NTNU

Norwegian University of
Science and Technology

Simulation of New Security Elements in an Ad Hoc Network

Syed Md. Ashraful Karim

Master of Science in Communication Technology
Submission date: March 2009
Supervisor: Svein Johan Knapskog, ITEM
Co-supervisor: Nyre Åsmund Ahlmann, SINTEF ICT
Peter Sjödin, KTH, Sweden

Norwegian University of Science and Technology
Department of Telematics

Problem Description

The candidate will configure a simulation of an ad hoc network for first responders in a crisis scenario using the NS2 simulation platform. The task will involve adjustment and experimentation with simulation parameters. Finally, new security protocol elements developed in the OASIS project at SINTEF ICT will be introduced through modification of the protocol definitions for NS2 written in C++. The work will be performed at SINTEF ICT.

Assignment given: 14. October 2008
Supervisor: Svein Johan Knapskog, ITEM

NTNU

Innovation and Creativity

Simulation of New Security Elements in an Ad Hoc Network

Syed Md. Ashraful Karim

Master of Science Thesis

Supervisor : Åsmund Ahlmann Nyre, SINTEF
Professor : Svein Knapskog, NTNU and
Peter Sjödin, KTH.

Norwegian University of Science and Technology (NTNU)
Department of Telematics

Royal Institute of Technology (KTH)
Department of Internetworking

Trondheim, March 16, 2009

Abstract

Several routing protocols and security extensions have been proposed to specifically address the area of Mobile Ad Hoc Networks (MANETs). However, due to the difference in nature of the various application areas and the wide variety of different network topologies, it has been proven difficult to find a single protocol to handle everything.

In this thesis we investigate a security extension to the OLSR routing protocol developed by SINTEF ICT tailored for use by first responders in emergency and rescue scenarios. We implement the protocol in the NS-2 simulation environment in order to verify correctness and measure the effect on key performance metrics. Our results indicate that delay and packet drop rate are not affected by introducing security, while the control data overhead is increased. Although overhead is significant, we argue that the protocol extension is well suited for MANETs of moderate size and density. We also consider that the protocol is suitable not only for first responders, but also for other similar application areas requiring restricted access to the network.

Keywords: MANET, routing, security, OLSR, encryption, simulation.

Problem Description

The candidate will configure a simulation of an ad hoc network for first responders in a crisis scenario using the NS-2 simulation platform. The task will involve adjustment and experimentation with simulation parameters. Finally, new security protocol elements developed in the OASIS project at SINTEF ICT will be introduced through modification of the protocol definitions for NS-2 written in C++. The work will be performed at SINTEF ICT.

Acknowledgment

First and foremost, I would like to express my heartiest gratitude to my supervisor Åsmund Ahlmann Nyre of SINTEF for his suggestions, guidance and constant encouragement throughout the progress of the thesis. I am greatly indebted to him for reviewing, analyzing the thesis structure and giving me valuable recommendations. I would also like to express my sincere thanks to my professor Svein Knapskog of the Norwegian University of Science and Technology and Peter Sjödin of the Royal Institute of Technology for their valuable advices and all-out cooperation. My heartiest gratitude also to Martin Gilje Jaatun for reviewing my final manuscript and for giving me the chance to work on this project with the research team at SINTEF ICT.

Preface

The thesis is done as the partial fulfillment for the degree of Masters of Science in Security and Mobile Computing (NordSecMob). This is a two year Erasmus Mundus Master Program that leads to two officially recognized M.Sc. degrees (120 ECTS) issued by the home and host university. The home university is the Royal Institute of Technology (KTH), Sweden and the host university is the Norwegian University of Science and Technology (NTNU), Norway. It was carried out at SINTEF ICT, Norway, in collaboration with the Department of Telematics at NTNU and the Department of Information Communication Technology at KTH during the autumn semester 2008.

Table of Contents

Chapter 4. Optimized Link State Routing Protocol (OLSR)

Chapter 5. New Security Extensions to OLSR

Chapter 6. Implementation of the New Security Extensions to OLSR in NS-2

Appendix A: Scripting Code

Appendix B: Algorithms

List of Figures

List of Tables

Abbreviations

ANSN	Advertising Node Sequence Number
AODV	Ad hoc On-demand Distance Vector
CA	Certificate Authority
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DoS	Denial of Service
DSDV	Destination Sequenced Distance Vector
DSR	Destination Source Routing
GUI	Graphical User Interface
IETF	Internet Engineering Task Force
MAC	Message Authentication Code
MANET	Mobile Ad-hoc NETwork
MPR	MultiPoint Relays
NAM	Network AniMator
NIA	Neighbor Interface Address
NS-2	Network Simulator version 2
OASIS	Open Advanced System for dISaster and emergency management
OLSR	Optimized Link State Routing
OTCL	Object TCL
PKI	Public Key Infrastructure
RREQ	Route REQuest
RREP	Route REPly
RERR	Route ERRor
SAODV	Secure Ad hoc On demand Distance Vector
SEAD	Secure Efficient Ad hoc Distance vector
SLSP	Secure Link State Protocol
SOLSR	Secured OLSR
TC	Topology Control
TCL	Tool Command Language
TORA	Temporary-Ordered Routing Algorithm

Chapter 1

Introduction

A Mobile Ad hoc Network (MANET) is a collection of communication devices or nodes that wish to communicate without any fixed infrastructure and pre-configured organization of available links. The nodes in a MANET are responsible for dynamically discovering other nodes to communicate. Although an ongoing trend is to adopt ad hoc networks for commercial uses, there are still concerns with respect to their vulnerability to security attacks. A number of challenges like open peer-to-peer network architecture, stringent resource constraints, shared wireless medium, dynamic network topology etc. are present in a MANET. The collaborative nature of ad-hoc networks and the necessity of each node to act both as an end system and to perform routing functions makes routing protocols the main target of security attacks.

1.1 Motivation

Mobile ad hoc networks (MANETs) are ideally suited for emergency response operations since they do not rely on fixed infrastructure nor are they are difficult to deploy whenever needed. All nodes in a MANET have to perform two roles: to act as an end-system and to perform routing functions. In addition to the inherent vulnerabilities of the wireless links, the routing mechanism of MANETs is one of the main areas of security risk. As an emergency service requires the transaction of valuable and private data while working in the field, all aspects of wireless data transfer need to be properly secured and private.

The motivation of the thesis is to design a secured ad hoc network that may be employed for field workers performing emergency rescue work during a crisis situation even if all other existing communication system are unavailable.

1.2 Scope

In our thesis work, we will base ourselves on a security extension to a routing protocol in mobile ad hoc networks. The security protocol was developed in the OASIS project [2] at SINTEF ICT. In the thesis, we will configure a simulation of an ad hoc network using the extension for first responders in a crisis scenario in the NS-2 simulation platform [1]. The new protocol will be implemented through modification of the protocol definitions for NS-2 written in C++ [26]. The task will involve adjustment and experimentation with simulation parameters.

1.3 Objective

Simulation can provide an insight into the basic operations and performance of an experimental protocol prior to performing a prototype implementation. Our goal is to implement the protocol so that,

- We can determine the appropriateness of the security extension through simulation.
- How much is the network performance affected by the deployment of the secured routing protocol compared to the original protocol or other routing protocol?
- Provide input to further development of the protocol and recommendations for real-world implementations.

1.4 Organization of the Thesis

The next chapters in this report are organized as follows:
Chapter two presents the background of the thesis work. The thesis work is done as a part of the OASIS project, which has provided an insight into the necessities and structure of the first responder network that we are working with.

Chapter three discusses on existing routing protocols for MANETs and proposed security extensions.

Chapter four presents the Optimized Link State Routing protocol (OLSR), on which the security extension is based. We also present a selection of exiting security extensions to OLSR.

Chapter five presents the new security extension to OLSR which is developed in the OASIS project. A brief overview of the access control, authentication mechanism and key establishment process is presented.

Chapter six outlines the details of the implementation in the source code of NS-2 to add the functionalities of the new security extension to OLSR. It presents the data structures and functions which handle new message formats and the security establishment process.

Chapter seven presents the simulation procedures, parameters, results from our simulation of the new protocol and the performance of the new protocol in wireless ad hoc networks.

Chapter eight discusses on the simulation result, their effects and reasons to choose the parameters used in the simulations.

Finally chapter nine briefly outlines the opportunities for further work and chapter ten concludes the thesis.

Chapter 2

Background

2.1 Context

The purpose of the thesis is to simulate a new security protocol in a wireless network infrastructure which is established during a crisis situation. The thesis is performed as a part of the Open Advanced System for dISaster and emergency management (OASIS) project at SINTEF ICT [2]. On response to a crisis situation a wireless ad-hoc network is formed by the field workers. The wireless network should be able to work without the presence of a central entity and also provide desired level of security. Along with that the wireless network should benefit from the central OASIS network services whenever possible. The proposed solution and simulation addresses both situations.

2.2 OASIS project

The OASIS Project addresses the Strategic objective, "Improving Risk Management" [2]. The objective of the project is to define and develop an Information Technology framework based on an open and flexible architecture that will be the basis of a European Disaster and Emergency Management system. SINTEF's part of the project was to devise a secured communication system for the first responders involved in an emergency rescue team.

2.3 Scenario

To consider a reference scenario, we have followed but not restricted to the “Fire in Chemical Plant” scenario described in the OASIS evaluation plan [3]. A sample communication scenario is shown in Figure 2.1. In our work on information security for the field workers, we have expanded the number of participants and other parameters to a more realistic level.

OASIS has performed several Pre-Operational System (POS) tests of different scenarios [4]. The hierarchical command structure of OASIS has three levels of operations: strategic (what we do?), operational (how we do it?) and tactical (do it!) [3]. The implementation of ad hoc networks falls into the tactical level.

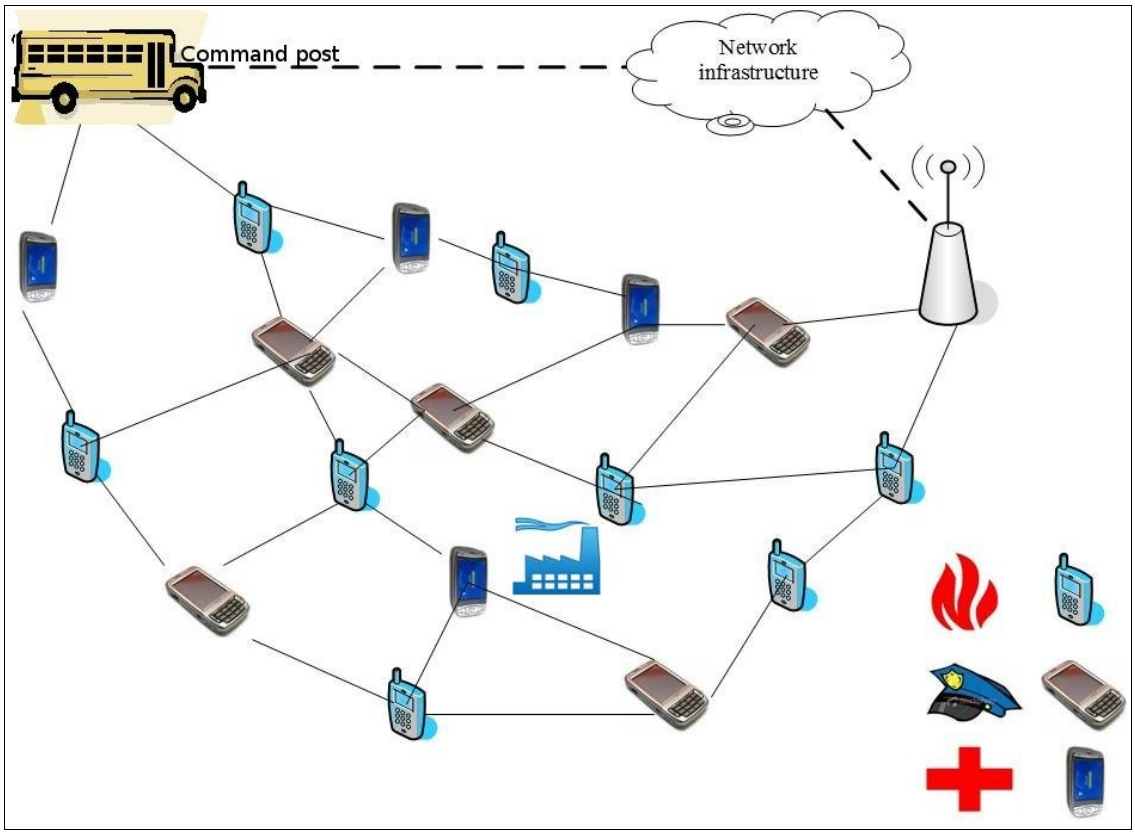


Figure 2.1: Communication scenario in OASIS project

To set up the ad hoc network, the most important assumption in the crisis scenario is that there are no available existing communication facilities. So the only default option for communicating among the field workers is an ad hoc network. Additionally any existing infrastructure may be available or used if there is any. The field workers need to participate in the ad hoc network. We assume that all field workers will be equipped with a portable unit that will cover all communications needs especially for mobile wireless communications.

2.4 Equipment

We assume that all field workers carry a portable computing device with IEEE 802.11g compatible wireless network interface and sufficient processing power, memory and battery life to sustain and perform the task needed for the processing, routing and storing data needed in the ad hoc network.

The command vehicle is equipped with an ad hoc network node as well as one or more pieces of long range infrastructure communication equipment from below:

- WiFi (IEEE802.11) link
- WiMAX link
- UMTS/GPRS link
- SatCom link

It is assumed in this scenario that the command vehicle has an unlimited power supply.

2.5 Communication needs

Field workers need to communicate with each other and need access to the central information repositories and the next tier of the command structure. The default behavior of the field workers' communication unit should be to join the “Oasis ad hoc network” if it is available. If a command post or communication infrastructure access point is available, a node may connect to it for bridging the communication.

In different circumstances, information may be transmitted using either the push or pull paradigm, that is, a node may attract the attention of another node to send some data (push) or a node may request an update on a particular situation to another node (pull).

2.6 Limitations

The field workers need to have some credentials to access into the network which are pre-configured. Such as,

- Assume all field workers have PKI certificates for authentication
- One of the challenges is to determine the revocation list of certificates
- Can assume that all certificates are issued by local authority
- Foreign certificates must be cross-signed by both of the local and border authorities.

Chapter 3

MANET Routing Protocols and Security Issues

A mobile ad hoc network (MANET) is a self configurable collection of any number of wireless mobile devices. All the nodes in a multi-hop wireless ad hoc network cooperate each other to form a network without the presence of any infrastructure such as access point or base station. The mobile nodes in this network require to forward packets for each other to enable communication among nodes outside the transmission range. The nodes in the network are free to move independently in any direction, leave and join the network arbitrarily. Thus a node experiences changes in its link states regularly with other devices. Eventually, the mobility in the ad hoc network, change of link states and other properties of wireless transmission such as attenuation, multipath propagation, interference etc. create a challenge for routing protocols operating in an ad hoc network. The challenges are enhanced by the various types of devices of limited processing power and capabilities that may join in the network. Figure 3.1 shows a small example of a mobile ad hoc network.

Figure 3.1: A mobile ad hoc network

Mobile ad hoc network is becoming popular in many application areas such as military operations, vehicular networks, wireless sensor networks, emergency services, students in classrooms and meeting attendees. For all of the above and any other areas of ad hoc networks, security in the routing protocol is essential to protect against attacks on routing protocol such as, diffusing incorrect routing information.

3.1 Routing protocols for MANET

For the nature and challenges found in designing an ad hoc network routing protocol, a large amount of work has been done in the research community to find a perfect routing protocol for wireless ad hoc networks. The research has resulted to a number of routing protocols which can be classified as *topology-based* routing protocols and *position-based* routing protocols as shown in Figure 3.2. Topology based routing protocol uses the traditional routing concept such as maintaining a routing table or distributing link state information. Position based routing protocol uses the geographical physical position of the mobile nodes to route the data packets to the destination.

Topology based routing protocols are further divided into two groups: *proactive* and *reactive* protocols. Proactive protocols try to maintain a consistent, up-to-date routing information within the system so that at any time, every node knows how to route packets to the other nodes in the network. In contrast to this, reactive routing protocols find a link between the source and the destination when it is needed. Reactive protocols are also known as on-demand routing protocols.

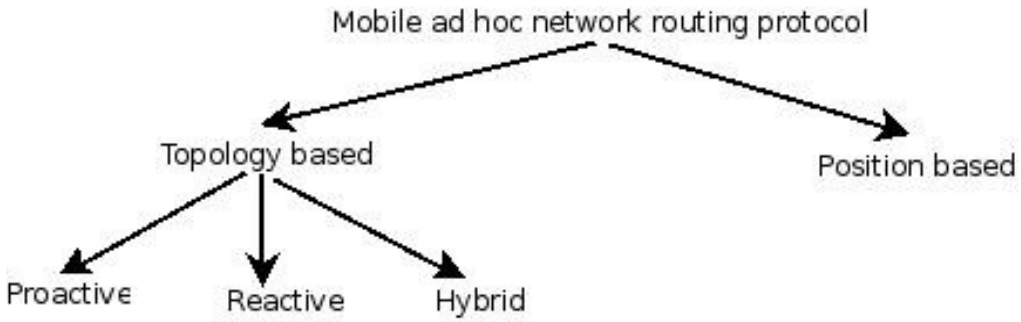


Figure 3.2: Classification of mobile ad hoc network routing protocols

Proactive routing protocols usually require periodic exchange of messages and routing information to maintain an updated information of the links. If only a few pair of nodes are communicating in a large network then most of the periodical exchanged information is useless and hence proactive protocol can waste a lot of bandwidth and other resources. On the other hand, as routing information is always updated and readily available, packets can be delivered to the destination virtually without any delay. In contrast to this, in case of reactive protocols, the nodes waste their resources to find out the routes when it is necessary. There exists some hybrid protocol that uses a combination of reactive and proactive approach to maintain routes. The hybrid technique is to use proactive approach for the nodes in the local neighborhood, that is, for the nodes up to a certain hops and use reactive approach when the destinations nodes are far away.

In case of position based routing, the nodes use the information about the physical location of other nodes to route data packets to their destinations. Each node in the network is aware of their own position by means of GPS or some other devices and obtains the location information of other nodes via a location service that is provided by the nodes themselves. When sending a data packet to a destination, the source node obtains the position of the destination node by the location service and includes this information in the header of the packet. Then, each intermediate node that receives the packet gets the location information of the destination from the packet and uses it to forward the packet comparing to its own location.

The advantage of position-based routing is that the nodes do not need to maintain routing information or to discover routes explicitly, which greatly reduces control traffic overhead. This relieves the protocols from bearing large control overhead in the packet header. However there is still some overhead to find the location service and get location information from the location service. The disadvantage of position-based routing is that the node needs to install some sort of hardware which will provide the precise geographical location information of the node itself e.g. a GPS device. Another problem here is that satellite-based positioning systems do not work well under ground, say there is a fire in a tunnel, then the fire fighters cannot rely on GPS for positioning.

In this chapter our goal is to present a short description of some basic routing protocols to discuss and understand the security measures and implications in wireless ad hoc network routing protocols. A comprehensive description of all existing ad hoc network routing protocols is considered out of scope.

3.2 Reactive routing protocols

Reactive routing protocols are also known as on-demand routing protocols. Reactive routing protocols can use the source routing or distance vector routing mechanisms.

3.2.1 Dynamic Source Routing (DSR)

To illustrate an on-demand source routing protocol, we briefly present the Dynamic Source Routing (DSR) protocol. DSR is one of the very first protocols proposed for mobile ad hoc networks. A detailed description of DSR can be found in [16].

DSR is a source routing protocol which means every packet will carry the list of the nodes that it will traverse to reach the destination. Every node which receives the packet first verifies whether it is the destination of the packet. If it is not, it checks its own identifier in the list of nodes carried by the packet. If the node finds itself in the list then it forwards the packet to the next node in the list which must be a direct neighbor. Otherwise it drops the packet.

Three main advantages of DSR are:

1. It is trivial to detect routing loops by identifying the repeating values in the list of node identifiers in the packet header.
2. The forwarding nodes need not to keep an updated routing information to forward the packet towards the destination as it is available in the packet header and is provided by the packet source.
3. Each node that receives the packet can extract routes from the packet header and cache it locally for future use.

The main disadvantage of DSR is the communication overhead that each packet has to carry. This limits the applicability of the protocol in highly resource constrained environment such as sensor network and in large network where routes can be very long.

3.2.1.1 Route discovery and maintenance in DSR

When using source routing, the source of the packet embeds the full route to the destination in the packet header. Hence the source must know the route to the destination before it starts sending a packet. In DSR this route can be obtained by the local route cache. If an appropriate route is not found in the route cache then DSR initiates a *route discovery* mechanism. In addition to route discovery, DSR has a *route maintenance* mechanism that allows the source to detect if a route is broken.

The DSR route discovery mechanism is based on flooding the entire network with a route request (RREQ) and returning some route replies (RREP) messages. A route request contains the identifier of the source and destination and a record listing the identifiers of every intermediate node that forwarded this particular request. Each intermediate node that receives the request verifies whether it has received the request before, if it has not, it appends its identifier in the request and re-broadcasts the request to its neighbors. This procedure is repeated until the request reaches the destination. When it reaches the destination, the node generates a route reply copying the recorded list of the intermediate nodes and unicasts back to the source node. DSR requires each intermediate node to make sure that the data packet is forwarded to the next hop properly and is ensured by hearing an acknowledgment. If it does not receive any acknowledgment, the intermediate node generates a route error (RERR) message indicating the next link is not functioning and sends this error back to the source of the packet.

3.2.2 Ad hoc On-demand Distance Vector (AODV)

Apart from the source routing, on demand distance vector routing protocols make routing decision based on traditional routing tables. As the routing table is updated on demand basis, this type of reactive protocol is named as on-demand routing protocol. On demand routing protocols do not maintain routing information for the nodes that are

not in active communication. A well known on demand, table based wireless ad hoc routing protocol is Ad hoc On-demand Distance Vector (AODV) protocol. A detailed description of AODV can be found in [19].

In AODV, each node maintains a routing table where each entry of the table contains information related to a particular destination. The entries include the following information: the identifier of the destination, the number of hops needed to reach the destination, the identifier of the next hop towards the destination, the list of hops that can forward the packet to the destination, and a destination sequence number.

AODV tuple = <ID, number of hops, next hop ID, list of hops, Dest_SEQ>

When an intermediate node receives a packet, it looks into the routing table to find the next hop towards the destination and forward the packet accordingly. This process is repeated until the packet reaches the destination. Obviously the routing mechanism needs a valid entry in the routing table to forward the packet. This is ensured by route discovery and route maintenance strategy similar to DSR routing protocol.

3.2.2.1 Route discovery and maintenance of AODV

In the route discovery mechanism of AODV, when a source wants to send a data packet to a destination, and it does not have a valid entry for that destination in its routing table, it generates and broadcasts a route request (RREQ) message. The RREQ message contains the identifier of the requested source and the destination node, a hop count, and two sequence numbers in which the first is the current sequence number of the source and the other is the last known sequence number of the destination routing table entry.

Upon receiving the route request message, an intermediate node determines if the request is a duplicate or not checking the sequence number that is maintained for each request. If the request is not a duplicate, it looks for a valid entry for the destination requested in its own routing table. If it does not have a valid entry, it re-broadcast the RREQ message after incrementing the hop count in it. On the other hand, if the

intermediate node has a valid entry then it generates a route reply (RREP) message. Obviously, if the request reaches the destination, it will generate a RREP. In the process of route discovery, the intermediate nodes also update their routing table using the predefined algorithm based on sequence numbers and hop count. AODV route maintenance scheme uses route error (RERR) message to indicate broken links.

3.3 Proactive routing protocols

Proactive routing protocols maintain up-to-date routing information for all possible destinations in the network. Thus the nodes using a proactive routing protocol have comprehensive understanding of the whole network topology. These protocols are usually based on a periodic exchange of routing information. Proactive protocols can be of two types : *link state protocols* and *distance vector protocols*.

In the link state protocols, each node periodically flood the network with a message containing the state of the links of that node. As the messages are propagated inside the entire network by all other nodes, each node in the network gets the knowledge of the links of all other nodes and thus each node has a full view of the entire network topology. Then each node independently apply central shortest path algorithm to calculate the shortest route based on the metrics it chooses and manages the routing table of the best routes to other reachable nodes in the network.

In contrast to this, in distance vector approach, the node uses a distributed shortest path algorithm to determine the best routes to every other node in the network. For this purpose, each node periodically sends its current routing table to the neighbor nodes. So each node learns about the network by trusting its neighbors. And looking into the routing table of the neighbors a node can find and update possible best routes to the other nodes. When it finds a better route than it has already, the node updates its routing table to incorporate new information. By repeating the process of sending and updating the routing table, the system converges to a stable state when all routing information becomes correct. Obviously, distance vector routing table takes some time to converge when network topology changes abruptly.

Example of proactive routing protocols are Optimized Link State Routing (OLSR) protocol [5], Destination-Sequenced Distance Vector (DSDV) [20] etc. We will have more discussion on OLSR in the next chapter. A brief description of DSDV is presented here.

3.3.1 Destination-Sequenced Distance Vector routing

Destination-Sequenced Distance Vector (DSDV) is a proactive, topology based routing protocol for mobile ad hoc networks. DSDV has special feature of adding sequence number to prevent routing loop in comparison to other distance vector routing protocols. DSDV is a predecessor of AODV and employs a similar mechanism to refresh routing updates using sequence number for each entry in the routing table. So if a routing update has a greater sequence number than it has in the table entry then the entry is updated considering the routing metric is preferable to the node. For each change in the routing table, the sequence number is increased. DSDV can be optimized by sending only incremental updates that lists only the changes occurred in the route since the last full update sent to the given node rather than sending the full list of all destinations.

3.4 Vulnerabilities of routing protocols

Attacks on routing protocol primarily aim at routing disruption or resource consumption of the mobile devices. In case of routing disruption attacks, the attacker attempts to cause the legitimate data packets to be routed in a dysfunctional way. On the other hand, in case of resource consumption attacks, the attacker injects packets into the network to consume valuable resources in the network, such as, bandwidth or node resources like memory, battery power or computational time. The most common attack types are mentioned below [12] :

Spoofing: An adversary can create control packets using a fake identity.
Impersonation: This attack is performed using spoofing IP or ARP packets and gain access as a legitimate source.
Packet forgery: Fabricating control packets containing fake routing information.
Sniffing and traffic analysis: An intruder listens to the messages and exploits them.

Replay: Keeping the sequence numbers of the packets, attackers replay sessions by replacing original message by a new false one.
Modification, insertion and deletion: Attackers capture the messages and then modify or insert false message or delete the message without forwarding.
Denial of Service: A traditional DoS attack, consumes all the resources of a node or bandwidth by sending or flooding excess number of packets.
Black hole or Selfishness: This attack occurs when nodes refuse to forward control messages of the other nodes and drop them intentionally.
Tunneling: When two adversarial nodes pass control packets back and forth between each other encapsulating them in normal data packets and use the routing service of the network is called tunneling.
Wormhole: A wormhole is similar to tunneling except it works in the physical layer. It can be implemented with two radio transceiver connected through an out of band channel.

All these types of attack can be used for route disruption, creation of incorrect routing state, route diversion, generation of extra control traffic, creation of gray hole etc. [12].

3.5 Secure routing protocols

The standard MANET routing protocols such as DSR, AODV etc. do not address the security of routing information allowing attackers to launch attacks easily and disrupt routing, partition the network, create black hole to interrupt any communication. Thus providing security to ad hoc network routing protocol is an active research area and many extensions to the protocols is proposed along with new secured routing protocols. Here we will present a brief overview of the proposed extensions.

3.5.1 Secure Routing Protocol (SRP)

SRP is an on demand source routing protocol based on symmetric key cryptography [18]. In SRP, only the source and the destination node share a key resulting to a strict end-to-end exchange of routing information between the source and the destination, and

end-to-end authentication of routing control packets. The design of SRP is influenced by the observation that due to the mobility of the nodes, it would be impractical that the source or destination shares keys with all intermediate nodes on a route. Therefore sharing the key only between the source and destination simplifies the key management considerably. In SRP intermediate nodes do not send replies to route discovery messages and they do not cache route information from overheard routing control packets.

S → * : (rreq, S, D, id, sn, mac-S, [])
F1 → * : (rreq, S, D, id, sn, mac-S, [F1])
F2 → * : (rreq, S, D, id, sn, mac-S, [F1, F2])
D → F2: (rrep, S, D, id, sn, [F1, F2], mac-D)
F2 → F1: (rrep, S, D, id, sn, [F1, F2], mac-D)
F1 → S : (rrep, S, D, id, sn, [F1, F2], mac-D)

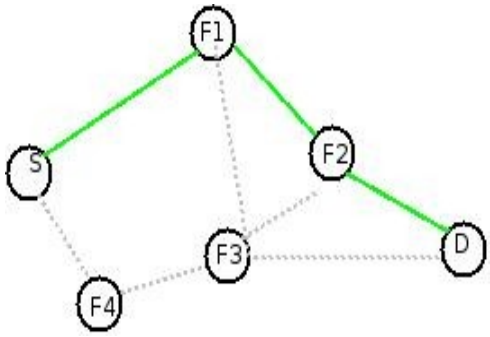


Figure 3.3: SRP message format and forwarding example

An operational example and message format is shown in Figure 3.3. S denotes the source identifier, D for destination identifier, F1 and F2 are intermediate nodes, '*id*' is a randomly generated query identifier and '*sn*' is a query sequence number maintained by S and D, *mac-S* is computed by S taking the fields *rreq*, S, D, *id* and *sn* and *mac-D* is the MAC computed by D that covers the fields *rrep*, S, D, *id*, *sn* and [F1, F2]. The source S generates a route request message and broadcasts it to its neighbors. The integrity of the route request is protected by a MAC that is computed by the pre-shared key between the source and the destination. Each intermediate node that receives the route request for the first time appends its identifier to the request and re-broadcasts it.

SRP is a very efficient protocol as the route request and route reply messages contain only a single MAC value. Moreover, the MAC value is not processed by the intermediate nodes. SRP is resistant to attacks aiming at route disruption and route diversion from a single adversarial node but not against attacks mounted by colluding

adversaries. SRP assumes that a shared secret is established between the source and destination but do not solve the problem of key agreement.

3.5.2 SEAD

Secure Efficient Ad hoc Distance vector (SEAD) routing protocol is a robust protocol against multiple uncoordinated attacks creating incorrect routing state in any other node, in spite of active attackers or compromised nodes in the network [13]. The design of SEAD is partly based on the concept of DSDV. The protocol uses one way hash functions and do not use asymmetric cryptographic operations. Thus it provides efficient support for nodes with limited CPU processing capability and guard against DoS attacks in which an attacker attempts to cause other nodes to consume excessive network bandwidth.

Each node in SEAD uses a specific single next element from its hash chain in each routing update that it sends about itself. The one-way hash chain is formed from one-way hash function. To create a one-way hash chain, a node choose a random bit string x of length m, computes the list of values $h_0, h_1, \ldots, h_n$, where $h_0 = x$, and $h_i = H(h_{i-1})$ for $0<i<=n$, for some n.

3.5.3 Ariadne

Ariadne is a secure on-demand routing protocol that protects against node compromise and relies on highly efficient symmetric cryptography [14]. It discovers route on demand and the concept is primarily based on DSR. Ariadne can authenticate routing messages in the following ways:

- shared secret between each pair of nodes,
- shared secret between communicating nodes combined with broadcast authentication, or
- using digital signature.

In Ariadne with digital signature not only the source and destination nodes authenticate the messages, but also the intermediate nodes insert their own digital signatures in route requests. In addition, Ariadne uses per-hop hashing to prevent removal of identifiers

from the list of routes in the route request. Ariadne with TESLA [21] is an efficient broadcast authentication scheme that requires loose time synchronization. Use of pair wise shared keys can avoid the need for time synchronization but it costs a higher key-setup overhead. However, it does not elaborate the solution for key agreement to establish the pre-shared secret key between the source and destination nodes.

3.5.4 ARAN

An attempt to secure AODV is the Authenticated Routing for Ad hoc Network (ARAN) which is proposed in [22]. According to the ARAN protocol, each node has a certificate signed by a trusted authority which associates its IP address with a public key. Like AODV, it is an on-demand protocol that maintains route information using route discovery and route maintenance scheme.

To find a destination, the source node broadcasts a signed RREQ message that includes a target, its own certificate, a nonce and a timestamp. The nonce and timestamp together ensure the freshness of the RREQ when used in a network with a limited clock skew. Each intermediate node that forwards the packet checks the signature or signatures. After a successful verification of the signature, an intermediate node signs the original RREQ, removes last forwarder's signatures and includes its own certificate. Since ARAN uses public-key cryptography for authentication, it is computationally intensive. It is particularly vulnerable to DoS attacks based on flooding the network with bogus control packets for which signature verification is required.

3.5.5 SAODV

Secure Ad hoc On demand Distance Vector (SAODV) is another attempt taken by [23] to secure AODV. The idea behind SAODV is to authenticate most immutable fields of a route request (RREQ) and route reply (RREP) and to use hash chain to authenticate the hop count. SAODV adds a single signature extension in RREQ packet (RREQ-SSE) to prevent impersonation attack. Based on the expected network diameter, the source chooses a maximum hop count and generates a one-way hash chain of length equal to

the maximum hop count plus one. This one-way hash chain is much like SEAD which is used as a metric authenticator.

3.5.6 SLSP

Like the on demand routing protocols, measures have been taken to secure proactive protocols too. Secure Link State Protocol (SLSP) uses digital signatures and one-way hash chains to ensure the security of link-state updates. SLSP is a periodic protocol that receives link information through a periodic Neighbor Location Protocol (NLP). As a part of NLP, each node broadcasts a signed pairing between its IP address and its MAC address. SLSP secures non-mutable data using signatures and mutable data using hash chains.

3.6 Summary

We have seen that reactive protocol such as AODV and DSR have their security extensions respectively in the form of SAODV and Ariadne. SAODV uses digital signature for control message and Ariadne authenticate sender by clocked synchronization and delayed key disclosure. ARAN uses authenticated route discovery. DSDV has secure version SEAD which uses hash chain for message authentication.

A lot of research have been done on designing a robust routing protocol for mobile ad hoc network and provide the security in the routing procedure to withstand the attacks launched to disrupt the routing. A number of challenges remain in the area of security in the wireless networks. Firstly the networking environment is not well modeled and thus lacks a formal method to make a comprehensive evaluation of the proposed security protocol. Another problem is to design an efficient routing protocol that has both strong network security and high performance. Although researchers have designed security extensions for several existing protocols, many of these extensions remove important performance optimizations.

Chapter 4

Optimized Link State Routing Protocol (OLSR)

4.1 Introduction

The Optimized Link State Routing (OLSR) protocol is one of the four routing protocols for mobile ad hoc network considered by the IETF MANET working group [6]. OLSR is a proactive routing protocol based on the link state algorithm [8]. The protocol optimizes the retransmission of routing control messages in comparison to other link state protocols. It provides optimal routes calculating the distance of the routes in terms of the number of hops a packet has to travel to reach the destination. Since all nodes in an ad hoc network are routers, compromising any of the nodes could potentially disrupt the routing functionality of the entire network. OLSR is vulnerable to security attack as it does not provide any security measures in its original proposal [5]. As a result, providing security to OLSR is an active field of research and several proposals to extend OLSR exists. In this chapter, we will briefly present the operation of the OLSR routing protocol and a selection of the proposed extensions to secure the protocol.

4.2 Working principle

As like other proactive routing protocols, OLSR uses control messages to broadcast the link states of the network. The two types of control messages that are used by OLSR are HELLO message and Topology Control (TC) message.

4.2.1 HELLO message

HELLO messages are sent periodically by a node and are used for neighbor sensing and MutiPoint Relay (MPR) selection (MPR is described in section 4.2.3). HELLO messages are broadcasted within only 1-hop neighbor and are not retransmitted further.

It contains the list of neighbors from which control traffic is being heard, the list of neighbors with which a symmetric link is established and the list of MPR set that has been selected by the originator of the message.

Upon receiving the HELLO message, a node examines the list of addresses. If it finds its own address in the list then the node assumes that a bi-directional link can be established with the originator of the message. Besides sensing the link status with the neighbors, periodical exchange of HELLO messages gives information about the nodes that are two-hops away. This information is stored as *2-hop neighbor set* and is used for the selection of MPR set.

4.2.2 TC message

TC messages are sent periodically just like HELLO messages but the interval is larger than HELLO messages. The purpose of TC messages is to diffuse link state information to the entire network that will be used for routing table calculation. The message contains the list of bi-directional links between a node and some of its neighbors. The novelty of OLSR lies into the broadcast technique of TC messages as it is flooded into the network exploiting the MPR optimization and thus reducing the number of messages flooded into the network. Only the nodes that are selected as MPR generate and broadcast TC messages.

An individual OLSR packet can contain multiple HELLO or TC messages. All messages are uniquely identified by its originator address and message sequence number from the message header. The basic layout of an OLSR packet is shown in Figure 4.1.

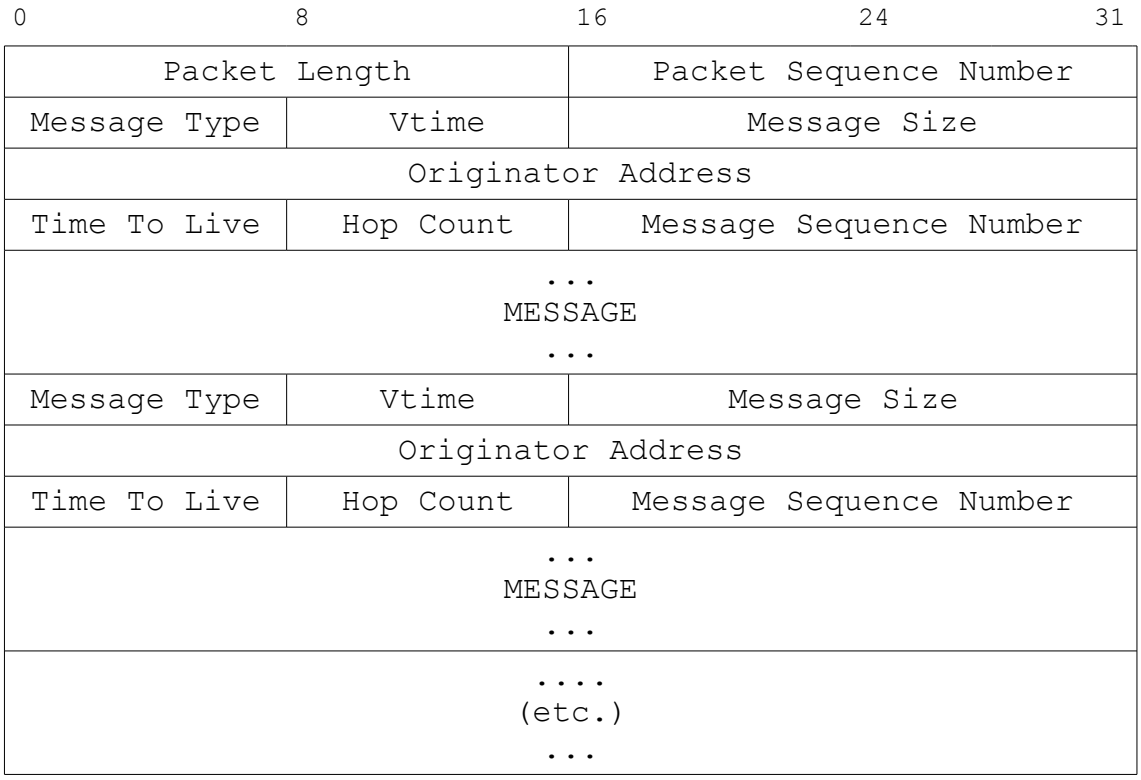


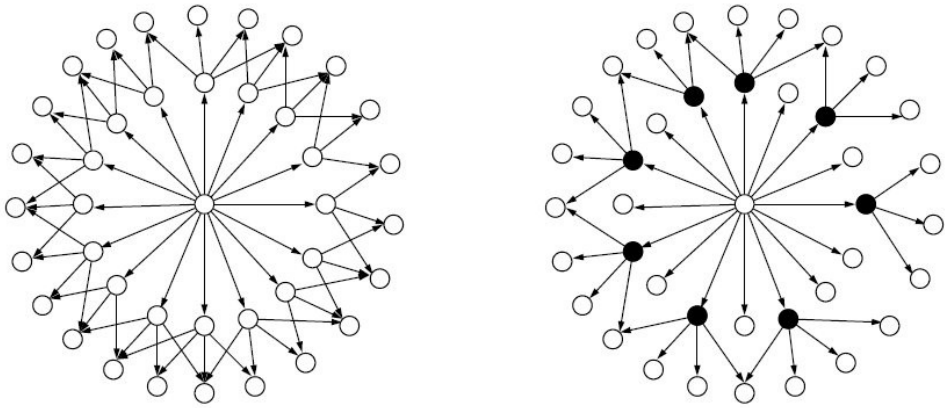
Figure 4.1: OLSR packet format

4.2.3 MPR selection

The core optimization in OLSR is done by the selection of multipoint relay (MPR) nodes. Each node in the network independently selects a set of neighbors which retransmit its packets. This minimizes the broadcast of packets in the network by reducing the duplicate transmission of control messages in the same region. The set of selected neighbor node is called multipoint relays (MPRs) of that node [8] .

MPR selection is done in such a way that all the 2-hops neighbors are reachable from the MPRs in terms of radio range [7] . The 2-hop neighbor set found from the exchange of HELLO messages is used to calculate the MPR set and the nodes signal their MPR selections through the same mechanism. Each node in the network maintains an *MPR selector set*, the set of neighbors that have selected this node as an MPR. This information is broadcasted by the periodical TC messages.

Figure 4.2 (b) shows a typical selection of MPRs around the central node covering all its two-hop neighbors in its radio transmission range. The solid circle represents the multipoint relays which will only broadcast the control messages. Figure 4.2 (a) illustrate the same broadcast domain without the use of MPRs.



(a) (b)

Figure 4.2: The broadcast from the central node is transmitted: (a) by all its neighbors and in (b) by its MPRs only denoted by solid black circles.[28]

4.3 Security extensions

4.3.1 Integrity protected routing

An approach based on the authentication checks of information message injected into the network is proposed in [11]. But it cannot prevent replay attacks. So they also developed a distributed timestamp based approach to verify whether a message is old or new to prevent replay attacks. In this security framework it is assumed that the integrity of a trusted node is not compromised. So only the control traffic from the trusted node is considered in the network. The control messages are cryptographically signed. Both symmetric shared secret key or asymmetric public key cryptographic mechanisms may be deployed. The necessary keys to sign and verify messages are available to each node. A node will generate a signature by using its node id, key and message to send. The receiving node can verify the message by using the originator id, key, message and signature.

Injecting incorrect information into the network from malicious nodes are prevented by using a signature generated by the originator of each OLSR control message and transmitted with the control message. In addition, message freshness is determined by a timestamp associated with each signature. After receiving the control message a node can verify whether the message is from a trusted node and if its integrity is preserved. Signature is sent as a separate type of OLSR control message as shown in Figure 4.3.

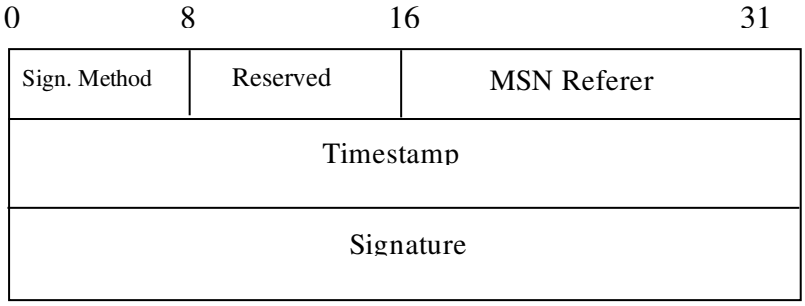


Figure 4.3: OLSR control message with signature

4.3.2 SA-OLSR

Security Aware Optimized Link State Routing (SA-OLSR) protocol is a proposed extensions to the plain OLSR based on exchanging acknowledgment (ACK) between 2-hop neighbors when the control traffic is successfully received. This approach protects many sophisticated attacks mentioned earlier such as, link spoofing, replaying and wormhole attacks without requiring any location information or any knowledge of complete network topology [17]. With an authentication mechanism in work to identify the exact origin of each packet which prevent malicious node from sending forged ACK, this method requires only 2-hop neighbors to send acknowledgment back to the TC source (ACK_{TC}), the originator of the TC message. It also maintains a trust table which contains 2-hop neighbor link tuple and its trust value in order to observe behavior of its own 1-hop and 2-hop neighbors. Hence to detect the unreliable links, two new elements are added:

- Acknowledgment message (ACK_{TC}): ACK_{TC} is used to assure that TC messages are successfully received at the node's 2-hop neighbors.

- Trust Table: Each node maintains a trust table which contains 2-hop neighbor link tuple and its trust value as shown in Table 4.1 to observe its own 1-hop and 2-hop neighbors. The initial trust value is set to 0.

Table 4.1: Example of Trust Table used in [17]

1-ho neighbors	2-hop neighbors	Trust value
N1_A	N2_A	0
...	...	...
N2_K	N2_K	0

The protocol works as following:

1. When a node receives a TC message, it checks whether the source node is its 2-hop neighbor. If the source node is a 2-hop neighbor then it generates an ACK_{TC}, otherwise it does not.
2. When a node receives an ACK_{TC} for its sent TC message, it considers the link towards the source of the acknowledgment truly exists and set the trust value of the 2-hop neighbor to 1.
3. If there exists any 2-hop neighbor link tuple with a trust value of 0, a node considers that this link is not reliable. It assumes that this 2-hop neighbor is fake or the packets might be dropped by that malicious node.
4. During MPR selection, a node avoids selecting the node in the suspicious link tuple as its only MPR.

Thus this approach can give protection against:

1. Link spoofing attack: Link spoofing will cause the trust value of the fake link to be 0 and hence can be detected.
2. Colluding misrelay attack: In case of colluding misrealy attack, a target node will not receive ACK_{TC} from its two hop neighbors and hence it can detect any anomaly due to this attack.
3. Wormhole attack: It is possible to detect wormhole attacks by calculating the delay between the time a node sends a TC and the time a node receives the corresponding acknowledgment ACK_{TC}.

4.4 Summary

OLSR is a robust routing protocol for dense mobile ad hoc network and many proposals for security extensions are available for it. Mobile ad hoc networks for emergency rescue operations or military usage demand a high level of security for seamless operation. But none of the above security extensions give a proper and robust solution which can be implemented in the OLSR protocol. Thus we present our proposal for a new security extension for secure routing with OLSR in the next chapter.

Chapter 5

New Security Extensions to OLSR

In the new security solution, OLSR is chosen as the base routing protocol. The solution provides access control to the network and confidentiality of data transmission. The access control is designed using PKI based protocol and the confidentiality in the link encryption is maintained using symmetric keys derived from the key agreement under PKI. The following sections describe how to establish the PKI and link encryption specified in [1]. Hence our thesis work is to simulate the new security extensions, here we are presenting a short description of the protocol taken from [1, 9-10].

5.1 PKI

The mobile network is established without possible present of an infrastructure network and having nodes with limited processing power. So it is not feasible to set up a full PKI based first responder ad hoc network. Thus we define that all units in a certain geographical area must have a certificate issued by a common Certificate Authority (CA). These certificate should have a limited validity (e.g. six months) and should be renewed within a very short period of time (e.g. three months). It is an inherent advantage of public key cryptography that the certificates require no special protection and could be transmitted over any network and subsequently be verified by the client. The self-signed CA certificate must have a significantly longer validity period.

The nature of a first responder network in our discussion often requires a cooperative network of cross border units. In this case both units need to have a cross-signed certificate signed by both authorities. In such areas, the personnel will routinely acquire cross signed certificates. Moreover if the command vehicle has access to the infrastructure network, it can acquire cross signed certificates immediately. Usually these certificates have a longer validity period while certificates with shorter validity period can be issued for the voluntary workers or to other nodes that cannot acquire

cross-signed certificates due to connectivity or any other problems. The cross signed certificate scheme can handle the certificate revocation list in a manageable way.

5.2 Access control

Access control is essential to prevent the misuse and exhaustion of the sparse resources – computing power, battery lifetime, bandwidth etc. available for the mobile nodes in the network. The access control mechanisms rely on the PKI scheme for secure authentication of the users. A node wish to enter into the network first presents its credentials and role in the form of certificate. The certificate is signed by a certificate authority that is trusted by the nodes in the network so that the authenticity of the requesting node can be verified. The nodes will subsequently enter into a challenge-response protocol described next to prove that it holds the private key corresponding to the public key certificate. If this protocol ends in a success then the node is granted access to the network according to the policy defined for the credentials and roles presented in the certificates.

5.3 Protocol description

Here we will present the proposed solution from [1] that we have implemented in NS-2. To refer to the new security protocol, we will refer it as SOLSR (Secured OLSR). As described in chapter 4, the routing data of OLSR is diffused through HELLO and TC messages. Hence we have provided security to the following steps in the routing process,

- Key establishment during link sensing and neighbor discovery
- After key establishment, protection of HELLO messages
- Protection of TC messages during topology discovery

5.3.1 General protocol operation

5.3.1.1 Packet format

The new protocol runs over the standard OLSR routing protocol. It does not need to change the general packet format of OLSR or the assigned UDP port by which routing messages are sent. The protocol only redefine OLSR messages to allow transmission of encrypted routing information. The new message types will be accommodated in the `Message Type` field of the OLSR packet presented in Figure 4.1 in Chapter 4.

5.3.1.2 Encrypted message

The link encryption will be provided by sending the new messages in an encrypted format. A general encrypted format is defined to accommodate the link encryption. The general format has not any associated Message Type but can be identified from the Message Type (e.g. `OLSR_ENC_HELLO`) field in OLSR packet. The general message pattern which is used to encrypt new messages is shown in Figure 5.1.

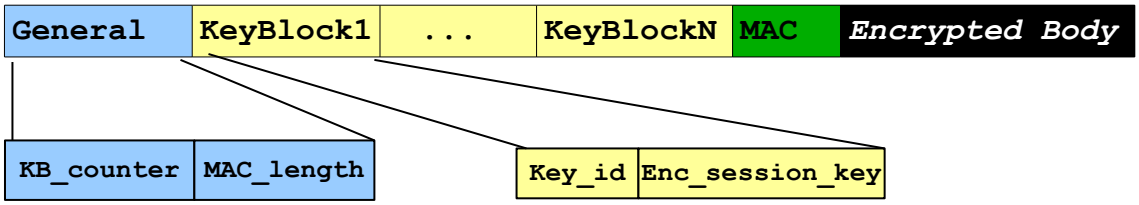


Figure 5.1: General encrypted message format for SOLSR

The message format consists of the following:

- General part - The general part of the message contains the number of key blocks (`KB_counter`) followed by the length of the message authentication code (`MAC_length`).
- Key blocks - There would be `KB_counter` (s) key blocks followed by the general header of the message. Each key block contains the key identifier (`Key_id`) that is used to encrypt the session key and the encrypted session key (`Enc_session_key`) itself.
- `MAC` - The `MAC` is computed over the header and the body part of the message

concatenated with the session key.

- `Encrypted body` - The encrypted body part is the encrypted OLSR message. The message type that is encrypted can be identified from the Message Type in the message header.

5.3.1.3 Message processing

Upon receiving a packet, a node examines each of the message headers. The Message Type field determines how the further processing will be accomplished. The general processing and forwarding conditions of OLSR will be applied to this protocol with some additional conditions for encrypted messages.

For an encrypted message as shown in Figure 4.1, a node first checks whether it shares a session key for encryption with the sender node. If it does not share such a key, the message must be discarded and depending on the message type, a key establishment process will be started. If a shared key has already been established, the node will decrypt the associated session key. The node could find the `Key_id` in the key blocks by matching the `Key_id` that it has stored during the key establishment phase with the sender node. In the next step it must compute the message authentication code and verify the integrity and authenticity of the message. If the MAC verification fails, the message is discarded. On success, the encrypted body is decrypted and further processing is dependent on the message type.

5.3.2 Information repositories

In this section, we describe which information is needed to be stored by each node according to the new protocol. In the OLSR protocol, four main information bases are defined,

- Multiple Interface Association Information Base
- Local Link Information Base
- Neighborhood Information Base
- Topology Information Base

Additionally the new protocol defines

- Group member association base (for multicast)

5.3.2.1 Local link information base

The local link information base stores information about links to one-hop neighbors. In the new protocol we need to store the `keyIDs` for each of the trusted neighbors and their key values. Hence, local link information base is modified to include this information. This data is called *Link Set* and each *link tuple* consists of,

- `L_local_iface_addr` - The interface address of the local node.
- `L_neighbor_iface_addr` - The neighbor interface address.
- `L_SYM_time` - The time until which the link is considered symmetric and key is shared.
- `L_ASYM_time` - The time until which the link is considered heard.
- `L_time` - The time until the entry expires and must be removed.
- *`L_local_KID` - The local part of the key ID.*
- *`L_neighbor_KID` - The neighbor part of the key ID*
- *`L_key_value` - The value of the shared secret key*

New fields

5.3.2.2 Neighborhood information base

The neighborhood information base stores information about neighbors, 2-hop neighbors, MPRs and MPR selectors respectively called as *neighbor set*, *2-hop neighbors set*, *MPR set* and *MPR selector se*t. In the new protocol, only a field is added to the information base of *neighbor set*. A neighbor tuple in the neighbor set consists of the following fields:

- `N_neighbor_main_addr` - The main address of the neighbor.
- `N_status` - Specifies whether the neighbor is symmetric or not.
- `N_willingness` - The nodes willingness to forward packets on behalf of others denoted by an integer value between 0 to 7.

Additionally it will contain,

- *`N_description` - The description derived from the nodes access credentials supplied during the security handshake.*

5.3.3 Link sensing and neighbor discovery

Due to the link encryption scheme in the new protocol, the process of link sensing and neighbor discovery is slightly different from the OLSR protocol. In OLSR, link sensing and neighbor discovery is done through periodical broadcast of HELLO messages containing known links to one-hop neighbors. However, the new link encryption scheme requires the establishment of shared secret key prior to any regular message processing. The protocol handles the situations where HELLO message is received and the sender and receiver do not share a key.

Reserved	LC Counter	Htime	Willingness
Link Code	Reserved	Link Message Size	
Neighbor Interface Address (NIA)			
Neighbor Interface Address			
...			
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			
...			

Figure 5.2: OLSR HELLO message

5.3.3.1 HELLO message

The new encrypted (`OLSR_ENC_HELLO`) message follows the general encrypted message format depicted in Figure 5.1. After decryption of the encrypted body, the HELLO message will be similar to the message defined in OLSR as shown in Figure 5.2.

5.3.3.2 Link codes

Link code in HELLO message gives information about the type of links and neighbors of the following neighbor interface addresses with the node that is advertising the links. Link code currently uses the lower 4 bits of the 8 bits allocated in the message format.

7	6	5	4	3	2	1	0
0	0	0	0	Neighbor Type		Link Type	

Figure 5.3: Link codes in the HELLO message

The definition of link is modified in the new protocol as following:

- UNSPEC_LINK - indicating that no specific information about the links is given.
- ASYM_LINK - indicating that the links are asymmetric (i.e., the neighbor interface is "heard"), or symmetric, but no shared key has been established.
- SYM _LINK - indicating that the links are symmetric with the interface and a shared key has been established.
- LOST_LINK - indicating that the links have been lost.

Neighbor types are described as the following:

- SYM_NEIGH - indicating that the neighbors have at least one symmetrical link with this node.
- MPR_NEIGH - indicating that the neighbors have at least one symmetrical link and have been selected as MPR by the sender.
- NOT_NEIGH - indicating that the nodes are either no longer or have not yet become symmetric neighbors.
- RES_SYM_NEIGH - indicating that the neighbors have at least one symmetrical link and have only restricted access to the network.

5.3.3.3 Encrypted HELLO message

HELLO messages are sent periodically for link sensing, building 2-hop neighbor set, and signaling MPR selection. After decryption, encrypted HELLO messages are processed in the same way as the original OLSR messages. The generation of Encrypted HELLO messages differs from the OLSR message in the way it set the link types.

The information bases are updated from the encrypted HELLO and TC messages. If a pair of node does not share a key, then key establishment process is initiated before accepting any HELLO or TC message. Information bases such as link set and neighbor set are updated and new entries are added during the key establishment process.

5.3.4 Key establishment

In order to establish a share key between 1-hop communicating neighbors, the key establishment process must be completed. The process is triggered by the reception of a encrypted HELLO message and the receiving node finds that it do not have shared key with the sender node. The overview of the main steps is depicted in Figure 5.4.

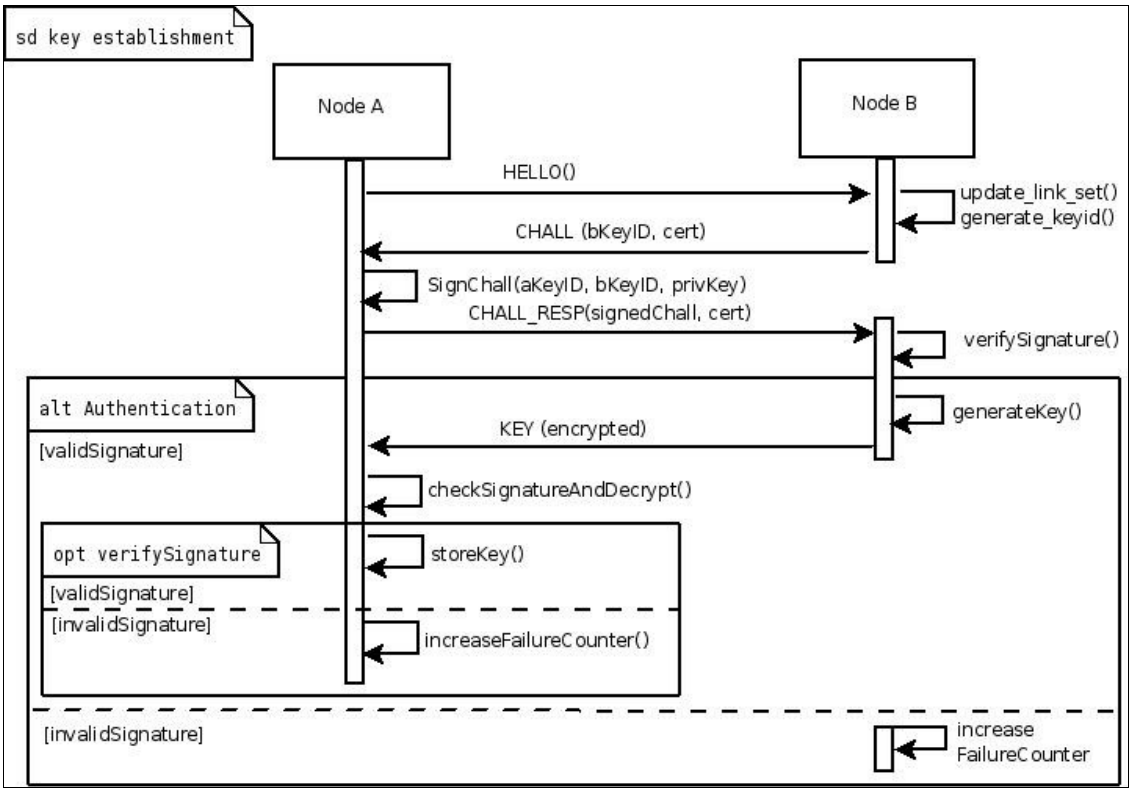


Figure 5.4: Key establishment process

The above process can be described shortly as:

1. Node B receives a HELLO message from node A and discovers that they do not share a secret key.
2. Node B sends a CHALLENGE containing a randomly generated nonce that works as a unique key identifier, and its public key certificate.

3. Node A verifies that the certificate is genuine and generates its own unique key identifier with node B.
4. Node A signs and sends the nonce and both key identifiers along with its certificate.
5. Node B verifies the certificate and signature, and generates a random key.
6. Node B encrypts and signs the generated key and sends it to node A.
7. Node A verifies the signature and stores the key and key identifiers.
8. In the case of a failure in the signature verification process, a failure counter is stepped and further communication is halted.

5.3.5 Message format

In this section we describe the format of the messages that are outlined in the previous sections. We have already mentioned the structure of ENC_HELLO message. The other messages introduced in the protocol are CHALLENGE, RESPONSE and KEY messages. These messages can be stuffed into the general OLSR packets. Though most of the OLSR control messages are delivered after a certain time interval, in case of the messages that are used here for security handshake, these messages must be delivered immediately.

5.3.5.1 CHALLENGE message

The structure of CHALLENGE message is shown in Figure 5.5.

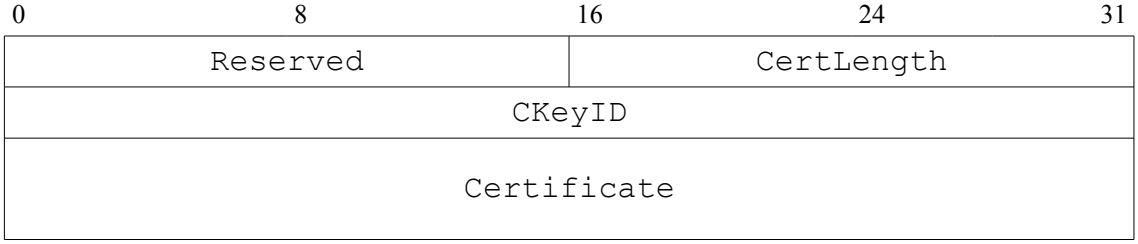


Figure 5.5: Format of CHALLENGE message

The fields are:

- `Reserved` – This field can be used for future extensions. Now it must be zero to adhere to this specification.
- `CertLength` - The length in bytes of the enclosed certificate.

- `CKeyID` - A randomly generated key identifier that is unique for the originator of the message.
- `Certificate` - The public key certificate of the originator.

5.3.5.2 RESPONSE message

The RESPONSE message is sent after getting a certified challenge message from a peer node. At that time the challenger's `keyID` is available. A node sends its own key identifier and credentials in the RESPONSE message. The format is shown in Figure 5.6.

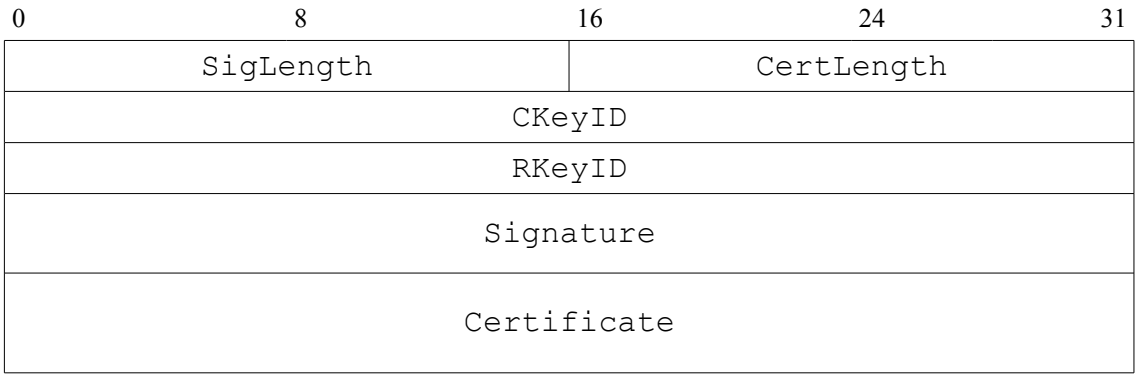


Figure 5.6: Format of RESPONSE message

The fields used in the RESPONSE message are:

- `SigLength` - The length in bytes of the enclosed signature.
- `CertLength` - The length in bytes of the enclosed certificate.
- `CKeyID` - The unique key identifier provided in the corresponding challenge.
- `RKeyID` - A randomly generated key identifier that is unique for the originator of the RESPONSE message.
- `Signature` - A digital signature of the message header and preceding contents.
- `Certificate` - The digital certificate of the originator.

5.3.5.3 KEY message

KEY message is used to send the shared symmetric key by the challenger in the communication process. This is the last message in the security handshake protocol. It contains the encrypted key, key identifier and signature. The message format is shown in Figure 5.7.

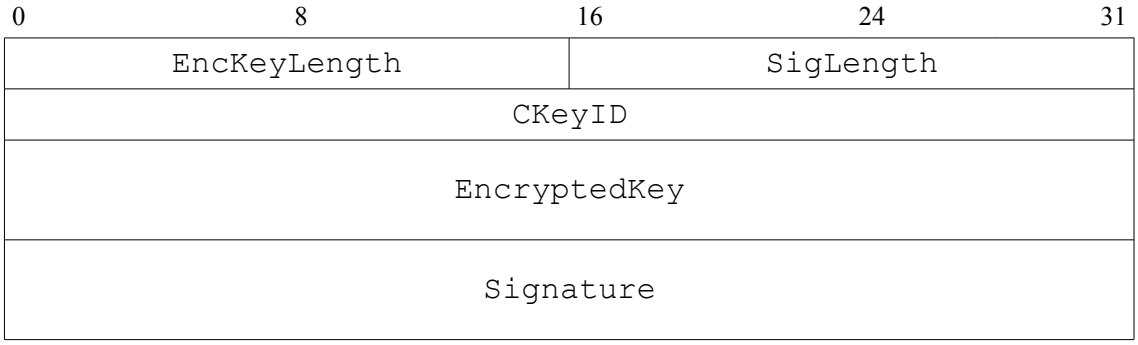


Figure 5.7: KEY message format

A short description of the fields that are used in KEY message format is:

- `EncKeyLength` - The length in bytes of the enclosed encrypted key.
- `SigLength` - The length in bytes of the enclosed signature.
- `CKeyID` - The unique key identifier provided in the corresponding CHALLENGE and RESPONSE messages.
- `EncryptedKey` - The session key encrypted using the public key from the certificate received in the corresponding RESPONSE message.
- `Signature` - A digital signature of the message header and preceding contents.

5.3.6 Message processing and generation

Here we describe shortly the generation and processing of key establishment messages. A detail description of the algorithms used in message generation and processing is added in Appendix B. The CHALLENGE, RESPONSE and KEY messages are not forwarded. Moreover, unlike other broadcast messages used in OLSR, these messages are sent as unicast message to a particular node with which the conversation is going on.

5.3.6.1 CHALLENGE message

A CHALLENGE message is generated when a node receives a HELLO message from a source that it does not share a key with. A node examines the shared link by looking at its Link Set. If it does not have a shared link key with the sender node, the node initiate the challenge-response protocol to establish the shared key. At this time it also updates its Link Set entry with the sender node address and key identifier that it generates. The CHALLENGE message is sent only to the sender node as a unicast message.

Upon receiving a CHALLENGE message the validity of the message is checked and the node updates its Link Set with the neighbor information included in the message. The node also generates a unique local key identifier to identify the shared key with the neighbor.

5.3.6.2 RESPONSE message

A RESPONSE message is generated after a successful reception and processing of CHALLENGE message. The message is sent to give proof of the public key certificate that a node contains and it synchronizes the key identifier with the neighbor node too.

The processing of RESPONSE message after validation includes the update of neighbor entry, Link Set and generation of KEY message.

5.3.6.3 KEY message

Key message is generated after a successful reception and processing of RESPONSE message. To generate a KEY message, a random key is generated and the Link Set is update with the key and its identifier, then the key is encrypted with the neighbor's public key extracted from the neighbor's certificate.

Upon receiving the KEY message, a node updates its Link Set adding the decrypted shared key with the corresponding key identifier and further routing messages are encrypted by this shared secret key.

5.3.7 Topology discovery

Topology Control (TC) messages are used to discover the overall network topology outside the 2-hop neighbors of a node. In the new protocol, TC messages are encapsulated in the same way as HELLO messages. A decrypted TC message takes the form like an OSLR TC message except it has an extra field of Node Description as shown in Figure 5.7.

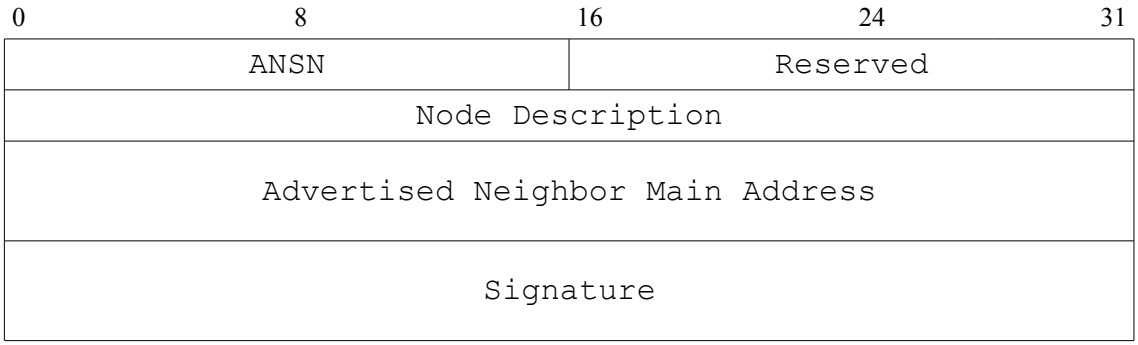


Figure 5.8: Structure of a TC message after decryption

The fields are:

- `ANSN` - Advertising Node Sequence Number is used to prevent the processing of updates that are out of order. This is incremented after every change in advertised neighbor set.
- `Reserved` – This field is reserved and must be set to zero to be in compliance with this specification.
- `Node Description` - A bit string representing the node description extension of the advertising node's certificate.
- `Advertised Neighbor Main Address` - The main address of a neighbor that the advertising node shares a symmetric link with.
- `Signature` - The digital signature of the node sending the message.

TC messages are generated in the same way as it is in the original OLSR protocol, except from that it adds a new field for node description. Special care has to be taken so that no node can change the unsigned node description.

Upon receiving a TC message, the validity of the message is checked and the contents are decrypted to update the Topology Set. A TC message received from a node with no shared secret key is silently discarded.

5.4 Summary

In this chapter we have presented a short description of the new security extension to optimized link state routing protocol. We have discussed how the public key infrastructure control the access to the secured wireless network and provide credentials to validate the nodes that are willing to join the network. The routing message generation and processing is also discussed. New routing messages are used to complete the security handshake to establish a shared key between two nodes in the network. When the symmetric key is established, subsequent routing messages are encrypted with this symmetric key. In the next chapter we will present how we have implemented the security extension in the network simulator.

Chapter 6

Implementation of the New Security Extensions to OLSR in NS-2

The implementation of the new security extension is written as a modified protocol of existing OLSR implementation by University of Madrid [26]. This chapter briefly presents the implementation and modification of NS-2 source code. The elegant design of NS-2, it modifiability and open source property enable us to extend the simulator to adapt with the new protocol. We will discuss on the new data structures and functions to be added for the new secured routing agent.

6.1 Network Simulator 2 (NS-2)

The Network Simulator 2 (NS-2) is a popular discrete event simulator developed mainly for networking research [24]. It is an open source software developed at USC/ISI. It provides an extensive simulating environment for applications, protocols, data sources, network types, traffic models and network elements. In NS-2, the system is modeled as sequential events which happens in an instant of virtual time, but takes an arbitrary amount of real time. The software is designed to separate the processing of data and control. NS-2 is designed having a dual approach as, C++ for core functionality and OTcl for scripting purposes. The core of NS is written in C++, which handle data processing. The Object TCL (OTCL) scripting language is used for writing control script to run the simulation. The rationale for this is that the protocol implementation requires a powerful language (here C++) for faster per packet processing. And the use of script language makes the writing and change of simulation configuration faster to adjust with desired parameters.

NS-2 is also accompanied by the network animator (NAM) that gives a Graphical User Interface (GUI) and visualization of the network that is designed and simulated using

NS-2. For MANET, NS-2 provides a comprehensive library for ad hoc routing and mobile IP, topology generators, propagation models, mobility models and data sources. To run any simulation in NS-2, the scenario is defined using TCL script. The simulation generates a trace file containing data about packets sent, received, forwarded, dropped, size of packets, type of packets etc. for further analysis.

Although NS-2 provides a great number of networking models and protocols, it does not provide all. Our implemented protocol is based on OLSR routing protocol. But it is not included in the original NS-2 (version 2.33). We have implemented our protocol extension based on the OLSR implemented by University of Madrid (UM-OLSR) [26]. Our implemented source code and compiled executables with the security extension can be found with the accompanying compact disc.

Since NS-2 is an open source software, it allows modification and adding new protocols or any other components to it. In our way to add the new security protocol to NS-2, in this chapter, we will discuss the basic structure of NS-2 and the modification that we have done.

6.2 NS-2 directory structure

To ease the installation of NS version 2, it comes with a release called *ns-allinone*, so that all the dependent packages are installed from the one integrated package. NS-2 needs TCL interpreter to run the script and network animator to show the network design and data transmission in a graphic screen. Figure 6.1 shows a typical directory structure of NS-2 installed in our Ubuntu Linux machine. The basic C++ source code is placed in the `ns-2.33` directory. Some example and test codes are given in `tcl` directory. We have added the UM-OLSR [26] source code in the `ns-2.33/olsr` directory and performed necessary modification for our protocol. Recompiling the source code by `make` command in the `ns-2` directory, creates the executable code for the modified protocol.

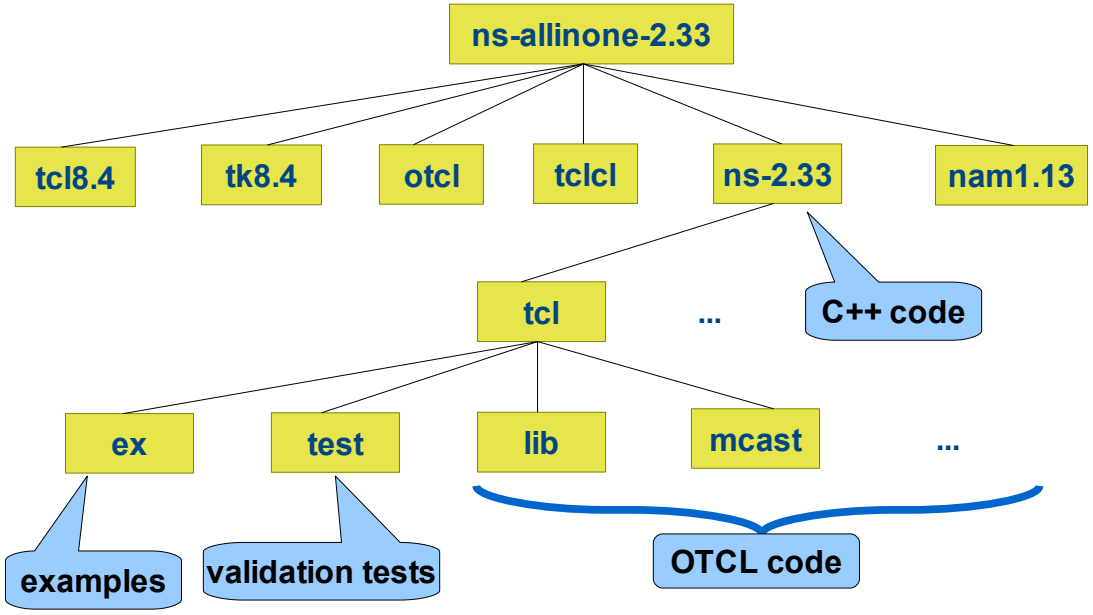


Figure 6.1: NS-2 directory structure

6.3 Class hierarchy in NS-2

Figure 6.2 shows a partial class hierarchy of the NS components. The full description and complete class hierarchy can be found at [25]. In the class hierarchy, the root of all OTcl library objects is the `TclObject`, such as scheduler, timers, network components, NAM related objects etc. `NsObject` is the superclass of all basic network components that handle packets and it is further divided into two descendants: `Connector` and `Classifier`. The `Connector` class represents the network objects that have only one output data path. All network nodes that handle the packets are subclass of `Agent` and we have to modify one of the agents called `OLSR` to add new packet processing functionalities for our protocol.

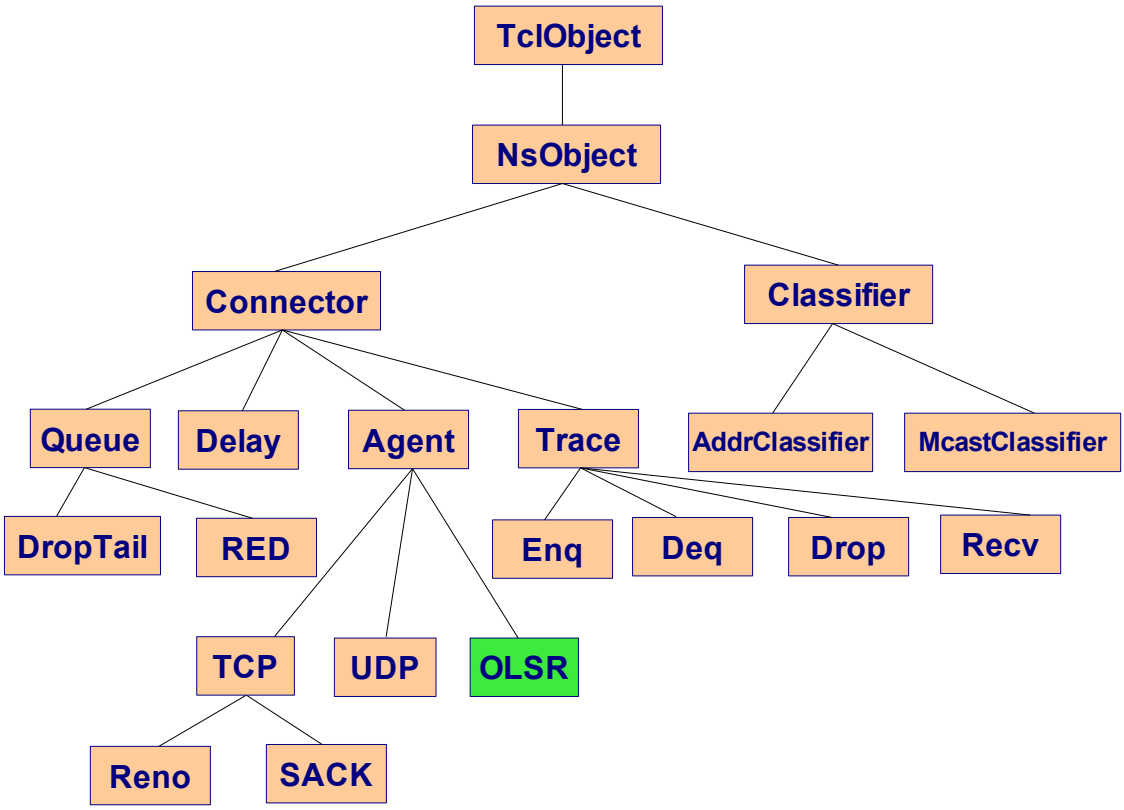


Figure 6.2: Class hierarchy in NS-2

6.4 Header file extension

The following header files contains all the declarations of OLSR related functionalities and are modified to include all related constants and macros.

- `OLSR.h` – for OLSR agents and related classes, variables and functions.
- `OLSR_pkt.h` - for OLSR packets and message types.
- `OLSR_repositories.h` – for OLSR information base.

6.4.1 New message headers

To accommodate our new messages, five new message types are added in `OLSR_pkt.h` file. We have numbered the message from 130 as the first 127 numbers are reserved for original OLSR messages, though only the first three numbers of 127 were used for HELLO, TC and MID messages.

```
#define OLSR_ENC_HELLO_MSG          130
#define OLSR_ENC_TC_MSG             131
#define OLSR_ENC_CHALLENGE_MSG      132
#define OLSR_ENC_RESPONSE_MSG       133
#define OLSR_ENC_KEY_MSG            134
```

6.4.2 New constants

The following new constants are also declared in order to define the data structure needed to handle new message types and relevant information.

```
#define OLSR_ENC_HELLO_HDR_SIZE     4
#define OLSR_KEY_ID_SIZE            4
#define OLSR_MAX_KB                 16
//size in bytes, 32x8=256 bits
#define OLSR_MAX_MAC_LENGTH         32
#define OLSR_MAX_SESSION_KEY_LENGTH 16
#define OLSR_MAX_ENC_HELLO_BODY     OLSR_HELLO_HDR_SIZE +
      (OLSR_HELLO_MSG_HDR_SIZE + OLSR_MAX_ADDRS*4)*OLSR_MAX_HELLOS
//assumption on certificate length, size in bytes, 256*8= 2048 bits
#define OLSR_MAX_PUBLIC_KEY_LENGTH  256
#define OLSR_MAX_CERT_LENGTH        2*OLSR_MAX_PUBLIC_KEY_LENGTH
#define OLSR_MAX_SIGN_LENGTH        2*OLSR_MAX_PUBLIC_KEY_LENGTH
```

6.5 Data structures

Following data structures are added to define new packet format and message types. Adhering to the definition style of NS, we have added the data structures in `OLSR_pkt.h` header file.

The key block structure is used to include the key blocks as presented in section 5.3.1.

```
typedef struct OLSR_key_block{
      u_int32_t   key_id_;
      typedef char      session_key_t[OLSR_MAX_SESSION_KEY_LENGTH];
      session_key_t     enc_session_key_;

      inline u_int32_t&       key_id()    { return key_id_;}
      inline session_key_t&   enc_session_key() {
            return enc_session_key_;
      }

      inline u_int32_t size() {
            u_int32_t sz      =     OLSR_KEY_ID_SIZE;
            sz                +=    OLSR_MAX_SESSION_KEY_LENGTH;
            return sz;
      }
} OLSR_key_block;
```

The message formats described in section 5.3.5 are accommodated in the following data structures.

The encrypted hello message contains a collection of key blocks, the original hello message encrypted with symmetric key and message authentication code.

```
typedef struct OLSR_enc_hello{
      u_int16_t KB_counter_;  // number of key blocks in this message
      u_int16_t MAC_length_;
      OLSR_key_block key_block_[OLSR_MAX_KB];
      typedef char MAC_t [OLSR_MAX_MAC_LENGTH];
      MAC_t MAC_;
      OLSR_hello hello_;
      inline u_int16_t& KB_counter()      { return KB_counter_; }
      inline u_int16_t& MAC_length()      { return MAC_length_; }
      inline MAC_t&          MAC()        { return MAC_; }
      inline      OLSR_hello& hello()     { return hello_; }
      inline OLSR_key_block&  key_block(u_int16_t i)        {
            return key_block_[i];
      }
      inline u_int32_t size() {
            u_int32_t sz = OLSR_ENC_HELLO_HDR_SIZE;
            for (int i = 0; i < KB_counter_; i++)
                  sz += key_block(i).size();
            sz+= OLSR_MAX_MAC_LENGTH;
            sz+= hello_.size();
            return sz;
      }
} OLSR_enc_hello;
```

The `enc_tc_msg` will be included in `enc_tc` message.

```
typedef struct OLSR_enc_tc_msg {
      u_int16_t   ansn_;      //Advertised Neighbor Sequence Number.
      u_int16_t   reserved_;
      u_int32_t   node_description_;
      /// List of neighbors' main addresses.
      nsaddr_t    nb_main_addrs_[OLSR_MAX_ADDRS];
      /// Number of neighbors' main addresses
      int         count;

      inline      u_int16_t&  ansn()            { return ansn_; }
      inline      u_int16_t&  reserved()        { return reserved_; }
      inline      u_int32_t&  node_description(){return
            node_description_; }
      inline      nsaddr_t&   nb_main_addr(int i)     { return
            nb_main_addrs_[i]; }
      inline      u_int32_t size() { return OLSR_TC_HDR_SIZE + 4 +
            count*ADDR_SIZE; }
} OLSR_enc_tc_msg;
```

The enc_tc is a collection of enc_tc_msg in general encrypted message format defined in section 5.3.1.

```
typedef struct OLSR_enc_tc{
      u_int16_t KB_counter_;  // number of key block in this message
      u_int16_t MAC_length_;
      OLSR_key_block key_block_[OLSR_MAX_KB];
      typedef char MAC_t [OLSR_MAX_MAC_LENGTH];
      MAC_t MAC_;
      OLSR_enc_tc_msg tc_;
      inline u_int16_t& KB_counter()      { return KB_counter_; }
      inline u_int16_t& MAC_length()      { return MAC_length_; }
      inline MAC_t&           MAC()       { return MAC_; }
      inline      OLSR_enc_tc_msg&  tc()        { return tc_; }
      inline OLSR_key_block&  key_block(u_int16_t i)        {return
            key_block_[i];}
      inline u_int32_t size() {
            u_int32_t sz = OLSR_ENC_HELLO_HDR_SIZE;
            for (int i = 0; i < KB_counter_; i++)
                  sz += key_block(i).size();
            sz+= OLSR_MAX_MAC_LENGTH;
            sz+= tc_.size();
            return sz;
      }
} OLSR_enc_tc;
```

The structure for challenge message is following:

```
typedef struct OLSR_enc_challenge{
      u_int16_t reserved_;
      u_int16_t cert_length_;
      u_int32_t key_id_;
      typedef char incumbent_cert_t[OLSR_MAX_CERT_LENGTH];
      incumbent_cert_t incumbent_cert_;

      inline u_int16_t& reserved()  {return reserved_;}
      inline u_int16_t&   cert_length()   { return cert_length_;}
      inline u_int32_t& key_id()    {return key_id_;}
      inline incumbent_cert_t& incumbent_cert() {return
            incumbent_cert_;}
      inline u_int32_t size() {
            u_int32_t sz = 4;//header
            sz+=4;      //key id
            sz+=OLSR_MAX_CERT_LENGTH;
            return sz;
      }
} OLSR_enc_challenge;
```

Response message is defined as follows:

```
typedef struct OLSR_enc_response{
      u_int16_t sign_len_;
      u_int16_t cert_len_;
      u_int32_t ckey_id_;
      u_int32_t rkey_id_;
      typedef char joiner_sign_t[OLSR_MAX_SIGN_LENGTH];
      joiner_sign_t joiner_sign_;
      typedef char joiner_cert_t[OLSR_MAX_CERT_LENGTH];
      joiner_cert_t joiner_cert_;

      inline u_int16_t& sign_len()  {return sign_len_;}
      inline u_int16_t& cert_len()  {return cert_len_;}
      inline u_int32_t&       ckey_id()   {return ckey_id_;}
      inline u_int32_t&       rkey_id()   {return rkey_id_;}
      inline joiner_sign_t&   joiner_sign()     {return joiner_sign_;}
      inline joiner_cert_t&   joiner_cert()     {return joiner_cert_;}

      inline u_int32_t size() {
            u_int32_t sz=0;
            sz+=4; // sign and cert length
            sz+= 8;//A and B key id
            sz+=OLSR_MAX_SIGN_LENGTH;
            sz+=OLSR_MAX_CERT_LENGTH;//joiner certificate
            return sz;
      }
} OLSR_enc_response;
```

The definition for sending key message is following:

```
typedef struct OLSR_enc_key{
      u_int16_t   key_length_;
      u_int16_t   sign_length_;
      u_int32_t key_id_;
      typedef char enc_key_t[OLSR_MAX_PUBLIC_KEY_LENGTH];
      enc_key_t enc_key_;
      typedef char signature_t[OLSR_MAX_SIGN_LENGTH];
      signature_t signature_;

      inline u_int16_t& key_length()      {return key_length_;}
      inline u_int16_t& sign_length()     {return sign_length_;}
      inline u_int32_t& key_id()    {return key_id_;}
      inline enc_key_t& enc_key()   {return enc_key_;}
      inline signature_t& signature()     {return signature_;}

      inline u_int32_t size() {
            u_int32_t sz =    4;    // header
            sz+=  OLSR_KEY_ID_SIZE;
            sz+=  OLSR_MAX_PUBLIC_KEY_LENGTH;
            sz+=  OLSR_MAX_SIGN_LENGTH;
            return sz;
      }
}OLSR_enc_key;
```

6.6 Message handling functions

The main behavior of OLSR agent in NS-2 is implemented in `OLSR.cc` file. We have redefined the behavior of OLSR agents so that it includes the security mechanism that has been defined in our new protocol. For this purpose, new functions are added to the source code and a brief discussion is presented in this section. The functions are written to handle the key establishment protocol defined in 5.3.4.

6.6.1 Send encrypted packet

Functions to send HELLO and TC messages in OLSR are written to send broadcast messages to announce an agent's neighbors and MPR selectors. In our security extension, during the security establishment process, an agent needs to send unicast messages to the node with which a key negotiation is going on. For that purpose, we have written a new function for OLSR agent, `void OLSR::send_enc_pkt()` that create a packet for a particular message and send immediately to the destination specified in the function parameter.

6.6.2 Send encrypted hello

OLSR is a proactive protocol that sends regular HELLO messages for link sensing after the HELLO interval is expired. In the implementation it is triggered by the `OLSR_HelloTimer::expire()` function. So when the HELLO interval time is expired, we call the `send_enc_hello()` function to broadcast the encrypted HELLO message from our OLSR agent. The HELLO message is generated following the algorithm presented in Appendix B.

6.6.3 Process encrypted hello

The receive function `recv()` of the `Agent` class handle any packet received by this particular agent. The OLSR agent checks if the received packet is an OLSR packet and if it is then it starts processing the packet in the `recv_olsr()` function. OLSR packet can contain multiple type of OLSR messages specially HELLO and TC messages. Depending on the type of messages, corresponding message handling functions are called. In case of `OLSR_ENC_HELLO` message is received, the function

`process_enc_hello()` is called to process the encrypted HELLO message. This time a unique key identifier is generated to be included in the challenge message that will be sent by this node.

6.6.4 Send challenge

When processing the HELLO message, if the agent finds that there is no shared secret key with the node that sends the HELLO message, it have to send a challenge to the source of the message to start the key establishment process. The `send_enc_challenge()` function sends a challenge to the source of the HELLO message. Note that, following this function the new transmissions are unicast message to a particular node in the security handshake process, it will not be a broadcast message like HELLO or TC messages.

6.6.5 Process challenge

Upon receiving a challenge message, a node call its `process_enc_challenge()` function to handle the challenge message. Usually when a node gets a challenge message from a node, the source node will not be in its Link Set, since a node in the Link Set denotes that the node is already trusted and security handshake is finished with it. Therefore, receiving a challenge message from a node means that the node is a new neighbor and the Link Set of this node has to be updated to add the source of the challenge message.

6.6.6 Send response

After processing the challenge, a node has to send a response to the source node of the challenge message. The `send_enc_response()` function is written to send a response to the node from where a challenge is received. The function prepares a proper message header for `OLSR_ENC_RESPONSE` message and adds signature and certificate of the node to authenticate itself to the challenger.

6.6.7 Process response

When the challenger in key establishment process gets the response message of its corresponding challenge message, it starts processing the response message by calling the `process_enc_response()` function. This function verifies the signature of the neighbor node. If the signature is valid then it updates the neighbor key identifier that is sent in the response message. A valid signature in the response message authenticates the neighbor as a trusted participant in the network. At this point the neighbor set is also updated with the neighbor's information.

6.5.8 Send key

The `send_enc_key()` function generates an encryption key and send it to the neighbor from which a successful response message is received. The function adds generated link encryption key and corresponding key identifier to produce the `OLSR_ENC_KEY` message and sends it to its newly trusted neighbor.

6.5.9 Process key

When a node receives an `OLSR_ENC_KEY` message it call its `process_enc_key()` function to finish the key establishment process. The node has to update its link tuple to add the encryption key with the corresponding neighbor tuple. The neighbor set is also updated to include the new neighbor. At this stage the key establishment process is finished.

6.5.10 Send TC

The topology control (TC) messages are sent by an agent when the TC timer is expired, i.e. `void OLSR_TcTimer::expire` function is called. The `expire()` function calls the `send_enc_tc()` function to broadcast encrypted TC message. TC message are sent following the same procedure as in the original OLSR except that the message is encrypted.

6.5.11 Process TC message

When a node receives an `OLSR_ENC_TC` message it calls the `process_enc_tc()` function that decrypts the TC message and update the topology set to include the information received in the TC message.

6.6 Trace output format for new messages

When we run the simulation with our new extension to the OLSR routing protocol, the simulator will produce a descriptive trace file while sending and receiving the packets at the nodes configured in the simulation. In NS-2, the trace format for wireless network is defined in the `cmu-trace.cc` file. Our implementation can produce trace information for both old and new format. We have to add the following code in the **`void CMUTrace::format_olsr`** method of the **`ns/cmu-trace.cc`** source file.

6.6.1 Tagged packet format

For tagged packet we have add the following conditions (from line number 965) in the source code.

```
else if (op->msg(i).msg_type() == OLSR_ENC_HELLO_MSG)
      s = "-olsr:t EHELLO -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_TC_MSG)
      s = "-olsr:t ETC -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_CHALLENGE_MSG)
      s = "-olsr:t ECH -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_RESPONSE_MSG)
      s = "-olsr:t ERES -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_KEY_MSG)
      s = "-olsr:t EKEY -olsr:o %d -olsr:h %d -olsr:ms %d ";
```

6.6.2 Supporting new trace format

To support the new trace format in `cmu-trace` file, our protocol extension needs the following code to be added in the trace file generator source code (relative line number 999 when the above code for tagged condition is added).

```
else if (op->msg(i).msg_type() == OLSR_ENC_HELLO_MSG)
      s = "-olsr:t EHELLO -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_TC_MSG)
      s = "-olsr:t ETC -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_CHALLENGE_MSG)
      s = "-olsr:t ECH -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_RESPONSE_MSG)
      s = "-olsr:t ERES -olsr:o %d -olsr:h %d -olsr:ms %d ";
else if (op->msg(i).msg_type() == OLSR_ENC_KEY_MSG)
      s = "-olsr:t EKEY -olsr:o %d -olsr:h %d -olsr:ms %d ";
```

6.6.3 Old trace format

Usually we use the default old trace format for wireless network simulation. The following lines will produce information about the packets in the trace file. (The code has to be added at line number 1033, considering the above modified code is inserted).

```
else if (op->msg(i).msg_type() == OLSR_ENC_HELLO_MSG)
      s = "[EHELLO %d %d %d]";
else if (op->msg(i).msg_type() == OLSR_ENC_TC_MSG)
      s = "[ETC %d %d %d]";
else if (op->msg(i).msg_type() == OLSR_ENC_CHALLENGE_MSG)
      s = "[ECH %d %d %d]";
else if (op->msg(i).msg_type() == OLSR_ENC_RESPONSE_MSG)
      s = "[ERES %d %d %d]";
else if (op->msg(i).msg_type() == OLSR_ENC_KEY_MSG)
      s = "[EKEY %d %d %d]";
```

6.6.4 Trace file output

When we modify the trace file accordingly, we can get a trace file in the output as shown in Table 6.1. [*Sample output in the table is taken from a trace file and is truncated and some intermediate output is skipped too.*]

Table 6.1: Trace file output

```
1. s 0.000169789 _0_ RTR  --- 0 UM-OLSR 84 [0 0 0 0] ------- [0:255 -
   1:255 32 0] [1 0 [EHELLO 0 0 0]]
2. r 0.001765900 _8_ RTR  --- 0 UM-OLSR 84 [0 ffffffff 0 800] -------
   [0:255 -1:255 32 0] [1 0 [EHELLO 0 0 0]]
3. r 0.023441458 _0_ RTR  --- 1 UM-OLSR 564 [13a 0 8 800] -------
   [8:255 0:255 32 0] [1 0 [ECH 8 0 1]]
4. s 0.023441458 _0_ RTR  --- 9 UM-OLSR 1080 [0 0 0 0] ------- [0:255
   8:255 32 8] [1 2 [ERES 0 0 2]]
5. s 0.058175084 _8_ RTR  --- 14 UM-OLSR 820 [0 0 0 0] ------- [8:255
   0:255 32 0] [1 1 [EKEY 8 0 2]]
6. r 0.218348707 _7_ RTR  --- 42 UM-OLSR 212 [0 ffffffff 2 800]
   ------- [2:255 -1:255 32 0] [2 4 [EHELLO 2 0 0][EHELLO 2 0 5]]
7. r 5.047169022 _1_ RTR  --- 224 UM-OLSR 500 [0 ffffffff 7 800]
   ------- [7:255 -1:255 32 0] [4 28 [ETC 6 1 19][ETC 7 0 33][ETC 8 1
   26][EHELLO 7 0 34]]
8. r 5.255878480 _5_ RTR  --- 237 UM-OLSR 168 [0 ffffffff 8 800]
   ------- [8:255 -1:255 32 0] [2 23 [ETC 2 1 28][ETC 0 1 22]]
```

Line 1 shows the EHELLO message (encrypted HELLO) that is sent as a broadcast message. Line 2 shows that node 8 received an EHELLO message from node 0. Upon receiving an EHELLO from node 0, node 8 initiated a challenge-response by sending a challenge message (ECH) to node 0. Node 0 sent a response message (ERES) and after receiving the proper response Node 8 sent the encrypted key in the EKEY message as shown in line 5. We know that in OLSR it is possible to encapsulate multiple HELLO and TC message in one packet and broadcast them after the HELLO or TC time expires [5]. In line 6 and 8 we can find that multiple EHELLO and ETC messages are sent in a packet and both EHELLO and ETC messages are advertised bundled in one packet which is shown in the trace file at line 7.

Chapter 7

Simulation and Performance of SOLSR

The main purpose of the simulation of our security extension in the ad hoc network routing protocol is to perform a feasibility study prior to further development of the modified protocol. We have prioritized the route discovery and unicast communication but not the group communication which is also proposed in the security extension [1]. In this chapter we will present the simulation results that we have performed on Security extension to OLSR (SOLSR) and OLSR with varying metrics.

7.1 Simulation model

In the simulation, cryptographic operations on the messages are not performed, but it is simulated by adding some delay during message processing. Contents of routing messages are not encrypted or signed. Instead, flags are used to indicate whether the signature is valid or not. The data overhead for signatures and encrypted payload is added in the routing packets that enable us to measure the performance, throughput and delay in the routing protocol.

7.2 Simulation parameters

Along with usual configuration of the wireless network simulation in NS-2, we have to set the routing protocol as OLSR in the node configuration command as,

```
$ns_ node-config -adhocRouting OLSR
```

The size and frequency of HELLO and TC messages contribute to the routing data overhead. It is possible to configure the HELLO or TC message emission interval from the TCL code. The default values can be set in the `/ns/tcl/lib/ns-defaul.tcl` file or in the running TCL source code. Configuration for all nodes

using OLSR routing protocol can be set by the following way,

```
Agent/OLSR set debug_          false
Agent/OLSR set use_mac_        false
Agent/OLSR set willingness_    3
Agent/OLSR set hello_ival_     2
Agent/OLSR set tc_ival_        5
Agent/OLSR set mid_ival_       5
Agent/OLSR set signed_         1
Agent/OLSR set enc_rate_       100       ;# in MBps
```

The `signed_` variable is used to denote whether the node is signed or not. Only the signed nodes can participate in the network communication. The expected delay for encryption is calculated from the `enc_rate_` given in mega bytes per second.

It is also possible to print the routing table and other internal data structures in the output trace file. This entries are preceded by a `P` (other standard prefix are *D* for dropped, *r* for received, *s* for sent packet, *f* for forwarded and *M* for mobility) in the trace file. Following are the list of functions that print corresponding data structures for node 0 (in the `node_` array).

```
$ns_ at 10.0 "[$node_(0) agent 255] print_rtable"
$ns_ at 15.0 "[$node_(0) agent 255] print_linkset"
$ns_ at 20.0 "[$node_(0) agent 255] print_nbset"
$ns_ at 25.0 "[$node_(0) agent 255] print_nb2hopset"
$ns_ at 30.0 "[$node_(0) agent 255] print_mprset"
$ns_ at 35.0 "[$node_(0) agent 255] print_mprselset"
$ns_ at 40.0 "[$node_(0) agent 255] print_topologyset"
```

An example TCL file is included in Appendix C to run a wireless simulation which uses OLSR as the ad hoc routing protocol.

The details of the computer system that we have used to compile NS-2 and run the simulation are presented in Table 7.1.

Table 7.1: System configuration

Processor	Intel Core 2 Duo1.66 GHz
Operating System	Ubuntu Release 8.04 (Hardy) Kernel Linux 2.6.24-23-generic
Memory	1 GB
C++ Compiler	gcc version 4.2.4 (Ubuntu 4.2.4-1ubuntu3)
TCL/TK version	8.4.18
NAM version	1.13
xgraph version	12.1

Each node in the simulation moves according to the random waypoint model, i.e. after it arrives at a random location, it stays there for a pause time seconds before moving to the next random location and repeats the same process. Here we set the pause time to be 2 seconds. The initial position of the nodes in the dimension area, the start time for moving toward the destination and their speed of moving in the range of 2 to 10 meters per second are set randomly. The code and command parameters that we have used are given in Appendix A.

Table 7.2: Simulation parameters in NS-2

Simulator	Network Simulator version 2.33 (NS 2.33)
Simulation time	200 seconds
Transmission range	200 meters
Traffic type	CBR
Data payload	512 bytes
Packet rate	4 – 2000 packets/sec
Number of total nodes	2/5/10/20
Maximum number of connections	1/8/15/30
Area/Dimension	1000x1000 meters
Node speed/mobility	2-10 m/s

7.3 Performance

The performance of routing protocol in wireless ad hoc networks depends upon several factors:

Number of Nodes: The number of nodes present in the network, which influence the routing table size and transmission of routing updates.

Data rate: When we tried with high data rate the packet drop rate increased. We run our simulation mainly with CBR traffic so that we can compare the routing protocols with the same scenario and with constant data transmission rate.

Mobility: Mobility is the main factor that affects the routing table in wireless networks. Some protocols converge with mobility faster and some protocols do not. We have found that link state protocols converge faster in highly mobile network.

Queue size: Implicitly in NS-2, the queue size for the mobile nodes affect the performance of the routing protocols. If the data rate is low, packet arrived for the nodes that are not in the routing table can be kept in the queue and when the mobile nodes come into range, the packets can be delivered. But with high data rate the queue overflows quickly and if the destination is not reachable then packets are dropped.

We have performed several tests on our routing protocol to observe its performance such as, data overhead, end to end delay etc. comparing to other routing protocols in particular OLSR on which the extension is based. We have mainly focused on OLSR to see how much the routing mechanism is affected by adding security elements to the protocol.

A general case for stationary nodes is presented first. Then Section 7.3.2 to 7.3.4 discusses the performance on data overhead, packet dropping rate and end to end delay for randomly moving nodes in wireless ad hoc network.

7.3.1 Stationary nodes

We have found that when the nodes are stationary, the performance of all the routing protocols is nearly same, in terms of routing table update and packet delivery. We are not considering the routing message overhead here. If there exists a route to the node,

packets are delivered to the target node properly. Although link state protocols normally perform better than on-demand protocols, it has been found that the link state protocols initially drop packets destined to the far away nodes. The reason we have found that link state protocols uses periodical topology control messages to update routes and to gain the knowledge of the whole network. So when the nodes are far away for instance, more than three hops away from a node, it takes some time to receive the topology messages from their neighbor nodes. On average, for the nodes at *n* hops distance, it takes (*n*-1) topology messages to update their route in the routing table. Hence, nodes at (*n*-1) hops away are updated in the routing table after [(*n*-1)* TC_INTERVAL] time for OLSR.

Additionally, nodes with slow mobility exhibit the property of stationary node in a mobile ad hoc network until it crosses the transmission range of their neighbors. Moreover, if the nodes are moving around in the transmission range of their neighbors and do not demand a hand-off to other nodes, the routing table need not to change for those nodes. As a result, routing protocols for nodes moving within a restricted area or nodes moving in low speed in the ad hoc network show nearly similar performance.

7.3.2 Data overhead for OLSR vs. SOLSR

Figure 7.1 shows the data overhead only for routing packets in case of both OSLR and SOLSR protocol when they are used as the routing protocol. The figure presents the graph for a network where 20 nodes are present. It is obvious from the the graph that when SOLSR is used, after the first encrypted HELLO message, all nodes get involved in establishing the encryption key using the challenge-response mechanism and start sending the public key certificates. That is why the data transfer rate is high at the initial one or two seconds. When the key is established, the data transfer rate converged to a minimal constant state for sending EHELLO and ETC packets. We found that SOLSR has more overhead than OLSR as per our expectation. This is for the encryption keys and MAC sent in the routing messages for SOLSR.

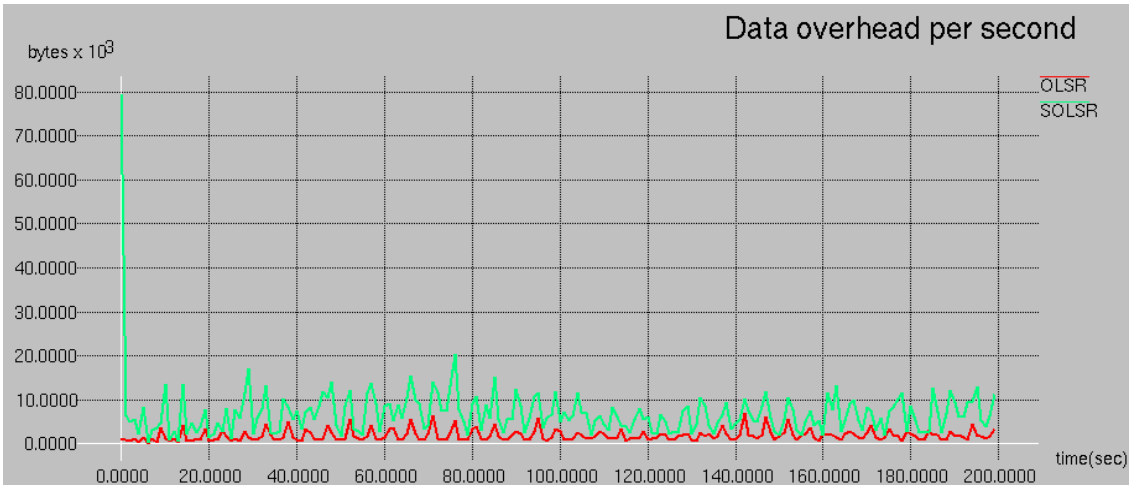


Figure 7.1: Data overhead for OLSR and SOLSR in a 20 nodes network

When we increase the number of nodes, the number of EHELLO messages and size of ETC messages is increased due to the numerous routes. Subsequently, the number of packets for routing messages also increases. Figure 7.2 depicts the change of data rate due to the increase of nodes in the network. In case of SOLSR, with the increase of nodes, there are more public keys and certificates that are needed to transfer and hence the data overhead increases faster than OLSR.

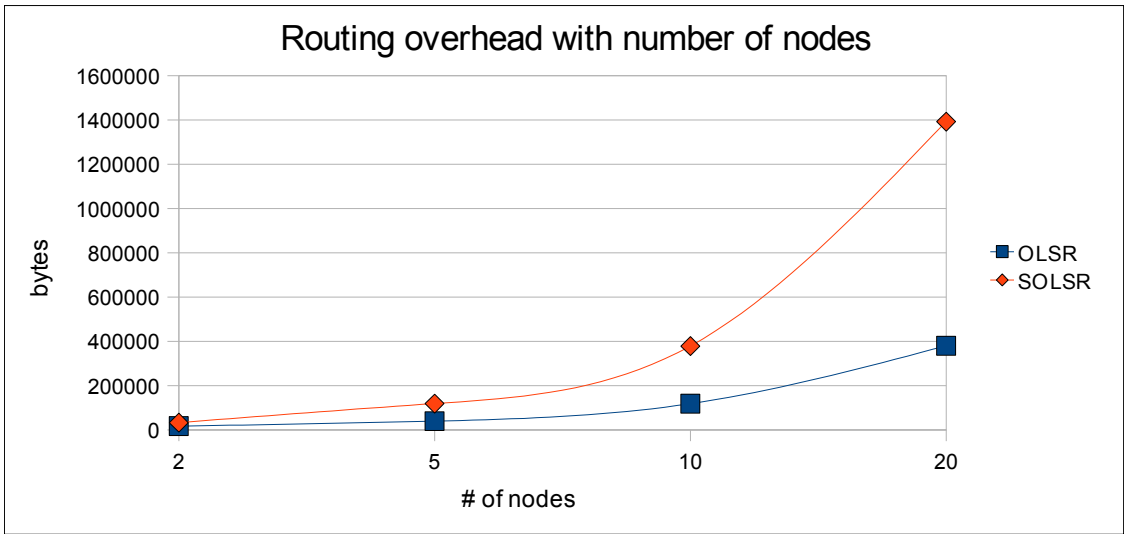


Figure 7.2: Routing overhead with increasing number of nodes.

In particular the challenge (ECH) message carries public key certificate and response messages (ERES) carries public key signature and certificate have a greater impact on the total data overhead.

7.3.3 Packet dropping rate

We have also performed a test with other mobile ad hoc routing protocols on the packet dropping rate in CBR connection as shown in Figure 7.3. The simulation is performed with the parameters mentioned in Table 7.2 having 20 nodes of which 12 nodes were actively transferring data and 19 CBR connections were established among them. The packets were mainly dropped due to the NRTE (no route), that is, the destination nodes were not in the routing table of the node which is routing the packet and eventually drops the packets. Obviously OLSR and SOLSR performed better as the routing table is updated periodically and route to destination is available faster than on-demand routing protocols.

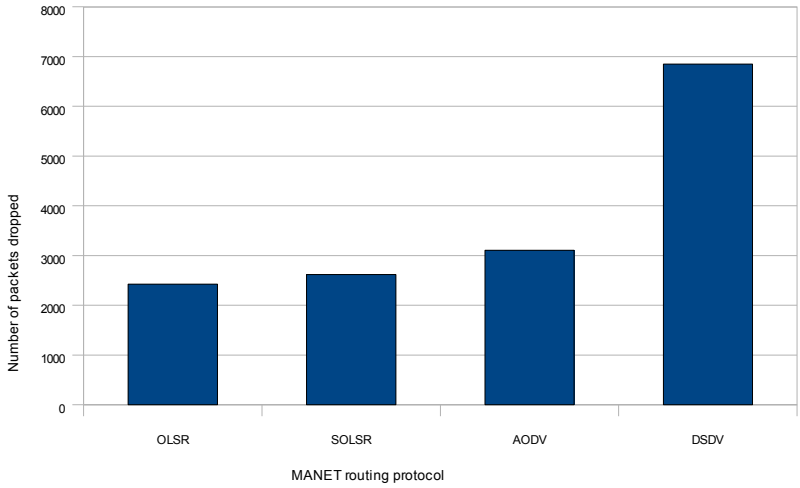


Figure 7.3: Packet dropping rate of different routing protocols, for 20 mobile nodes transferring CBR traffic in 12 active connections among them.

Earlier we have seen that SOLSR introduces more routing data overhead than OLSR to carry signature and message authentication code. In the next step, we were interested to see the packet drop rate of SOLSR with comparison to OLSR. In case of CBR traffic, nodes in communication generate constant traffic to a target node. Figure 7.4 shows the packets dropped by OLSR and SOLSR with increasing data rate. We have also added the packet drop rate of AODV to have a general comparison with other protocol. To optimize the simulation, we have limited the number of packets sent by each node to 10000. That is, when a node starts generating CBR traffic, it will stop after sending

10000 packets. Default simulation parameters are as mentioned in Table 7.2. Additionally the packet rate is gradually increased, 10 nodes are in the network of which 6 nodes are in active communication with 9 CBR connections among them. Nodes starts generating CBR traffic at a random time between 0 and 180 seconds. We found that the packet drop rate is not much affected in SOLSR. Most of the packets were dropped for NRTE (no route). When a node using OLSR successfully gets a route to a destination, using SOLSR it also get a route to the destination. Though a node using SOLSR uses the first HELLO message from its neighbor to start the security handshake protocol, eventually it processes the update from the next messages.

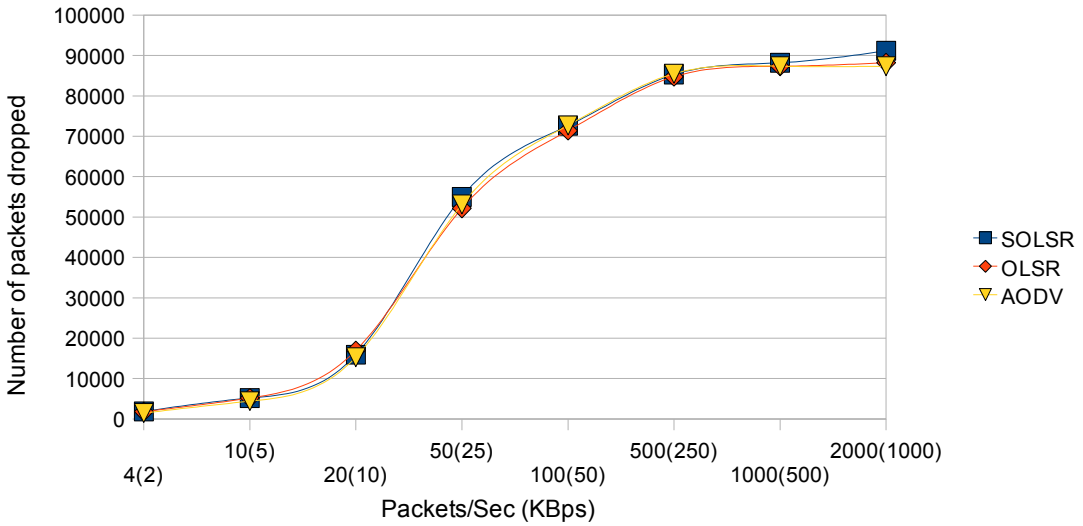


Figure 7.4 : Packet dropping rate with increasing data rate

In the graph it is shown that when the packet generation rate is high as 1000/2000 packets/seconds, the drop rate is decreasing. In contrast, with increasing packet rate, drop rate will drastically increase. But as we have mentioned, we restricted our nodes to generate no more than 10000 packets in the total simulation time, the nodes stop CBR traffic after generating 10000 packets. As a result, no packet is generated and dropped after that time.

7.3.4 Delay

In our simulation, we have introduced a time delay to represent the variable amount of time that may be needed to encrypt the HELLO and TC messages. The time needed for encryption is configurable from the TCL script for the OLSR agents by setting the `enc_rate` in mega bytes per second.

```
#Agent/OLSR set enc_rate_ 0.1      ;# in MBps = 100 KBps
```

Or,

```
#Agent/OLSR set enc_rate_ 100      ;# in MBps
```

The delay for message encryption is calculated as `Message_size * 1.0 / (enc_rate * 10^6)` seconds. We have run the simulation with different encryption rates considering the encryption algorithms, processing power etc. A list of approximate encryption rates can be found at [27].

Figure 7.5 shows an example of packet dropping rate against the various encryption rates for 20 nodes in a wireless network using SOLSR routing protocol.

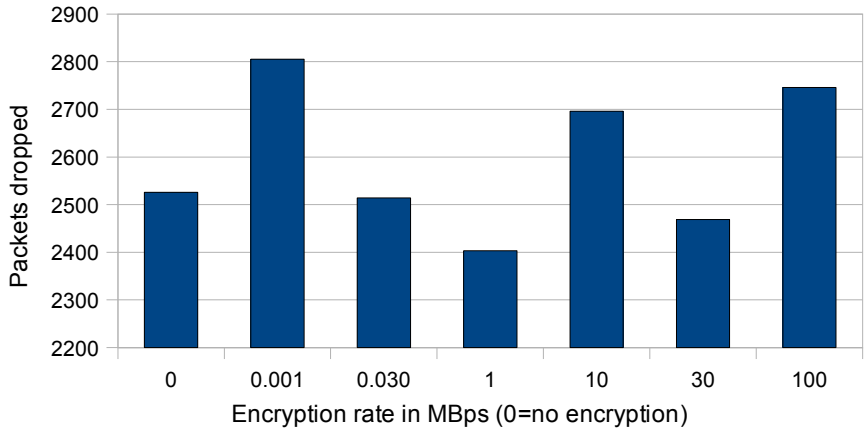


Figure 7.5: Packet dropping rate for SOLSR with varying encryption speed

We found that the packet dropping rate is not always increased with decreasing encryption rate. It is obvious that if the encryption rate is low, it will take more time to encrypt a control packet. To find any anomalies with the packet processing of SOLSR, we run a similar experiment with OLSR and introduce some delay in packet processing. In case of OLSR with 10 nodes, we found a similar result as shown in the Figure 7.6 below. The used delay values are in seconds and varied from no delay (0.000000 seconds) to 10 milliseconds. As for references, an OLSR HELLO message of size 68 bytes and with encryption rate of 100 MBps, the delay will be only 67 nanoseconds, which can be considered 0 second with respect to microsecond granularity. It is also found that for OLSR the packet dropping rate is not steadily increased.

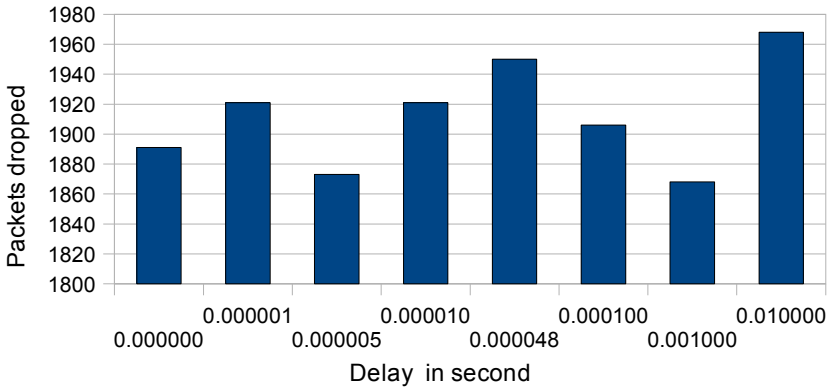


Figure 7.6: Packet dropping rate for OLSR when delay is added before sending the routing control message

Figure 7.7 shows the distribution of end to end packet delivery delay for SOLSR routing messages. Average end-to-end delay for SOLSR having 20 nodes and with no delay in encryption is 0.0169306 seconds. When we set an average encryption rate of 50 MBps the average end-to-end delay for control messages becomes 0.0193318 seconds. This is an increment of 14% delay for SOLSR protocol. Average end-to-end delay for OLSR having 20 nodes is 0.0167806 seconds. To make a clearer view of the lower values, the graph in Figure 7.7 shows only the delay between 0 and 1 second and values beyond one second is cut in the y-axis.

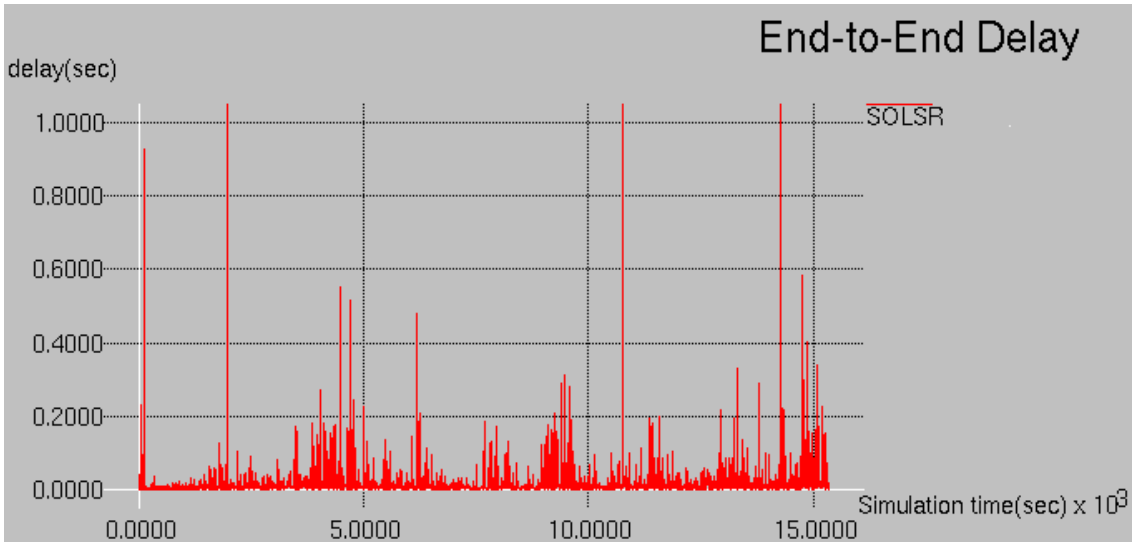


Figure 7.7: End-to-end delay distribution for SOLSR

We have observed that the packet delivery delay more than the average delay is mainly found for the key establishment messages specially for the challenge and response messages as they carry more data than usual control messages. As HELLO and TC messages are sent as broadcast message and received by multiple nodes in the network, we took the time for the farthest neighbor that receives the message. Table 7.3 presents comparative delay values of OLSR and SOLSR. The higher standard deviation for SOLSR demonstrates that some of its messages make more delay which we can also see in Figure 7.7.

Table 7.3: Statistical analysis of OLSR and SOLSR

Parameter	**OLSR**	**SOLSR**
Average	0.0167806	0.0193318
Maximum delay	1.025634	2.151732
Minimum delay	0.000949	0.001257
Standard deviation	0.0572818	0.104691

Though the average values of end to end delay for both protocols do not differ too much, the maximum delay and standard deviation are doubled for SOLSR. Very few packets contribute to the higher value of maximum delay of SOLSR. That is why it has little impact on the average end to end delay. However, the delay is only for the

dissemination of routing control messages. The data packet is forwarded as usual in both protocols when the destination address is in the routing table.

7.4 Network Animator (NAM)

Network Animator (NAM) is used to observe the nodes movement in the allowed dimension and transfer of routing packets and data packets for the whole simulation time graphically [29]. Figure 7.8 shows a screen from the network animator of one of our mobile networks that we have used in our simulation. While running simulation in NS-2, trace files for NAM can be generated to see the node movement and packet transfer.

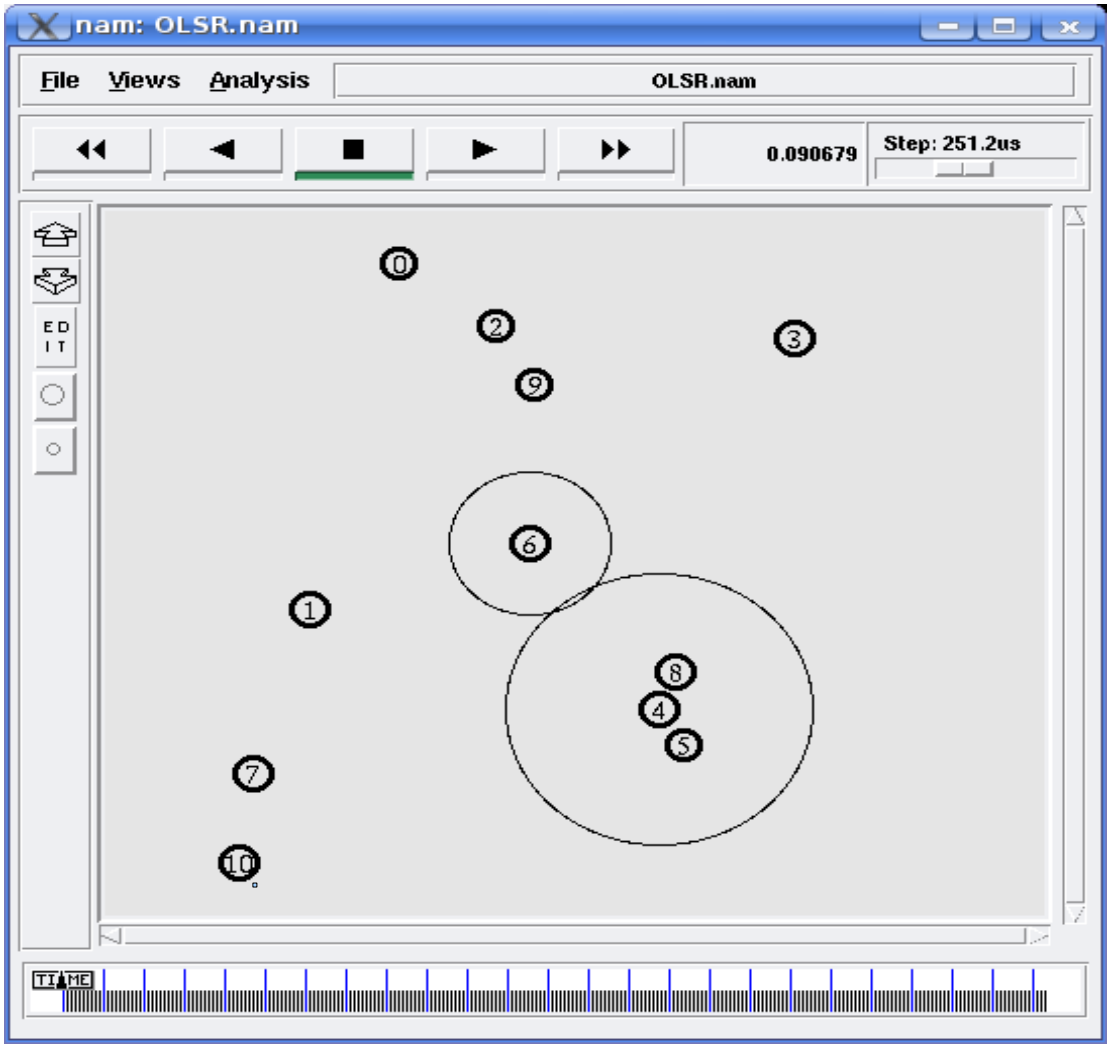


Figure 7.8: A mobile ad hoc network of 10 nodes in NAM using SOLSR routing protocol. Node 6 and 4 are broadcasting periodic EHELLO messages and a security handshake is going on between node 7 and 10.

Chapter 8

Discussion

The thesis is focusing on exploring the new security elements for the OLSR routing protocol by simulating its behavior using the network simulator NS-2. We have written the code module modifying the existing protocol definition and run simulations using different scenarios in wireless ad hoc networks. In this chapter we will discuss the various aspects of the simulations and their results.

8.1 Validity of results

The simulations were run to verify the routing mechanism of the new protocol in some MANET scenarios. The number of nodes is varied from 2 to 20 nodes. Since one of our targets was to compare the routing behavior with OLSR, we kept the same network scenario and applied both the SOLSR and OLSR routing protocols. With such a number of nodes, the change of routing mechanism can be verified by analyzing the trace files produced by NS-2.

In the simulation, the source nodes and destination nodes were randomly selected to generate CBR data traffic. With constant traffic rate and communicating nodes in the simulation, we were able to verify the packet drop rate, delay and routing overhead. The traffic load was varied from 4 packets to 2000 packets per second with a packet size of 512 bytes. Thus the highest data rate was approximately 8 Mbps, simulating nearly the maximum capacity of a typical IEE802.11b wireless link [30].

The dimension of the simulation was set to 1000x1000 meters. As the transmission range of the wireless nodes were 200 meters, the scenario was set such that it would represent both a dense network and a network where communicating nodes are several nodes away. Thus the routing table of the nodes was updated not only with HELLO messages but also with TC messages coming from other nodes. It has also helped us to verify the selection of MPR and MPR selector sets of the nodes.

The nodes' mobility is another important property that affects the routing table as the neighbor set experiences changes in time. For stationary nodes or nodes with low mobility most of the routing protocols shows nearly same behavior. When nodes are moving in a restricted area so that their neighbor set is not changed, in terms of routing the nodes can be considered as stationary nodes too. We have used the random waypoint model to simulate the mobility for the nodes. The nodes were moving towards a random destination with a constant speed of 2 to 10 meters per second, stop there for some seconds and again move to another random destination. Thus the routing table of the nodes were changed, expired routing entries were removed and hand-offs for the communicating nodes are performed.

8.2 Overhead

We found that the security extension to OLSR possesses similar routing capability with some extra data overhead to carry security parameters such as public key signatures, authentication codes etc. At the initial stage of the network setup, in a dense network, all the nodes in the network start to establish symmetric keys after listening to the HELLO messages. At that time a large amount of public key signatures are transferred and create a significant overhead on the routing control messages. In particular the CHALLENGE and RESPONSE messages contain the public key certificate and signature and carry more data than usual control messages. However, these messages are only transferred once between a pair of nodes at the first time of being a neighbor. Periodical HELLO and TC messages also contain encrypted keys and MAC and have more data than original OLSR. But this security payload is essential for the sake of security in the routing process.

The length of the public key certificate and signature influence the amount of overhead in the packet. In our simulation we consider 2048 bits public key cryptography and the certificate having its double length. It is possible to reduce the key length using different public key cryptography such as, Elliptic curve cryptography so that smaller key length can provide better security and hence reduce data overhead and signature length. In this way, the overhead in the protocol can be reduced.

With increasing number of nodes in the wireless network, the size of hello messages are mostly influenced by network density (i.e. number of neighbors), while TC messages are mostly influenced by network size (i.e. total number of nodes). In a dense network, the possibility to have more neighbor nodes is increased. This is also found in the simulations when we gradually increase the number of nodes and run the simulations. Though the size of the symmetric key or public key is not included in the specification [1], if we consider a 128 bit AES key as a symmetric key then for each neighbor node 160 bits overhead is added as key identifier and session key in the general encrypted HELLO and TC messages.

As presented in section 7.3.2, we found that each node of the mobile ad hoc network of 20 nodes had only 348 bytes/second routing data overhead on average. Since a typical IEEE 802.11g [30] network has a maximum data rate of 54Mbps (Mega bits per second), the overhead found here for a node is a very small fraction of its total bandwidth. We can calculate a theoretical threshold for a denser network. For instance, if a node in a network has 100 neighbors and in each encrypted TC message it sends topology data containing 100 neighbors then each of the encrypted HELLO or TC message size will be 16 KB (Kilo Bytes) with keys, MAC and encrypted data. By default, in 10 seconds the node will send 5 HELLO and 2 TC messages totaling 112 KB which accounts for 1.12 KBps, less than 0.02% of the total bandwidth of a 54Mbps wireless link. However, the overhead does not have any effect on the data packets or data transfer rates unless a large number of new neighbors start authenticating each other instantly.

8.3 Drop rate

We also found that the packet drop rate is not adversely affected when we introduce the secured routing mechanism. When a node is unreachable, packets destined to that node are dropped silently. We observed nearly similar drop rate for the security extension to OLSR as in the original OLSR. That is, the dissemination of topology information and route discovery process work well in the secured version of OLSR. To make a fair comparison, we also tested the same scenario with some other wireless ad hoc routing

protocols and found that AODV and DSDV drop more packets than SOLSR or OLSR in that particular network scenario. Obviously with higher packet generation rate in CBR, in case of an unreachable node, more packets are generated towards it and so packet drop rate will be higher.

8.4 Delay

We observe that the nominal delay (microseconds for encryption) that is introduced by encryption in SOLSR during sending of the control message does not have much effect on the performance of the new routing protocol. As such a delay in the emission of HELLO and TC messages is similar to the random *jitter* that is normally experienced with the OLSR protocol.

Jitter is added in OLSR to avoid collisions depending on the underlying link layer. The emission of control message may be synchronized such that several neighbor nodes attempts to send control traffic at the same time and may lead to a collision in the link layer. To avoid such collisions an amount of jitter is introduced to the message interval as [5]:

```
Actual message interval = MESSAGE_INTERVAL – jitter
```

The value of jitter is a random value between `0` and `MAXJITTER`. The proposed value of `MAXJITTER` in RFC 3626 is `HELLO_INTERVAL/4` and the proposed value of `HELLO_INTERVAL` is 2 seconds. That is, the jitter for message interval is a value between 0 to 500 milliseconds. Since the delay for encryption usually is less than the jitter time, the effect of encryption delay does not have a significant effect on packet dropping rate or routing table update in SOLSR.

We have also seen that small changes in the HELLO interval do not have a significant effect on the packet dropping rate. Figure 8.1 shows the packet dropping rate with increasing HELLO intervals for a 20 nodes wireless network using the SOLSR routing protocol. We can find a significant increase in the packet dropping rate when the interval reaches 5 seconds.

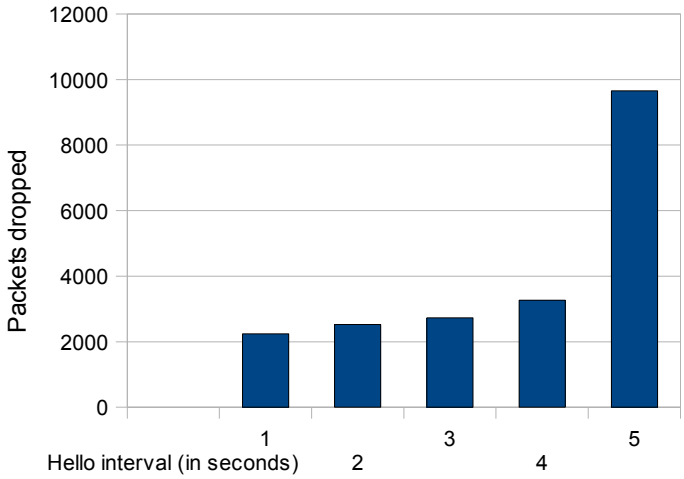


Figure 8.1: Effect of HELLO intervals on packet dropping rate

When we have measured the end-to-end delay of only routing packets, the distribution of delay values found were very high for some of the packets. We have observed that the end-to-end delay for challenge and response messages were higher than the delay for usual control messages. The average delay for SOLSR is a little higher than original OLSR. We also found that the distribution of SOLSR packet delay is sparser than OLSR as registered by the measured standard deviation of their delay values. We assume that the reason behind it is that SOLSR transfers larger packets for security handshakes and spends time in encryption scheme.

8.5 Application areas

The protocol depends on a PKI based system for access control and authentication of wireless nodes in the network. Primarily it was designed for a closed organizational network for rescue operations. The field workers working in such a crisis situation must have a public key certificate to be able to access in the secured network. The protocol can also offer a better service for ad hoc networks in military operation.

In case of a public open mobile ad hoc network, it may not be possible to issue valid public key certificates for all the participating nodes. However, if the nodes in the network somehow manage public key certificates and the nodes in the network can authenticate each other by means of the certificates, they can utilize the power of the secured routing protocol among themselves.

Chapter 9

Future Work

Security in MANET routing protocols is an open research issue, and investigative work is still ongoing. In the new security extension, a provision for group multicast is introduced too. Implementation of multicast in the new security extension will enrich it with more capabilities in the simulation of group communication. A prospective future improvement of the simulation would be to include the cryptological functions, such as verifying the public key certificates, encrypting the messages and check for MAC validity. Thus the keys, certificates and signatures can be configured from the TCL script while the simulation runs. It is our opinion that introducing the security protocol in real wireless devices will fulfill the objective of the protocol to secure the mobile ad hoc network.

Chapter 10

Conclusion

In this thesis a new security extension to the OLSR routing protocol for MANETs has been implemented in the NS-2 simulation environment. Through simulation experiments we have verified the correct behavior of the routing protocol with regards to building and maintaining routes in the network. Compared to the original unsecured version of OLSR, the new security protocol shows only little or no increase in delay and packet drop rate. Control data overhead however, do show a significant increase and is also assumed to increase further with larger and more densely populated networks. For MANETs of moderate size and density our simulation indicates that it could be very well suited. The security extension was originally proposed for use in emergency response scenarios; however it may also be applied to other application areas where closed networks and networks of similar size are to be deployed.

References

[1] Ingrid S. Svagard, "Information security for field workers in crisis situations", OASIS, available at http://www.oasis-fp6.org/documents/ OASIS_SP24_DDD_253_security_SIN_1_0_pub.pdf.

[2] The OASIS project web page. Available at http://www.oasis-fp6.org/.

[3] E. Wilkinson, "Evaluation Plan for POS1", OASIS, Deliverable D-TA2_04 OASIS_TA26_PLN_043_CRU, March 3 2006.

[4] E. Wilkinson, "POS1 Evaluation Report", OASIS, Deliverable D-TA2_10_OASIS_TA26_RPT_185_CRU, December 21 2006.

[5] T. Clausen (ed) and P. Jacquet (ed). Optimized link state routing protocol (OLSR), October 2003. RFC 3626, Experimental.

[6] IETF MANET working group website. http://www.ietf.org/html.charters/manet-charter.html. Last modified 2008-08-21.

[7] Amir Qayyum, Laurent Viennot, and Anis Laoutti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, Hipercom Project, INRIA Rocquencourt, 2000. INRIA RR-3898.

[8] P. Jacquet, P. Miihlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. Hipercom Project, INRIA Rocquencourt.

[9] Åsmund Ahlmann Nyre, Martin Gilje Jaatun, Inger Anne Tøndel. A secure MANET routing protocol for first responders, Accepted for publication, *1st International Workshop on Security and Communication Networks*, May 20-22, 2009, Trondheim, Norway. IEEE.

[10] Inger Anne Tøndel, Martin Gilje Jaatun, Åsmund Ahlmann Nyre. Security requirements for MANETs used in emergency and rescue operations, Accepted for publication, *1st International Workshop on Security and Communication Networks*, May 20-22, 2009,Trondheim, Norway. IEEE.

[11] Cedric Adjih, Thomas Clausen, Philippe Jacquet, AnisLaouiti, Paul Muhlethaler, and Daniele Raffo. "Securing the OLSR protocol". *In Proceedings of Med-Hoc-Net*, Mahdia,Tunisia, June 25–27 2003.

[12] Levente Buttyan and Jean-Pirre Hubaux. "Security and Cooperation in Wireless Networks". Cambridge University Press. 2007.

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

[14] Y.C. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proceedings of the 8th Annual International Conference Mobile Computing and Networking (MobiCom 2002)*, ACM Press, 2002, pp.12-23.

[15] L. Ji and M. S. Corson, "Differential destination multicast – A MANET multicast routing protocol for small groups,". *INFOCOM 2001. 20th Annual Joint Conference of the IEEE Computer and Communication Societies. Proceedings.* 2001.

[16] D. Johnson and D. Maltz, Dynamic source and routing in wireless ad hoc networks, In T. Imilienski and H. Korth, editors, Mobile Computing. Kluwer Academic Publishers, 1996.

[17] Kannhavong, B. Nakayama, H. Jamalipour, A. "SA-OLSR: Security Aware Optimized Link State Routing for Mobile Ad Hoc Networks". *IEEE International Conference on Communication*, 19-23 May 2008. Pages 1464-1468.

[18] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. *In Proceedings of SCS communication Networks and Distributed Systems Modeling Simulation Conference (CNDS)*, 2002.

[19] C. Perkins, E. Belding-Royer and S. Das. Ad-hoc on-demand distance vector (AODV) routing. Internet RFC 3561. July 2003.

[20] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *In proceedings of the SIGCOMM'94 Conference on Communications Architectures, Protocols, and Applications*. August 1994.

[21] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. *In proceedings of the IEEE Symposium on research in Security and Privacy*, May 2000.

[22] K. Sanzgiri et al., "A Secure Routing Protocol for Ad Hoc Networks." *Proceedings of the 10th International conference on Network Protocol (ICNP '02)*, IEEE Press, 2002, pp. 78-87.

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

[24] Official NS-2 website. http://www.isi.edu/nsnam/ns/.

[25] The NS-2 manual. http://www.isi.edu/nsnam/ns/doc/index.html.

[26] UM-OLSR. http://masimum.dif.um.es/?Software:UM-OLSR. Retrieved on February 25, 2009.

[27] Speed comparison of popular crypto algorithms. Retrieved from http://www.cryptopp.com/benchmarks.html on March 02, 2009.

[28] Daniele Raffo, Cédric Adjih, Thomas Clausen, Paul Mühlethaler. An advanced signature system for OLSR. *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks.* October 25-25, 2004. Washington DC, USA.

[29] Nam: Network Animator. http://www.isi.edu/nsnam/nam/

[30] IEEE 802.11 working group. http://grouper.ieee.org/groups/802/11/

Appendix A: Scripting Code

A.1 Scene generation

We have generated scenario file for 2, 5, 10, and 20 mobile nodes. We have a used random traffic generator script to setup connection between mobile nodes. The script is available in the *ns* directory, **~ns/indep-utils/cmu-scene-gen/** directory**.**

The following command creates a CBR connection among 5 nodes. The data rate is 4 packets per seconds. Generated scenario file is saved in the `cbr-5.sc` file. This and the other scenario files are given in the accompanying CD (or as attachment in DAIM).

```
$ ns cbrgen.tcl -type cbr -nn 5 -seed 1.0 -mc 8 -rate 4.0
> /home/sislab/ns/test4/cbr-5.sc
```

Default values for packets size is 512 bytes, and CBR start time is a random value between 0 to 180 seconds. We can change the time in the source code line,

```
set stime [$rng uniform 0.0 180.0]
```

One thing we must remember that the nodes are numbered through ***0*** to ***nn***, not starting with our usual node number from ***0*** to ***nn-1***.

A.2 Generating node movement for wireless network

To generate random node movement, we have used the utility code comes with NS-2. We have compiled the C++ source code and used the following commands several times changing the parameters for our particular scenarios. Generated node movement files are given in the accompanying CD (or as attachment in DAIM).

<original 1999 CMU version (version 1)>

```
$./setdest -v <1> -n <nodes> -p <pause time> -M <max
speed>  -t <simulation time> -x <max X> -y <max Y>
```

Or,

<modified 2003 U.Michigan version (version 2)>

```
$./setdest -v <2> -n <nodes> -s <speed type> -m <min
speed> -M <max speed>    -t <simulation time> -P <pause
type> -p <pause time> -x <max X> -y <max Y>
```

The following command will generate node movement pattern file for 5 nodes moving with a speed of 2.0 - 10.0 m/s with an average pause between movements being 2.0 seconds. We want the simulation to stop after 200 seconds and the topology boundary is defined as 1000x1000.

```
$ ./setdest -v 2 -n 5 -m 2.0 -M 10.0 -t 200 -p 2.0 -x 1000
-y 1000 > /home/sislab/ns/test4/mobile-5.sc
```

A.3 *awk* scripts

Data extraction from the generated trace file by NS-2 is done using Linux tools *awk*, *grep* etc.

The following *awk* script will extract only routing packets sent per second from a standard wireless trace file.

Listing: time-vs-data.awk

```
/^s.*RTR.*UM-OLSR/ { i= int($2); val[i]+=$8;}
END{
n=200

for(i=0; i<=n; i++)
  if(val[i]>=1)
    print i " " val[i]
  #else print i " " 0
}
```

To keep the data to a file for plotting graph, we can run the following command,

```
$awk -f time-vs-data.awk OLSR-20.tr > tile-olsr-20.dat
```

Then, we can use the *xgraph* tool to draw a graph on the data found, as like the following command,

```
sislab@sislab1:~/ns/test4$ xgraph time-olsr-20.dat time-
solsr-20.dat -x "time(sec)" -y bytes -nb -lw 2 -t "Data
overhead per second" -0 OLSR -1 SOLSR
```

An example to calculate the total routing data transmitted from the above file can be get by,

```
$ awk '{n+=$2} END {print n}' time-solsr-2.dat
32308     [output]
```

End-to-end delay

End to end delay can be calculated from a NS-2 trace file by the following code:

```
sislab@sislab1:~/ns/test4$ cat end-to-end.awk
BEGIN {
   maxpack_id=0;
}
/.*UM-OLSR.*HELLO|TC/ {  event = $1;
   time = $2;
   pack_id = $6;
   if ( pack_id > maxpack_id ) maxpack_id = pack_id;
   if ( ! ( pack_id in tempIn ) ) tempIn[pack_id] = time;
   if ( event == "r" ) timeReached[pack_id] = time;
}
END {
  for ( pack_id = 0; pack_id <= maxpack_id + 1; pack_id+
+ ) {
      duration =  timeReached[pack_id] - tempIn[pack_id];
      if ( duration > 0 ) printf("%d %f \n", pack_id,
duration);
}
exit 0
}
# awk -f end-to-end.awk OLSR-10.tr > end-to-end-olsr-
10.dat
```

The average end to end delay is calculated by,

```
# awk 'BEGIN {count = 0; total = 0;} {count = count+1;
total = total + $2; } END { print "Average =
"total/count}' end-to-end-olsr-10.dat
Average =  0.0193318 [output]
```

After getting the average value, we can calculate the standard deviation. The codes are given next.

```
$ cat stddev.awk
{ln++}
{d = t - $2; s = s + d*d}
END {print "Standard deviation = " sqrt(s/ln)}

$ awk -v t=0.0193318 -f stddev.awk end-to-end-solsr-20-
50mbps.dat
```

The maximum and minimum values are also calculated as,

```
$ awk 'BEGIN {max = $2} { if (max<$2) max=$2} END {print
"Maximum delay = " max}' end-to-end-solsr-20.dat
Maximum delay = 2.151732 [output]

$ awk 'NR==1 {min=$2} { if (min>$2) min=$2} END {print
"Minimum delay = " min}' end-to-end-solsr-20.dat
Minimum delay= 0.001257 [output]
```

Appendix B: Algorithms

B.1 CHALLENGE message generation

A CHALLENGE message is generated whenever a node receives an encrypted message from a source that it does not share a key with. A node is said to share a link with a source if there exist an entry in the Link Set such that:

1. L_local_iface_addr = node address AND,
2. L_neighbor_iface_addr = Source address AND,
3. L_KEY_time > current time (i.e. has not expired)

If this condition is not met, the following procedure is done.

1. Create (or locate) an entry in the Link Set where
   a. L_local_iface_addr = node address AND
   b. L_neighbor_iface_addr = Source address
2. Set the L_ASYM_time = L_time = current time + validity time
3. Set the L_SYM_time = current time -1 (expired)
4. Set the L_local_KID = random bit string, and verify its uniqueness in the Link Set (i.e. that no other L_local_KID is identical)
5. Create a new CHALLENGE message with
   a. Vtime = validity time
   b. CKeyID = L_local_KID
   c. Certificate = node certificate
6. Transmit the message to the Source address by unicast.

B.2 CHALLENGE message processing

Upon receiving a CHALLENGE message the validity time is computed from the Vtime field of the message header and the message is processed according to the following description:

1. If the node shares a key with the source OR the contained certificate is not valid, the message is silently discarded.
2. Otherwise, create (or locate) an entry in the Link Set where
   a. L_local_iface_addr = node address AND

b. L_neighbor_iface_addr = Source address

3. Set the L_ASYM_time = L_time = current time + validity time
4. Set the L_SYM_time = current time -1 (expired)
5. Set the L_local_KID = random bit string, and verify its uniqueness in the Link Set (i.e. that no other L_local_KID is identical)
6. Set the L_neighbor_KID = CKeyID
7. Node description is extracted from the certificate.
8. The Neighborhood information base is updated such that it contains an entry with:
    a) N_neighbor_main_ addr = Originator Address
    b) N_status = NOT_SYM_LINK
    c) N_willingness = 0 (willingness is not known)
    d) N_access_level = 1 (if node description contains regular node) or 0 otherwise.

9) The Node Description Set is updated so that it contains an entry with:
    a) ND_main_addr = Originator Address
    b) ND_description = node description

B.3 RESPONSE message generation

After a successful challenging process, a RESPONSE message generation is done as follows:

1. Set CKeyID = L_neighbor_KID AND RKeyID = L_local_KID
2. Set Vtime = validity time
3. Create a digital signature over the message header and message contents fields.
4. Set Certificate = node certificate

B.4 RESPONSE message processing

Upon receiving a RESPONSE message validity time is computed from the Vtime field and message processing done as the following:

1. If the nodes share a secret key, OR the certificate is not valid, OR the signature

is not valid, the message is silently discarded.

2. If there exist no entry in the Link Set where
    a) L_ local_iface_addr = node address AND
    b) L_neighbor_iface_addr = source address AND
    c) L ASYM time> current time AND
    d) L_local_KID = CKeyID,

    The message is silently discarded.
3. Otherwise, the Link Set entry is updated such that
    a) L_ASYM_time = L_SYM_time = L_time = validity time
    b) L_neighbor_KID = RKeyID
    c) L_key_value = random bit string
4. Node description is extracted from the certificate.
5. The Neighborhood information base is updated such that it contains an entry with:

a) N _neighbor _ main_ addr = Originator Address
b) N status = SYM LINK
c) N _willingness = 0
a) N_access_level = 1 (if node description contains regular node) or 0 otherwise.

6. The Node Description Set is updated so that it contains an entry with:
    a) ND_main_addr = Originator Address
    b) ND_description = node description

B.5 KEY message generation

A KEY message is created upon the reception and successful processing of a RESPONSE message. The generation process is as follows:

1. Set KeyID = L_local_KID and Vtime = validity time.
2. Encrypt L_key_value using the public key of the receiver
3. Create a signature over the message header and message contents fields.

B.6 KEY message generation

Upon receiving a KEY message, validity time is computed from the Vtime field and message processing done as the following:

1. If the nodes share a secret key OR the signature is not valid, the message is silently discarded.
2. If there exist no entry in the Link Set where
   a) L_local_iface_addr = node address AND
   b) L_neighbor_iface_addr = source address AND
   c) L ASYM time > current time AND
   d) L_neighbor_KID = CKeyID

   The message is silently discarded.
3. Otherwise, the shared key is decrypted
4. The Link Set entry is updated such that

   a. L_ASYM_time = L_SYM_time = L_time = validity time

   b. L_key_value = key
5. The entry in the neighborhood information base is updated such that

   a. N Status = SYM_LINK

Appendix C: Sample TCL Code

```
# Example TCL file to run the simulation.
# Parameters like number of nodes and routing protocol
# should be changed.
# scenario file val(cp) and val(sc) should be changed
# according to the number of nodes.

#===================================================
# Define options
#===================================================

set val(chan)       Channel/WirelessChannel
set val(prop)       Propagation/TwoRayGround
set val(netif)      Phy/WirelessPhy
set val(mac)        Mac/802_11
set val(ifq)        Queue/DropTail/PriQueue
set val(ll)         LL
set val(ant)        Antenna/OmniAntenna
set val(x)          1000   ;# X dimension of the topography
set val(y)          1000   ;# Y dimension of the topography
set val(ifqlen)     50            ; # max packet in ifq
set val(seed)       0.0
set val(adhocRouting)   OLSR
#set val(adhocRouting)  DSDV
#set val(adhocRouting)  AODV
set val(nn)             21
set val(cp)         "cbr-20.sc"     ;# connection pattern
set val(sc)         "mobile-20.sc"  ;# mobility
set val(stop)        200.0      ;# simulation time

#===================================================
# Main Program
#===================================================

set ns_   [new Simulator]
set topo  [new Topography]
$topo load_flatgrid $val(x) $val(y)
set god_ [create-god $val(nn)]
#create-god $val(nn)

# create trace object for ns and nam
puts "Routing Protocol is $val(adhocRouting)"
set tracefd  [open $val(adhocRouting).tr w]
set namtrace [open $val(adhocRouting).nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

```
# node configuration
$ns_ node-config -adhocRouting $val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF\
                 -movementTrace ON

# common initialization should be done
# before creating the nodes.

if { $val(adhocRouting) == "OLSR" } {
#    Agent/OLSR set debug_ true
#    Agent/OLSR set hello_ival_ 2
#    Agent/OLSR set enc_rate_ 50
# in MBps, 0.0 means no ecnryption.
#    Agent/OLSR set delay_enc_session_key_ 0.0
# 0.0 means no encryption.
}

#  Create the specified number of nodes [$val(nn)]
for {set i 0} {$i < $val(nn) } {incr i} {
     set node_($i) [$ns_ node]
#    $node_($i) random-motion 0 ;# disable random motion
}

puts "Loading connection pattern..."
source $val(cp)

puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 30
}

# Specific settings for OLSR
#if { $val(adhocRouting) == "OLSR" } {
#    $ns_ at 1.0 "print_all_for_all_nodes";
#    $ns_ at 105.0 "print_all_for_all_nodes"
```

```
#     $ns_ at 160.0 "print_all_for_all_nodes"
#     $ns_ at 135.0 "print $node_(8)"
#     $ns_ at 135.0 "print $node_(4)"
#     $ns_ at 155.0 "print $node_(7)"
#}


# Tell nodes when the simulation ends

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ; $ns_
halt"

puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp
$val(adhocRouting)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."
$ns_ run


#=========================================================
# utility procedures
#=========================================================

# print all possible OLSR repository for this node.

proc print { node } {
     [$node agent 255] print_rtable
     [$node agent 255] print_linkset
     [$node agent 255] print_nbset
     [$node agent 255] print_nb2hopset
     [$node agent 255] print_mprset
     [$node agent 255] print_mprselset
}


# print routing table for all nodes in this simulation

proc print_all_rtable { } {
     global node_ val
     for {set i 0} {$i < $val(nn)} {incr i} {
     [$node_($i) agent 255] print_rtable
     }
}
```

```
# print all nodes tables
proc print_all_for_all_nodes { } {
     global node_ val
     for {set i 0} {$i < $val(nn)} {incr i} {
          print $node_($i)
     }
}

###  END of script.
```

Appendix D: Source Code

The accompanying compact disk (CD) (or the attachment in DAIM) contains the implemented source code and executables. The sample script files that are used to test the protocol are also included for reference.

Steps for installation:

1. The modified implementation of security extension to OLSR is included in 'olsr' directory. The original UM-OLSR is enhanced to incorporate the security standard. After installing ns-allinone-2.33, copy this directory in the 'ns' (or whatever directory the NS-2 source code is installed) directory. Running the included patch file will perform modifications in the NS-2 source code.
2. Copy the 'cmu-trace.cc' file in the 'ns/trace' directory. Overwrite the existing file to get a cleaner view of the new messages in the trace file.
3. Copy the 'ns-default.tcl' file in 'ns/tcl/lib' directory. Overwrite the existing file or add the new default values for OLSR in script code.
4. Recompile NS-2 by 'make clean' and 'make'.