Security-Aware Optimized Link Routing Protocol for Mobile Ad-Hoc Networks

Amandeep Dhir[1] and Jyotsna Sengupta[2]
[1]Research and Development Team, Newgen Software Technologies
New Delhi, 110020 – India
[2]Department of Computer Science, Punjabi University, Patiala
Punjab, 147001 – India
[e-mail: amannewgen@gmail.com]
*Corresponding authors: Jyotsna Sengupta

Received November 21, 2008; revised January 16, 2009; accepted January 28, 2009;
published February 23, 2009

Abstract

In this technical report, we have examined the basic building blocks of mobile ad-hoc networks. The paper discusses various security requirements of ad-hoc networks, attacks in ad-hoc networks, Security Implementation and Routing Protocols. The primary purpose of the paper is to address the Optimized Link State Routing (OLSR) protocol in detail, along with the various possible attacks. Finally, algorithms for securing OLSR are proposed, via the addition of digital signatures, as well as more advanced techniques such as cross checking of advertised routing control data with the node's geographical position. The main aim of this research work is the addition of security features to the existing OLSR protocol. In order to effectively design a secure routing protocol, we present a detailed literature survey of existing protocols, along with the various attacks. Based on the information gathered from the literature survey, a secure routing protocol for OLSR is proposed. The proposed secure routing protocol involves the addition of a digital signature as well as more advanced techniques such as the reuse of previous topology information to validate the actual link state. Thus, the main objective of this work is to provide secure routing and secure data transmission.

Keywords: Ad-hoc networks, simulation, threshold, secure communication, non-repudiation and routing

---

This research was funded by UGC (University Grants Commision) for development of efficient wireless networks at Department of Computer Science, Punjabi University, Patiala, India.

DOI: 10.3837/tiis.2009.01.003

1. Introduction

Security in mobile ad-hoc networks (MANETs) is an important issue that needs a solution. As much of the research has focused on the efficiency of the network, there are a number of routing protocols that have excellent efficiency. But the use of wireless links makes MANETs susceptible to attack. Eavesdroppers can access secret information, violating network confidentiality. Adversaries can directly attack the network, in order to delete messages, inject erroneous messages, or impersonate a node, which violates availability, integrity, authentication, and non-repudiation. Compromised nodes also can launch attacks from within a network. Security has radically changed the situation; the existing routing protocols are based on the assumption that the participating nodes of the network environment do not harm security. This is in stark contrast to reality. Existing secure routing protocols have the following disadvantages:

- Use of asymmetric encryption primitives too expensive for energy-constrained devices in an ad-hoc network. Hence, from the economic perspective, this is infeasible.
- Requiring global clock synchronization.
- Providing flat security services, i.e. not coping with the security requirements of the applications.

2. Security Requirements in MANETs

This section discusses why security is needed in MANETs, as well as various security issues and mechanisms for providing security. One of these mechanisms is used to secure the routing algorithms in MANETs. The use of wireless links makes MANETs susceptible to attack. Eavesdroppers can access secret information, thus violating network confidentiality. Adversaries can directly attack the network to delete messages, inject erroneous messages, or impersonate a node, which violates availability, integrity, authentication, and non-repudiation. Compromised nodes also can launch attacks from within a network. The fundamental aspects of computer security; confidentiality, integrity, authentication and non-repudiation, are also considered when the protection of routing in MANET networks is discussed [1].

Thus the MANET routing protocol must not be based on assumptions about availability of specific nodes at certain times. The routing framework must guarantee the robustness of the routing fabric, so that the services are available even in dynamically changing conditions where there is the possibility of compromised nodes. Moreover, the routing fabric must scale efficiently when the network topology changes, e.g. due to network partitions or mergers. The distribution of services is naturally an essential property of MANET networks. Mobile ad-hoc networks are highly dynamic; topology changes and link breakages frequently occur. Therefore, we need a security solution that is also dynamic. Any malicious or misbehaving nodes can generate hostile attacks. These types of attacks can seriously compromise basic aspects of security, such as the integrity, confidentiality and privacy of the node. Current ad-hoc routing protocols are completely insecure. Moreover, existing secure routing mechanisms are either too expensive or have unrealistic requirements.

Security requirements for ad-hoc routing protocols include [2]:

- Certain discovery, which means the route is always found if it exists between two nodes
- Isolating misbehaving nodes, which means ensuring that misbehaving nodes are always identified and isolated from routing.

- Location privacy, which means protecting information about the node location and network structure.

3. Attacks on Ad-Hoc Routing Protocols

Here we focus on attacks on the routing protocol in ad-hoc networks. These attacks may aim to modify the routing protocol so that traffic flows through a specific node controlled by the adversary. An attack may also aim to impede the formation of the network, make legitimate nodes store incorrect routes, and more generally cause perturbation of the network topology. Because of the specific architecture of ad-hoc networks, they are more vulnerable to attacks than wired networks. These attacks can be of two kinds; [3] Active and Passive attacks. The passive attacks aim to discover valuable information by listening to the traffic, while active attacks try to disrupt the operation of the MANET protocols. A good protocol must safeguard the network from malicious nodes, particularly in the route discovery phase. Various types of active attacks are briefly described below [4]:

3.1 Modifying Route Request Packets

In this type of attack the adversary tries to announce itself as having shorter routes to the destination. The shorter route can either be identified by the number of hops or announcing a better route metric in the reply packets to the sender, in the route discovery phase. Similarly an intruder can make itself a part of the route and start discarding traffic by employing a DOS (Denial of Service) attack for the packets received from the sender.

3.2 Spoofing

The adversary can merely start spoofing the valid IP addresses and isolate the nodes from the remainder of the network. This vulnerability is easily exploitable in many protocols.

3.3 Fabrication

The adversary can intentionally float error messages on the network, thus falsifying the existence of valid routes. An adversary can mount a replay attack by advertising stale routes and the adversary can even advertise a zero metric for all destinations, causing all nodes to route packets to it and thus creating a black hole.

3.4 Incorrect Traffic

This category includes attacks that involve sending false control messages: i.e. control messages sent on behalf of another node (identity spoofing), or control messages that contain incorrect or outdated routing information. The network may exhibit Byzantine behavior, i.e. conflicting information in different parts of the network. The consequences of this attack are degradation of network communications, unreachable nodes, and possible routing loops.

3.5 Relaying Incorrect Traffic

In this type of attack, network communications from legitimate, protocol-compliant nodes may be affected by misbehaving nodes.

4. Related Work

The Optimized Link State Routing (OLSR) Protocol [5] is a proactive routing protocol, so the routes are always immediately available when needed. OLSR is an optimization version of a pure link state protocol. So the topological changes cause the flooding of the topological information to all available hosts in the network. In order to reduce the possible overhead in the network, the protocol uses Multipoint Relays (MPR). The principle of MPR is to reduce flooding of broadcasts by reducing the same broadcast in some regions in the network. Another method of reduction is to provide the shortest path. Reducing the time interval for the control messages transmission can result in more reactivity. OLSR uses two kinds of control messages: HELLO and Topology Control (TC). HELLO messages are used for finding information about the link status and host's neighbors. With the HELLO message the Multipoint Relay (MPR) Selector set is constructed, which describes which neighbors have chosen this host to act as MPR. Based on this information the host can calculate its own set of MPRs. The HELLO messages are only sent one hop, but the TC messages are broadcast throughout the entire network. TC messages are used for broadcasting information about their own advertised neighbors, which includes at least the MPR Selector list. The TC messages are broadcast periodically and only the MPR hosts can forward the TC messages. There is also the Multiple Interface Declaration (MID) message, which is for informing other hosts that the announcing host can have multiple OLSR interface addresses. The MID message is broadcast throughout the entire network only by MPRs. There is also the Host and Network Association (HNA) message, which provides external routing information by enabling routing to external addresses. The HNA message provides information about the network- and net mask addresses, so that the OLSR host can consider that the announcing host can act as a gateway to the announcing set of addresses. The HNA is considered as a generalized version of the TC message, where the only difference is that the TC message can inform about route canceling, while HNA message information is removed only after the expiration time.

4.1 Limitations of OLSR

Security measures were not included in its design, like in other primitive routing protocols. An adversary performing identity spoofing or message replay needs to change the Message Sequence Number field of the spoofed or replayed message. Otherwise, nodes that have already received a message with the same originator and MSN (according to their Duplicate Set) will drop the malicious message. Furthermore, accepting the malicious message causes message loss when a legitimate message with the same originator and MSN is received by the victim nodes, and dropped according to the protocol.

The various types of attacks on OLSR protocol are as follows [6]:

1. **Incorrect traffic generation:** One way in which a node can misbehave is by generating control messages in a way that does not conform to the protocol. Incorrect message generation can be due to an incorrect HELLO message generation, incorrect TC message generation, incorrect MID/HNA message generation and ANSN attack
2. **Incorrect traffic relaying:** If control messages are not properly relayed, network malfunctions are possible. Incorrect traffic relaying may be due to a Blackhole attack, Replay attack, Wormhole attack and MPR attack

Cedric, D. Raffo and P. Muhlethaler [6], examined the security issues, and described the architecture for securing mechanisms. They provided details on the types of attacks prevented by this architecture, along with details about protocols, algorithms, mechanisms and implementation.

5. Security Implementation Issues in MANETs

Securing wireless ad hoc networks is particularly difficult for many reasons, including the following:

- Vulnerability of channels. As in any wireless network, messages can be eavesdropped and fake messages can be injected into the network, since there is no need to have physical access to network components.
- Vulnerability of nodes. Since the network nodes do not usually reside in physically protected locations, such as locked rooms, they can be more easily captured and fall under the control of an adversary.
- Absence of infrastructure. Ad-hoc networks are supposed to operate independently of any fixed infrastructure. This means that the classical security solutions based on certification authorities and on-line servers are inapplicable.
- Dynamically changing topology. In mobile ad-hoc networks, permanent changes of topology require sophisticated routing protocols, for which security is an additional challenge. A particular difficulty is that incorrect routing information can be generated by compromised nodes or as a result of various changes of topology, and it is hard to distinguish between the two cases.

5.1 Basic Security Mechanisms:

The various types of mechanisms [7] that can be used for providing security in the routing algorithms in MANETS are as follows:

5.1.1 Cryptography

Cryptography [7] is the heart of security. If privacy is required, we need to encrypt our message at the sender site and decrypt it at the receiver site. The original message, prior to transformation, is called plaintext. After the message is transformed, it is called cipher text. An encryption algorithm transforms the plaintext to cipher text, whereas a decryption algorithm transforms the cipher text to plaintext. The sender uses an encryption algorithm and the receiver uses a decryption algorithm. A key is a number (value) that the cipher, as an algorithm, operates on. In order to encrypt a message, we need the encryption algorithm, encryption key and plaintext. These create the ciphertext. In order to decrypt a message, we need the decryption algorithm, decryption key and cipher text. Cryptographic Algorithms can be studied as Symmetric-Key Cryptography and Public-Key Cryptography.

5.1.2 Cryptographic Hash Functions

Secure communication requires that data transmitted is not altered by any entity. Hash functions are the security primitives that ensure data integrity. The hash function is often called a one-way hash function, because it is quite difficult to compute the inverse function. For example, for the cube function $y = x^3$, it is quite easy to compute y given x. But the inverse function, *3xy*, is much more complicated to compute. The most common use of a hash function is to create a digital signature and provide data integrity. It is also used for entity authentication [6]. For a digital signature, a hash function is applied to the entire message. Then the hashed value is signed. After reception, the hash value is recomputed and it is verified that the received signature is unaltered from the original source. This saves both time and space, as only the hashed value is signed, instead of the entire message. This is widely used to provide data integrity. The sender computes the hashed value over the data and sends it along with the original message to the receiver. The destination entity computes the hash value from the transmitted message and compares it with the hashed value (transmitted). The hash function can be public (no key) or it can contain a key. The most common hash functions are MD5 (Message Digest 5) and SHA (Secure Hash Algorithm).

5.1.3 Digital Signatures

A digital signature is an important cryptographic primitive used for authentication, authorization and non-repudiation [7]. A digital signature is the best use of public key cryptography. An asymmetric encryption algorithm such as RSA can be used to create and verify a digital signature. In practice, digital signature creation and verification are performed using the combination of a hash function and asymmetric encryption. In order to create a digital signature the sender first computes the message authentication code (MAC) or hash of the original message and appends the code to the message. Then the hash code is encrypted using asymmetric encryption. At the reception end the receiver uses the same hash algorithm to compute the hash code of the message, decrypts the encrypted message using the corresponding public key, and compares the hash value.

6. Routing Protocols

In this section, the routing protocols used in conventional networks and ad-hoc networks are explained. Also, routing algorithms for ad-hoc networks and their disadvantages from the security perspective are discussed.

6.1 Basic Routing Schemes

Routing protocol schemes are effectively distributed database systems. They propagate information about the topology of the network among the routers within the network. Each router in the network then uses this distributed database to determine the best loop free path through the network to reach any given destination. There are two fundamental ways [7] to distribute the data through a network:

By distributing vectors, each router in the network advertises the destinations it can reach, along with information that can be used to determine the best path to each reachable destination. There are two types of vector-based protocols: distance vector and path vector.
By distributing the state of the links attached to the routers; each router floods (advertises to all other routers in the network, whether directly adjacent or not), the state of each link to which it is attached. This information is used independently by each router within the routing domain to build a tree representing the topology of the network (the shortest path tree). Routing protocols that distribute the state of attached links are called link state algorithms. In the subsequent sections, we discuss these two schemes and various new protocols exclusively designed for ad-hoc networks

6.2 Classification of Routing Algorithms for MANETs

Efficient routing of packets is a primary MANET challenge [8]. Conventional networks typically rely on distance-vector or link-state algorithms, which depend on periodic broadcast advertisements of all routers to keep routing tables up-to-date. In some cases, MANET also uses these algorithms, which ensure that the route to every host is always known. However, this approach has several problems:

- Periodic updating of the network topology increases bandwidth overhead;
- Repeatedly awakening hosts to receive and send information quickly exhausts batteries.
- The propagation of routing information, which depends on the number of existing hosts, causes overloading, thereby reducing scalability.

- Redundant routes accumulate needlessly; and communication systems often cannot respond to dynamic changes in the network topology quickly enough.

MANETs use multi-hop rather than single-hop routing to deliver packets to their destinations. These routing protocols [9] may generally be categorized as:

- Table–driven protocols
- Source-initiated (demand-driven) protocols
- Hybrid protocols

7. Theoretical Formulation of Protocol

After discussing the basis of mobile ad-hoc networks we explore the routing protocol under investigation. The objective is to achieve basic understanding of the optimized link state protocol, limitations of OLSR, and our proposed changes to it, for achieving a secure protocol.

7.1 OLSR Packet Format

OLSR control messages are communicated using a transport protocol defined by a general packet format, given in **Fig. 1**. Each packet encapsulates several control messages into a single transmission. Control traffic in OLSR is exchanged through two different types of messages: HELLO and TC (Topology Control) messages. HELLO messages are exchanged periodically among neighbor nodes, in order to detect links to neighbors and signal MPR selection. TC messages are periodically flooded to the entire network, in order to diffuse link state information to all nodes. The other OLSR control messages are MID (Multiple Interface Declaration) and HNA (Host and Network Association). MID and HNA messages are only emitted by nodes that have multiple interfaces. To avoid collisions, the OLSR protocol adds a certain amount of jitter to the interval in which all control messages are generated.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Packet Length         |    Packet Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
```

Fig. 1. OLSR Packet Format

The various important constituents of the OLSR Packer format are as follows:

1. Packet Length: The length, in bytes, of the entire packet, including the header.
2. Packet Sequence Number: Incremented by one each time a new OLSR message is transmitted. A separate Packet Sequence Number is maintained for each interface, so that packets transmitted over an interface are sequentially enumerated.
3. Message Type: Integer identifying the type of message. Message types of 0-127 are reserved by OLSR, while the 128-255 space is considered private and can be used for custom extensions of the protocol.
4. Vtime: A field indicating the interval after reception for which a node will consider the information contained in the message as valid. The time interval is represented in a mantissa-exponent format.
5. Message Size: The size of the message, including the message header, in bytes.
6. Originator Address: The main address of the originator of the message.
7. Time To Live: The maximum number of hops the message can be forwarded. Using this field one can control the radius of flooding. Before a message is retransmitted the time-to-live must be decremented by one. When a node receives a message with a time-to-live equal to zero or one, the message must not be retransmitted under any circumstances. Normally, a node does not receive a message with a TTL of zero.
8. Hop Count: A field indicating the number of times the message has been forwarded. Initially, this is set to zero by the originator of the message.
9. Message Sequence Number: This is used to ensure that a given message is not transmitted more than once by any node. While generating a message, the originator node assigns a unique identification number to each message. This number is inserted into the sequence number field of the message. The sequence number is incremented by one for each message originating from the node.

While messages may potentially be broadcast to the entire network, packets are only transmitted between neighbor nodes. The unit of information forwarded is called a message. An individual OLSR control message can be uniquely identified by its Originator Address and Message Sequence Number (MSN), which are both from the message header. The Originator Address field specifies the originator of a message. This does not change as the message is relayed around the network; the address contained in this field differs from the IP header source address (except at the first hop, when the message is created), which is changed at each hop to the address of the retransmitting node.

A node may receive the same message several times. Therefore, to avoid processing and sending the same message multiple times, the node records information about each received message. This information is stored in a tuple consisting of the message's originator address, the MSN, a Boolean value indicating whether the message has already been retransmitted, the list of interfaces for which the message has been received, and the tuple's expiration time. All tuples are maintained in the Duplicate Set (Duplicate Table) of the node.

The common packet format allows individual messages to be piggybacked and transmitted together in one emission, if the MTU size can support this. Therefore different kinds of control messages can be emitted together, although they are processed and forwarded differently in each node; e.g. HELLO messages are not forwarded, while all other control messages are. OLSR does not handle unicast communications: a message from a node is either transmitted to all its neighbors or to all nodes in the network.

7.2 OLSR Control Traffic

Control traffic in OLSR is exchanged through two different types of messages: HELLO and TC messages. HELLO messages are exchanged periodically among neighbor nodes, in order to detect links to neighbors, detect the identity of neighbors and signal MPR selection. TC messages are periodically flooded to the entire network, in order to signal link state information to all nodes.

HELLO Messages: HELLO messages are emitted periodically by a node, including its own address, as well as encoding three lists: a list of neighbors, from which control traffic has been heard (but where bi-directionality is not yet confirmed), a list of neighbor nodes with which bi-directional communication has been established, and a list of neighbor nodes that have been selected to act as MPR for the originator of the HELLO message. HELLO messages are only exchanged between neighbor nodes. After receiving a HELLO message, a node examines the lists of addresses. If its own address is included, it is confirmed that bi-directional communication is possible between the originator and recipient of the HELLO message. When a link is confirmed as bi-directional, this is advertised periodically by a node with a corresponding link status of symmetric. The HELLO message format is shown in Fig. 2.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Reserved             |     Htime     |  Willingness  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Link Code   |   Reserved    |       Link Message Size       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Neighbor Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Neighbor Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                             . . .                             :
 :                                                               :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Link Code   |   Reserved    |       Link Message Size       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Neighbor Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Neighbor Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                                                               :
```

Fig. 2. HELLO Message Format

The main constituents of the Message Format are:

1. Reserved field must be set to 0000000000000 to comply with the specification.
2. Htime field specifies the HELLO emission interval used by the node for the particular interface i.e. time before the transmission of the next HELLO.
3. Willingness field specifies the willingness of a node to carry and forward traffic for other nodes. A node with willingness WILL_NEVER must never be selected as MPR by any node. A node with willingness WILL_ALWAYS must always be selected as MPR. By default, a node must advertise a willingness if WILL_DEFAULT.
4. Link Code field specifies information about the link between the interface of the sender and the following list of neighbor interfaces. It also specifies information about the status of the neighbor.
5. Link Message Size contains the size of the link message, in bytes, from the beginning of the Link Code field until the next Code Field.

6. Neighbor Interface Address field contains the address of an advertised neighbor.

In addition to information about neighbor nodes, periodic exchange of HELLO messages allows each node to maintain information describing the links between neighbor nodes and nodes that are two hops away. This information is recorded in a node's 2-hop neighbor set and is utilized for MPR optimization.

TC Messages: Diffuse link state information, i.e., information about the last hop, to the entire network. A TC message contains a set of symmetric neighbors (i.e. neighbors which have at least one symmetrical link to the originator of the TC message), each one contained in an Advertised Neighbor Main Address field. TC messages are periodically flooded to the entire network, exploiting MPR optimization. Only nodes that have been selected as an MPR generate (and relay) TC messages.

The TC message includes an ANSN field that contains the Advertised Neighbor Sequence Number. This number is associated with the node's advertised neighbor set, and is incremented each time the node detects a change in this set. The TC message format is shown in **Fig. 3**, which shows the reserved bits and main address clearly.

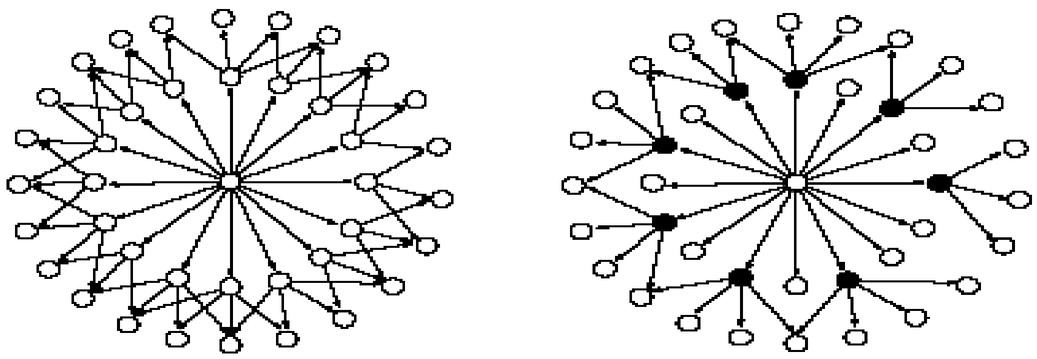
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ANSN             |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Advertised Neighbor Main Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Advertised Neighbor Main Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 3. TC Message Format

MID Messages: Are only emitted by a node with multiple OLSR interfaces, in order to announce information about its interface configuration to the network. A MID message contains a list of addresses, each address belonging to an OLSR interface of the sending node.

HNA Messages: Are only emitted by a node with multiple non-MANET interfaces. They provide connectivity from an OLSR network to a non-OLSR network. The gateway sends HNA messages containing a list of addresses of the associated networks and their net masks.

Multipoint Relay Selection and Signaling: The core optimization in OLSR is that of Multipoint Relays (MPRs). The principle is as follows: each node must select MPRs from among its neighbor nodes such that a message emitted by a node and repeated by the MPR nodes will be received by all nodes two hops away. MPR selection is based on the 2-hop neighbor set received through the exchange of HELLO messages, and is signaled through the same mechanism: a link status of MPR specifies that the link between the originator of the HELLO message and listed address is symmetric, and that the node with the included address is selected as MPR by the originator. Thus, each node maintains an MPR selector set, describing the set of nodes that have selected it as MPR. After receiving an OLSR control message, a node checks the MPR selector set to determine whether the message is to be retransmitted: if the last-hop of the control message is an MPR selector, then the message is retransmitted – otherwise it is not retransmitted. Fig. 4 shows a node with neighbors and 2-hop neighbors. In order to achieve a network-wide broadcast, it suffices that a broadcast transmission is repeated by a subset of the neighbors. This subset comprises the MPR-set of the node [10]. MPR selection is based on the 2-hop neighbor set received through the exchange of HELLO messages, and is signaled through the same mechanism. Each node maintains an MPR selector set, describing the set of nodes that have selected it as MPR.



(a) The case where all neighbors retransmit a broadcast

(b) The case where only the MPRs of a node retransmit the broadcast

Fig. 4. Two hop neighbors and multipoint relays (solid circles) of a node

7.3 Vulnerabilities in OLSR

Here we discuss various security vulnerabilities in the proactive routing protocol for ad-hoc networks. These are examples of vulnerabilities common to all proactive routing protocols. One vulnerability, common to all routing protocols operating a wireless ad-hoc network, is jamming - i.e. a node generates massive numbers of interfering radio transmissions, which restricts legitimate traffic (e.g. control traffic for the routing protocol as well as data traffic) on part of the network. This vulnerability cannot be dealt with at the routing protocol level (if at all), thus, the network is unable to maintain connectivity. Jamming is somewhat similar to that of network overload: a sufficiently significant amount of routing protocol control traffic is lost. Our assumption is that the routing protocol can consistently provide a correct view of the network topology in each network node. This assumption implies that all nodes in the network correctly implement the routing protocol-specifically that each node correctly processes and emits control traffic. Thus an attack on the ability to provide connectivity in the network must result from the incorrect behavior of at least one node in the network. In this context, incorrect means that the node does not process and emit control traffic in accordance with the routing protocol specifications. We note that in most cases such non-conforming behavior of a node is due to malice - i.e. specially targeted to interfere with network connectivity. The node responsible for this incorrect behavior may be either an intruder (i.e. a node that is not supposed to be in the network) or a compromised node (i.e. a node, that is supposed to be in the network, but which has been modified to be non-conforming with the routing protocol). We also note that non-conforming behavior of a node may be without malice - e.g. due to a simple malfunction of a node. When an ad-hoc network is operating under a proactive routing protocol, each node has two different (related) responsibilities. Firstly, each node must correctly generate routing protocol control traffic conforming to the protocol specification. Secondly, each node is responsible for forwarding routing protocol control traffic on behalf of other nodes in the network. Thus, incorrect behavior of a node can result either from a node generating incorrect control messages or from incorrect relaying of control traffic from other nodes. This is illustrated in **Fig. 5**.

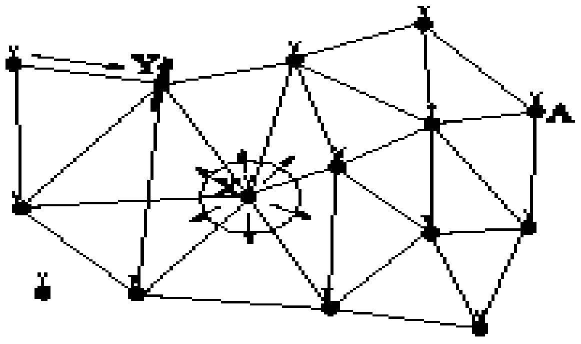


Fig. 5. Two kinds of attacks in a proactive routing protocol: node X generates incorrect information (e.g. advertises node A as a neighbor) while node Y does not relay control traffic for other nodes.

Here we investigate how these incorrect behaviors appear in OLSR. We note that while we employ OLSR in our investigations, much of the following also applies to other proactive routing protocols.

7.3.1 Incorrect Control Traffic Generation

OLSR basically employs two different kinds of control traffic: HELLO messages and TC messages. Here we describe how a non-conforming node may affect the network connectivity through incorrect generation of HELLO and TC messages. In general, in terms of control traffic generation, a node may misbehave in two different ways: by generating control traffic impersonating another node (i.e. Identity Spoofing) or by advertising incorrect information (links) in the control messages (i.e. Link Spoofing).

7.3.2 Incorrect HELLO Messages

Identity Spoofing: A node may send HELLO messages impersonating the identity of another node. e.g node X sends HELLO messages with the originator address set to that of node A, as illustrated in **Fig. 6**. This may result in the network containing conflicting routes to node A. Specifically; node X will choose MPRs from among its neighbors, signaling this selection impersonating the identity of node A. The MPRs subsequently advertise that they can provide a last hop to node A in their TC messages. This may result in conflicting routes to node A, with possible loops.

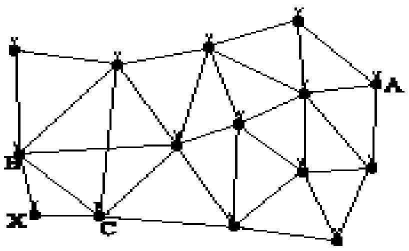


Fig. 6. Identity Spoofing of HELLO messages: node X assumes the identity of node A for sending HELLO messages. Nodes B and C may subsequently announce reachable to node A through their TC messages.

Link Spoofing: A node may send HELLO messages, signaling an incorrect set of neighbors. This may take either of two forms: if the set is incomplete, i.e. a node ignores some neighbors; the network may not have connectivity to these ignored neighbors. Alternatively, a compromised node advertising a neighbor relationship to non-present nodes may cause inaccurate MPR selection, and as a result, some nodes may not be reachable in the network.

7.3.3 Incorrect TC Messages

Identity Spoofing: A node may send TC messages impersonating the identity of another node. Effectively, this implies link spoofing, since a node assuming the identity of another node effectively advertises incorrect links to the network. Link Spoofing: A node may send TC messages advertising an incorrect set of links. This may take either of two forms: if the set is incomplete i.e. a node ignores links to some nodes in its MPR selector set, the network may not have connectivity to these ignored neighbors, nor to neighbors which are reachable only through the ignored neighbors. A node may also include non-existing links (i.e. links to non-neighbor nodes) in a TC message. This is illustrated in **Fig. 7**. Link spoofing in TC messages may yield routing loops and conflicting routes in the networks.

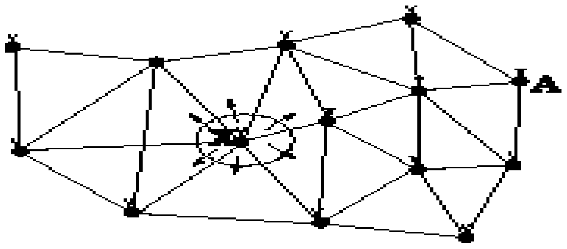


Fig. 7. Node X generates incorrect TC messages, e.g. advertising a link between node X and node A

7.3.4 Incorrect Control Traffic Relaying

If TC messages (or routing protocol control messages in general) are not properly relayed, a loss of connectivity may occur. In networks where there is no redundancy e.g. a strip network, a loss of connectivity will surely occur, while other topologies may provide redundant connectivity. Similarly if a node does not forward data packets e.g. if intra-node forwarding is impaired, a loss of connectivity may occur.

8. Algorithm for Securing OSLR Protocol

OLSR is a proactive routing protocol designed specifically for large and dense networks. OLSR provides optimal routes in terms of the number of hops. There are various possible attacks in OLSR that can hinder the proper functioning of the protocol. In this chapter, we present a framework that enables OLSR to resist the various attacks discussed in the previous chapter. The security architecture proposed is mostly cryptography-based i.e. a few constraints are enforced for the cryptographic system employed to secure the OLSR. Any cryptographic system satisfying the following two requirements may be employed:

- A signature for a message can be generated in a node using a function sign i.e., node id, key, and message.
- A signature for a message can be verified using verification i.e., originator id, key, message, and signature.

8.1 OLSR Signature Message

We have designed an infrastructure [6][11] to protect OLSR. To prevent malicious nodes from injecting incorrect information into the OLSR network, an additional security element called a signature is generated by the originator of each control message, and transmitted with the control message. A timestamp is associated with each signature, in order to estimate message freshness. Thus, after receiving the control message, a node can determine whether the message originates from a trusted node, or whether message integrity is preserved.

Signatures are inherently, separate entities from OLSR control traffic: while OLSR control messages fulfill the goal of acquiring and distributing topological information, signatures serve to validate information origin or integrity. For this reason, a signature is implemented as a separate type of OLSR message (a SIGNATURE message), instead of appending the timestamp and signature to the control message. The resulting signature message is considered and handled in the same manner as any other OLSR standard message. Furthermore, while this implementation slightly increases the total message size, it does not require great modifications to the standard OLSR protocol, as it uses the standard format for the control messages. It is mainly used to prevent the injection of incorrect information in the network.

8.1.1 Specifications of Signature Message

For each control message i.e. HELLO, TC, MID or HNA, generated, a corresponding SIGNATURE message is generated, which is sent in the same packet as the one containing the control message, immediately before it. Signatures are used by a receiving node to authenticate the corresponding OLSR control message: every control message without a matching corresponding signature is dropped. In this architecture, a signature corresponds to a message, rather than an entire packet. It is not possible to sign or digest an entire OLSR packet, because it may change in transit from one node to another. This is because a packet may contain TCs, which are flooded in the network, as well as HELLOs, which are not forwarded further. Hence, after a few hops, the packet might no longer have a valid signature, because it was computed for the original packet. The control message and its SIGNATURE message are sent in the same OLSR packet, in order to simplify handling of the messages: the packet first contains the SIGNATURE message, followed immediately by the corresponding control message. If these messages were not sent in the same packet, their order of arrival could not be guaranteed. Therefore each node would need a buffer to temporarily store them after reception, before trying to couple them.

8.1.2 Format of the Signature Message

The SIGNATURE message is encapsulated and transmitted as the data portion of the standard OLSR packet format. The Message Type field is set to the SIGNATURE constant value; this value may also include information about the cryptographic primitives and keys used. The Time To Live and Vtime fields are set to the values of the Time To Live and Vtime fields of the message associated with the signature. The other fields of the message header are set as usual. An old version [11] of the SIGNATURE message is shown in **Fig. 8**. The message contains a MSN Referrer field, in order to identify the bijection between a control message and its SIGNATURE message.

The *Sign. Method* field specifies which method from a predefined set is used to generate the signature. This includes information about keys, cryptographic functions, and timestamp algorithms. The *Reserved* field is used for padding, to ensure that all fields are 32-bit aligned. It is set to zero and reserved for future use.

The *MSN Referrer* field of the SIGNATURE message contains the *Message Sequence Number* of the control message associated with this signature. The correspondence achieved by the Message Sequence Number is unique only if possible overflow and wraparound of the 16-bit field is disregarded; however this is not a problem, since a node uses further signature (and timestamp) verification to check the correspondence between the control message and signature message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign. Method  |   Reserved    |         MSN Referrer          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Signature                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 8. Old SIGNATURE message format

The *Timestamp* and *Signature* fields are the same as those in the actual version of the message. The implementation of the aforementioned approach makes it unnecessary to send the SIGNATURE message and its associated control message in the same packet, as the messages can be reordered and re-associated later. However, this means that every node would need to store the received messages i.e., control and signature messages in a buffer. This requires additional system resources and is more prone to failure and DoS attacks (for control messages for which the signature is lost, or vice versa). Furthermore, the delay is unfavorable when a message and its signature are not aggregated in the same packet. Thus, this approach was abandoned and a simplified version was designed, as explained below, by merging the different fields.

The actual format of a SIGNATURE message is specified in **Fig. 9**.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Signature                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 9. SIGNATURE Message Format

The *Timestamp* field contains the timestamp itself, in seconds. This is the timestamp of both the SIGNATURE message and associated control message. For compatibility reasons, the size of the timestamp is 32 bits. This represents the standard Unix time, which is encoded in a 32-bit signed integer2 data type. The current time is obtained from the node's internal BIOS clock. The *Signature* field contains the signature, computed for the sequence of bits comprising the following fields:

- The message header (80 bits) of the control message, excluding the *Time To Live* and *Hop Count* fields. These fields are not considered in the signature computation, because they are modified while the message is in transit (the *Time To Live* is decremented by one and the *Hop Count* is incremented by one at each hop). Subsequently, the signature of the message would be invalidated;
- The control message (variable size);
- The message header (80 bits) of the SIGNATURE message, excluding the *Time To Live* and *Hop Count* fields;
- The *Timestamp* field (32 bits).

8.1.3 The Timestamp Field

A common problem in distributed systems with non-secure communication channels is that even when it is assumed that a signature message is checked, it is possible for an intruder to replay previously transmitted messages. For instance, an intruder in a network may replay one day of control messages. If they cannot be identified as old they will be accepted as valid, as they are properly signed. This may easily disrupt the protocol functioning and thus the network integrity. Timestamps are a commonly used means to prevent replay attacks. They provide the proof of freshness, so that older pieces of information can be detected and rejected. The criterion for a stale timestamp is:

$$| \text{Timestamp} - t_0| \leq \Delta t$$

where t_0 is the current time at the receiving node and Δt is the accepted value for discrepancy, including the difference in the synchronization of clocks. A strict clock synchronization of the nodes is not necessary; the timestamp is used to disambiguate possible wraparound of the Message Sequence Number. The replay check in the above expression can be complemented, in order to prevent replays over a small time scale i.e. replays over a delay of less than Δt, by maintaining a signature table. The signature table contains the signatures of the most recently received messages, for an interval greater than or equal to Δt. If the signature of a received message is already in the signature table it is ignored, since the message has already been received and processed.

This security architecture [6] relies on the use of asymmetric cryptography. An offline Certification Authority has the duty of assigning an identity-based key pair for each participating node. Before joining the network, a node contacts the Certification Authority through a secure channel, and obtains a global key. The node also generates a key pair, and diffuses its public key (local key) through the network via a specific key exchange protocol: it originates Key Distribution messages, signed with its global key, which are spread by pure flooding. Subsequently, the node uses its local key to sign its control messages.

8.2 Modifications to the Standard OLSR Protocol

This security architecture is not interoperable with the standard OLSR. The non-secured nodes, which are unable to check the signature, because of limited computing power or lack of knowledge of the key, could simply drop SIGNATURE messages after reception. However, their unsigned control messages would be dropped by secured nodes. So securing the OLSR protocol involves modifying some parts of its basic functioning. The modifications made to the standard OLSR are as follows:

8.2.1 Sending a Signed Control Message

In brief, to compute a signature corresponding to a control message, the following protocol is used:

1. The node creates the control message;
2. The node retrieves the current time, and puts it in the Timestamp field;
3. The node computes the signature, and puts it in the Signature field;
4. The node puts the SIGNATURE message and control message in the packet, in this exact order.

Then, the node sends the packet, or repeats the protocol for another control message before sending the packet.

8.2.2 Changes to the Duplicate Set

The Duplicate Set of the standard OLSR is modified to include a new field *D_timestamp*. This field stores the value of the Timestamp field, once the match between the SIGNATURE message and control message has been found. The *D_timestamp* field is filled with the same value for the control message and it's SIGNATURE. Incoming messages are recorded in the Duplicate Set as usual.

8.2.3 Receiving and Checking a Signed Control Message

After receiving a control message with its SIGNATURE message, a node processes both. The outline of protocol is given below:

1. The node processes the SIGNATURE message, checks the timestamp, and keeps the SIGNATURE in memory;
2. The node checks the signature of the control message;
3. If the timestamp is fresh and the signature is valid, the control message is accepted and processed according to the standard OLSR specifications for the message type. Otherwise, both the control message and SIGNATURE message are dropped.

To fit the secured infrastructure, some modifications also need to be made to the packet processing algorithm described in the standard specifications [10]. We briefly describe these modifications. A receiving node must process an incoming packet according to the following algorithm:

- If the packet contains no messages, silently drop the packet; (As in standard OLSR)
- If the TTL of the message is ≤ 0 or if the message was sent by itself, silently drop the packet.

The various processing conditions are as follows:

a. If there exists a tuple in the Duplicate Set where *D_addr = Originator Address* and *D_seq_num = Message Sequence Number* and *D_timestamp = Timestamp* then do not process this message, because it has already been processed.

b. Else process the message according to its Message Type. If the message is a SIGNATURE, then
   - If the timestamp (from the *Timestamp* field) is fresh, then maintain the SIGNATURE message (with its header) in memory. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set;
   - Else if the message is of another Message Type which has already been implemented, then
   - If the *Message Sequence Number* of the message = *Message Sequence Number* of the SIGNATURE in memory +1, then continue. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set; (This step is optional)
   - If the computed signature (from the Signature field) is valid, then flush the SIGNATURE message from memory, and process the message according to the

standard OLSR specifications. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set.

The forwarding condition is as follows

- If there exists a tuple in the Duplicate Set where *D_addr = Originator Address* and *D_seq_num = Message Sequence Number* and *D_timestamp = Timestamp* and the receiving interface address is listed in *D_iface_list* then do not retransmit this message, because it has already been considered for forwarding;
- Else forward the message according to its Message Type, or to the standard forwarding algorithm if its Message Type is not implemented. (As in standard OLSR)

Erasing the Duplicate Tuple corresponding to bad messages i.e. with a stale timestamp or an invalid signature, ensures that only good messages in the Duplicate Set are tracked. This is to avoid a DoS attack from a malicious node that floods the network with junk messages not coupled to a signature message (or coupled to an invalid signature message). These junk messages fill the Duplicate Set of receiving nodes, which causes receiving nodes to reject valid messages containing the same MSN as a previously received junk message.

Table 1. Protection against various OLSR attacks by SIGNATURE message

Attack on OLSR Protocol			Protection by SIGNATURE
Incorrect Traffic Generation	Incorrect HELLO Generation	ID Spoofing	Yes
		Link Spoofing	No
	Incorrect TC Generation	ID Spoofing	Yes
		Link Spoofing	No
	Incorrect MID/HNA Generation		Yes
	ANSN Attack		Yes
Incorrect Traffic Relaying	Message Tampering		Yes
	Black hole Attack		No
	Replay Attack		Yes
	Wormhole Attack		No
	MPR Attack		No

8.3 Requirements of Signature Algorithm

Generally, it is desirable that the signature algorithm used in ad-hoc networks has these characteristics:

- A short signature (in bits), to minimize the message overhead;
- A fast signature verification time, to prevent an intruder from performing a DoS attack merely by sending a large number of false signatures.
- Faster verification than signing, because a message generated and signed by a single node has to be verified by several (or all) nodes in the network.
- Low complexity, because of the CPU power limitations of nodes in MANETs.

8.3.1 Use of Multiple Signatures in OLSR

To resist the link spoofing attack, a protocol was designed that uses multiple signatures (generated by different nodes) to validate link state information [12]. This protocol relies on creating and sending a new additional message (in conjunction with routing control messages) called an ADVSIG (for ADVanced SIGnature) message. This approach is based on authentication checks of new information injected into the network, and reuse of this information by a node, to prove its link state at a later time. In general, HELLO and TC control messages have the semantics of the originator advertising, “I have a link with these other neighbor nodes”. The signature for these messages, introduced in the previous section, serves to verify that the originator is indeed the one claiming that such a link exists. The task is then to validate that the other nodes also believe that such a link exists. This solution does not require modification to the standard OLSR control messages.

In OLSR, the network topology is related to the network topology at the previous instant. This is because the link state at a given time t depends on the link state at the time immediately preceding it; $t - \Delta t$. We can use this fact to avoid the injection of false routing information into the network. The rationale is that every node stores the most recent link state information about itself, as received by its neighbors, then, the node reuses this information by including it as a proof in its control messages. By this means a node can prove that it supplies routing information that is in accordance with and consistent with its previous neighborhood status.

8.3.2 Link Atomic Information

It would be inefficient to sign and redistribute an entire HELLO message as a proof, because each HELLO contains many links related to many nodes. As OLSR control messages are not modified, this data must be split into reusable pieces of information. In order to keep the protocol as light and simple as possible, a minimum of exchanged link state information must be identified. The *link atomic information* generated by a node A concerning a neighbor node B consists of:

- The address of A as the originator node
- The address of B as the advertised node B
- B’s link state with respect to A
- The timestamp of the creation time
- The signature (computed by A) of these four fields

The address of the originator node is found in the message header as the Originator Address field, and is part of the standard packet. The address of the advertised node and its link state are exchanged through a HELLO message, respectively, in the Neighbor Interface Address and Link Code fields. The timestamp and the signature are contained in the ADVSIG message coupled to the HELLO. Depending on its use, this atomic information is called either a *Certificate* or a *Proof.*

Hence, after reception of a HELLO and its companion ADVSIG message, a node extracts the information about itself from them (i.e. where the advertised field contains the node’s address). When used in this manner, the aforementioned atomic information is called a *Certificate*. The Certificates are stored by the node in a Certiproof Table. Subsequently, when the node sends a HELLO or TC message, it selects the relevant Proof from its Certiproof Table and includes it in the ADVSIG message coupled to that HELLO/TC message.

The same atomic piece of information is called a Certificate when it is created and supplied in order to inform about the neighborhood, as fresh reusable topology information; it is called a Proof when it is reused and supplied in order to prove a link state. The Certiproof Table of node B only contains Certificates signed by various neighbors of B; in each of these Certificates, the advertised field contains the address of B.

8.3.3 Required Proofs

As mentioned, if a node wishes to report a link with a neighbor node in a HELLO/TC message, the required proof must be built using elements of the HELLO message and the accompanying ADVSIG message that were recently sent by the node . The proofs are then stored (as Certificates) in the Certiproof Table and reused (as Proofs) whenever necessary. The proofs must be sent in a new companion ADVSIG message, with the new HELLO/TC messages they are intended to prove.

8.3.4 Certiproof Table

When a node B receives a HELLO and its accompanying ADVSIG message from A, it extracts any information about itself from them, and stores the tuple (i.e, originator, advertised, link state, timestamp, and signature) in its Certiproof Table. This tuple is later resent by B as a Proof, but the same information is called a Certificate when sent by A. As explained, originator contains the address of A, advertised contains the address of B, link state is B's link state with respect to A, timestamp is the time when A generated the HELLO and ADVSIG messages, and signature is the signature computed by A for the four fields originator, advertised, link state and timestamp. The key of the tuple is the originator address. Only one tuple for each originator is maintained in the table: when B receives a subsequent HELLO message (with its ADVSIG) from A, it updates the tuple entry with the freshest information, which is determined by comparing the timestamp fields. In this manner, node B only stores the most recent Certificate about itself in the Certiproof Table, as given by a neighbor.

8.3.5 ADVSIG Message

The format of this security-enhanced ADVSIG message is shown in **Fig. 10**. An ADVSIG message must be generated and sent with every HELLO or TC message, possibly in the same packet. However, there is a difference between HELLOs and TCs: while both message types always require Proofs, HELLOs can contain Certificates whereas TCs do not. Hence the Signature of Certificate #i fields only exist in ADVSIG messages that are coupled to HELLOs.

The *Global Timestamp* is the timestamp of this ADVSIG message and of the coupled HELLO/TC. The *Global Signature* is computed for the sequence of bits comprising the entire HELLO/TC message (header included) and the associated ADVIG message, excluding the *Global Signature* field itself. The *Time To Live* and *Hop Count* fields are considered as set to zero, because they change in transit.

The *Signature of Certificate # i* is only present when the ADVSIG is coupled with a HELLO. This field contains the signature of the Certificate related, respectively, to the *Neighbor Interface Address* at position i in the HELLO coupled message.

An exception is the *Signature of Certificate #0* field, which is not related to any advertised neighbor link, but is always included in ADVSIGs that are coupled to HELLOs. The three subsequent fields (i.e., *Link Code # i*, *Timestamp of Proof # i* and *Signature of Proof # i*) correspond to the Proof related to the neighbor node declared in the HELLO/TC message. The *Reserved # i* field is used to ensure that all fields are 32-bit aligned, and may be reserved for future use.

The *Link Code # i* is the link state in the Proof related to the neighbor node *i*. The Timestamp of *Proof # i* and *Signature of Proof # i* are the timestamp and signature of the Proof related to the neighbor node *i*.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Global Timestamp                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                         Global Signature                      :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:   Signature of Certificate #0 (always present, HELLOs only)   :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:            Signature of Certificate #1 (HELLOs only)          :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:            Signature of Certificate #2 (HELLOs only)          :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              . . .                            :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:            Signature of Certificate #n (HELLOs only)          :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Link Code #1 |                  Reserved #1                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Timestamp of Proof #1                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                       Signature of Proof #1                   :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Link Code #2 |                  Reserved #2                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Timestamp of Proof #2                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                       Signature of Proof #2                   :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              . . .                            :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Link Code #n |                  Reserved #n                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Timestamp of Proof #n                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                       Signature of Proof #n                   :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 10. ADVSIG Message Format

8.4 Generating ADVSIG messages

When a node generates a HELLO or TC message, it must also generate an ADVSIG message by following this protocol:

1. Create the HELLO/TC message;

2. Write the timestamp;
3. If the message is a HELLO then compute the signature of the Certificate Zero and, for each advertised link, compute the signature of the Certificate and add the relevant required Proof;
4. Else if the message is a TC then merely add the relevant required Proof;
5. Compute the signature;
6. Send the HELLO/TC and ADVSIG messages.

When a node receives the control message ADVSIG, it must follow this protocol:

1. Check the validity of the timestamp;
2. Check the validity of the signature;
3. If the message is a HELLO then, for each advertised link, check the validity of the Proof, and extract the Certificate regarding itself, if any, or the Certificate Zero if there is no Certificate regarding itself;
4. Else if the message is a TC then, for each advertised neighbor, merely check the validity of the Proof.

If any of the previous checks fail, the HELLO/TC and ADVSIG message must be dropped. For this method, we have proposed a mechanism to enhance the security of the OLSR Protocol against external attacks. This solution is based on recording recent routing information and reusing this information, to prove the link state of a node at a later time.

8.5 Securing OLSR Using Node Location

The node's geographical location is useful information that can be included in the node's messages, in order to achieve redundancy and enhanced security. We propose a protocol [9][13] that enhances security by including and processing the geographical position of the sending node in its control messages. This solution may also be applied to other link state protocols. It is inspired by the work of Hu et al. about packet leashes [5]. Here, we assume that the geographical information is obtained from a safe source, such as an embedded GPS device.

Several attacks can be thwarted if we possess information about the node position, i.e. if every node knows the correct geographical position of every other node in the network. Nodes can then compare this geographical data with the received control messages containing topology data (i.e., the neighbor and link set). If contradictory information is found, the false control message is detected and discarded. Besides, the availability of geographical information about nodes in the network in this method enables new predictions on possible new features in the standard OLSR, such as improved MPR selection and link breaking forecast. For instance, when two linked nodes are moving in opposite directions (i.e., the distance between the two nodes is rapidly increasing), a link break will soon occur. Therefore, neither of the two nodes should select the other as a MPR. This protocol is robust against two of the most serious types of attacks: link spoofing and wormhole.

8.5.1 Specifications

A control message called SIGLOC (Signature and Localization), which is based on the SIGNATURE message, is added to provide the security features. The SIGLOC message contains an additional GPS Localization field with the current geographical position of the sending node, obtained from the GPS facility. It is included in the signature computation. This field has a size of 32 bits, which is sufficient to define the position over an area of more than

4,200 square km, with a precision of 1 m using Cartesian coordinates; a more efficient representation can also be used.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       GPS Localization                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Signature                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 11. SIGLOC message format

The format of a SIGLOC message is given in **Fig. 11**. All other fields, as well as the mechanisms of signature computation and verification, are the same as for the SIGNATURE message. A SIGLOC message is generated and sent along with each HELLO or TC message. A node informs the other nodes about its current geographical position via this SIGLOC message. The receiving node verifies the correctness of the timestamp and signature as previously specified, and extracts the timestamp and information relative to the position of the originator node. This data is stored as a tuple (i.e., *address*, *position*, *timestamp*) in a *Position Table* maintained by each node. The position table stores the most recent position of every other node in the network. Note that geographical information is propagated in the network via SIGLOCs coupled to TCs, as HELLO messages are not distributed to more than one hop.

The advantage in knowing the geographical position of nodes is that we can predict whether communication of a message from a sender node S is likely to be heard by a receiver node R Let p_s and p_r be the current position of the sender and receiver and T_r be the current time according to the receiver's clock. Also let Δt be the discrepancy in the clocks' synchronization, Δd the maximum absolute error in the position information, v_{max} the maximum velocity of any node, and r_{max} the maximum transmission range. Based on the timestamp T_s of the sender's message, a lower bound is computed for the distance d_{SR} between the sender and receiver. This is given by

$$r_{max} \geq d_{sr} \geq \| p_{r} - p_s \| - (T_R - T_S + \Delta t) \,.\, 2v_{max} - \Delta d \tag{1}$$

where the radius r of the circle is the quantity on the right-hand side of the equation:

$$r = \| p_{r} - p_s \| - (T_R - T_S + \Delta t) \,.\, 2v_{max} - \Delta d$$

If (1) is not satisfied, this means that the receiver node is too far from the sender node to be able to hear its transmission; therefore such a transmission is highly suspicious and might be a fake.

Note that the standard OLSR already defines some checks to be performed at message reception; if the sender S of a TC/MID/HNA message is not a symmetric neighbor of receiver R, the latter must drop the message. A node that has the Tuple (X, p_X, T_X) in its location table and receives a SIGLOC message from X that contains the geographical data p'_X and the timestamp T'_X must check if the following condition is satisfied:

$$\| p'_X - p_X \| \geq (T'_X - T_X) \,.\, v_{max} \tag{2}$$

If (2) is not satisfied, this means that node X is pretending to be in a location it could not reach according to its maximum velocity; therefore either p_X or p'_X are likely to be false.

8.6 Protection against Link Spoofing

For any communication between a sender and receiver, Equation (1) must be satisfied, and this obviously also applies to a link state. Therefore, it is possible to detect the case in which a misbehaving node X falsely advertises a link (in a HELLO message) with the non-neighbor node N, or declares N as a neighbor (in a TC message). In the case of such a false declaration, Equation (1) is not satisfied with respect to the distance d_{XN} as evaluated by the receiver A of the message.

8.7 Protection against Wormhole Attacks

When a message is being maliciously tunneled between legitimate nodes A and B, Equation (1) is not satisfied with respect to the distance d_{AB} as measured by A.

8.8 Generating SIGLOC

When node A generates a HELLO or a TC message, it must also generate a SIGLOC and perform the following steps:

1. Create the HELLO/TC and SIGLOC.
2. Insert the GPS Localization from the GPS device output.
3. Insert the Timestamp from the actual time.
4. Compute and write the SIGLOC of HELLO/TC.
5. Send the HELLO/TC and SIGLOC.

When a node B receives a SIGLOC and its HELLO/TC from node A , it handles them in the same manner as a SIGNATURE message, performing the same tests (matching the SIGLOC with the companion HELLO/TC, timestamp validity check, and signature verification). Note that A is the last hop to B, and O is called the originator of the control message relayed by A . If the control message is a HELLO, then O is surely A, because we know HELLOs are not relayed; if the control message is a TC, then O may or may not be A (depending on whether A is MPR or not). The protocol performs the following steps:

1. Correctly pair the HELLO/TC with its SIGLOC companion, by matching the Message Sequence Number with the MSN Referrer.
2. Check the freshness of the timestamp.
3. Check the validity of the signature.
4. Check the validity of the HELLO message with respect to its originator node using (1) and the validity of links advertised in the HELLO/TC message.
5. Store the tuple (i.e., address, node location, timestamp) in the position table.

If any of the previous checks fail, node B must stop processing the HELLO/TC and SIGLOC, and discard them. Note that this algorithm assumes that B has entries for A, O and N, in its position table; this may not be the case at network initialization. Immediately after network bootstrapping, during the initial formation of the network, if B lacks the entry needed for the test it must skip the relevant step.

9. Experiments and Results

The variation in the performance of OLSR and Secure OLSR was studied via the NS-2 simulation environment. The results are discussed in the present section. We performed Mobile and Network Size simulations. In the case of the Mobile simulation we varied the mobility to determine the impact on various measured metrics. In the case of the Network Size simulations we varied the network size to determine the impact on the performance of an ad-hoc network.

9.1 Simulation Environment

For the simulation environment, many types of field configurations were simulated: the configurations were made by varying the following parameters: Network size, Number of nodes, Node movement and location, and Packet Size. The throughput of the sent and received packets was measured under varying environmental conditions. In all simulations, the simulation time was 150 seconds. Initially, using a fixed scenario file, we simulated OLSR routing algorithms for the network size of five nodes. We used the same scenario file and parameters for all four cases. **Table 2** shows the values of all parameters used during the simulation.

Table 2. Parameters

Parameter	Value
Environment size	500 x 400
Simulation time	150 sec.
Number of nodes	10

The adjustable parameters used during various simulations are as follows:

- Maximum speed: The speed the node is moving in the topography.
- Number of nodes: Constant for all simulations, except for the size simulations.
- Environment size (Topography): The size of the environment. We used a size of 500 X 400.
- Simulation Time: The time period of the simulation.

9.2 Simulation Details

In order to obtain meaningful performance and comparison results, a simulation study of a OLSR and Secure OLSR was performed for various traffic patterns and network sizes. As per the modifications to OLSR, two fields were added to the standard packet format, Timestamp (32 bits) and Signature (80 bits). The networks scenarios simulated are summarized in **Table 3**.

Table 3. Scenarios

Scenario	Packet Size
1	64 bytes : Standard OLSR
2	68 bytes : Timestamp added
3	88 bytes : Signature added
4	92 bytes : Both Timestamp

- **Scenario 1:** This scenario involves the transmission of packets that conform to the standard size, in this case 64 bytes.
- **Scenario 2:** In this scenario, the timestamp field is added to the OLSR packet. The timestamp field has a size of 32 bits; therefore four bytes are added to the existing packets. The packet size increases to 68 bytes.

- **Scenario 3:** The OLSR signature format consists of various fields, which have a total size of 192 bits; therefore 24 bits are added to the existing packet size. This scenario involves the transmission of packets 64 bytes in size.
- **Scenario 4:** This scenario involves the transmission of packets that contain both the timestamp and signature fields. The size of packets in this scenario is 92 bytes.

Each of these scenarios was simulated and the performance metrics were measured. Conclusions were drawn from the measured values of the metrics. Graphs depicting the throughput are provided in **Fig. 12-15**.

9.2.1 Throughput of Sending Packets : The total number of packets that are sent from source nodes as a fraction of the number of packets that are generated.

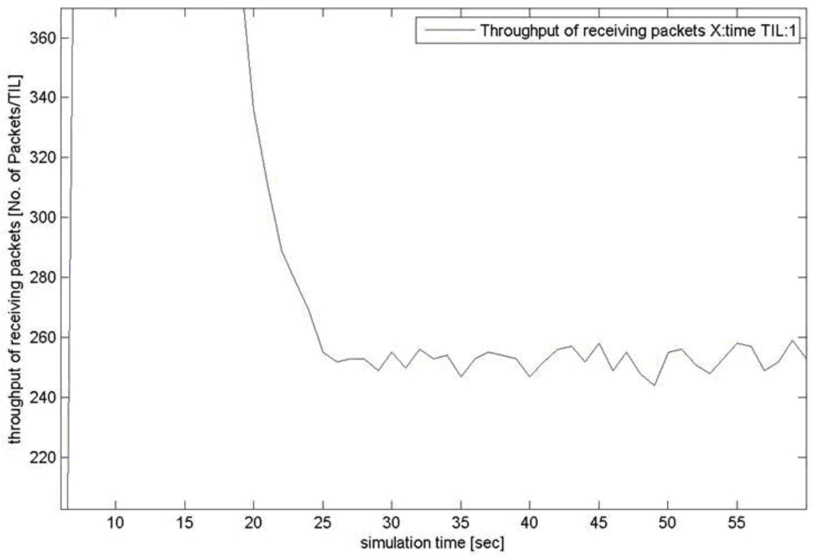


Fig. 12. Throughput of Sending Packets – Scenario 1

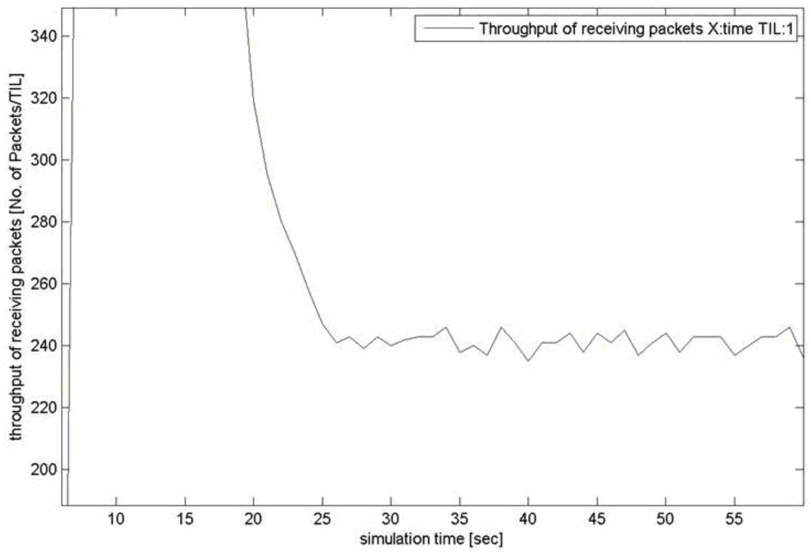


Fig. 13. Throughput of Sending Packets – Scenario 2

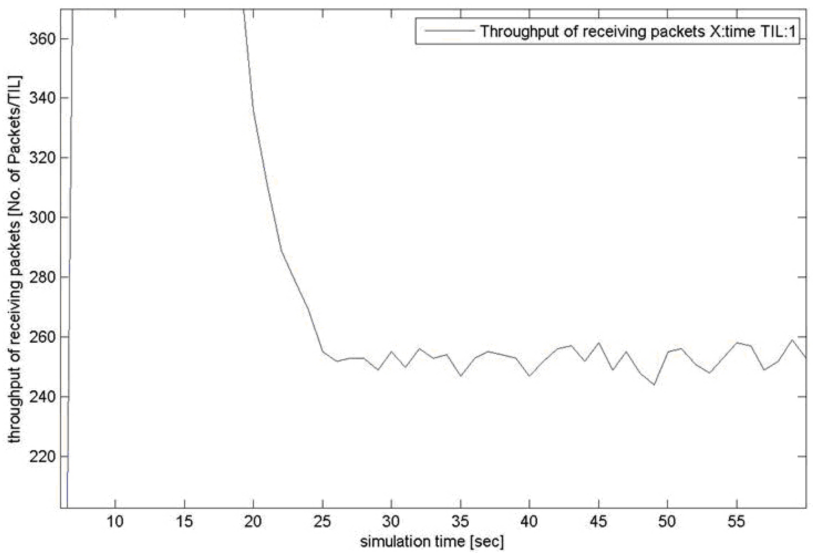


Fig. 14. Throughput of Sending Packets – Scenario 3

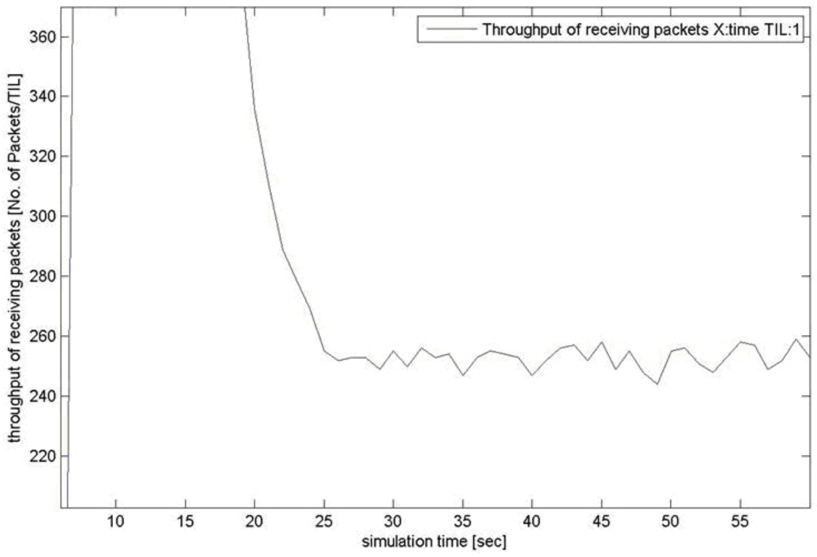


Fig. 15. Throughput of Sending Packets – Scenario 4

The throughput of the sending packets is the sum of the number of packets sent per unit time (1 second). As shown in Figure 13-16, there is an overall decrease in the throughput of the sent packets. Initially, the difference is negligible, but during the second half of the simulation period there is a noticeable drop in the throughput.

9.2.2 Throughput of Receiving Packets

The total number of packets that are received at the destination nodes as a fraction of the number of packets that are generated.

The throughput of the receiving packets is the sum of the number of packets received per unit time (1 second). There is an overall decrease in the throughput. Initially, the difference is negligible, but during the second half of the simulation period there is a noticeable drop in the throughput, as shown in **Fig. 16-19**.

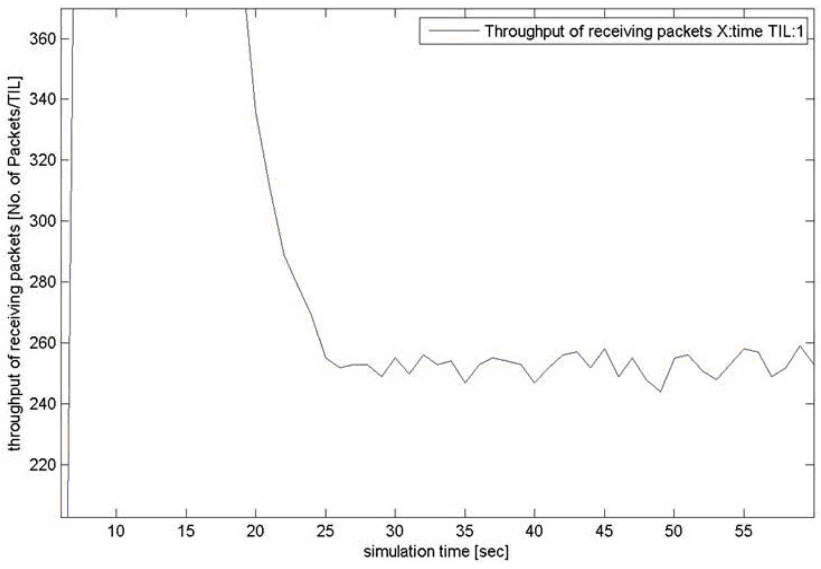


Fig. 16. Throughput of Receiving Packets – Scenario 1

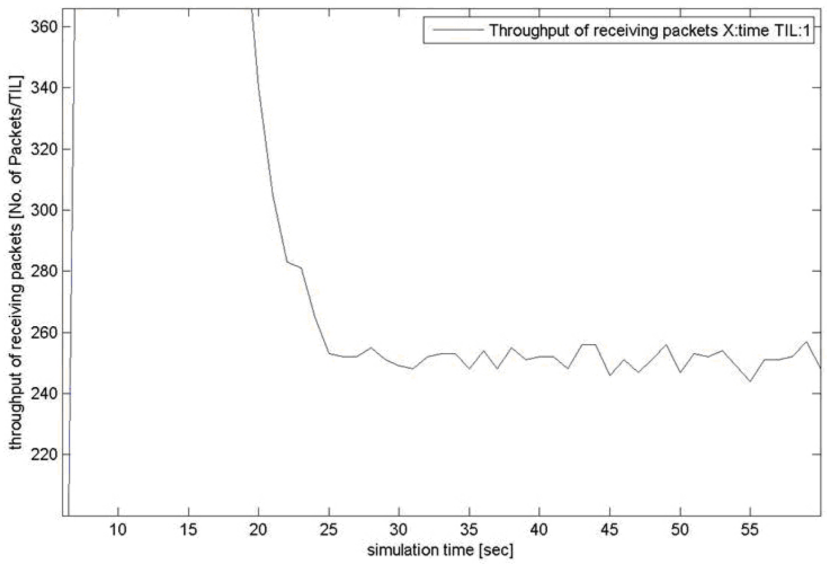


Fig. 17. Throughput of Receiving Packets – Scenario 2

9.3 Investigation

In this paper, we presented an overview of the security problems in wireless networks, focusing on the routing protocols in ad hoc networks. We contributed a proposed new algorithm named Secure OLSR, which is a secure form of the OLSR protocol. The solutions include the addition of a digital signature (SIGNATURE) to the control traffic, which is mainly used to prevent the injection of incorrect information in the network. For each control message (i.e., HELLO, TC, MID or HNA) generated, a corresponding SIGNATURE message is generated and sent in the same packet as the one containing the control message, immediately before it. Signatures are used by the receiving node to authenticate the corresponding OLSR control message, and every message without a matching, corresponding signature is dropped.

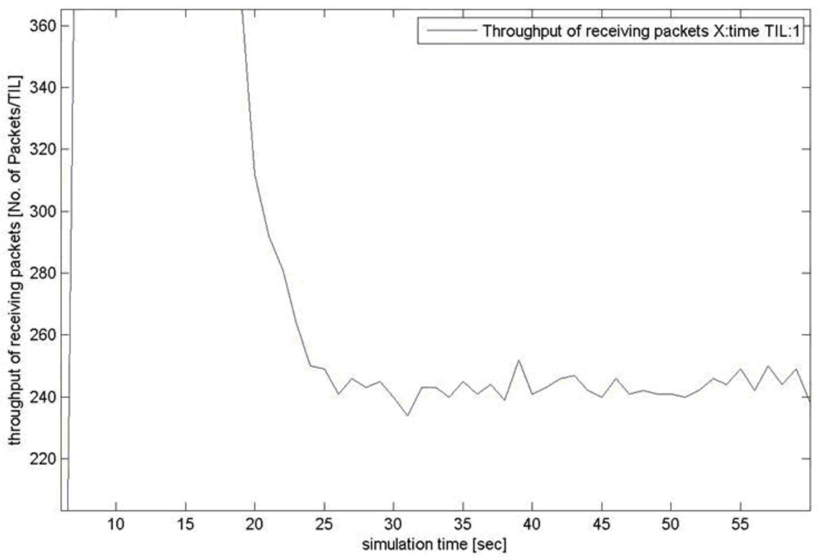


Fig. 18. Throughput of Receiving Packets – Scenario 3

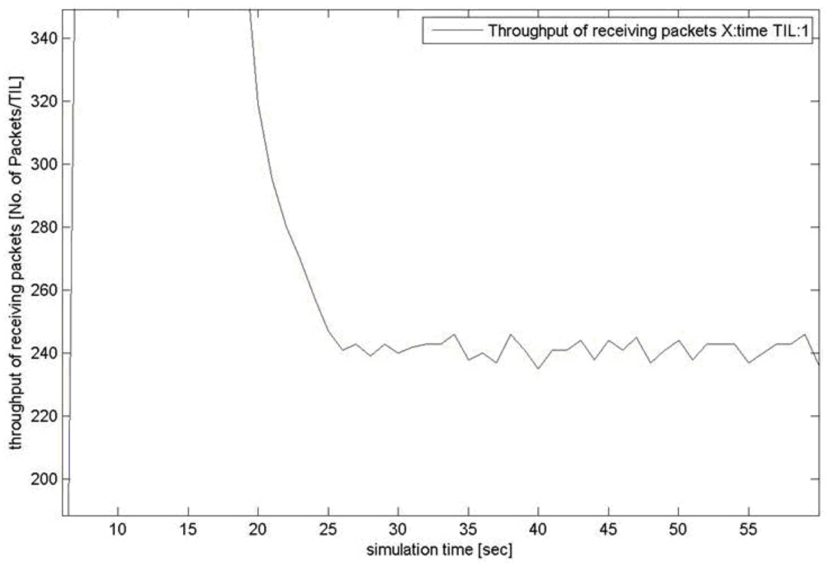


Fig. 19. Throughput of Receiving Packets – Scenario 4

The other solution utilizes the node's location to prevent the link spoofing and wormhole attack. If information about the node's geographical position is available, several types of attacks can be resisted, as nodes can then compare this geographical data to the received control messages containing topological data. If contradictory information is found, the false control message is detected and discarded. However, the increased security provided by the proposed solutions has the cost of a greater message overhead, as the exchanged control messages have a larger size and involve additional processing in both the originating and receiving node. These systems aim for the protection of network topology information.

Various results were obtained from simulations of various scenarios. First, results were obtained for the standard OLSR protocol, by varying the number of nodes in the network. The location of the nodes was then modified, i.e. there was node mobility. Finally, a simulation of the secure OLSR was performed, and we measured a drop in the performance when the packet size was increased to accommodate the added security related features such as signatures. However, this drop in performance is negligible, and it is acceptable considering the security that is provided.

10. Comparison with Existing Solution

We have provided an overview of the security problems in wireless networks, focusing on the routing protocols in ad hoc networks. We contributed some solutions that make the OLSR protocol more secure. The solutions include the addition of a digital signature (SIGNATURE) to the control traffic, which is mainly used to prevent the injection of incorrect information in the network. For each control message (i.e., HELLO, TC, MID or HNA), generated, a corresponding SIGNATURE message is generated and sent in the same packet as the one containing the control message, immediately before it. Signatures are used by the receiving node to authenticate the corresponding OLSR control message, and every message without a matching corresponding signature is dropped.

The use of multiple signatures generated by different nodes to validate link state information is provided as a solution to resist the link spoofing attack. It relies on creating and sending a new additional message called ADVanced Signature (ADVSIG) in association with routing control messages. This approach is based on authentication checks of new information injected into the network, and reuse of this information by a node, to prove its link state at a later time. The third solution utilizes the node location to prevent the link spoofing and wormhole attack. If information about the node's geographical position is known, several types of attacks can be resisted, as nodes can then compare this geographical data to the received control messages containing topological data. If contradictory information is found, the false control message is detected and discarded. **Table 4** summarizes the various attacks that the enhanced OLSR protocol is likely to resist:

Table 4. Protection against various OLSR attacks.

Attack on OLSR Protocol			Protection by SIGNATUR	Protection by ADVSIG	Protection by SIGLOC
Incorrect Traffic Generation	Incorrect HELLO	ID spoofing	Yes	Yes	Yes
		Link Spoofing	No	Yes	Yes
	Incorrect TC Generation	ID Spoofing	Yes	Yes	Yes
		Link Spoofing	No	Yes	Yes
	Incorrect MID/HNA Generation		Yes	Yes	Yes
	ANSN Attack		Yes	Yes	Yes
Incorrect Traffic Relaying	Message Tampering		Yes	Yes	Yes
	Black hole Attack		No	No	No
	Reply Attack		Yes	Yes	Yes
	Wormhole Attack		No	No	Yes
	MPR Attack		No	No	No

The enhanced security provided by the proposed solutions has the cost of a greater message overhead, as the exchanged control messages have a larger size and require additional processing by both the originating and receiving node. This may be unsuitable for a network composed of nodes with insufficient computational power, for a network requiring QoS that must guarantee high performance in terms of the data rate, or a network that simply does not need enhanced security. On the other hand, these techniques can be combined in order to provide a higher security level. These systems aim for the protection of network topology information.

11. Conclusions

This technical report was concerned with the domain of wireless computer networks. We have addressed the security issues related to the OLSR (Optimized Link State Routing) protocol. Various algorithms have been proposed, to provide the security features in the existing OLSR protocol. It is anticipated that future work can extend this; by adding additional fields to the message, the overall message overhead and complexity is increased, but the security is enhanced. So in the near future, new protocols may be designed that provide the security with minimal message overhead and complexity. The study of better cryptographic algorithms (in terms of a smaller signature size, reduced computation complexity, and greater speed) will increase the applicability of the proposed OLSR security architectures to a practical ad-hoc protocol. A detailed investigation of the principles of mobile ad-hoc networks, their types, architectures, and routing mechanisms has been discussed in this technical report. The current research survey shows that there has been very little work related to the security of algorithms in MANETs. In the present research, these algorithms have been studied from the security perspective. The report also provided a detailed study and classification of the types of attacks against the Optimized Link State Routing (OLSR) protocol. In addition, proposed solutions that are likely to resist various attacks were also presented.

References

[1] Y.C. Hu, D.B. Hohnson, and A. Perrig. SEAD: “Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks,” Proceedings of *4th IEEE Workshop on Mobile Computing Systems and Applications* (*WMCSA 02*), pp.3-13, 2002.

[2] M. Guerrero Zapata and N. Asokan, “Securing Ad Hoc Routing Protocols,” Proceedings of *ACM Workshop on Wireless Security* (*WiSe*), pp.1-10, 2002.

[3] A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song, “Efficient authentication and signing of multicast streams over lossy channels,” *IEEE Symposium on Security and Privacy*, pp.56-73, 2000.

[4] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, “The Zone Routing Protocol (ZRP) for Ad Hoc Networks,” draft-ietf-manet-zone-zrp-04.txt, July, 2002.

[5] Vesa Karpijoki, “Signalling and Routing Security in Mobile and Ad-hoc Networks,” Department of Computer Science Helsinki University of Technology, May, 2000.

[6] Cedric Adjih, Daniele Raffo, Paul Muhlethaler, “Attacks Against OLSR: Distributed Key Management for Security,” INRIA, Domaine de Voluceau, France.

[7] Behrouz A. Froouzan, “Data Communications and Networking,” 3rd Edition, Tata McGraw Hill Edition, 2001.

[8] T. Clausen and P. Jacquet, “Optimized Link State Routing Protocol (OLSR),” RFC 3626, IETF Network Working Group, Oct. 2003.

[9] Daniele Raffo, Cedric Adjih, Thomas Clausen, and Paul Muhlethaler, "OLSR with GPS information," Proceedings of the *2004 Internet Conference* (*IC 2004*), Tsukuba, Japan, Oct. 28-29 2004.
[10] Navid Nikaen, "A short tutorial on routing issues in wireless ad hoc networks".
[11] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo, "Securing the OLSR Protocol," Proceedings of the *2nd IFIP Med-Hoc-Net 2003*, Mahdia, Tunisia, June 25-27, 2003.
[12] Daniele Raffo, Cedric Adjih, Thomas Clausen, and Paul Muhlethaler, "An advanced signature system for OLSR," Proceedings of the 2004 ACM Workshop on Security *of Ad Hoc and Sensor Networks* (*SASN '04*), pp.10-16, Washington, DC, USA, Oct. 25, 2004.
[13] Daniele Raffo, Cedric Adjih, Thomas Clausen, and Paul Muhlethaler, "Securing OLSR using node locations," Proceedings of *2005 European Wireless* (*EW 2005*), pp. 437-443, Nicosia, Cyprus, Apr. 10-13, 2005.

Amandeep Dhir did his bachelor's in the Department of Computer Science and Engineering, College of Engineering, Punjabi University, Patiala, India. He has been awarded as First Class First Degree with Distinction by Punjabi University. He has published many research papers in International Journals and Conferences. He is currently working with Newgen Software Technologies, New Delhi as a software development engineer. His job deals with wide categories of software development with great emphasis on security and quality. His key research areas are secure internet architecture, security from attacks and threats, and mobile adhoc networks.

Jyotsna Sengupta received her B. E. degree in Electronics and Communications, and her Ph. D degree in Computer Science and Engineering from Thapar Institute of Engineering and Technology, Patiala, India, in 1982 and 2002, respectively. She received her M.S. in computer Science and Engineering from University of Santa Clara, California, USA, in 1985. She worked as a Software Engineer in EXL, Ltd., Sunnyvale, California, in 1985-86. During 1986-87, she worked in the Department of Computer Science, Delhi University, India. Since 1989, she has been in the faculty of Punjabi University, Patiala, India where she is currently a key leader in the Department of Computer Science. Her current research interests primarily lie in the area of network security and secure computing.