Securing Link State Routing for Wireless Networks against Byzantine Attacks: A Monitoring Approach

Babatunde Ojetunde, Naoki Shibata, and Juntao Gao
Nara Institute of Science and Technology, Nara, Japan
Email: ojetunde.babatunde.nq3@is.naist.jp; n-sibata@is.naist.jp; jtgao@is.naist.jp

***Abstract*—Byzantine attacks are constant threats in many applications such as disaster management, and battlefield applications. In this paper, we introduce a monitoring scheme 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. 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.**

Keywords – Secure, routing, routing protocol, routing attack, monitoring, byzantine attacks, Link State Routing

1. Introduction

Routing security is an important aspect of wired and wireless networks that has been given wide consideration over the years. The significant improvement in the role that such security features play in the whole network cannot be overemphasized. In the past, several security mechanisms have been proposed for various routing protocols. 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]. However, routing protocols are still subject to one form of attack or another, such as the Byzantine attacks.

In this paper, we propose a monitoring scheme to secure the link state routing protocol (LSR) against Byzantine attacks except Denial of Services (DoS) attacks. Here, a DoS attack is an attack where one or more malicious nodes transmit too many 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 approach 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 - 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 - 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. 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 LSR 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. Harshavardhan [2] 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. Then they analyzed various security threats and techniques to mitigate them. Ali *et. al.* [3] also surveyed security challenges in mobile ad-hoc networks (MANETs) and discussed various mitigating approaches against attacks in MANETs. They introduced three important security parameters, and further divided security aspects into two areas, which are security services and attacks. Mojtaba *et. al.* [4] 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.* [5] surveyed routing attacks in MANETs. They investigated various security issues in MANETs and examined routing 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.* [6] surveyed various DoS attacks that are security concerns in MANETs and some of the proposed solutions to identify and prevent such attacks. Zapata *et. al.* [7] 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. Allegedly *et. al.* [8] 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. Baadache *et. al.* [9] 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.* [10] 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. 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. 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.* [11] proposed a different mechanism from their previous work. 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.

3. Overview of Routing Protocol and Byzantine Attacks

In this section, we describe the link state routing protocols 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. 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.

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.

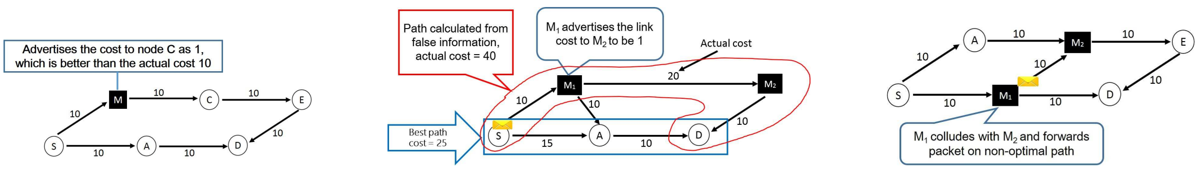
In these attacks, the goal of a malicious node is to corrupt the routing table, either by falsifying neighbor information, or by capturing and modifying the neighbors' link information broadcast by a benign node. Doing this can cause the routing protocols to maintain the wrong information in the routing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

3.2.2. Black Hole Attacks.

In this 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.

3.2.3. Wormhole Attacks.

In this 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


Advertises the cost to node C as 1, which is better than the actual cost 10
M
C
E
S
A
D
10
Path calculated from false information, actual cost = 40
M₁ advertises the link cost to M₂ to be 1
Actual cost
Best path cost = 25
M₁
M₂
20
15
M₁ colludes with M₂ and forwards packet on non-optimal path

(a) Corruption of routing table (b) Wormhole attack (c) Colluding attack

Figure 1: Various types of Byzantine attacks.

in the network. This is also a form of colluding attack. Figure 1b illustrates an example of a wormhole attack.

3.2.4. Colluding Attacks.

In a colluding attack [12], 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. Figure 1c shows an example of a colluding attack.

4. Proposed Secure Routing Protocol Based on Monitoring Scheme

In this section we describe our secure routing protocol.

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.
- A benign node generates one pair of public/private key.
- A key pair is kept secret by a benign node.
- Links are not stable, i.e., not all packets are received by neighboring nodes.
- All benign nodes know the link states of all neighbors.
- 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 (FIFO) 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. 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. 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.

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 the 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 4.3.3) 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 if 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.

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:

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

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.

4.3.3. 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 FIFO manner. 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.

Algorithm 1 Monitoring packet dropping

```
Input: $q$ : the probability of a packet being dropped
       $\alpha$ : 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: $n_d \leftarrow 0$;
 2: $j \leftarrow 1$;
 3: while $j \leq N$ do
 4:     The monitoring node observes how the monitored node
        handles a received packet not destined for himself;
 5:     $j \leftarrow j + 1$;
 6:     if The monitored node drops the received packet then
 7:         $n_d \leftarrow n_d + 1$;
 8:     end if
 9: end while
10: Calculate $p$ according to $(1)$;
11: if $p \leq \alpha$ then
12:     return Reject $H_0$;
13: else
14:     return Accept $H_0$;
15: end if
```

4.4. Preventing Byzantine Attack

In this section we explain how our monitoring scheme prevents Byzantine attacks.

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.

As shown in Figure 1b, if a node selected to take part in the routing of packets 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, 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.

4.4.3. Black Hole Attack.

If a malicious node *M* decides to drop or ignore packets, thereby carrying out a black hole attack. In this case, source node *S* and other neighbor nodes will not overhear the packet. After a predetermined time interval without the neighboring nodes overhearing the packet, the statistical method described earlier detects that node *M* is malicious. Then the links between node *S* and other neighboring nodes 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.

4.4.4. Preventing Colluding Attacks.

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* 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.

4.4.5. Preventing Intentional Packet Delay Attacks.

In this situation, a malicious node M deliberately delays packets in the network, the 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.

5. Evaluation

In this section, we evaluate our proposed monitoring scheme in terms of the security goals it achieves.

5.1. 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 the 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 destination node *D's* public key so each intermediate node on the route to the destination in the network cannot 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 ensures the integrity of the packets and the information in the packet history field.

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. 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, (ii) Packet history field monitoring and (iii) Statistical hypothesis testing. Also, our monitoring scheme uses RTS/CTS to identify when a node is intentionally delaying packets in the network. In addition, our monitoring scheme achieved relatively low communication overhead with an average of 124bytes packet history field size and 186bytes 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 16-S38, 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, *(IJCCIE 2016)*, Int'l Journal of Computing, Communications and Instrumentation Engineering, Vol. 3, Issue 1, 2016.

[2] 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.

[3] 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).

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.