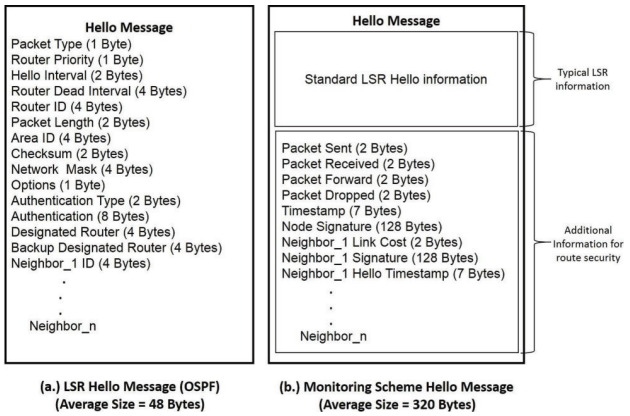


Fig. 4 An example of Hello message.

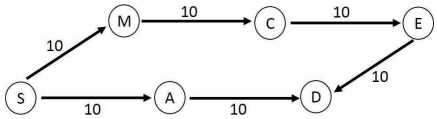


Fig. 5 Packet route in a network.

Each node floods the link state information of its neighbor to other nodes in the network. As part of this flooding, we use acknowledgment and retransmission because links are not reliable. Each node maintains its routing table using the neighbor information in the hello message. Since benign nodes are connected, all neighbor information of benign nodes reaches all benign nodes. For each link, the quality of that link is reported twice from two nodes. If the information from two nodes is different, we adopt the worse one. After a node collects all topology information, each benign node calculates the best logical path to every possible destination with the information collected from the hello messages. Then it uses the best paths to each destination to form its routing table.

After neighbor nodes receive a hello message, each neighbor node responds to the hello message by sending an acknowledgment to confirm receiving the hello message. Within the replies, each neighbor node identifies itself with its node ID and digital signature. The node that initiates a hello message can use the information from the neighbors to confirm that the hello messages were received. Also, when a node receives a new hello message from its neighbor, after authenticating the neighboring node with its signature and node's ID, the node will then check the timestamp to confirm that an old hello message has not been replayed.

4.4 Monitoring Scheme for LSR Protocol

In a LSR protocol, to send a packet from a source to a destination, the routing protocol finds the shortest path to the destination using the information in the source node's routing table. However, a malicious node that is included in the route to the destination may attack the route. To prevent such attacks, we introduce a statistical method and a mutual monitoring scheme.

4.4.1 Overview of Proposed Scheme

Let's consider a situation such as that in **Fig. 5**, in which node S is sending a packet to destination *D* with *S* - *A* - *D* as the shortest path to the destination.

In our method, we only ensure communication among benign nodes. The first thing we should avoid is forwarding packets along a wrong route, or dropped by malicious nodes. In order to prevent this, first, surrounding nodes compare the optimal packet route against the route history. The optimal path is calculated at the source node and stored in each packet in our protocol. Previous nodes and other neighbor nodes in the network overhear when the packet is forwarded by each node. Then, the nodes checks if the packet is forwarded on a wrong route or dropped. The monitoring node checks the packet history to verify if it is correctly signed by the forwarding node, the packet signing and verification uses the same process explained in Section 4.2.

If the node fails to correctly sign the packet, the results of the monitoring are reported to other nodes in the network. Recording the packet route history of packets allows other nodes to track the past events of packets sent. In order to confirm that the packet is delivered to the destination, the destination node sends an acknowledgement packet through the reverse route. If the source node does not receive this acknowledgement packet after a certain amount of time elapses, the source node asks the nodes along the route to show the signature from the next node.

In addition, it is possible that some packets may be dropped due to poor link quality. A malicious node may also drop the packet, and state poor link quality as the reason for the packet loss, as a result of this we need to know if a node is intentionally dropping packets. Therefore, we use a statistical method explained in the next section to determine if a node is intentionally dropping packets.

When some node reports another node to be malicious, we need to handle the cases where a malicious node is reporting a benign node to be malicious. Our goal, that is to maintain communication among benign nodes, can be achieved by separating malicious nodes from benign nodes. When some node reports one of its neighboring node to be malicious, we can be sure that at least one of them is malicious. Thus, we separate those two. In our protocol, the link between two nodes is advertised to the whole network, and it will not be used in the future.

In a situation when a malicious node decides to keep rejoining the network with a new address after being excluded from the network, then such malicious node is not immediately included in the routing of packets. We wait for some time after a new node joins the network, during this period this node is not used as a part of a route. In addition, a malicious node might intentionally delay packets, expecting that the packet delay would be hidden by delays due to transmission conflicts with other nodes. Also, if one of the neighboring nodes is communicating with other nodes, that node cannot start sending out packets. This cannot be observed by other nodes because of the hidden/exposed terminal problem. We introduce a signed Request to Send/Clear to Send (RTS/CTS) mechanism (explained in Section 4.4.4) to detect if a node intentionally delays packet forwarding and to solve the hidden/exposed terminal problem in this case. Before sending a data packet, each node first sends an RTS packet to the next hop node and only transmits the data packet after a CTS packet has been received from the next hop node. Other nodes, overhearing the RTS/CTS, refrain from sending any packets to the node until an acknowledgment packet is overheard. Then a node that is sus-

pected of intentionally delaying packets can show the RTS/CTS packets as a proof that there is no packet delay.

Since our protocol allows any node to create a pair of keys, a malicious node can pretend there are many nodes around it. Even some of the links are advertised to be invalid, there are still many links usable for malicious node. In order to handle cases like this, a node retransmits its packet using a 2-hop reactive mode. Using this reactive mode, a node will create a new packet history field indicating that the packet is being retransmitted with a reactive mode scheme and broadcast its packet to 2-hop neighbors. On receiving the packet, any node that is neighbor to both the source and the 2-hop destination node can forward the packet to the destination node. If a malicious node is trying not to forward the packet by pretending there are many nodes around it, all these links can be invalidated at the same time. The 2-hop reactive mode is only used when a packet has been dropped and the malicious node has been reported to other benign nodes in the network.

4.4.2 Monitoring Packet Dropping

A monitoring node observes the packet dropping behavior of a monitored node and adopts the approach of statistical hypothesis testing to determine if the monitored node is a malicious node.

The statistical hypothesis testing approach: First, the monitoring node makes a hypothesis H_0 that the node being monitored is a benign node and sets the value of significance level α (as a common practice $\alpha = 5\%$). Second, the monitoring node observes the monitored node for N packets and counts the number n_d of packets dropped by the monitored node. Third, the monitoring node calculates the *P-value* p using the following formula

$$p = \sum_{i=n_d}^{N} \binom{N}{i} q^i (1-q)^{N-i}. \quad (1)$$

where p is the right-tailed *P-value*, and n_d denotes the number of packets dropped among N packets where N is the number of packets being monitored at each link. Since some packets may be dropped due to poor network connection between nodes, we need to identify packets that are dropped in this manner as against packets that are dropped intentionally by malicious nodes. Therefore, we set a probability that a packet will drop due to channel fading and denote it by q.

If $p \leq \alpha$, the monitoring node rejects the hypothesis H_0, meaning that the monitored node is identified as a malicious node. Otherwise, the monitoring node accepts the hypothesis H_0. The whole process is summarized in Algorithm 1.

All nodes are first identified as benign nodes, which means no packet is expected to be dropped intentionally by a benign node. In such case, the *P-value* is calculated such that $p > \alpha > 0$. Contrarily, a malicious node is expected to drop more packets to reduce network performance. Therefore, the more the number of packets dropped, the more $p \to 0$. With this approach intentional dropping of packets can be easily detected on every link in the network using the calculated *P-value* p.

4.4.3 Packet History Field Monitoring

Each data packet contains a route history in the packet history field of the packet header, which records all events occurring to the packet, such as receiving or forwarding of a packet.

Algorithm 1 Monitoring packet dropping

Input: q : the probability of a packet being dropped
α : level of significance
N : sample size of observed packets
Variables: n_d : the number of dropped packets
p : *P-value*
j : counter
Output: Reject H_0 or Accept H_0

```
1: n_d ← 0;
2: j ← 1;
3: while j ≤ N do
4:     The monitoring node observes how the monitored node handles a received packet not destined for himself;
5:     j ← j + 1;
6:     if The monitored node drops the received packet then
7:         n_d ← n_d + 1;
8:     end if
9: end while
10: Calculate p according to Eq. (1);
11: if p ≤ α then
12:     return Reject H_0;
13: else
14:     return Accept H_0;
15: end if
```

To achieve this, a node creates a packet history field which is added to the packet header. The packet history field consists of the packet route and node signature. Intermediate nodes on the route to the destination append their signatures to the packet history field when they receive the packet. The signature serves as a confirmation for accepting a packet. Similarly, the destination node appends its signature on the packet route history field and replies to the source node with an acknowledgment packet which is used to confirm end-to-end transmission delivery.

The packet history field also contains the source node signature. Each field used for signatures of the intermediate nodes is time stamped. This allows the neighboring nodes to determine the delays at each node and to prevent modifications. The following information are stored in the packet history: time stamp, packet route and node signature.

4.4.4 Detecting Intentionally Delayed Packets

When receiving a packet, a benign node will insert the packet at the end of a packet queue that is served in First-In-First-Out manner (FIFO). However, a malicious node may intentionally delay inserting or removing the received packet into/from the queue, resulting in additional packet delay at that node. Packet delay at a node is defined to be the time interval from the time a packet is received by the node to the time that packet is transmitted.

Nodes that overhear packets can determine the packet queue order of their neighbors by checking the timestamp each time a packet is forwarded by a neighboring node to another node. If the packet is not delayed, the order of the packets will not change. However, if a node intentionally delays a packet, the order of packets in the queue changes. This can easily be detected by a neighboring node that is overhearing packets. Our method also ensures that there is time synchronization between benign nodes. Neighbors with unsynchronized time are treated as malicious.

To detect a node that is intentionally delaying packets, the

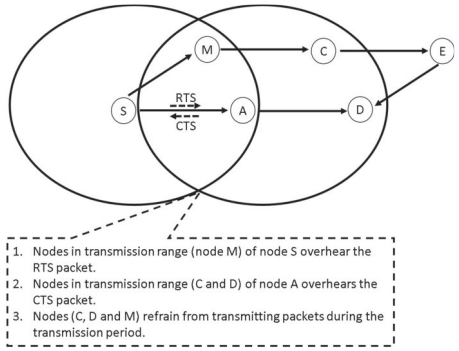


Fig. 6 An example of RTS/CTS transmission process.

RTS/CTS mechanism is used. The RTS/CTS packets contain information about the transmission duration of the data packet. This is used by other nodes to determine the estimated completion time of data transmission. If a CTS packet is overheard, other nodes wait until data packet, and ACK transmission are completed. Also, even if some links are busy and a node cannot send out packets, it can still receive a packet from another node.

In a situation where a node deliberately holds onto a packet after indicating its availability to receive and forward the packet by responding with a CTS packet. The previous node will not overhear the packet and then report such node as malicious. Hence, transmission collisions can be avoided and malicious nodes intentionally delaying packets are detected. **Figure 6** shows an example of RTS/CTS transmission.

In addition, a malicious node delay responding to an RTS packet might force the sending node to hold onto a packet, thereby delaying transmission. The sending node after the specified time elapses for receiving a CTS packet will select another route to send its packet.

4.5 Preventing Various Kinds of Attacks

In this section we explain how our monitoring scheme prevents Byzantine attacks. Specifically, we focus on preventing corruption of the routing table, wormhole attacks, colluding attacks, blackhole attacks, and delaying packets.

4.5.1 Corruption of Routing Table

A malicious node may try to corrupt the routing table information by advertising the wrong link delay to its neighbor or not adding a node as a neighbor in its hello message. In a situation where a node advertises a link delay that is better than the actual situation as described in Fig. 1 (a), where malicious nodes M advertises the link delay to node C to be 1, while node C advertises its link delay to the malicious node M as 10. Other benign nodes in the network will get conflicting information from the nodes connected to such a malicious node. In such a situation, nodes in the network will adopt the worse link delay. Similarly, if a node advertises a link delay that is worse than the actual situation while a neighbor node to such node advertise the actual delay cost, other nodes in the network will still adopt the worse link delay. This is not a problem since the link between these two nodes is a link that should be invalidated.

4.5.2 Wormhole Attack

As shown in Fig. 1 (d), if a node selected to take part in the routing of packets from the source to the destination decides to carry out a wormhole attack, it will do this by tunneling packets to another malicious node in the network which eventually drops the packets or selectively drops some packets. To prevent this, the neighboring node S overhears the packets, and detects the malicious action by observing how malicious node M_1 handles the received packet. Neighboring nodes such as node S also check the packet history field signed by the malicious node M_1 to determine the past activities of the packet, and check if node M_2 is part of the packet route by comparing the sending node address to the packet route information stored in the packet history field (e.g. node ID or MAC address in the packet header to the one stored in the packet history field). If node M_2 is not stored as part of the packet route information, the neighboring node reports that node M_1 is a malicious node to other nodes in the network. So recording and checking the route information prevents packet tunneling and wormhole attacks.

When this occurs, Node S will report that node M_1 is malicious to other benign nodes in the network, and the link between node S and malicious node M_1 will be excluded. Again, node S will afterwards select another path and retransmit its packets using the two hop reactive mode.

4.5.3 Black Hole Attack

As shown in Fig. 1 (c), if a node decides to drop or ignore packets, thereby carrying out a black hole attack, the source node S and node A will not overhear the packet. In this case, after a predetermined time interval without node S and node A overhearing the packet, and the statistical method described earlier detects that node M is malicious. Then the links between node S and node A to node M are excluded from the network, and the nodes report that node M is malicious to other nodes in the network. Afterwards, source node S selects another path for its packets and retransmits its packets using the two hop reactive mode.

4.5.4 Preventing Colluding Attacks

In our scheme, there are two main types of colluding attacks we address. The first colluding attack scenario is when only one node M_1 of the colluding nodes is part of the selected path and it forwards the packet to the second colluding node M_2 on a non-optimal path. This type of colluding attack is similar to the wormhole attacks discussed in Section 4.5.2 and can be prevented in a similar way as described above, that is neighboring nodes such as the source node S can check if the colluding node M_2 is part of the packet route by comparing node M_2 address (e.g., node ID or MAC address) to the packet route information stored in the packet history field. If node M_2 is not stored as a part of the packet route information, the neighboring node S reports that node M_1 is a malicious node to other nodes in the network and the link between node S and node M_1 is excluded from the network. Thereby this form of colluding attack is prevented.

In the second colluding attack scenario, suppose node S selected a route S - A - M_1 - M_2 - D which includes two malicious nodes M_1 and M_2 and the packet is dropped at node M_2 but node M_1 fails to report such malicious action. After a predefined time for receiving the ACK from the destination node D by node S has passed and the ACK is not received, node S requests from node A the overheard packet which includes the signed packet history

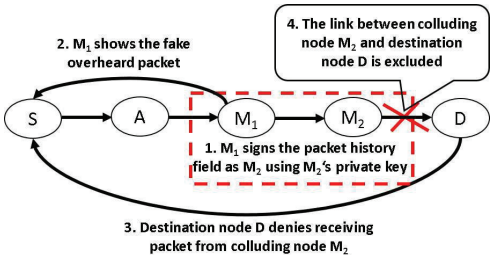


Fig. 7 Colluding attacks using fake overheard detection.

field that confirms that packet from node S is forwarded to the next hop by node M_1. In this situation, if node M_1 fails to show the overheard packet from node M_2, then node M_1 is reported as malicious and the link between node A and node M_1 will be excluded from the network.

Additionally, to prevent a situation in which node M_1 colludes with M_2 such that node M_1 gets node M_2's private key and uses it to make a fake overheard packet (i.e., node M_1 signs the packet history field as node M_2) and shows the fake overheard packet to node S as proof that the packet was forwarded from node M_2. Each node showing the overheard packet must also show the CTS packet received from the next hop node, which can be compared to the previous overheard RTS/CTS packets. After receiving the fake overheard packet, the source node S will request the destination node D to resend the ACK to confirm that the destination node D receives the packet from node M_2, then the destination node D can report not receiving the packet (see **Fig. 7**). Also, monitoring nodes can detect such packet dropping behavior by node M_2 by using the statistical hypothesis testing method. If the statistical hypothesis testing method confirms that node M_2 is malicious and there is no ACK received by the destination node D, then the link between node M_2 and the destination node D will be excluded from the network.

4.5.5 Preventing Intentional Packet Delay Attacks

In this situation, a malicious node M deliberately delays packets in the network, e.g., using the network in Fig. 1 (c). When this happens, source node S will overhear the packet, and check the order of packets in the packet queue for node M. Node S also checks whether it has previously overheard RTS/CTS packets from node M. If the queue order of the packets has changed and node M has not been sending and receiving RTS/CTS packets, then node S determines that node M is maliciously delaying packets. Node S reports node M as a malicious node to other nodes in the network.

5. Evaluation

In this section we first highlight the security goals that our proposed system achieves. Then we evaluate the performance of our proposed monitoring scheme for securing Link State Routing against Byzantine attacks using a custom simulator.

5.1 Security Goals

When our proposed monitoring scheme is run successfully, it achieves the following security goals.

- *Authentication:* In our scheme, there is no certificate authority, therefore each node creates its unique public and private keys in advance. The unique key pair is safely created from random numbers by any node. The unique key pair is used to generate and verify a digital signature which is used for authentication in our schemes. When a node sends/forward packets, the node will sign its ID which can be verified by other nodes to authenticate the sender of such packets. Also, the packet history field used to store the route history information is digitally signed with source node *S's* private key.
- *Confidentiality:* All data in the packet are encrypted and digitally signed by users. When the source node S sends a packet to destination node D, the data message in the packet is encrypted with a session key created using the symmetric key algorithm and the session key is encrypted with the destination node *D's* public key so none of the intermediate nodes on the route to the destination in the network can decrypt the message. Hence, confidentiality of the message between the source node S and the destination node D is maintained.
- *Non-repudiation:* When a packet is sent by a source node S to a destination node D, each intermediate node appends their signature to the packet history field as a confirmation of receiving the packets and records a timestamp when the node forwards the packet to confirm forwarding the packet. To achieve non-repudiation in the network, each intermediate node digital signature and timestamp is verified whenever the previous node overhears packet forwarding by the intermediate nodes. An intermediate node cannot deny appending its signature to the packet history field.
- *Integrity:* To ensure that the packet and the packet history field are not tampered with by a malicious node or an intermediate node, the source node S signs its signature on the packet history field which can be verified wherever the packets are overheard. The digital signature and timestamp from the source node S ensure the integrity of the packets and the information in the packet history field.

5.2 Simulation Configuration

The performance of our proposed monitoring scheme is evaluated using a custom simulator. In our simulation, we implement all parts of our proposed protocol and focus our evaluation on the packet dropping attacks such as black hole attacks, wormhole attacks and colluding attacks, where a malicious node selectively or completely drops received packets. The main objectives of our simulation are to validate (i) that our monitoring scheme guarantees communication among benign nodes, and (ii) detect if a node is dropping packets deliberately.

The simulated scenario is implemented to enable benign nodes to connect with each other easily within the transmission range, given that mutual monitoring is an important aspect in our protocol. Nodes are first evenly placed in a 3 km × 3 km area as shown in **Fig. 8** with malicious nodes randomly selected to maintain a uniform position in the network. We varied the number of malicious nodes from 2 to 12 out of the 20 network nodes. The skeleton map represents the network topology. Each node maintains low mobility in the network. The route is based on Dijkstra's shortest path algorithm. Each node randomly selected a destination node and calculates the shortest path to the desti-

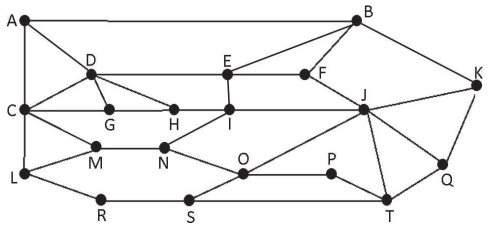


Fig. 8 Network topology for simulation.

Table 1 Simulation parameters.

Parameter	Value
Simulation time	3,600 seconds
Bandwidth	1 Mbps
Buffer Size	100–500 KB
Transmission range	250 m
Network size	3 km x 3 km
Mobility	Static
Number of nodes	20
Number of malicious nodes	2–12
Level of significance (α)	0.0001
Probability of packet dropped due to fading (q)	0.001
Probability of packet dropped by malicious nodes	0.5
Sample size of observed packets (N)	1,000
Total number of packet generated	50,000

nation node. Then each node generates a packet every second, the packet is routed to the selected destination node using the our secured link state routing protocol. The total number of packets generated in the network is 50,000 packets. In our simulation, we eliminate sending packets on selected paths with just 1 or 2 hops. We used paths with 3 hops and above to route packets. All nodes have the same buffer size and transmission range. We assume 802.11g wireless WiFi (802.11g comes with ad-hoc mode) is used for communication. The network bandwidth of our simulation is set to 1 Mbps. We set the logical packet size to 5 KB, a logical packet is a combination of several physical frames.

The probability that a packet is being dropped due to network fading is set to 0.001 while the probability that malicious nodes dropped a packet is set to 0.5. Each link in the network is monitored for 1,000 packets. Links between nodes are identified as a malicious link if the calculated P-value p is less than the level of significance α which is set to 0.0001. All simulated results in the figures below are the averages of 30 simulation runs. The summary of the default values used in our simulation is shown in **Table 1**.

The following metrics will be measured in our simulation.

Packet delivery ratio: This is the ratio between the number of packets successfully delivered to destinations and the number of packets generated by sources. Not all packets can be successfully delivered to destinations due to reasons like malicious node dropping packets, buffer overflow, etc.

False positive ratio: This is the ratio between the number of links between benign nodes that are falsely removed as malicious and the number of all network links. It is desirable to have a low false positive ratio.

Malicious link detection ratio: This is the ratio between the number of successfully detected malicious links and the number of all malicious links in the network.

Malicious node packet dropping ratio: This is the ratio between the number of packets dropped by malicious nodes and the number of packets generated by the sources.

5.2.1 Packet Delivery Ratio

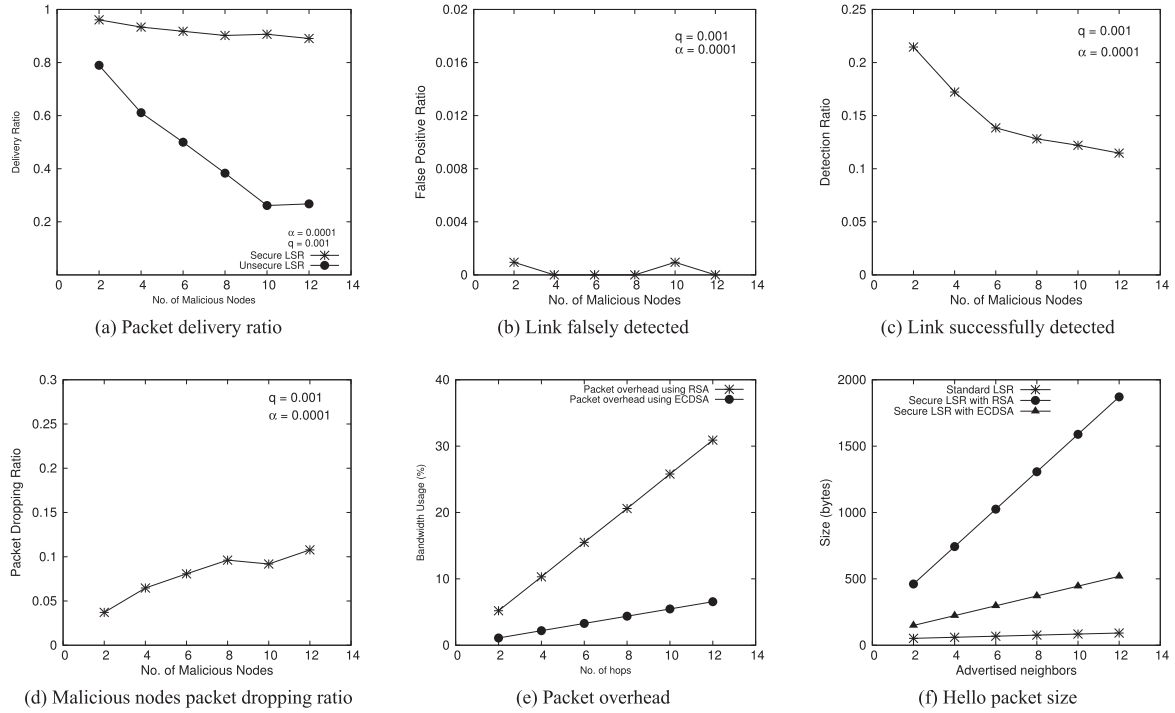
In our simulation, to show that our system guarantees communication among benign nodes in the network we estimate the packet delivery ratio. We set the probability that a packet is dropped due to channel fading to 0.1% and the probability that a packet is dropped deliberately by a malicious node is set at 50%. Each monitoring node observes the link between two nodes for 1,000 packets and counts the number of packets that is being dropped before calculating the *P-value* that such node is malicious or not. If the P value is greater than the level of significance, the link between the two nodes is further monitored for another 1,000 packets until a malicious node on the path is detected and the link is excluded. In our method, malicious links require 1,000 packets to be detected. We evaluate the number of packets that are successfully delivered to the destination node. As shown in **Fig. 9** (a), our proposed monitoring scheme achieved an average of 89% to 96% packet delivery ratio. The packet delivery ratio reduces as the number of malicious nodes in the network increases. The reason for the higher packet delivery ratio when the number of malicious nodes is set between 2–6 as compared to 12 malicious nodes is that the benign nodes have more alternative routes to send their packets to the destination nodes. Hence, more packets are delivered. The more the number of malicious nodes, the more the number of packets that are dropped and vice versa. In addition, our proposed scheme still achieved 89% packet delivery ratio even when more than half of the network nodes (i.e., 12 out of the 20 nodes in the network) are malicious.

Moreover, the proposed scheme is compared to a case without the proposed scheme. In such a situation, malicious nodes drop packets without being detected. As shown in the figure, the packet delivery decreases as the number of malicious nodes increases when there is no secure scheme adopted in the network. The proposed scheme achieves a better performance in packet delivery compared to when no secure scheme is used. Using the proposed scheme, the packet delivery ratio is increased by 17%, 32%, 42%, 52%, 64% and 62% compared to when no secure scheme is used with 2, 4, 6, 8, 10, and 12 malicious nodes respectively.

5.2.2 False Positive Ratio

In our scheme, each node monitors how the received packet is being handled by the intermediate node. When a monitoring node overhears a packet, the monitoring node calculates the P value p and compares it with the level of significance α. If the P value is less than the level of significance, the node being monitored is detected as malicious and reported to other nodes in the network. Therefore, the link between the malicious and the benign node is excluded from the network. However, it is possible that a malicious node reports other benign nodes as being malicious, thus we set the frequency of malicious nodes reporting other benign nodes as being malicious in the simulation to 0. Any malicious node with a frequency greater than 0 is excluded in the network. Therefore, such malicious node behavior is quickly detected and further influence of such malicious behavior is prevented.

In our simulation, we evaluate the false positive ratio of links among benign nodes that are falsely detected as being malicious



(a) Packet delivery ratio (b) Link falsely detected (c) Link successfully detected

(d) Malicious nodes packet dropping ratio (e) Packet overhead (f) Hello packet size

Fig. 9 Simulation results for the network.

links. Figure 9 (b) shows that the false positive in our scheme is significantly lower with an average of 0.1% false positive ratio when 2 and 10 of the network nodes are malicious.

5.2.3 Malicious Link Detection Ratio

Figure 9 (c) shows the detection ratio of malicious links that are excluded from the network after being detected as malicious links. The overall average malicious links in our network is 16 links. Our proposed monitoring scheme achieved an average of 11% to 21% malicious link detection ratio. The malicious link detection ratio decreases as the number of malicious nodes increases. The goal of our method is to detect malicious links that are utilized in routing a packet from a source node to a destination node and guarantee secure communication among benign nodes. By excluding 11% to 21% of the active malicious links in the network, our method achieves higher packet delivery ratio as shown in Fig. 9 (a). In addition excluding all malicious links all at once in the network may result in network partition.

5.2.4 Malicious Nodes Packet Dropping Ratio

As shown in Fig. 9 (d), the packet dropping ratio of malicious nodes increases as the number of malicious nodes increases. In our simulation, 10% of the packets are dropped when 8 out of the 20 nodes in the network are malicious while 11% of packets are dropped even when more than half of the nodes in the network are malicious (12 out of the 20 nodes). The reason for the lower packet drop rate in the network is due to the early exclusion of some malicious links which further isolates other malicious nodes from being selected in packet routing without causing a network partition among benign nodes.

5.3 Communication Overhead

In this section we discuss the overhead cost introduced by our monitoring scheme using the speed benchmark for cryptographic algorithm: River Shamir Adelman (RSA) with 1,024 bit key size and ECDSA 192 bit key size digital signature algorithm [16], run on an Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode x86/MMX/SSE2.

The packet overhead of our scheme is divided into three parts: the first is the packet history field size required to send a packet from a source node to a destination node. The second is the hello message size a node exchanged with its neighbor nodes and the last part is the computational cost of signing and verifying the digital signature of intermediate nodes by the monitoring nodes.

5.3.1 Packet History Fields Size

In our proposed scheme, for a node to transmit a packet to a destination, the node needs to create packet history fields which are used for monitoring if the packet is forwarded on the right route. Each intermediate node appends its signature on the packet history field, therefore, we need to calculate the additional overhead introduced by our monitoring scheme to transmit a packet from a source node to a destination node. First, we find the number of signatures appended to the packet history field between the source node to the destination node (N_S), and then the size of the signature appended by the intermediate node (s).

In addition, each source node stored the optimal path for sending its packet to the destination node in the packet route fields of the packet history. We also calculate the overhead introduced by this route specification (RS) in our protocol. The size of each next hop ID stored in the packet route fields from the source node to the destination node is 4 bytes. Therefore, the total overhead cost needed to route a packet to destination in our scheme is calculated as: $T_{overhead} = (N_S \times s) + RS$. The size of the packet history field increases as the number of nodes increases. We compare the size of the packet history field using the RSA (128 bytes) and ECDSA (24 bytes) signature. The average total overhead in term of size of our packet history field is 1,320 bytes and 380 bytes respectively, when the number of nodes in the route is 10.

Each packet needs to store extra information about intermediary nodes, which are the node ID of 4 bytes and signature of 24 bytes for ECDSA which form a total overhead of 28 bytes per hop. Let's say there are 10 hops between a source and destination. We assume that the packet size is 5 KB, therefore, the additional traffic generated will utilize 5.5% ($\frac{28\times10}{5\times1024}$) of the bandwidth. Figure 9 (e) shows the overhead of our secure monitoring scheme when the number of hops from source to destination is varied from 2 to 12. The scheme achieves better performance with an average of 7% maximum bandwidth utilization. There is a 4%, 8%, 12%, 17%, 21% and 25% decrease in bandwidth utilization using the ECDSA signature algorithm as compared to the RSA signature with hop numbers of 2, 4, 6, 8, 10, and 12, respectively.

5.3.2 Hello Message Size

In our protocol, each node exchanges hello messages with its neighbor nodes and appends to its hello message the information from its neighbor hello message such as neighbor ID, link cost, neighbor signature and timestamp of the hello message. Therefore, we evaluate the size of the hello messages in our protocol as compared to a typical hello message size of LSR protocol. The size of the hello message depends on the number of nodes in the network, each node distance, the number of hello message broadcast by neighbor nodes and their link status. To determine the size of the hello message at each node we estimate the size of the hello message in our scheme and compare it with that of the standard LSR without any security measure. The average size of a standard LSR hello message is 48 bytes when only one node is advertised as a neighbor while the hello message in our scheme is 320 bytes and 112 bytes using RSA and ECDSA signatures. Figure 9 (f) shows that the LSR hello message size increases from 52 bytes to 92 bytes as the number of nodes advertised as neighbor increases. Comparatively, the hello message size of our scheme increases from 461 bytes to 1,871 bytes when the RSA signature is used. However, by adopting the ECDSA signature we reduce the hello message size of our scheme by 67% to 72% when we varied the advertised neighbors from 2 to 12 nodes.

In addition, there are four additional packets (i.e. database descriptor, link state request, link state update and link state acknowledgment) in LSR protocol, that use a common 24-bytes header as the hello message. We adopt the implicit acknowledgment where a neighbor that received a packet makes a duplicate and encode it to its ACK, then send the ACK back to the sending node. Multiple neighbors can be acknowledged in a single multicast ACK packet.

5.3.3 Computation Overhead

Each node on the selected route generates its signature, which is appended to the packet history field and verified by monitoring nodes in the network. The average computation time for generating a signature by each node on the route is 0.002 seconds while the verification time by monitoring nodes that overhears the packet is less than 0.0001 seconds. Using the network in Fig. 5, with 5 nodes on the packet route, the total processing time used for generating signatures by each node on the route is 0.01 seconds while a total verification time used by nodes to verify the packet history field is 0.0003 seconds. Hence, the computational overhead of our scheme is adequately low.

5.4 Current Open Issues of Interest

In this paper, we described how our proposed monitoring scheme prevents Byzantine attacks. However, we did not consider DoS attacks such as overwhelming amount of traffic or jamming signals. In jamming signal attacks, a malicious node disrupts the communication of benign nodes by blocking the transmission of radio signals in the network. A jamming signal attack is an active attack, which can be carried out by an outside attacker. There is no way to prevent these kind of attacks.

6. Conclusion

In this paper, we propose a monitoring scheme to secure link state routing against Byzantine attacks. We adopt the statistical hypothesis testing to determine if a node is intentionally dropping packets. In addition, our monitoring scheme also uses hello message verification to validate the hello messages and to identify inconsistent information when a malicious node tries to corrupt routing table information, and packet history field. The monitoring node checks whether packets are forwarded correctly according to the stored optimal path. Also, our monitoring scheme uses RTS/CTS to identify when a node is intentionally delaying packets in the network.

Our approach guarantees communication among benign nodes as validated by simulation with an average of 89% packet delivery ratio even when 12 out of the 20 nodes in the network are malicious. Also, simulation results show a significantly lower false positive ratio with an average of 0.1% while the malicious link detection ratio is between 11% to 21%. Active malicious links to benign nodes are detected and excluded from the network, thereby guaranteeing communication among benign nodes. In addition, our monitoring scheme achieves relatively low communication overhead with an average of 299 bytes packet history field size and 519 bytes hello message size. Also, the communication overhead for generating and verifying signature by each node is less than 1 second (0.002 seconds and 0.0001 seconds respectively).

Acknowledgments This work is supported by the Otsuka Toshimi Scholarship Foundation reference number 17-S72 and JSPS KAKENHI Grant Numbers JP16K12421, JP15K15981, JP16K01288.

References

[1] Geetha, A. and Sreenath, N.: Byzantine Attacks and its Security Measures in Mobile Adhoc Networks, *Int'l Journal of Computing, Communications and Instrumentation Engineering* (*IJCCIE 2016*), Vol.3, No.1, pp.42–47 (2016).

[2] Ojetunde, B., Shibata, N. and Gao, J.: Securing Link State Routing for Wireless Networks against Byzantine Attacks: A Monitoring Approach, *Proc. IEEE 41st Annual Computer Software and Applications Conference* (*COMPSAC 2017*), Turin Italy, pp.596–601 (2017).

[3] Harshavardhan, K.: A Survey on Security Issues in Ad Hoc Routing Protocols and their Mitigation Techniques, *International Journal of Advanced Networking and Application*, Vol.3, No.5, pp.1338–1351 (2012).

[4] Ali, D., Seyed, K. and Esmaeil, K.: Security Challenges in Mobile Ad hoc Networks: A Survey, *International Journal of Computer Science and Engineering* (*IJCSES 2015*), Vol.6, No.1, pp.15–29 (2015).

[5] Mojtaba, S., Imran, G., Aida, H. and Seung, J.: Routing Attacks in Mobile Adhoc Networks: An Overview, *Science International* (Lahore), Vol.25, No.4, pp.1031–1034 (2013).

[6] Kannhavong, B., Nakayama, H., Nemoto, Y., Kato, N. and Jamalipour,

A.: A survey of routing attacks in mobile ad hoc networks, *IEEE Wireless Communications*, Vol.14, No.5, pp.85–91 (2007).
[7] Jhaveri, R.H., Patel, S.J. and Jinwala, D.C.: DoS Attacks in Mobile Ad Hoc Networks: A Survey, *Proc. 2012 Second International Conference on Advanced Computing and Communication Technologies*, Rohtak, Haryana, pp.535–541 (2012).
[8] Zapata, M.G. and Asokan, N.: Securing ad hoc routing protocols, *Proc. 1st ACM Workshop on Wireless security* (*WiSE '02*), pp.1–10, ACM, New York, NY, USA (2002).
[9] Alajeely, M., Ahmad, A. and Doss, R.: Malicious Node Detection in OppNets using Hash Chain Technique, *2015 4th International Conference on Computer Science and Network Technology* (*ICCSNT*), Harbin, pp.925–930 (2015).
[10] Baadache, A. and Belmehdi, A.: Fighting against packet dropping misbehavior in multi-hop wireless ad hoc networks, *Journal of Netw. Comput. Appl.*, Vol.35, No.3, pp.1130–1139 (2012).
[11] Papadimitratos, P. and Haas, Z.J.: Secure link state routing for mobile ad hoc networks, *Proc. IEEE Workshop on Security and Assurance in Ad hoc Networks*, in conjunction with the 2003 Symposium on Applications and the Internet, pp.379–383 (2003).
[12] Papadimitratos, P. and Haas, Z.J.: Secure data transmission in mobile ad hoc networks, *Proc. 2nd ACM Workshop on Wireless security* (*WiSe '03*), pp.41–50, ACM, New York, NY, USA (2003).
[13] Bhuiyan, M.Z.A. and Wu, J.: Collusion Attack Detection in Networked Systems, *Proc. 2016 IEEE 14th Intl. Conf. on Dependable, Autonomic and Secure Computing, 14th Intl. Conf. on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress* (*DASC/PiCom/DataCom/CyberSciTech*), Auckland, pp.286–293 (2016).
[14] Bonaventure, O.: Computer Networking: Principles, Protocols and Practice, pp.41–45 (2011), available from ⟨http://inl.info.ucl.ac.be/cnp3⟩.
[15] Johnson, D., Menezes, A. and Vanstone, S.: The Elliptic Curve Digital Signature Algorithm (ECDSA), *International Journal of Information Security*, Vol.1, No.1, pp.36–63 (2001).
[16] Al Imem, A.: Comparison and evaluation of digital signature schemes employed in NDN network, *International Journal of Embedded System and Applications* (*IJESA*), Vol.5, No.2, pp.15–29 (2015).

Babatunde Ojetunde received his HND degree in Computer Science from Osun State Polytechnic, Iree, Osun, Nigeria in 2003 and also received his PGD in Computer Science from Lagos State University, Ojo, Lagos, Nigeria in 2006. He received his M.E in Information Science from Nara Institute of Science and Technology, Japan in 2015. Currently, he is a Ph.D. student at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include mobile computing, and ubiquitous computing.

Naoki Shibata received his Ph.D. degree in Computer Science from Osaka University, Japan, in 2001. He was an assistant professor at Nara Institute of Science and Technology from 2001 to 2003 and an associate professor at Shiga University from 2004 to 2012. He is currently an associate professor at Nara Institute of Science and Technology. His research areas include distributed systems, inter-vehicle communication, mobile computing, multimedia communication, and parallel algorithms. He is a member of IPSJ, ACM and IEEE/CS.

Juntao Gao received his B.S. and M.S. degrees both in Computer Science from Xidian University, Xi'an, China, in 2008 and 2010, respectively, and received his Ph.D. degree from Graduate School of Systems Information Science at Future University Hakodate, Japan, in 2014. He is an assistant professor at Graduate School of Information Science at Nara Institute of Science and Technology, Japan. His research interests are in the areas of performance modeling and analysis, stochastic optimization and control in wireless networks, queueing theory and its applications.