Vol.2017-DPS-171 No.31
Vol.2017-MBL-83 No.31
Vol.2017-ITS-69 No.31
2017/6/2

A Proposed Monitoring Scheme to Prevent Byzantine Attacks on Link State Routing in MANETs

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

Abstract: Secure communication is essential in most applications such as battlefield and disaster management applications. Most existing protocols adopt cryptography based approach, trust based approach (reputation of nodes) and incentive based approach to detect and prevent attacks in such applications. However, such protocols are still subjected to drawbacks like expensive overheads, difficulty in maintaining secure key and session management, unsecured routes against Byzantine attacks and so on. Therefore, we introduce a monitoring scheme to secure packets route in link state routing protocol against Byzantine attacks.

1. Introduction

Secure timely delivery of packet is necessary in wireless network communication, however, it is impossible to achieve such timely delivery if the malicious action of nodes cannot be detected or prevented. There have been various security protocols proposed to secure a packet route against malicious attacks, but none of those protocols adopt the monitoring approach to detect various 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 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 works 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 our research, we proposed a monitoring approach to secure the link state routing protocol against Byzantine attacks [2]. In this paper, we show the overheads of our monitoring scheme. The goal of our proposed scheme is to guarantee communication among connected benign nodes in 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 - used to validate the 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 whole 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

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

to the destination only when needed. Examples include Ad-hoc On-Demand Vector (AODV) and Dynamic Source Routing. Finally, hybrid protocols are produced by combining both proactive and reactive protocols. 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.

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-hole, wormhole, replay, 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.* [2] 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.

Allegedly *et. al.* [9] proposed a new detection scheme for malicious nodes to detect packet faking by a malicious node. 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 are detected if the root of 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 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 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 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 from 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 Routing Protocol and Byzantine Attacks

3.1 Link State Routing Protocols (LSR)

Link state routing (LSR) protocols are proactive protocols in which a node creates a topology of the network and positions itself at the root of the tree. LSR protocols are based on a Shortest Path First algorithm, also known as Dijkstra's Algorithm, to find the best path to a destination. There is no hop count limit in LSR protocols. Examples of LSR protocols are open shortest path first and intermediate system to intermediate system.

In LSR, each node finds out the status and the cost of their neighbors' links, then creates a map of the network showing how nodes are connected to each other. This information is then broadcast to the entire network. Using this information, each node then calculates the best path to every possible destination. A node then collects the best paths to each destination to form its

routing table. When a node's link status changes, a routing update called a Link-State Advertisement (LSA) is exchanged between nodes. Whenever a node receives an LSA routing update, the link-state algorithm is used to recalculate the shortest path to the affected destinations.

Detecting an outsider attack on the packets exchanged by neighbors can be performed by using cryptographic schemes such as digital signatures or hash chains. However, it is difficult to prevent an insider attack such as falsifying routing information, dropping packets, faking packets or other Byzantine attacks. Such attacks are described in the following subsections.

3.2 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]. The malicious nodes can either 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.

3.2.1 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 1a shows an example of a corruption of routing table attack.

3.2.2 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 1b shows an example of a black hole attack.

3.2.3 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.

3.2.4 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 causes packet loss in the network.

3.2.5 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 each other as neighbors. The first colluding scenario is when a malicious node M_1 is part of the selected 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 Figure 1c, is when two malicious nodes advertise the wrong link costs, e.g. 1 and 2 respectively, in order to be included in the best packet route.

These types of Byzantine attacks are difficult to detect or prevented, 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 describe our secure routing protocol using monitoring designed to protect a network against Byzantine attacks. First 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.

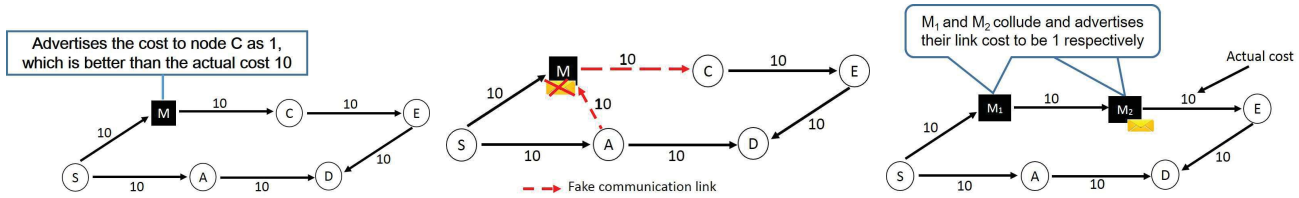
4.1 Assumptions

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 key.
- 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. 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.

4.2 Routing Table Formation

In our LSR protocol, each node broadcasts hello messages to its neighbors periodically, e.g. every 10 seconds, with a dead interval of 30 seconds. The dead interval is used to confirm if a node is still alive. If a node fails to receive hello messages from a particular neighbor within the dead interval, then that neighboring node is considered to be disconnected from the network. In addition to the information of 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



(a) Corruption of routing table attack (b) Black hole attack (c) Colluding by delaying packets

Fig. 1: Examples of some types of Byzantine attacks.

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 2 shows an example of a 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.

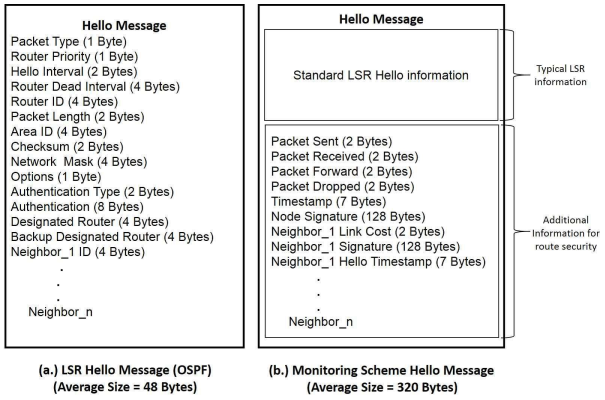


Fig. 2: An example of hello message

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, each node authenticates each other with a digital signature. A node will sign its signature on the ID which can be verified by other nodes. The public and private key scheme is used to generate a digital signature and 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. 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. In addition, any node can join the network without pre-registration.

4.3 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.

4.3.1 Overview of Proposed Scheme

Let's consider a situation 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 that packets are forwarded 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. 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 source node does not receive this acknowledgement packet for some time, 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 other 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 collisions from other nodes. Also, if one of neighboring nodes is communicating with other nodes, that node cannot start sending out packet. 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 4.3.4) 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 accused of intentionally delaying packets can show the RTS/CTS packets as a proof of not delaying packets.

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 a 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.

4.3.2 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

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

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.

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 (1);
11: if p ≤ α then
12:     return Reject H_0;
13: else
14:     return Accept H_0;
15: end if
```

4.3.3 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 packet received or packet forwarded. 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 reply 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.

4.3.4 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 and neighbors with unsynchronized time are treated as malicious.

To detect a node that is intentionally delaying packets, the 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 on to 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 is detected. Figure 3 shows an example of RTS/CTS transmission.

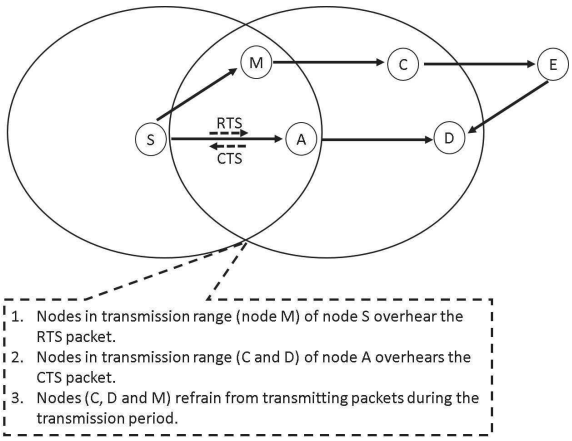


Fig. 3: An example of RTS/CTS transmission process

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

4.4 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.

4.4.1 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 reality as described in Figure 1a, 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 reality while a neighbor node to such node advertise the real 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.

4.4.2 Wormhole Attack

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, afterwards node S will select another path and retransmit its packets using the two hop reactive mode.

In addition, if a node decides to drop or ignore packets, thereby carrying out a black hole attack. In this case, source node S and node A will not overhear the packet. After a predetermined time interval without node S and node A overhearing the packet, 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.

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. Af-

ter 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 request from node A the overheard packet which includes the signed packet history field that confirms that node S packet 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.

4.4.3 Preventing Intentional Packet Delay Attacks

In this situation, a malicious node M deliberately delays packets in the network, e.g. using the network in Figure 1b. 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 he 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 evaluate our proposed monitoring scheme for securing Link State Routing against Byzantine attacks, in terms of packet overhead it adds.

5.1 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 1024 bit key size and ECDSA 192 bit key size digital signature algorithm [14], 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.

5.1.1 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), 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 1 byte. Therefore, the total overhead cost needed to route a packet to destination in our scheme is calculated as:

$$T_{overhead} = (N_S \times s) + RS$$

For example, using the scenario where S - M - C - E - D is the selected route to send a packet from a source to a destination. The packet is signed 5 times with the signature of each node from the source to the destination. Hence, using 1024 bit key size, the size of the packet history field is calculated as:

$$N_S = 5$$
$$RS = 4 \times 1$$

With RSA	With ECDSA
$s = 128bytes$	$s = 24bytes$
$T_{overhead} = (5 \times 128) + 4$	$T_{overhead} = (5 \times 24) + 4$

The total size of the packet history field needed to store each node signature and the packet route when sending a packet from source node S to destination node D is 644 bytes with RSA and 124 bytes with ECDSA. This is proportional to the number of nodes in the selected packet route.

5.1.2 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.

Suppose we have a network with 10 nodes (e.g. node A - J) and the average number of neighbors connected to by each node is 3. To determine the size of the hello message at each node we use the following, $LSR_{HelloSize}$: standard LSR hello message size, $hello_{info}$: the size of other information in the LSR hello message without the neighbor list (since the size of the neighbor list depends on the number neighbors of each node), $NeighborList_{size}$: size of each neighbor in the neighbor list and number of neighbors: n. The standard LSR Hello message size is calculated as follows:

$$LSR_{HelloSize} = hello_{info} + (NeighborList_{size} \times n)$$
$$LSR_{HelloSize} = 44 + (4 \times 3) = 56bytes$$

In addition to the standard LSR hello message size which is 48 bytes if a node is connected to just one neighbor, we introduced additional information to achieve routing security. The additional information using RSA and ECDSA signatures are $SNode_{info}$: sending node signature (RSA: 128 bytes, ECDSA: 24 bytes) and timestamp (7 bytes), N_{added}: each neighbor's link cost (2 bytes), signature (RSA: 128 bytes, ECDSA: 24 bytes) and timestamp of their hello messages (7 bytes).

With RSA	With ECDSA

$SNode_{inf} = 128 + 7 = 135$ $SNode_{inf} = 24 + 7 = 31$
$N_{added} = 2 + 128 + 7 = 137$ $N_{added} = 2 + 24 + 7 = 33$

Hence the size of hello message in our protocol:

$$P_{HelloSize} = LSR_{HelloSize} + SNode_{info} + (N_{added} \times n)$$
$$P_{HelloSize}(ECDSA) = 56 + 31 + (33 \times 3) = 186bytes$$
$$P_{HelloSize}(RSA) = 56 + 135 + (137 \times 3) = 602bytes$$

Table 1 shows the comparison of a typical standard hello message size for LSR and our proposed protocols when the number of neighbors is 3 using the size stated in Figure 2.

Table 1: Comparison of a typical hello message size of LSR and our monitoring scheme

Protocol	Key Type	Hello size (bytes)
Standard LSR	-	56
Proposed monitoring scheme	192 bit ECDSA key	186
	1024 bits RSA key	602

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.

5.1.3 Computation Overhead

To measure the computational overhead of our scheme, we used the speed benchmark for cryptographic algorithm for RSA 1024 bit key size. 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 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.

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.

5.2 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 too much 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 this kind of attacks.

6. Conclusion

In this paper, we proposed a monitoring scheme to secure link state routing against Byzantine attacks. The goal of our proposed scheme is to guarantee communication among connected benign nodes in the network. Our monitoring scheme adopts three main methods: (i) Hello message verification - to verify the validity of hello messages and to identify inconsistent information when a malicious node tries to corrupt routing table information, (ii) Packet history field monitoring - for neighboring nodes to check whether packets are forwarded correctly according to the stored optimal path and (iii) Statistical hypothesis testing - to know if a node is intentionally dropping packets. Also, our monitoring scheme uses RTS/CTS to identify when a node is intentionally delaying packets in the network.

Acknowledgments This work is supported by the Otsuka Toshimi Scholarship Foundation reference number 16-S38 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, *(IJCCIE 2016)*, Int'l Journal of Computing, Communications and Instrumentation Engineering, Vol. 3, Issue 1, 2016.

[2] Babatunde Ojetunde, Naoki Shibata, and Juntao Gao Consideration on Monitoring Scheme to Secure Link State Routing against Byzantine Attacks, *to appear in the proc. of COMPSAC 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. 03, Issue 05, pp. 1338-1351, March-April, 2012.

[4] Ali, D., Seyed, K., Esmaeil, K.: Security Challenges in Mobile Ad hoc Networks: A Survey, *(IJCSES 2015)* International Journal of Computer Science and Engineering, Vol. 6, No. 1, February, (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, in IEEE Wireless Communications, Vol. 14, no. 5, pp. 85–91, October 2007.

[7] Jhaveri, R., H., Patel, S., J., and Jinwala, D., C.: DoS Attacks in Mobile Ad Hoc Networks: A Survey, in Proceedings of 2012 Second International Conference on Advanced Computing and Communication Technologies, Rohtak, Haryana, 2012, pp. 535-541.

[8] Zapata, M., G., and Asokan, N.: Securing ad hoc routing protocols, in Proceedings of the 1st ACM workshop on Wireless security (WiSE '02). ACM, New York, NY, USA, 1–10, 2002.

[9] Alajeely, M., Ahmad, A., and Doss, R.: Malicious Node Detection in OppNets using Hash Chain Technique, 4th International Conference on Computer Science and Network Technology (ICCSNT 2015), Vol. 1. IEEE, 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, May 2012, pp. 1130-1139.

[11] Papadimitratos, P., and Haas, Z., J.: Secure link state routing for mobile ad hoc networks, in Proceedings of the IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 Symposium on Applications and the Internet, pp. 379–383, January, 2003.

[12] Papadimitratos, P., and Haas, Z., J.: Secure data transmission in mobile ad hoc networks, in Proceedings of the 2nd ACM workshop on Wireless security *(WiSe '03)*, ACM, New York, NY, USA, 41–50, 2003.

[13] Bhuiyan, M., Z., A., and Wu, J.: Collusion Attack Detection in Networked Systems, in Proceedings of 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, 2016, pp. 286-293.

[14] Al Imem, A.: Comparison and evaluation of digital signature schemes employed in NDN network, in International Journal of Embedded system and Applications (IJESA), Vol. 5, No. 2, June 2015.