Secure Ad hoc On-demand Distance Vec[illegible]or Routing Protocol

Developed by

Muhammad Asfand-e-yar

Supervised by

Assistant Prof. Dr. Muhammad Sher

Department of Computer Science
International Islamic University, Islamab[illegible]ad
(2004)

بسم الله الرحمن الرحيم

**In the name of ALMIGHTY ALLAH,
The most Beneficent, the
Merciful.**

Department of Computer Science,
International Islamic University, Islamabad.

18th February, 2004

Final Approval

It is certified that we have read the thesis, titled "**Secure Ad-hoc On-demand Distance Vector Routing Protocol**" submitted by **Muhammad Asfand-e-yar** under University Reg. No. **09-CS/MS/01**. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad, for the Degree of **Master of Science**.

Committee

External Examiner

Dr. Asmat Ullah Khan
Associate Professor,
Department of Electric Engineering,
COMSETS Institute of Information & Technology,
Islamabad.

[illegible]

Internal Examiner

Dr. Tauseef Ur Rehman
Head Department of Telecommunication and Computer Engineering.
International Islamic University,
Islamabad.

[illegible]

Supervisor

Dr. Muhammad Sher
Assistant Professor,
Department of Computer Science,
International Islamic University,
Islamabad.

[illegible]
18/02/04

A dissertation submitted to the
Department of Computer Science,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
Master of Science

Declaration

I hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have developed this software entirely on the basis of my personal efforts made under the sincere guidance of our teachers. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Muhammad Asfand-e-yar
09-CS/MS/01

Dedication

I have the honor to mention here that this thesis has been scribed today all because of my Mother, who always guided and inspired me to make hard effort in order to achieve this goal. I dedicate these efforts to my sweet Mother.

Muhammad Asfand-e-yar

Acknowledgements

All praise to the Almighty Allah, the most Merciful, the most Gracious, without whose help and blessings, I will be unable to complete the project.

Thanks to my Parents who helped me during my most difficult times and it is due to their unexplainable care and love that I am at this position today.

Thanks to my project supervisor Muhammad Sher, their sincere efforts helped me to complete my project successfully.

I acknowledge my teachers and my friends for their help in the project.

Muhammad Asfand-e-yar

Project in Brief

Project Title:	Secure Ad hoc On-demand Distance Vector Routing Protocol
Objective:	To Develop an efficient secure route for communication between source and destination
Undertaken By:	Muhammad Asfand-e-yar
Supervised By:	Muhammad Sher Assistant Professor, Department of Computer Science, International Islamic University, Islamabad.
Technologies Used:	Microsoft® Visual C++ 6.0,
System Used:	Pentium® III
Operating System Used:	Microsoft® Windows® 2000 Professional
Date Started:	1st November, 2002
Date Completed:	25th August, 2003

Abstract

An Ad hoc net work is a self organizing system of mobile nodes connected by wireless links. Ad hoc networks do not required any centralized access point. It maintains a routing table giving distance from itself to all possible destinations. In Distance Vector (DV) routing protocol loops are formed when communication is done by multiple paths from single source to single destination. These loops in DV routing protocol are avoided by different methods such as routing information protocol. To over come this deference's the On-demand Distance Vector Routing Protocol is introduced. Ad hoc On-demand Distance Vector Protocol (AODV) requires nodes to maintain routes to destinations, which are not actively used in communication. In AODV routing protocol the route discovery is done by route request and reply packets. After establishing a route the communication between source and destination takes place. The route is maintained by keep alive messages. If any link breaks in a route, then route error message is generated from broken point. The route error message stops the source from sending data. And make destination to establish link again. AODV routing protocol solves the loop formation. It works on wired as well wireless networks. This thesis report describes the AODV functionality and Security implementation. AODV is not standardized by IETF (Internet Engineering Task Force) therefore the tests are done by simulation. Here we have developed the simulation which will help us to test our research work. We make the route secure by introducing the dual signature technique. For signatures we use RSA, which are preferable, used in internet environment.

Abbreviations

AODV	Ad hoc On-demand Distance Vector
CSPEC	Control Specification
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
STD	State Transition Diagram
RSA	Reviest Shamir Adleman

TABLE OF CONTENTS

Chapter 1

Introduction

1. INTRODUCTION

The term MANET stands for Mobile Ad-hoc Network. This new networking concept defines simple mechanisms which enable mobile devices to form a temporary community without any planned installation, or human intervention. The idea is to form a totally improvised network that does not require any pre-established infrastructure. But, how can we make this possible? The answer is very simple. Each node acts as a host and a router at the same time. This means that each node participating in a MANET commits itself to forward data packets from a neighboring node to another until a final destination is reached. In other words, the survival of a MANET relies on the cooperation between its participating members if a source node wants to communicate with another node which is out of its transmission range, the former will send its packets to a neighboring node, which will send them, in its turn, to one of its neighboring nodes, and so on, until the destination node is reached.

Although, this new approach of networking brings a great flexibility to the world of wireless communications, it raises some new challenges among the research community as far as routing and security issues are concerned.

In fact, ad-hoc networks have certain characteristics (high degree of mobility, absence of centralized administration, limited resources) that impose some strong requirements with regard to the routing functionality. An adequate routing protocol for MANETs should not only react quiet rapidly to the dynamically changing topology of the network but it should also maintain a decent quantity of control traffic in order to preserve the available resources (bandwidth, battery power, etc). Classic protocols such as link-state or distance-vector would be too drastic for such a purpose. Both rely on flooding mechanisms, which would cost a lot in terms of resources and would take a long time to converge in case of higher mobility.

Therefore, the Internet Engineering Task Force has formed a new working group for Mobile Ad-hoc Networking, whose mission consists of providing a framework for developing IP-based routing protocols in ad-hoc networks. At this time, no standard protocol has been adopted yet but many of them are currently under study. One of the rising protocols is known as AODV, which stands for Ad-hoc On-demand Distance Vector.

In keeping with the general pattern of its policy, the Wireless Communication Technologies Group (WCTG) of the National Institute of Standards and Technology (NIST) intends to play an active role in the development of MANETs by investigating the performance of routing protocols for MANETs that have been submitted for possible standardization by the IETF.

1.1 General Concepts about Ad-hoc Networking

There are two approaches that allow two wireless stations to communicate with each other. The first one is to introduce a third fixed party (a base station) that will hand over the offered traffic from a station to another, as illustrated in Figure 1. This same entity will regulate the attribution of radio resources, for instance. When a node S wishes to communicate with a node D, the former notifies the base station, which eventually establishes the communication with the destination node. At this point, the communicating nodes do not need to know of a route for one to each other. All that matters is that both nodes source and destination are within the transmission range of the base station. If one of them fails to fulfill this condition, the communication will abort.

Note however that this form of centralized administration is very popular among wide cellular networks such as GSM, UMTS, etc.

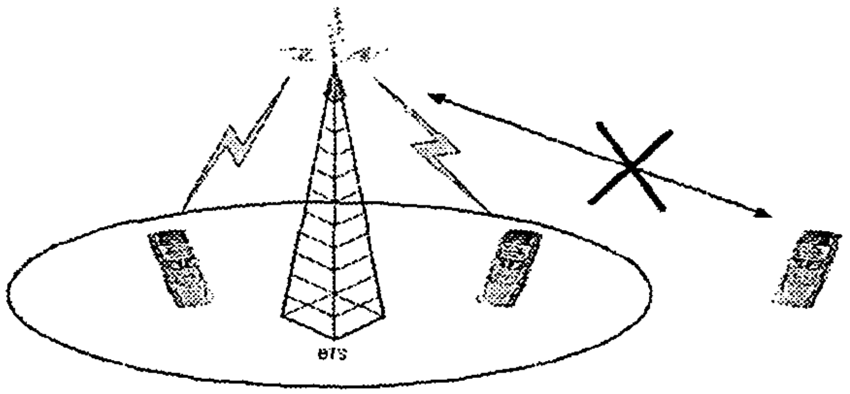


Figure 1 Infrastructure network

The second approach, called ad-hoc, does not rely on any stationary infrastructure. The concept behind these infrastructures less networks is the collaboration between its participating members, i.e., instead of making data transit through a fixed base station, nodes consequentially forward data packets from one to another until a destination node is finally reached. Typically, a packet may travel through a number of network points before arriving at its destination.

Figure 2 illustrates a simple 3-node ad-hoc network. In this figure, a source node S wants to communicate with a destination node D. S and D are not within transmission range of each other. Therefore, they both use the relay node R to forward packets from one to another. So, even though R is primarily a host, R is acting as a router at the same time.

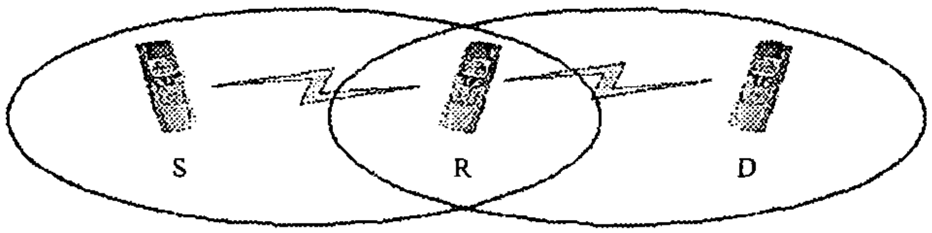


Figure 2 Infrastructure less networks

By definition, a router is an entity that determines the path to be used in order to forward a packet toward its final destination. The router chooses the next node to which a packet should be forwarded according to its current understanding of the state of the network.

Because of the improvised nature of ad-hoc networks, routes are built dynamically as and when nodes are regrouping (Discovery). Hence, ad-hoc networks are more responsive to topology changing than any wired networks. Consequently, routing protocols for ad-hoc networks should be able to cope with link breakages and make sure that the network won't collapse as nodes are moving or shutting down (Maintenance).

1.1.2 Routing protocols for Ad-hoc Networks

As we previously mentioned, ad-hoc networks have certain characteristics that put a lot of stress on the routing layer. Some of these characteristics are listed bellow:

Nodes in a MANET are connected by wireless links with a constrained bandwidth. Thus, an appropriate routing protocol for MANETs should imply a reasonable over-head in order to preserve the limited bandwidth. Message complexity must be kept very low.
Nodes in a MANET are likely to be hand-held devices and laptops with relatively constrained resources. These resources include namely storage capacity and battery power and should be used in a smart way. Therefore, a suitable routing protocol for MANETs should not make an excessive use of flooding or periodic update messages.
Nodes in a MANET are likely to be mobile. In order to cope with such a rapidly changing topology, a routing protocol for ad-hoc networks should be able to find alternate routes very quickly. Rapid convergence is the ultimate goal for an ad-hoc routing protocol.

1.1.2.1 Classic Protocols

Two routing algorithms, link-state and distance-vector, have been traditionally used in packet-switched networks. They both allow a node to determine the next hop along the "shortest path" toward a given destination. The shortest path is computed according to a specific cost, which is usually the distance in number of hops.

1.1.2.1.1 Link State Protocols

In the link-state approach, each node collects the state (cost) of its outgoing links (links toward neighboring nodes) and diffuses it to every other node in the network in the form of link state packets (LSPs). This is done periodically or whenever a change is detected in one of the outgoing links. Also, LSPs are tagged with a sequence number in order to indicate their freshness (most recent LSPs have greater sequence numbers).

On the other hand, as a node receives the LSPs, it updates its global view of the network topology in terms of link states. Given this map, the node runs a shortest path algorithm (usually, Dijkstra's shortest path algorithm [5]) and determines the optimal next hop for each destination in the network. Only the next hop and route cost for each destination are stored in the routing table.

Note however that nodes do not update their tables at the same time (due to the propagation time of LSPs). This can lead to the formation of some short-lived loops, which will disappear by the time the LSPs have reached every other node in the network.

1.1.2.1.2 Distance Vector Protocols

The distance vector algorithm, also called the Bellman-Ford algorithm, is based on a different approach. Each node advertises its entire routing table to its immediate neighbors only. As a node receives the routing information of its neighboring nodes, it updates its own routing table by looking at its neighbors' routing table. Basically, for each destination, a node looks at each of its neighbor's distance for that destination (distance from neighbor to destination) and increments it by the link cost to reach this same neighbor. The node, then, picks up the neighbor via which the distance is the smallest. Each node keeps track of its neighbor's distances and modifies its local vector if any changes occur.

The Bellman-Ford algorithm suffers from both short-lived and long-lived loops. Also, it is known for its "counting-to-infinity" problem, which occurs in case of link failure.

1.1.2.1.3 Critique

Both algorithms link-state and distance-vector have proven their efficiency in case of networks with very low mobility. However, they both collapse when it comes to frequently changing topology networks. They not only take a long time to converge, but they also consume a big part of the available resources in finding routes that would probably never be used. Plus, some sort of mechanism should be added in order to avoid the formation of neither short nor long-lived loops.

1.1.2.2 Enhanced Protocols for MANETs

With the growing interest for mobile ad-hoc networks, a large set of IP-based routing protocols was proposed to the MANET working group. No standard has been adopted yet, but some promising protocols are enthusiastically under study. One of the rising protocols is the subject of this report. It is called Ad-hoc On Demand Distance Vector (AODV) and finds some of its inspiration in the Destination Sequenced Distance Vector (DSDV) protocol.

1.1.2.2.1 Destination Sequenced Distance Vector Protocol

As the name indicates, DSDV is originally based on the distance-vector algorithm.

Two major modifications were added in order to make the protocol suitable more for mobile ad-hoc networks. The first modification was added to cope with the "counting to infinity" and "loop formation" issues. The idea is very simple and consists of tagging each route with a sequence number in order to indicate its freshness. Basically, a route with a greater sequence number is a route that was issued later in the time and has therefore more chances to be accurate with regard to the current topology of the network. The destination sequence number is originally issued by the destination node. This number is maintained within each node by incrementing it whenever a change occurs along this same route. The sequence number and the cost route are included in the update messages. Thus, when a node receives an update message for a given route, it first checks the sequence number that is included in it and compares it with the one that is stored in the routing table. Only update messages with greater sequence number are taken into account. In case of equality, the route with smaller metric is to be considered.

The second improvement that was added to the original algorithm aims at reducing the over-head generated by the protocol. It consists of the introduction of two types of update messages: full dump and incremental. A full dump packet is used to advertise all available routing information whereas an incremental packet has smaller size and is used to advertise only the changes between two full dumps.

Indeed, DSDV is much more stable than the basic Bellman-Ford algorithm. However, DSDV still relies on periodic messages to maintain its routing information. Therefore, a final adjustment was needed to minimize the amount of over-head traffic generated.

1.1.2.2.2 Ad-hoc On-demand Distance Vector Protocol

AODV brings another brick to the edifice. As an improvement on DSDV, AODV reduces the amount of control traffic by simply minimizing the number of enquired routes. Instead of building a route for all possible destinations in the network, a node only creates and maintains routes that it really needs. When a route is needed, a node initiates a request in order to locate its interlocutor node. On the other hand, when a route is no longer used, it is simply expunged from the routing table. This approach is known as source-initiated on-demand routing as opposed to table-driven routing. It is also known as reactive as opposed to proactive. For reasons of scalability, proactive protocols may not find the same response within the MANET community as reactive protocols, such as AODV, DSR, or OLSR would probably have.

Another adjustment was added as far as route maintenance is concerned. In fact, in case of link failure, the node upstream immediately broadcast an update message to the set of nodes that are truly affected. Periodic updates such as full dump packets in DSDV were completely eradicated.

Of course, AODV uses the same set of sequence numbers as in DSDV, which guaranties loop-freedom in its routes.

1.2 AODV Specifications

We mentioned in next section 1.3.2.2, the Ad-hoc On-demand Distance Vector protocol uses a whole different approach to build its routing information. As a matter of fact, route enquiries are initiated on an on-demand basis.

When a node wishes to send a packet to a destination node, it initiates a discovery process in order to locate it. If no route is found within a specific period of time, the initiator node assumes that the destination node is unreachable. The discovery process is aborted and the corresponding data packets are dropped. On the other hand, if the initiator node receives a route as a response to its enquiry, it updates its routing table by creating an entry for the destination node.

Once an entry is created, a maintenance process is triggered in order to monitor the status of the just created route if a route is no longer used, a node deletes from its routing table. If a failure occurs along an active route, the node upstream immediately notifies the earlier hops of such a breakage using a specific type of control packets. In the presence of packets still in need of a route, affected nodes may re-initiate new discovery activities in order to find a replacement route.

AODV defines the routing information in a distributed table-driven manner. This means that each node along a particular path has to maintain a routing table entry for the destination node down that same path. This is opposed to the source routing approach where only the source node is aware of the complete path (hop-by-hop) toward the destination. AODV also allows each node to maintain one and only route per destination.

It is good to know then that some other routing protocols allow multiple route discoveries. In this case, if a primary route collapses, an alternative one is used.

Routing table entries are defined in AODV in the form of tuples <destination address, next hop address, sequence number, distance, precursor list, expiration date>. Sequence numbers are used to guarantee loop-freedom and indicate the freshness of a route. Distances are expressed in number of hops. Whenever a route is invalidated, the destination sequence number is incremented by one and the distance is set to infinity. The precursor list designates the set of neighboring nodes that are effectively using that entry to forward data packets. Finally, the expiration date indicates the time upon which the entry must be deleted. Of course, the expiration date is extended every time the entry is used.

1.2.1 Route Discovery

The discovery process is a mechanism that allows each source node participating in a MANET to locate (given its IP address) a destination node for which it has a packet to send. Of course, nodes initiate discovery requests only for nodes that have not been previously located (those for which node has not an entry in its routing table). If an entry already exists, then data packets are immediately sent to their destination and the discovery step is skipped.

In order to initiate a discovery process, a node has to broadcast a route request packet (RREQ) to its neighboring nodes. The packet is propagated by every other node in the network until a route is found (see Figure 3).

1.2.1.1 Generating route requests

Basically, if a node source A is looking for a destination node B in a MANET, A would let every other node in the network know that it is looking for B by broadcasting a RREQ packet to its neighbors. The RREQ includes the IP address and sequence numbers of both source and destination. The destination sequence number refers to the sequence number associated to the last route to the destination node B that the source node A has been aware of. If no trace of the sequence number is found, a default value of 0 is used.

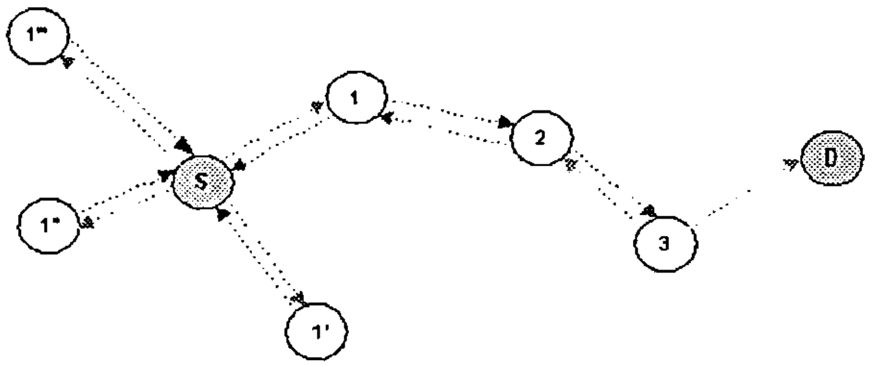


Figure 3 RREQ propagation

Each intermediate host which receives the broadcast RREQ has to re-broadcast it at a certain extent until the RREQ reaches either the destination node B or an intermediate node which is aware of a fresh enough route (according to the included destination sequence number) to the destination node B.

Two other fields of a RREQ are the time to live (TTL) and the broadcast ID.

The TTL field allows a discovery initiator to control the degree of dissemination of its RREQ within the network. For instance, a RREQ packet whose TTL field is set to 2 will travel through 2 hops at most from the source node. When a RREQ is broadcast, the source node sets its TTL field to an initial value in terms of hops and waits for a corresponding period of time before taking any action. If by any chance, a route is received before the waiting period is finished, then the discovery process is successfully terminated. On the other hand, if the waiting period arrives to its term without having received any response, the source node re-broadcasts the "same" RREQ packet and again waits for another period of time. This time, however, the RREQ has a bigger TTL value and the waiting period is consequently longer. Having a bigger TTL value, the new RREQ will reach a larger set of nodes and will hopefully lead to a route reply generation.

If still no reply is received, the source keeps on re-broadcasting the RREQ with an *incremented TTL up to a maximum number of retries*. Upon threshold, the discovery period is aborted. These mechanisms of broadcast control are also known as the expanding ring search technique.

Also, each RREQ packets is tagged with a sequence number, called Broadcast ID. This tag provides a mean for nodes to distinguish the different RREQs emanating from the same node and is incremented after each broadcast. A couplet <source IP address, Broadcast ID> uniquely identifies a RREQ and a RREQ with a greater Broadcast ID is fresher. As an intermediate node processes a RREQ issued from a particular node, it records the corresponding Broadcast ID. Later, the intermediate node will only process RREQs from a same source node with a greater Broadcast ID. Other RREQs with smaller Broadcast ID are simply discarded.

Finally, a RREQ carries a hop count field that records the number of hops that the RREQ has traveled through.

1.2.1.2 Forwarding route requests

An intermediate node **I** receives a RREQ emanating from a source node S which is requesting a route for a destination node **D**.

First of all, **I** check the Broadcast ID and the source node of the just received RREQ. If the RREQ is obsolete (smaller or equal Broadcast ID), then R does not process it. The RREQ is destroyed.

In case **I** has to process the RREQ, it first creates or updates a reverse route to the source node S (Figure 4 illustrates all the reverse routes that were created as the RREQ propagated in Figure 3). This route will eventually be used to carry the route reply back to the source node S. The next hop along this path is the node from which the RREQ was received and the corresponding sequence number is the source sequence number that is included in the RREQ.

Once the reverse route is created, **I** checks if it has a fresh enough route to **D**. If this is the case, **R** generates a route reply packet (see paragraph bellow) and unicasts back along the reverse route. At this point, the RREQ does no longer need to be re-broadcast. If **R** does not have a fresh enough route, it will re-broadcast the RREQ after having consumed one hop of its time to live (TTL decremented by one). A RREQ that has expired (TTL = 0) is not re-broadcast. In case of re-broadcast, the hop count field is also incremented by one.

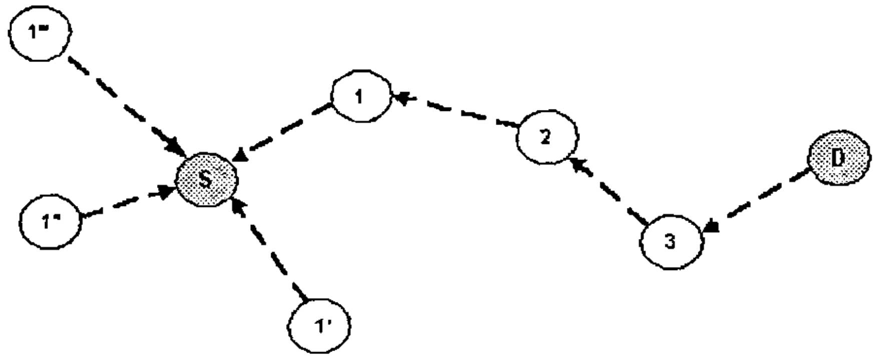


Figure 4 Reverse route creation

1.2.1.3 Generating route replies

When a node wants to make a route available (either a destination node or an intermediate node with a fresh enough route), it unicasts a route reply packet (RREP) back to the source node that initiated the discovery process.

The RREP contains IP addresses of both source and destinations nodes and the sequence number of the advertised route. It also includes a hop count field (similar to the one in RREQ packets) and a lifetime field whose value indicates the validity period of the advertised route.

1.2.1.4 Forwarding route replies

As it is illustrated in Figure 5, the forward path (from source to destination) is built as the route reply packet travels down along the reverse path. Each node receiving the RREP creates an entry for the destination node **D**. The destination sequence number and hop count are copied from the RREP itself and the next hop along this path is the last node that forwarded the RREP.

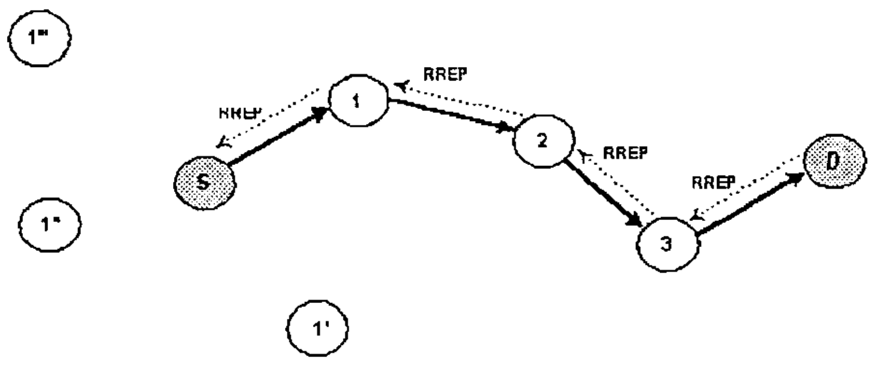


Figure 5 Forward route creation

If the RREP has not reached its destination **S** yet, then it has to be forwarded to the next hop along the reverse path. Of course, the hop count field is incremented first.

When the RREP finally reaches the source node, it does no longer need to be forwarded. After S has created a forward entry toward **D**, it automatically destroys the RREP packet. The discovery period is terminated and the newly established route can now be used to send the data packets waiting on the buffer.

So far, two limitations of AODV were pointed out. First, AODV discovers a single route between a pair source/destination. Second, AODV discovery procedure assumes that RREP packets will travel down the same path that RREQ packets went through, but in the other way. This assumption of bi-directionality makes AODV not suitable for situations where uni-directional links may exist. Yet, in a wireless environment, links are unlikely to be symmetric.

1.2.2 Local Connectivity Management

When a route toward a particular destination is created, a node may use a set of mechanisms in order to monitor its status. In other words, each node along a particular path tries to make sure that the next hop toward that destination remains available and viable as long as needed. If the next hop is active, the route is still valid. If not, the current node has to notify earlier nodes that the route is no longer valid.

Monitoring can be done in two different fashions: proactive or reactive. In the first approach, some kind of anticipation is introduced. Each node constantly monitors the actual state of activity of its neighbors by updating its local connectivity map whenever a broadcast packet is received. Said differently, whenever a broadcast packet is received the current node updates or creates its routing table entry for the originator node. This entry would have a short lifetime period which would correspond to the maximum period of time that a neighboring node is allowed to remain silent before the current node assumes that it is unavailable. Thus, as long as the neighboring node is available, the corresponding entry remains valid. On the other, if a routing table of a neighboring node happens to expire, the current node assumes that the downstream link has broken and a link failure notification procedure is immediately called.

Sometimes however, a node's silence is not an accurate indicator of its availability. Therefore, AODV makes use of periodic hello message broadcasts. Basically, if no broadcast packet has been emitted within the last hello interval period of time, a node advertises its presence by broadcasting a specific RREP packet containing its identity and sequence number. Hello messages were also introduced in order to make AODV independent from lower layer platforms local connectivity management can rely entirely on hello message exchanges. Lower layer messages can be ignored.

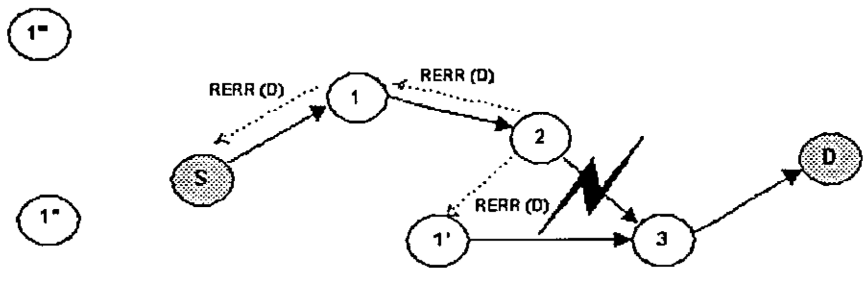
The second approach is called proactive. This means that a link breakage is only detected upon a data transmission failure. In this case, a failure may be detected a bit later. However, a reactive approach induces a quite heavy overhead which might not be adequate for mobile ad-hoc networking purposes.

1.2.3 Route Maintenance

When a link failure is detected along a route, the node upstream propagates a route error (RERR) packet in order to notify earlier nodes down the path of such a breakage. The RERR contains the list of all lost destinations along with their sequence numbers incremented by one.

As the RERR travels down the forward path (Figure 6), each affected node updates its routing table by invalidating the corresponding routes. For each destination included in the RERR packet, the current node sets the distance value to infinity and updates the corresponding sequence number by copying it from the RERR packet. Plus, if the precursor list is not empty, the current unreachable destination remains within the RERR packet which will be broadcast to predecessor nodes. Of course, the RERR will be re-broadcast only if at least one unreachable destination is remaining.

Also, note that each node invalidates its entry for a given destination only if the RERR was received from its own next hop toward that same destination. For instance, even though **1'** receives the RERR, **1'** will not invalidate its route for **D** because As a matter of fact, **1'** receives the RERR from **2** which is not, according to its routing table, the next hop toward **D**. Therefore, the RERR is simply destroyed.



Node 2 (upstream node): precursor list not empty => RERR generated
Node 1': not along the path => RERR destroyed
Node 1: along the path + precursor list not empty => RERR forwarded
Node s: source node + precursor list empty => RERR destroyed + discovery reinitiated if needed

Figure 6 RERR propagation

1.2.4 Route expiry and deletion

In keeping with the purely on-demand nature of AODV, each node gets rid of all the routes which are no longer used. Thus, if an active route is not used within the last period of time out, the current node first invalidates it by incrementing its sequence number and setting the hop count to infinity. The entry is said to be expired. At this point, the entry has not been removed from the routing table. In fact, the entry will stay in the expiry state for a delete time period, before being completely expunged from the routing table.

The reason behind the expiry/deletion mechanism is for a node to keep record of the sequence number of the expired route as long as possible. As a matter of fact, if a node deletes a route upon expiry (without transiting through a deletion period), it would lose track of its sequence number and the default value will be used. This situation can lead to the formation of loops.

In fact, if a node needs to keep track of the sequence number, it should never delete an expired entry. This would certainly be safer. However, such a solution may not be practical in a dynamic network where a node is likely to see lots of nodes for short periods of time. Also, rebooted nodes would have no recollection of any sequence number whatsoever. Therefore, a proper deletion mechanism has to be introduced.

1.3. The Project and Research Work

AODV routing protocol, is not standardized according to IETF (Internet Engineering Task Force). So, till now simulations are developed by different research institutes. In this project we have to work out for "Secure Route" in AODV protocol, to defend against network attacks, such as Interception and Modification. Here we used the signature technique which helps the source and destination node to communicate securely. For signature technique we use the authentication and encryption technique. These techniques are used in such a way that it does not affect the performance of AODV protocol. In encryption technique we used the RSA algorithm. This is light weighted algorithm then other encryption algorithm and also not easily breakable.

We used these techniques in such a way that hacker does not know the path and the source and destination. First AODV will make the route from source and destination. Then this route is sent back to source, where it will encrypt the path address including Hop Counts and append the public key to it. After this the message is transferred to the destination. When source receive the acknowledgement from destination then it will transfer the message without public key.

1.4. Objective

The objective of this study is the implementation, documentation and evaluation of a simulation model for the AODV routing protocol, with the ability of Secure Routing Protocol.

Chapter 2
System Analysis

2. SYSTEM ANALYSIS

At a technical level, software engineering begins with a series of modeling tasks that lead to a complete specification of requirements and a comprehensive design representation for the software to be built. The Analysis model, actually a set of models, is the first technical representation of a system. Over the years many methods have been proposed for analysis modeling. However two of them dominate the analysis modeling landscape now a day. The first, *structured analysis* is a classical modeling method and the other approach is *object oriented* method. We have used the first modeling technique for the analysis of the software Face Recognition through Infrared Image and Eigenfaces.

2.1 Structured Analysis

The Structured Analysis Model must achieve three primary objectives.

- Describe what the customer requires.
- Establish a basis for the creation of a software design.
- Define a set of requirements that can be validated once the software is built.

To accomplish these objectives, the analysis model derived during the structured analysis takes the form illustrated in Figure 2.1. At the core of the model lies the data dictionary a repository that contains description of all data objects consumed or produced by the software. Three different diagrams surround the core. The entity-relationship diagram (ERD) depicts relationships between data objects. The ERD is the notation that is used to conduct the data modeling activity. The attributes of each data object noted in the ERD can be described using a data object description. The Data flow Diagram (DFD) Provide an indication of how data are transformed as they move through the system and depict the functions and sub functions that transform the data flow.

The DFD provides additional information that is used during the analysis of the information domain and serves as a basis for the modeling of function. A description of each function presented in the DFD is contained in a process specification (PSPEC).

The State-transition diagram (STD) indicates how the system behaves as a consequence of external events. To accomplish this, the STD represents the various modes of behavioral modeling. Additional information about control aspects of the software is contained in the control specification (CSPEC).

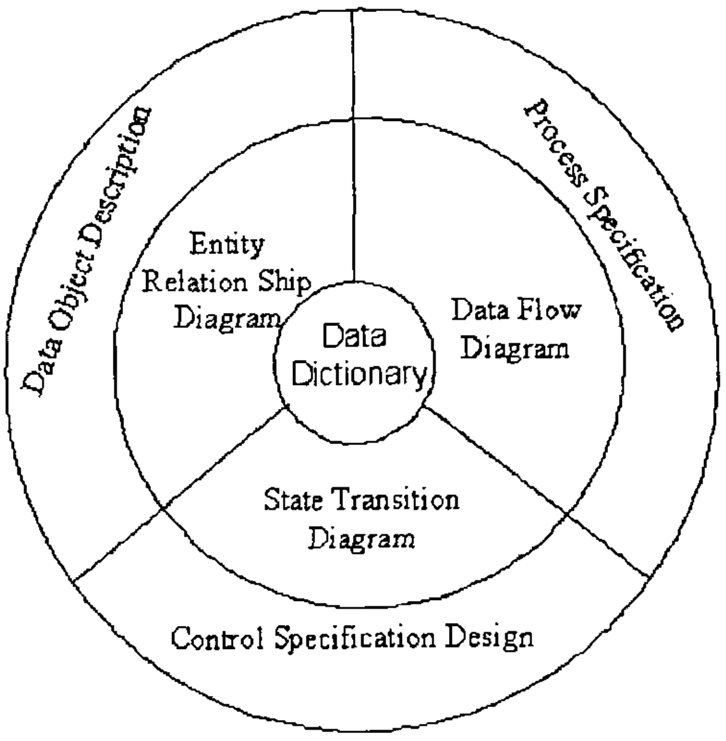


Figure 2.1 Analysis Model

2.1.1 Object Description

In Object Description of software keeps the information, which should be stored for further use. In Object Description ERD is use to model the software data base.

requirements. We do not require the Object Description phase or ERD modeling because it is not needed in our project.

2.1.2 Data Flow Diagrams (DFD)

As information moves through the software, it is modified by a series of transformations. A DFD is a graphical technique that depicts information flow and the transforms that are applied as data move from input to output. The DFD is also known as *Data flow graph* or a *bubble chart.*

Our software is expanded up to the second level DFD and all the functions shown in the DFDs are described in PSPEC.

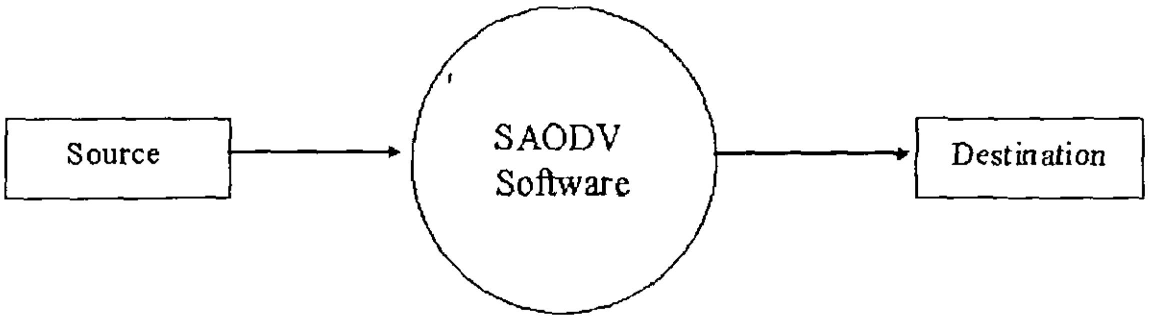


Figure 2.2 Level 0 DFD for Secure Ad hoc On-demand Distance Vector Routing Protocol.

Figure 2.2 shows the Context level DFD for the software Secure Ad hoc On-demand Distance Vector Routing Protocol. This level is the highest level of abstraction where no details are shown only the input to the software and output from software is shown. There is only one bubble which is the software and reveals no function of the software.

Now the DFD is expanded and level one shows the detail of the process or functions of the software. This level is called level 1 DFD for the software and reveals the function but not the sub function.

Figure 2.3 is the expansion of bubbles in level 0 DFD and here the level of abstraction decreases but only up to the functions still the sub functions are not reveled. It

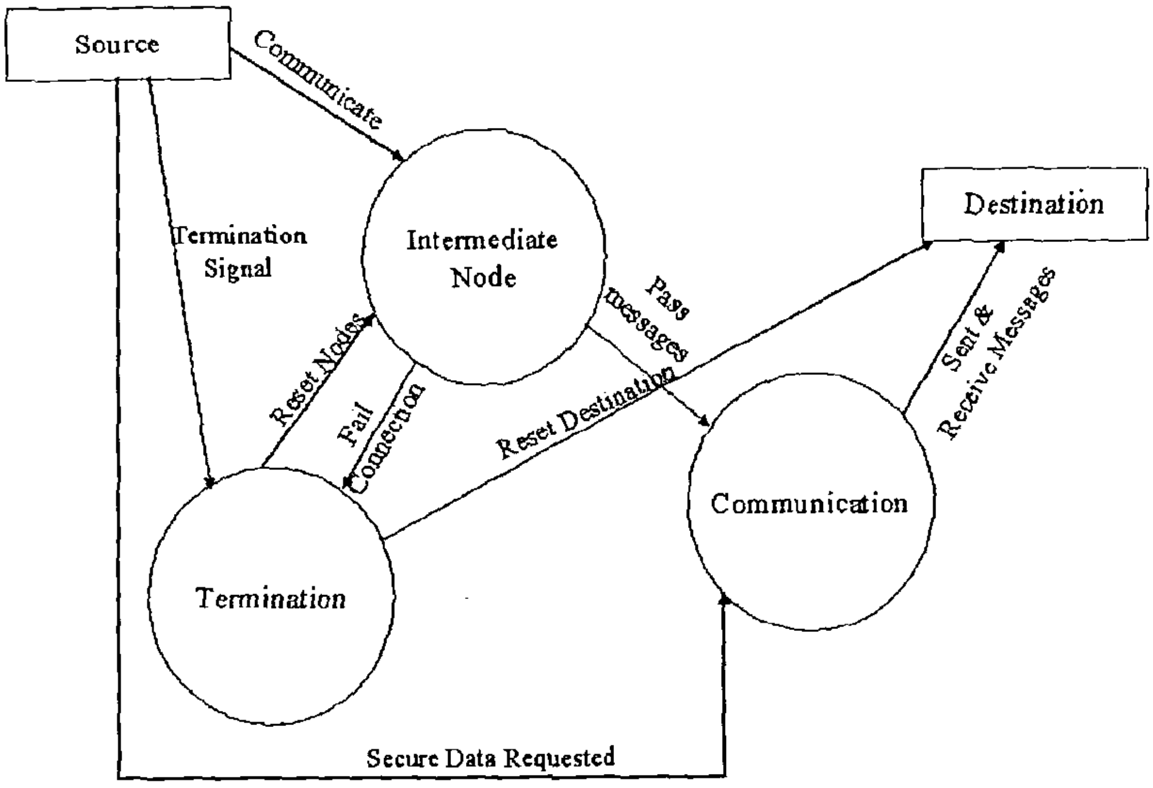


Figure 2.3 Level 1 DFD of Secure Ad hoc On-demand Distance Vector Routing Protocol.

also shows the data storage and the arrow in show which process store the data and which process uses the stored data. The Source shows the source node of router which makes connection with other nodes till the requested Destination is reached. Once the connection is established then the communication starts between the Source and Destination through Intermediate processes. If the connection failed due to any reason or reset is called then automatically every node is reset to make other connections.

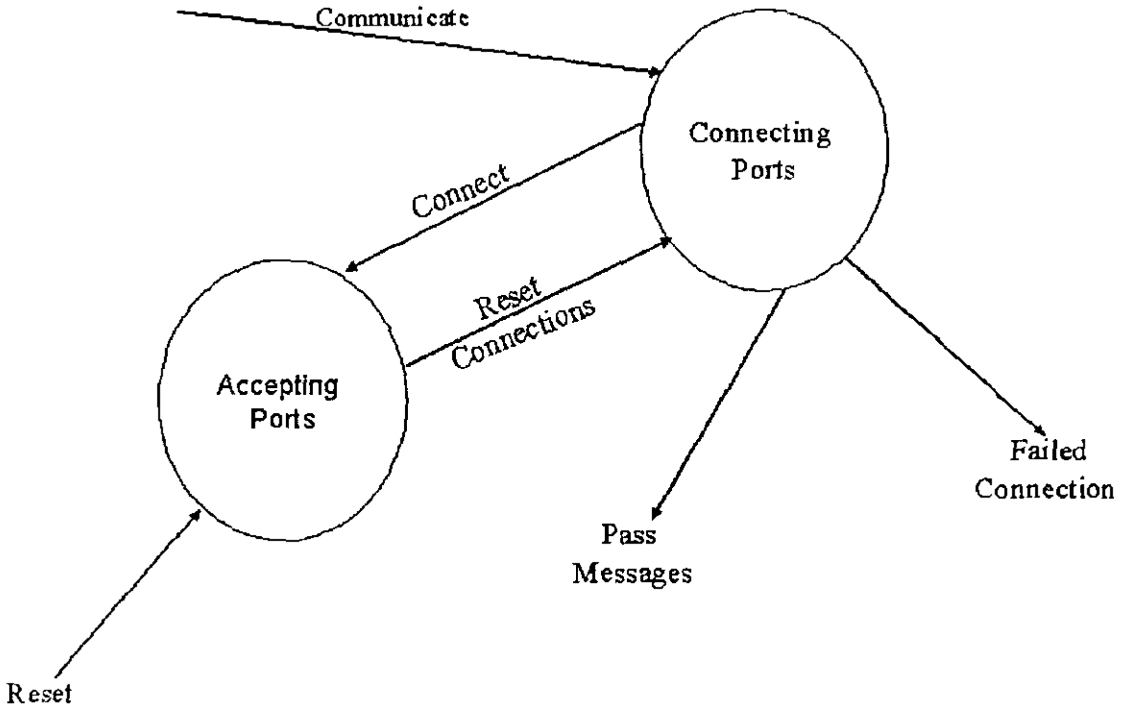


Figure 2.4 Level 2 DFD of Intermediate Node.

Level 2 DFD for intermediate node show functionality of Accepting and Connecting different nodes in the network. Source first connects its self to intermediate node and pass on the path information to it, which helps to connect to other nodes till destination is reached. When connection fails that is it does not find the path further then it pass information to End Communication for resetting all nodes. The connecting ports process passes the messages from source to communicating process which further send it to destination.

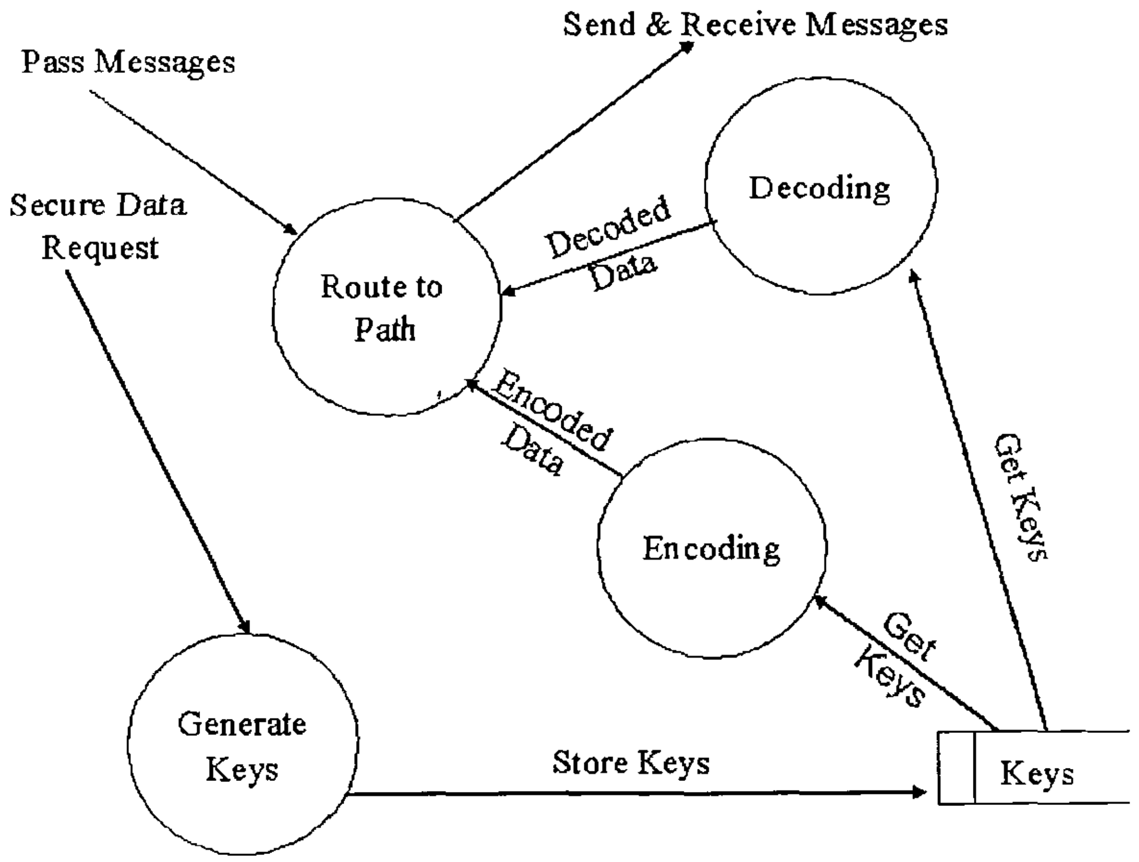


Figure 2.5 Level 2 DFD of Communication Process

In communication process the source and destination communicate with each other. When source needs to communicate with destination then it generates the key that is public and private keys by RSA algorithm and also make the check sum which is manually transfer to destination. Then source and destination communicate with each in a secure way.

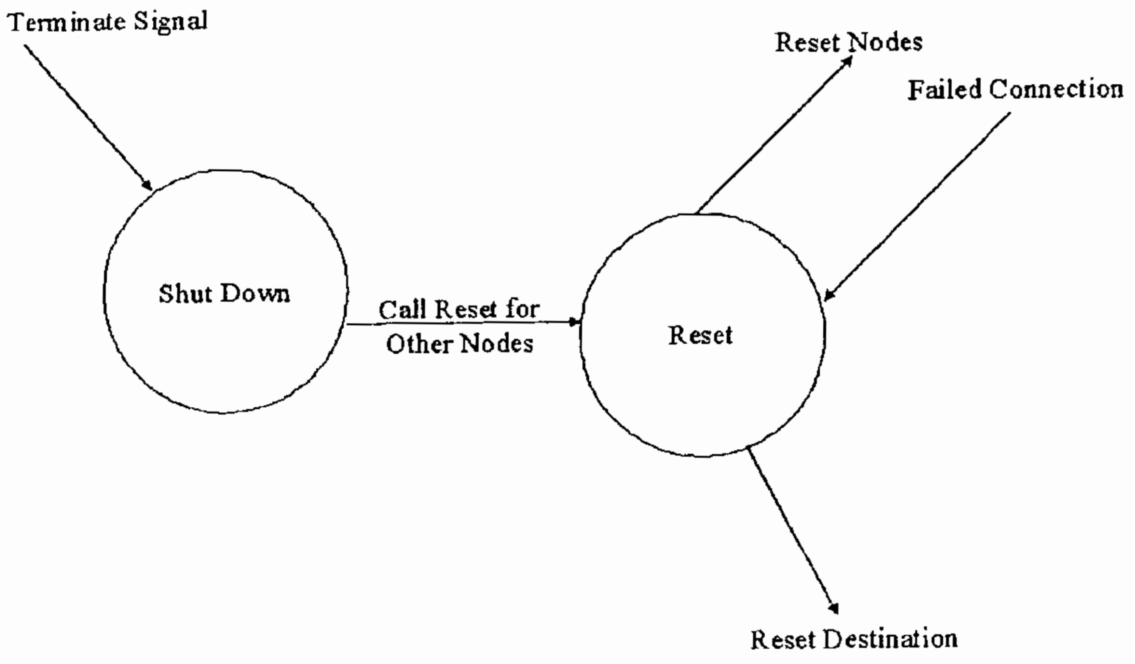


Figure 2.6 Level 2 DFD of Termination Process

In Termination process the main thing is resetting other nodes when ever shut down or connection failed occur. In shut down process the node it self shut down while other nodes are resetted.

2.1.3 Process Specification (PSPEC)

The PSPEC of our software in the form of PDL are:

2.1.3.1 Initialize the Router

```
Procedure Initialize the router;
        Read the information given to software;
        If command is valid then initialize the Router;
        Else do not initialize the server;
        End if;
        If information given in path nodes then source,
        Else intermediate node or destination.
        End if;
Endproc
```

2.1.3.2 Accepting ports

```
Procedure Accept Performance;
    Read the input data;
    If data is valid then accept the socket;
    Else do not accept the socket;
    End if;
Endproc
```

2.1.3.3 Connecting ports

```
Procedure Connecting to other node;
    Connect to other node by the port number received from source;
    If connection is valid then connect to the node;
    Else pass the connection failed message;
    End if;
    Send the source information to other connected node till the destination is
    reached.
Endproc
```

2.1.3.4 Perform Communication

```
Procedure Communication;
    Pass on the send and receive data from source to destination through
    intermediate nodes
    If secure route needed then generate key;
    Else transfer data without security;
    End if;
Endproc
```

T-925

2.1.3.5 Perform Key Generation

```
Procedure Key generation;
        Use the RSA to generate keys, one for public use and other for private use
        Generate keys; Get the port number and total hop counts in source,
        Encode data and Decode at Destination.
Endproc
```

2.1.3.6 Encoding Decoding

```
Procedure Encoding Decoding;
        Get the data from user and encode it at source side and decode it at
        destination node.
Endproc
```

2.1.3.7 Termination

```
Procedure Termination;
        If shutdown then shutdown itself and reset other nodes;
        Else If reset then reset itself and other nodes;
        Else reset all;
Endproc
```

2.1.4 State Transition Diagram (STD)

The State Transition Diagram indicates how the system behaves as consequence

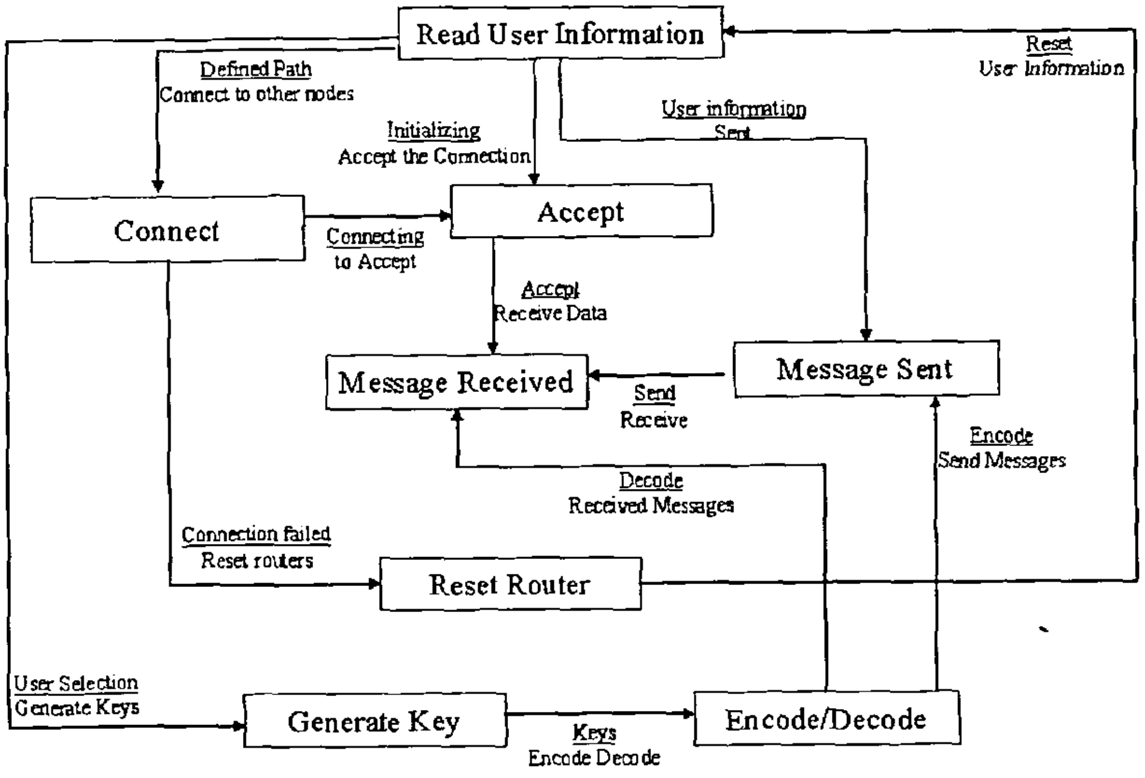


Figure 2.7 State Transition Diagram

of external events. By studying STD, a software engineer can determine the behavior of the system and can ascertain whether there are "holes" in the specified behavior. Figure 2.7 shows the State Transition Diagram for the software Secure Ad hoc On-demand Distance Vector Routing Protocol.

2.1.5 Control Specification (CSPEC)

The Control Specification (CSPEC) represents the behavior of the system in two different ways. One is called specification behavior and the second one is the combinational specification. The CSPEC does not provide any information about the inner working of the processes that are activated as a result of this behavior.

2.1.6 Data Dictionary

The Data Dictionary is an organized listing of all data elements that are pertinent to the system, with precise, rigorous definitions. There is no such Data Storage in this Software.

Chapter 3
System Design

Figure 3.1 shows the realtion of Analysis model to Design model and the arrows shows which of the information from the analysis model is necessary for which design. Data Design is created using the DataDictionary and Entity-Relationship Diagram information of Analysis model. Archicectural Design is creared using the information from Data Flow Diagram of the Analysis model. Interface Design is also created using the infromation from data flow diagram. Procedural Design uses the information from CSPEC, PSPEC and state-transition diagram of the Analysis model.

3.2 Design Types

There are four types of Designs.

Data Design

Interface Design

Architectural Design

Procedural Design

3.2.1 Data Design

The Data Design transform the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity-relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design.

Data Design is the first of four design activities that are conducted during software engineering. The primary activity during data design is to select logical representation of data objects identified during the requirement definition and specification phase. The selection process may involve algorithmic analysis of alternative structures in order to determine the most efficient design or may simply involve the use of a set module that provide the desired operations upon some representation of an object.

Our project does not contain any data base information, so we do not include it in this section.

3.2.2 Interface Design

The interface design describes how the software communicates within itself, to systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g. data and/or control). Therefore, the data and control flow diagrams provide the information required for interface design. The interface Design can be seen in Appendix A.

3.2.3 Architectural Design

The Architectural Design defines the relationship among major structural elements of the program. This design representation the modular framework of a computer program can be derived from the analysis model(s) and the interaction of subsystem defined within the analysis model.

The Program Structure of the software is show in Figure 3.3, 3.4, 3.5, 3.6, 3.7 of DFD level 1 and 2.

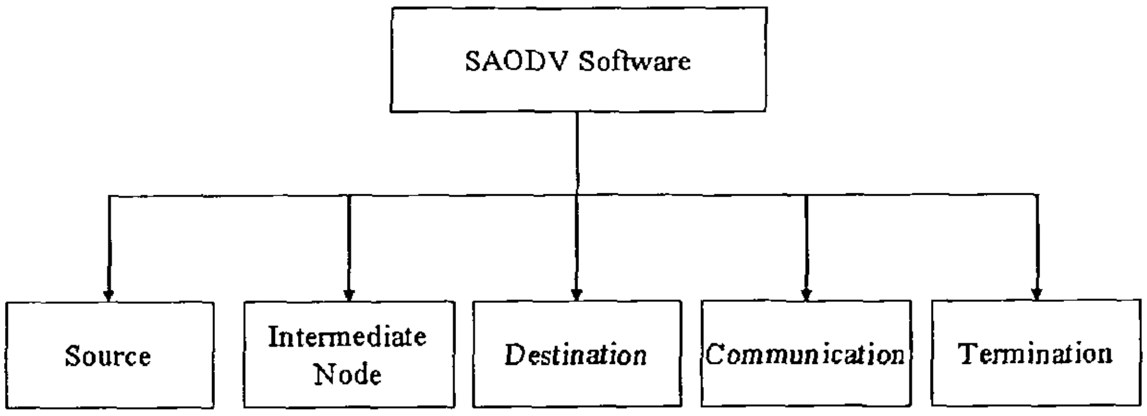


Figure 3.2 Program structure

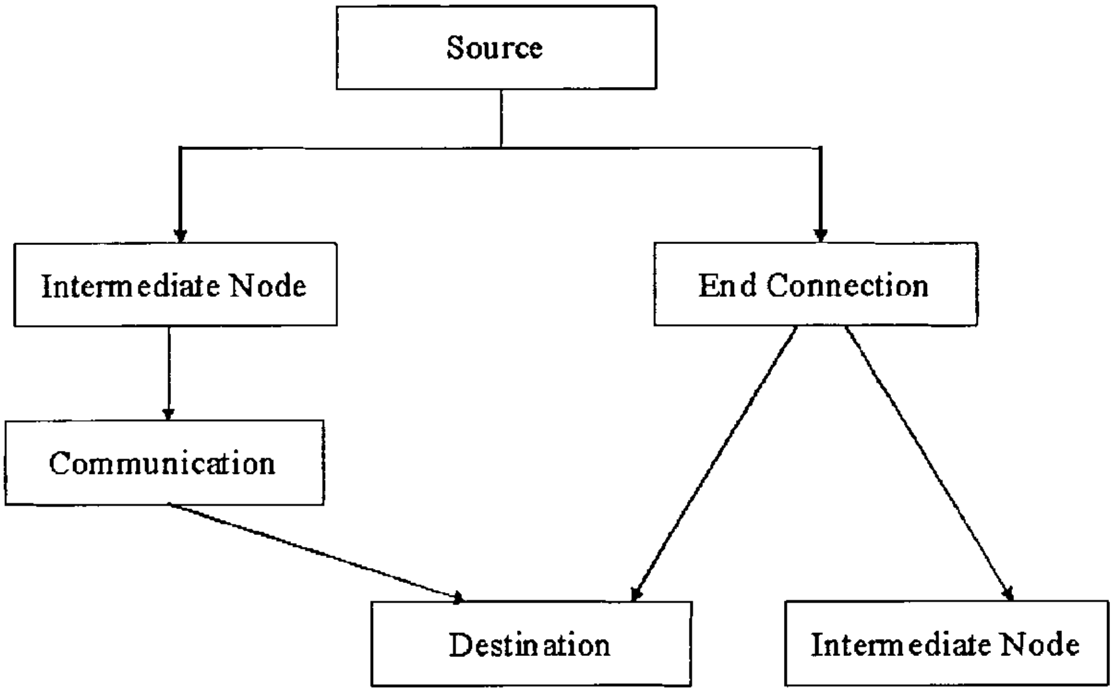


Figure 3.3 Program Structure of SAODV process

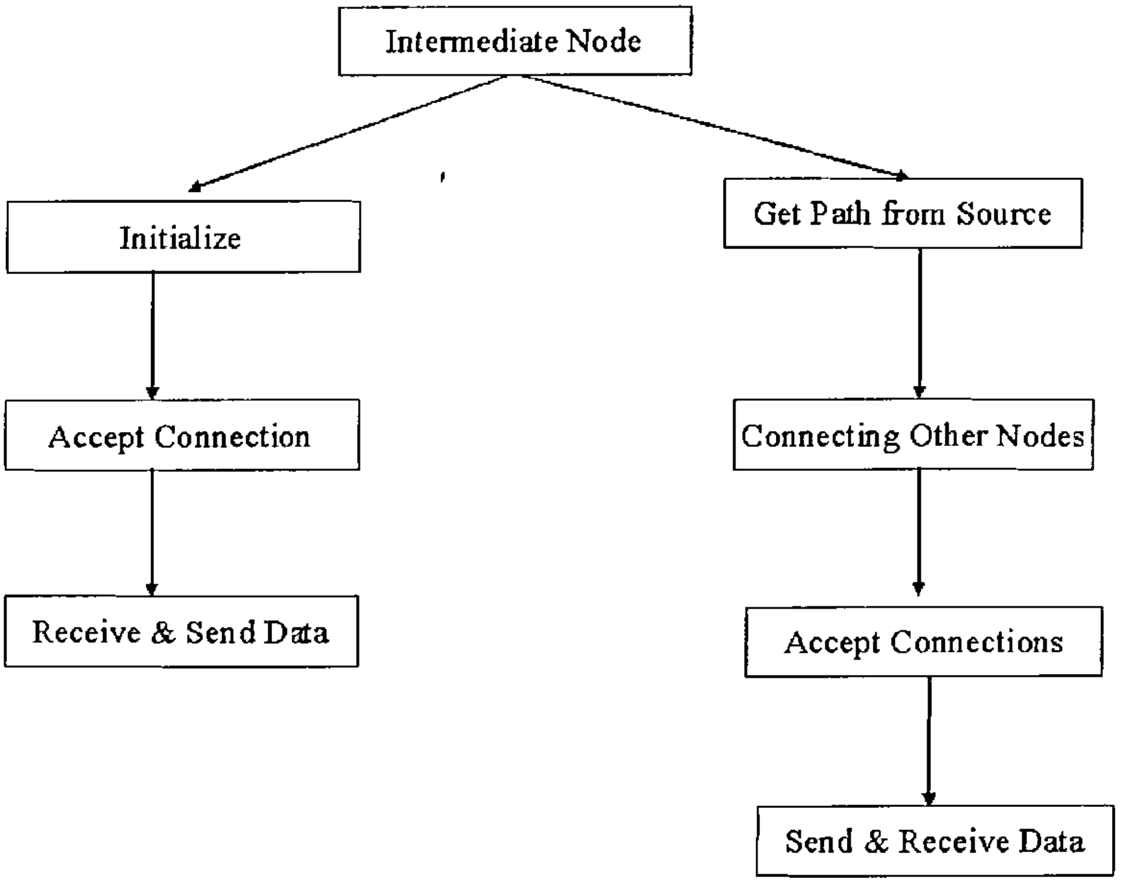


Figure 3.4 Program structure of Intermediate Node process

3.2.4 Procedural Design

The procedural design transforms structural elements of the program architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC and STD serve as the basis for procedural design.

The Procedural Design for the software is shown in the Figures 3.7, 3.8, 3.9 and 3.10.

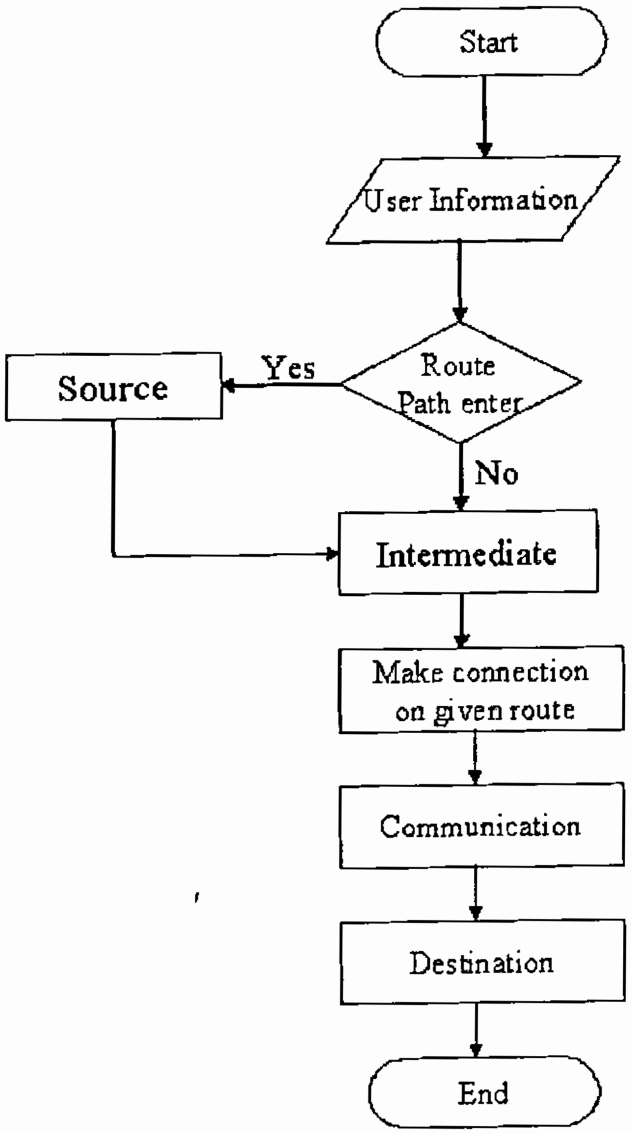


Figure 3.7 Procedural Design

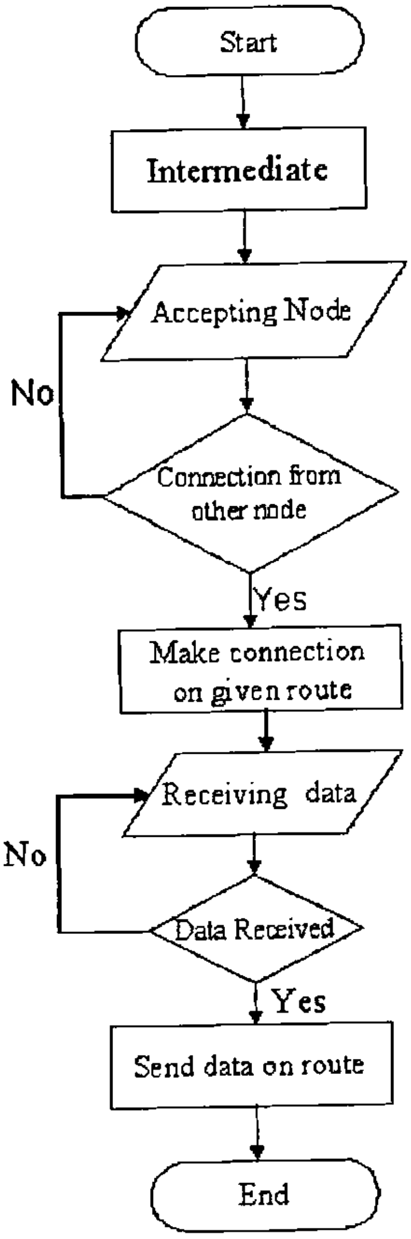


Figure 3.8 Procedural Design of Intermediate Process

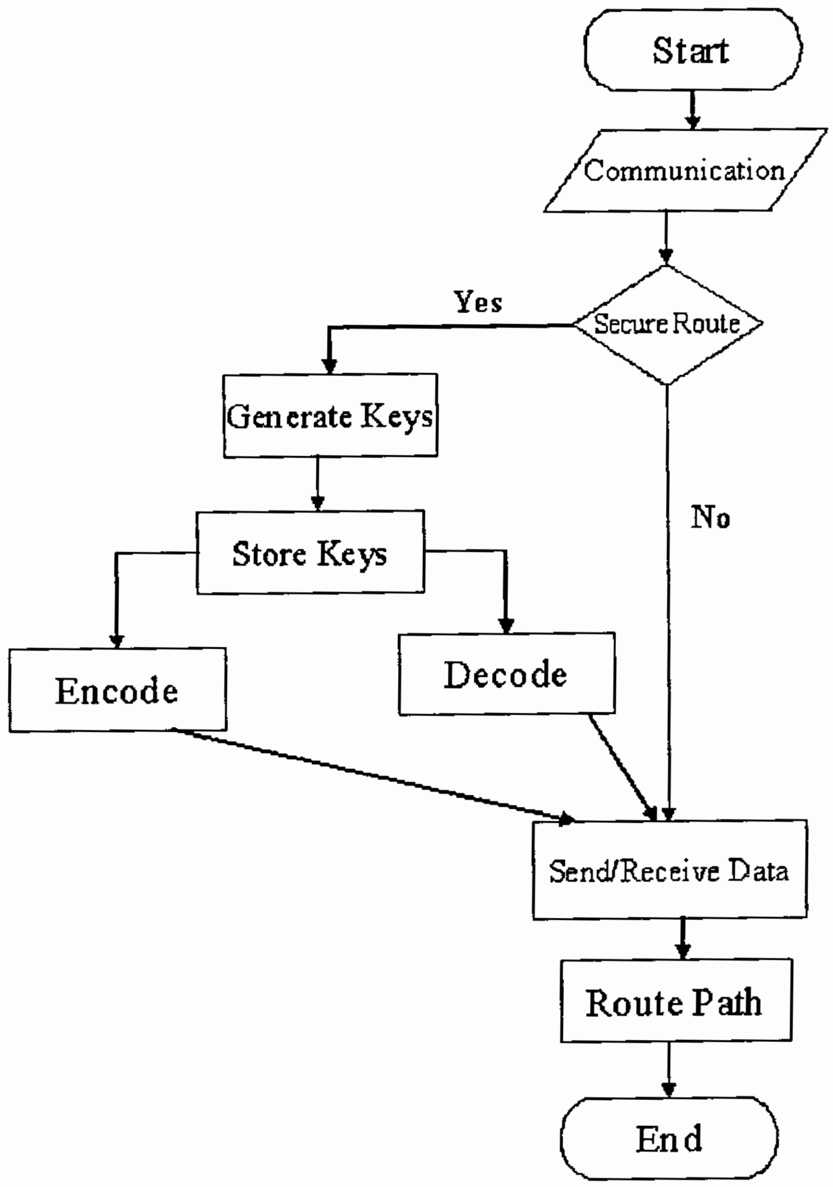


Figure 3.9 Procedural Design of Communication Process

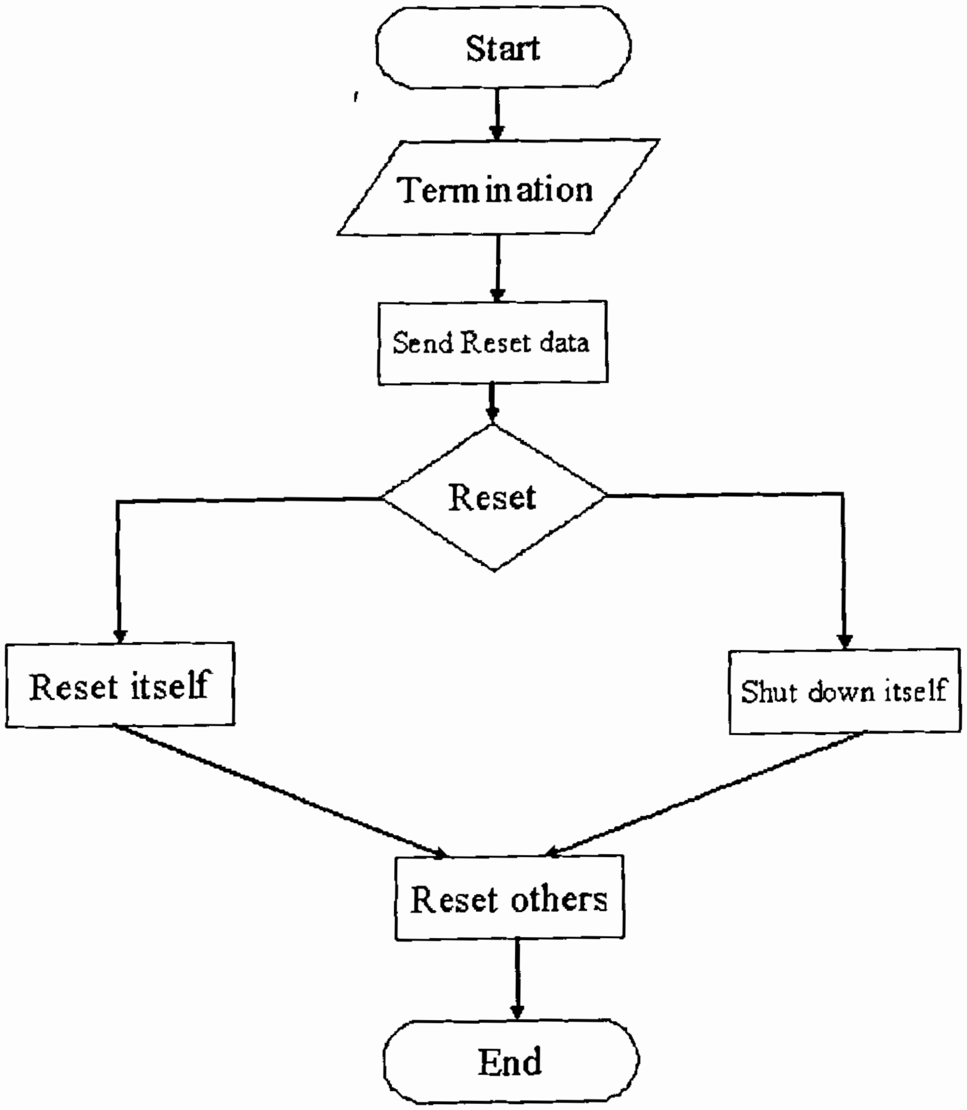


Figure 3.10 Procedural Design of Termination process

The Design Activity is completed here and can easily be mapped to coding in implementation section which is last activity of the software development Process.

4. SYSTEM DEVELOPMENT AND IMPLEMENTATION

The second last activity in the project, comes before testing of the whole program, however the partial testing can be done during implementation after completion of every module.

Visual C++ 6.0. is used to developed the software. The TCP/IP Socket APIs are used to communicate with different application. For secure route we used RSA and SHA algorithms, which are used for signatures.

4.1 Sockets

The Windows Sockets specification defines a binary-compatible network programming interface for Microsoft Windows. Windows Sockets are based on the UNIX sockets implementation in the Berkeley Software Distribution (BSD, release 4.3) from the University of California at Berkeley. The specification includes both BSD-style socket routines and extensions specific to Windows. Using Windows Sockets permits your application to communicate across any network that conforms to the Windows Sockets API. On Win32, Windows Sockets provide for thread safety.

Windows Sockets under network protocols including Transmission Control Protocol/Internet Protocol (TCP/IP), Xerox Network System (XNS), Digital Equipment Corporation's DECNet protocol, Novell Corporation's Internet Packet Exchange/Sequenced Packed Exchange (IPX/SPX), and others. Although the present Windows Sockets specification defines the sockets abstraction for TCP/IP, any network protocol can comply with Windows Sockets by supplying its own version of the dynamic link library (DLL) that implements Windows Sockets. Examples of commercial applications written with Windows Sockets include X Window servers, terminal emulators, and electronic mail systems.

The Windows Sockets specification, Windows Sockets: An Open Interface for Network Computing under Microsoft Windows, we use version 2.2, was developed as an open networking standard by a large group of individuals and corporations in the TCP/IP community and is freely available for use. The sockets programming model supports one "communication domain" currently, using the Internet Protocol Suite. The specification is available in the Win32 SDK.

4.1.1 Definition of a Socket

A socket is a communication endpoint an object through which a Windows Sockets application sends or receives packets of data across a network. A socket has a type and is associated with a running process, and it may have a name. Currently, sockets generally exchange data only with other sockets in the same "communication domain," which uses the Internet Protocol Suite.

Two socket types are available:

- Stream sockets

  Stream sockets provide for a data flow without record boundaries a stream of bytes. Streams are guaranteed to be delivered and to be correctly sequenced and unduplicated. We use stream sockets in our project.

- Datagram sockets

  Datagram sockets support a record-oriented data flow that is not guaranteed to be delivered and may not be sequenced as sent or unduplicated.

Both kinds of sockets are bi-directional: they are data flows that can be communicated in both directions simultaneously (full-duplex).

4.1.2 Socket Address Families

There are total 24 types of socket address families, in which AF_INET is used in this project. AF_INET is basicaly used for internetwork communication like UDP,TCP/IP,etc.

4.1.3 Socket Types

Socket API uses five types of socket that is STREAM for TCP/IP, DGRAM for UDP, RAW for raw protocol interfaces, RDM for relibility delivery messages and SEQPACKET for sequenced packet stream.

There are different types of other sokect functionalities which are not needed in this project.

4.2 Functionality Used in Visual C++ for Socket Implementation

Visual C ++ 6.0 provides two library for socket implementation i.e. "winsock2" which is used for consol base application and "afxsock" which is used for windows application. For connecting two applications with each other by sockets following steps are used.

- Socket initialization
- Bind Socket
- Listen Socket
- Accept Socket
- Connect Socket
- Send / Receive Messages
- Close Socket

4.2.1 Socket initialization

WSAstartup is used for starting up sockets. When socket is started then it is created by following command.

```
SOCKET socket (int family, int type, int protocol);
```

In socket family we use PF_INET which is basically for TCP/IP and UDP connections its type is integer. STREAM is the socket type which we have used for TCP/IP sockets and in protocol parameter we use 0, which is for dummy IP's. We use dummy IP's in this application because it is single PC document. Socket function returns the SOCKET which is further used for connecting two applications.

4.2.2 Bind Socket

The Windows Sockets bind function associates a local address with a socket. The bind function is used on an unconnected socket before subsequent calls to the connector listen functions. It is used to bind to either connection-oriented (stream) or connectionless (datagram) sockets. When a socket is created with a call to the socket function, it exists in a name space (address family), but it has no name assigned to it. Use the bind function to establish the local association of the socket by assigning a local name to an unnamed socket.

A name consists of three parts when using the Internet address family:

- The address family
- A host address
- A port number that identifies the application

For binding socket we use the following function.

```
int bind  (SOCKET s,
           const struct sockaddr FAR *name,
           int namelength);
```

In above function socket which is created is pass, the structure socket address contains socket address family i.e. AF_INET, in name length parameter we pass structure length. Bind function returns integer, if this integer is 0 it means that binding of socket is successful.

4.2.3 Listen Socket

When binding socket is successful then the TCP/IP socket will listen the acceptance of connect request from other side. Listen socket returns 0 when listing is successful, following parameters are passed to this function.

```
int listen  (SOCKET s,  int backlog );
```

The socket which is created and bind will listen the connect request from other application. In second parameter, queue of connecting sockets is maintained which will be accepted by accept function.

4.2.4. Accept Socket

The Windows Sockets accept function permits an incoming connection attempt on a socket. The accept function extracts the first connection on the queue of pending connections on socket s. It then creates a new socket and returns a handle to the new socket. The newly created socket is the socket that will handle the actual connection and has the same properties as socket s.

```
SOCKET accept  (SOCKET s,
                struct sockaddr FAR *addr,
                int FAR *addrlength);
```

The accept function can block the caller until a connection is present if no pending connections are present on the queue, and the socket is marked as blocking. Therefore, we use Threads which are helpful in handling the blocking state. First we create thread in initialize function and start the thread.

```
void CSockDlg::ThreadAccept(CSockDlg *pDlg)
{
        char Buff[]="I am on line";
        DWORD dw;
        int retcode;
        while(1)
        {
                retcode = sizeof(pDlg->Asin);
                pDlg->AS3 = accept(pDlg->LS, (struct sockaddr *) &pDlg->Asin,
                                        &retcode);
                if(pDlg->AS3 != INVALID_SOCKET)
                {
                        if(pDlg->Acall==0)
                        {
                                pDlg->RecAS1th=CreateThread
                                (0,NULL,(LPTHREAD_START_ROUTINE)ThRecAS2,pD
                                lg,0,&dw);
                                pDlg->Acall=1;
                                retcode = send(pDlg->AS3,Buff,sizeof(Buff),0);
                                break;
                        }
                        if(pDlg->Acall==1)
                        {
                                pDlg->RecAS2th=CreateThread
                                (0,NULL,(LPTHREAD_START_ROUTINE)ThRecAS2,p
                                Dlg,0,&dw);
                                pDlg->Acall=2;
                                break;
                        }
                        if(pDlg->Acall==2)
                        {
```

```
                    pDlg->RecAS3th=CreateThread
                    (0,NULL,(LPTHREAD_START_ROUTINE)ThRecAS2,p
                    Dlg,0,&dw);
                    pDlg->Acall=3;
                }
            }
            if(pDlg->exitflage = = 1 || pDlg->Acall = =3)
                break;
            Sleep(500);
        }
        ExitThread(0);
}
```

4.2.5 Connect Socket

The connect function is used to create a connection to the specified destination. If socket *s*, is unbound, unique values are assigned to the local association by the system, and the socket is marked as bound. For connection-oriented sockets (for example, type SOCK_STREAM), an active connection is initiated to the foreign host using *name* (an address in the name space of the socket). The Windows Sockets connect function establishes a connection to a specified socket.

```
    int connect (SOCKET s,
            const struct sockaddr FAR *name,
            int namelength);
```

If no error occurs, connect returns zero, otherwise, it returns SOCKET_ERROR. In project the connection socket function is used as follows.

```
    CS1= socket(PF_INET, SOCK_STREAM, 0);
            if (CS1 = = INVALID_SOCKET)
            {
```

```
        AfxMessageBox("Connect Socket Err");
        Reset();
}
memset(&Csin1, 0, sizeof(Csin1));
Csin1.sin_family = AF_INET;
int portno=0;
portno = BufToInt(buf);
if(portno = = 0)
{       retcode=closesocket(CS1);
        return 0;
}
Csin1.sin_port = htons(portno);
CString ServerName = "localhost";
if (pHostEnt = gethostbyname(ServerName))
{       memcpy(&Csin1.sin_addr, pHostEnt->h_addr_list[0],
        pHostEnt->h_length);
}
else
{       AfxMessageBox("CS1 ServerName LocalHost");
        Reset();
}
retcode = connect(CS1, (struct sockaddr *)&Csin1, sizeof(Csin1));
if (retcode == SOCKET_ERROR)
{       AfxMessageBox("CS1 Connect");
        errexit();
}
```

4.2.6. Send / Receive Messages

The windows sockets send function sends data on a connected socket. If no error occurs, send returns the total number of bytes sent, which can be less than the number indicated by length for non blocking sockets, otherwise, a value of SOCKET_ERROR is returned

```
        int send (SOCKET s, const char FAR*Buffer, int lengthBuffer, int flags );
```

The Windows Sockets receive function receives data from a connected socket. If no error occurs, receive function returns the number of bytes received. If the connection has been gracefully closed, the return value is zero, otherwise, a value of SOCKET_ERROR is returned

```
        int recv (SOCKET s, char FAR *Buffer, int lengthBuffer, int flags );
```

In project we use send and receive function in threads which receive data as the data or message is send from other node, and like wise send data or message as the data is received.

```
void CSockDlg::ThRecAS2(CSockDlg *pDlg)
{       SOCKET s=pDlg->AS3;
        char Buffer[70];
        int nret;
        while(1)
        {
                nret = recv(s,Buffer, sizeof(Buffer),0);
                if(nret !=SOCKET_ERROR)
                {
                        if(checkpoint==0)
                        {       nret = pDlg->Connect(1,Buffer);
                                checkpoint =1;
                        }
                        temp0.Format("%d",nret);
                        temp1 = "Received From CS1 : " + temp0;
                pDlg->m_Display.AddString(temp1);

                pDlg->m_Display.AddString(Buffer);
                pDlg->m_Display.AddString(" ");

                Buffer[0]='\0';
            }
            if(pDlg->exitflage = = 1)
                break;
            Sleep(500);
        }
        ExitThread(0);
    }
```

The above function is the receive thread function of the accepting socket. When the connection is accepted then this thread is created and respond on receiving data.

```
void CSockDlg::ThRecCS1(CSockDlg *pDlg)
{
        int checkpoint=0;
        CString temp0;
        CString temp1;
        char Buffer[70];
        char Buff[70];
        int retcode;
        while(1)
        {
            retcode = recv(pDlg->CS1,Buffer, sizeof(Buffer),0);
            if(retcode!=SOCKET_ERROR)
            {   temp0.Format("%d",retcode);
                temp1 = "Byte Received From CS1 : " + temp0;
                pDlg->m_Display.AddString(temp1);
                pDlg->m_Display.AddString(Buffer);
                pDlg->m_Display.AddString(" ");
                if(checkpoint = = 0)
                {       char LBuff[] = "I am on line";
                        if(strcmp(Buffer,LBuff) = = 0)
                        {       temp0 = pDlg->FinalBuf();
                                strcpy(Buff,temp0);
                                retcode = send(pDlg->CS1,Buff,sizeof(Buff),0);
                        }
                        checkpoint = 1;
                }
                Buffer[0]='\0';
        }
        if(pDlg->exitflage  = = 1)
                break;
        Sleep(500);
    }
    ExitThread(0);
}
```

The above function is the receive thread function of connecting socket. When the socket connect function is called the receive thread function of connecting socket is resumed. This thread function is created in initialize function and after creation it is suspended.

4.2.7 Close Socket

The windows socket close function uses close socket for closing the created socket. After closing socket WSAClean up function is called which finally clean all sockets which were started up.

```
{
    if(Acall = = 1)
```

```
        {
                retcode=closesocket(AS);
                if(retcode == SOCKET_ERROR)
                        ErrorExit();
        }
        if(Ccall == 1)
        {
                retcode=closesocket(CS);
                if(retcode = = SOCKET_ERROR)
                        ErrorExit();
        }
        if(Lcall = = 1)
        {
                retcode=closesocket(LS);
                if(retcode == SOCKET_ERROR)
                        ErrorExit();
        }
```

The following functions are used for closing handle of Threads which were created and then exited.

```
        CloseHandle(Accepth);
        CloseHandle(RecASth);
        CloseHandle(RecCSth);

}
```

4.3. Functionality Used in Visual C++ 6 for Signatures

For signatures we use RSA and SHA algorithms. These are fast and efficient secure algorithms. The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n. because this is the only widely accepted approach to public key cryptography, we examine it in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cyptanalytical implications of RSA. The SHA was developed by the National Institute of Standards and Technology. This algorithm takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 160 bit message digest. The input is processed in 512 bit block.

4.3.1 RSA Algorithm

In this project RSA Algorithm is developed by our own hard work, we did not use any MFC classes of Microsoft. The prime is calculated by RSA Algorithm as follows:

```
for (int i = min ; i <= max ; i++)
        {
                for(int j =2;j <= i/2 ; j++)
                {

                        if(i%j = = 0)
                                break;
                        if(j = = i/2)
                        {
                                m_nArray.Add(i);
                        }
                }

        }
```

In the following code we calculate the power of cipher text in decryption or power of plain text in encryption methods.

```
        while ( pow != 1)
        {       a = pow%2;
                if (a = =1)
                        rem +="1";
```

```
            else
                    rem+="0";
            pow /= 2;
            if( pow = = 1)
            {       rem +="1";
            }
    }
    int m = rem.GetLength();
    DWORD c,d;  c=0;  d=1;
    for( int  i = m-1 ; i >= 0 ; i--)
    {       c = c*2;
            d = (d*d) % n;
            if ( rem[i] = = '1')
            {       c = c+1;
                    d = (d * Text) % n;
            }

    }
```

Encryption of plain text is done by the following method.

```
for(int i = 0; i <= str.Length() ; i++)
    {
            Encrypt(chBuf[0]);
            wNum.wWORD=m_cText;
            chBuf[0]=wNum.wStruct.Byte0;
            chBuf[1]=wNum.wStruct.Byte1;
    }
```

The above values are passed to the following function which encrypts the plain text for sending it to other node.

```
bool CRSA::Encrypt(UCHAR a)
{
        unsigned short b=(unsigned short)a;
        m_cText=CalcPower(b,m_eNum,m_nNum);
        return true;
}
```

The decryption is done by the same way but only the difference is in passing the public key in stead of private key in encryption algorithm.

Chapter 5
Testing and Results

5. TESTING AND RESULTS

The overall objective of the testing process is to identify the maximum number of errors in the code with a minimum amount of efforts. Finding an error is thus considered a success rather than failure. On finding an error, efforts are made to correct it.

5.1 Testing the Software

Our software is developed in Visual C++ 6.0 and is divided in three different phases' source, destination and intermediate node. These phases are working under same application. It is difficult to maintain these three different phases under same application because at one time it is working as a source, destination or intermediate node and all of them are communicating with each other. To solve the problem we make the destination and intermediate node run first so, that they should accept the connecting socket. In source node we have to give the route path and the destination. When it is initialized, first it comes on accepting state then it reads the route path and connect to the node whose port number is giving after its own port number. The process is going on till the destination is reached. After acquiring destination it stop connecting other nodes.

Then the communication starts between source and destination. The communication is both secure and non secure. The check box is used to trigger between the secure and non secure communication.

Our Software is basically divided into five components or modules (nodes configuration, communication between nodes, route path finding, secure communication and termination). Initially unit test was performed on every unit. Syntax errors were removed and the validation checks were tested and corrected entirely. For semantic errors, every program unit was tested with the help of test data.

After this all three modules are combined in one complete software as the integration completed. Again the Syntax and semantic errors were checked and removed. After the completion of individual testing of all the modules, the modules were integrated and testing phases were applied, then errors were checked and removed.

In the next step, the system test was applied and all the errors were checked and removed. When all these errors were detected and removed, the final test was applied to the system, to check whether these changes are affecting the remaining parts of the program unit or not. The errors occurred were removed. After this checking, all the functionality was checked and found correct.

To Test our software the two tests have been chosen to verify that our software is working good or not and named:

Test 1.

Test 2.

5.1.1 Test 1

First test is about, that nodes are communicating with each other or not. By pressing the connect button to connect other node and accept button to accept the connecting socket. If they are connected with each other then they can transmitte data to one an other or not. By pressing send button to send message and receive button to receive message or buffer.

Then we tested that if thread are introduced in software then they can accept the connecting socket automatically i.'e. with out pressing accept button.

If threads are introduced for the receiving buffer of connected socket. Then they can receive buffer or message of the other connted node, when message is send from connected (i.e. client) socket.

If the node accept the connecting socket automatically by the help of thread, when connect button is pressed. Then it is tested that all node|(i.e. upto 6) can connect each other by pressing only one initialize button. For this we used three threads (accept thread, receive accept thread message and receive connect thread message), which help us to connect all nodes with each other. In this method we use the defined route path in the source node, which may also be called shortest path.

In this test we had tested, that if destination address is define before other port number, then is it possible to stop connecting other nodes when destination is reached, or if the destination is the sixth node then connect all the six nodes. And also tested that send and receive messages is between the source and destination through the intermediate nodes.

5.1.2 Test 2

In this test, the secure communication between the source and destination. The security is done by RSA algorithm. This algorithm is the internet standard algorithm which is used in diffenert types of communications forexample, in militry communication. This algorithm is first developed and tested for single plain text, means that to encrypte and decrypt the plain text.

This algorithm is then used in this main software. When the RSA algorithm check box is checked then the required message is encrypted and send to destination.

According to the research work when the secure communication is needed then the public key of RSA algorithm is embeded in the request packet (which is for the destination to receive the encryted messages). This packet also includes the encrypted data of number of HopCounts and the port numbers through which the data is received (i.e. source port number).

Communication is done by encrypting the required messages with appended public key of RSA algorithm. After transmiting the public key to destination then the key are not shifted (for security purposes).

5.2 Results

Following are the results obtained after applying the test set to the software "Secure Ad hoc On-demand Distance Vector Routing Protocol".

5.2.1 Results of Research work

In Ad hoc networks there must be two security systems, one to protect the data transmission and other to make the routing path secure. Here we are concerned with the secure route communication not with the protection of data transmission. First it should be described that what security measures are needed for AODV routing protocol. It is needed to have the authentic route. It is also desired to avoid tampering attacks like reusing of packet; this can be solved by using the sequence number in AODV. In the Ad hoc network it is impossible to prevent denial of service attacks.

AODV protocol is under consideration of IETF, therefore it is not standardized for the internet or intranet communication . We made changes in the routing packet header of AODV for secure route communication. In the request and reply packets we introduced the signatures.

Following are the results of tests performed. In this test we have made the a routing path secure. In test set 1 we defined the route path as shown in following figure 5.1.

In this training set the source connects with the destination by defining the route path. In first transmission the request packet is transmitted which connects to the neighboring node and requset for the destination. The intermediate node transfers the packet to next node till the destination is reached. When destination is reach, it stop connecting to other nodes and reply to the source node for communication.

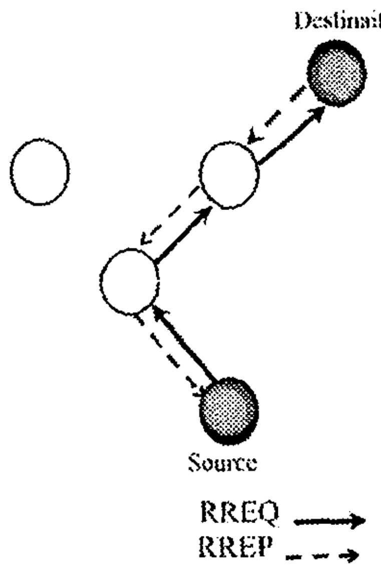


Figure 5.1 Defined route path between Source and Destination.

The destination will informe the source node that destination is reached. When the secure route is needed, source will encode the source address and destination address. The public key is embeded in the packet header and transferred to the destination. The destination decodes source address. This conforms that the data trasmitted from the authentic source. The public key which is with the desitnation makes shoure that data received from the authentic destination. This is shown in following figure 5.2.

This route path is responsible for the communication between the source and destinaiton. The source node sends the packet with the sequence number 1 and destination give acknowledgement of receiving packet and also request for new packet. The route table is maintained between the source, destination and intermediate routers. This route table contains the hop count, source port no, destination port no, sequence number, acknowledgement number, type of packet. These entries are maintained in all router application. When the secure route is requested then the hop counts are not mention in the route table because it is secure in the data packet.

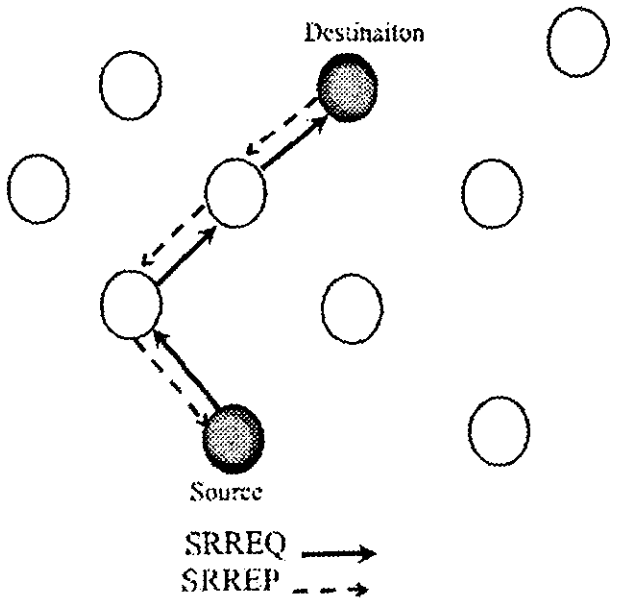


Figure 5.2. Secure route path between source and destination.

After defining route path the source node encrypts the source address, destination address and hop count by RSA algorithm, then embed the public key and encrypted data with in the packet header. This security is done so, that any other router may not easily configure the source address, destination address and the hop counts to the destination node. The secure request packet (SREQ) is shown in figure 5.3. and (SREP) packet header in figure 5.4.

Type 8bits	J 1bit	R 1bit	Reserved 14 bits	Offset Hop Count 8 bits	Broad cast ID 32 bits
Destination IP Address 32 bits	Destination Sequence No. 32 bits	Get Source IP Address 32 bits	Source Sequence No. 32 bits		
Signatures (Encrypted: Source Add and Hop Count) 80 bits	Public Key 8 bits				

Figure 5.3. Secure Request Packet.

Type 8bits	R 1bit	Reserved 10 bits	Pfx Length 5 bits	Offset HopCount 8 bits	Destination IP Address 32 bits
Destination Sequence No. 32 bits	Signature (Encrypted: Source Add and HopCount) 80 bits	Source Sequence No. 32 bits	Life Time		

Figure 5.4. SREP packet header.

5.2.2 Results of software developed

After performing all tests, we have developed the software without any fault remaining. The user interface of software is shown in figure 5.5.

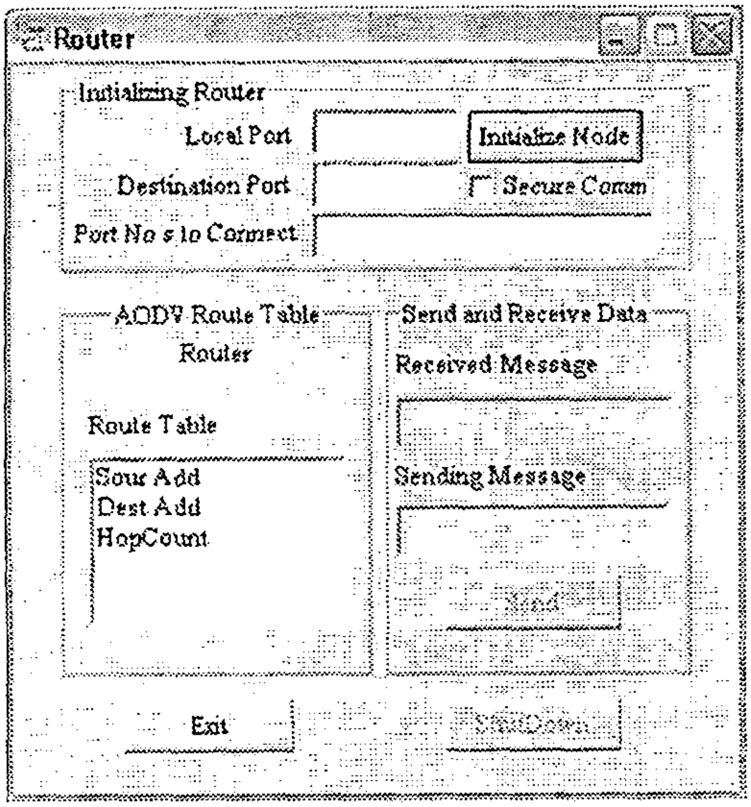
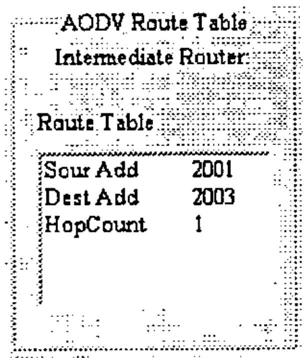


Figure 5.5. Main window.

Every part is separated for user understandings. The group box of initializing router on given port number which is the address of that application. If both destination and port number to connect edit boxes are filled then that application becomes the source node. When the source node is defined, then the send messages edit box and send button is enabled for communication. The receive messages edit box display every message pass through it or reached to it. As like the destination node's send message edit box and button is enabled. The shut down button is used for resetting the application, when this button is pressed then all the global variables are initialized. The route table is updated when any message is passed by application. The route table is set after completing the route path and destination acknowledgement.

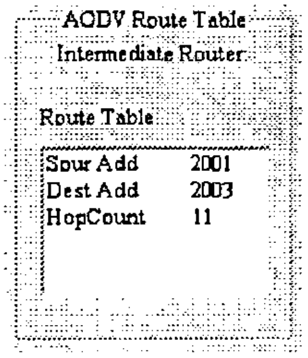
In non secure communication the route table is maintained with actual hop counts, and original source address. The intermediate node knows the whole route path of source and destination. The route table of the intermediate node is shown in figure 5.6, here the address of source is 2001, destination address is 2003 and intermediate address is 2002.



Sour Add	2001
Dest Add	2003
HopCount	1

Figure 5.6. Route Table of non secure path

In secure communication the route table of intermediate node is different for non secure communication. Because source node passes the offset hop count to intermediate nodes, so that hacker can be able to get to source. Source sends the offset hop count 10 which is incremented by 1 as the packet reaches to other node. The route table is shown is figure 5.7.



Sour Add	2001
Dest Add	2003
HopCount	11

Figure 5.7. Route Table of secure path

The communication between the source and destination is to be secure or non secure. For evaluating performance software we use graph to make distinguish between the secure and non secure route path. We use two types of graphs, one describe the performance of number of packet transferring between source and destination at speed and other describes the attaching rate on the packets, here the number of nodes used is six. In figure 5.8 the dark line shows the number of route request packets transmission and dotted line shows the number of route reply packets transmission. In this graph the two lines are constantly moving up which shows with the passage of time the speed increases as the number of packets are increasing.

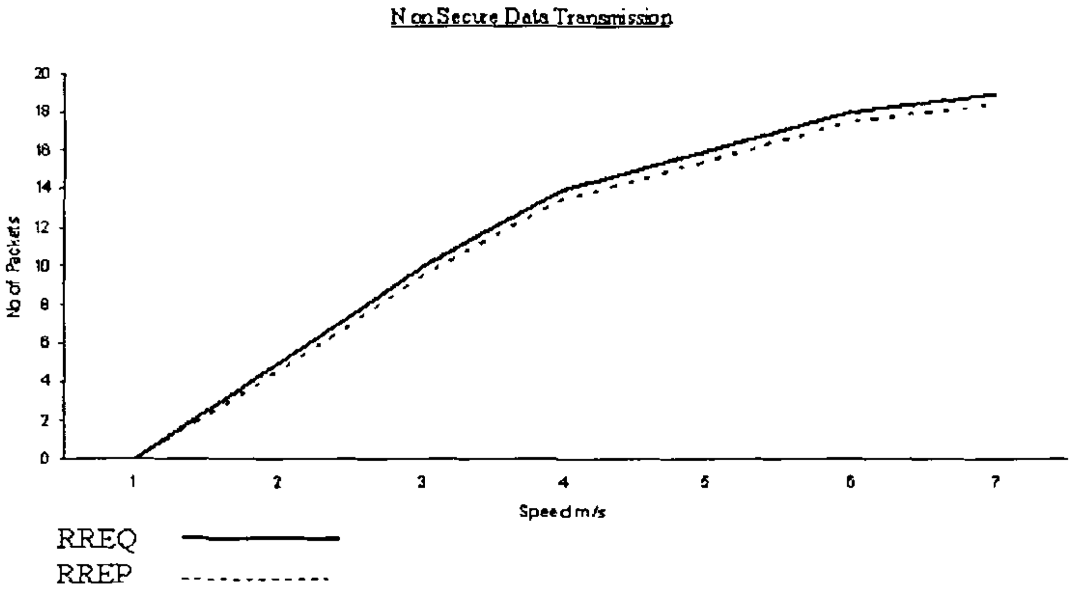
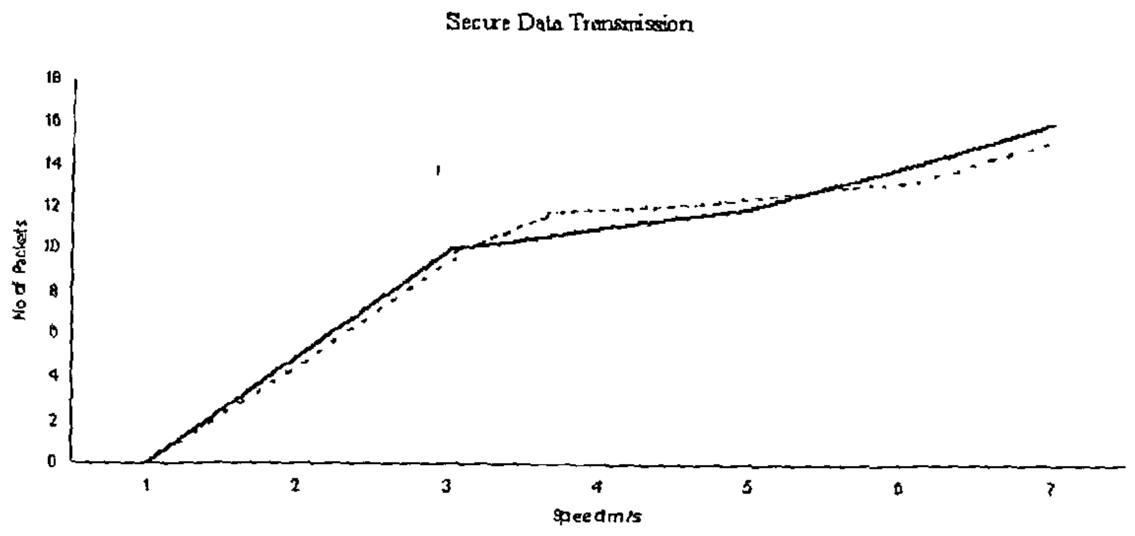


Figure 5.8: Non Secure Data Transmission Graph.

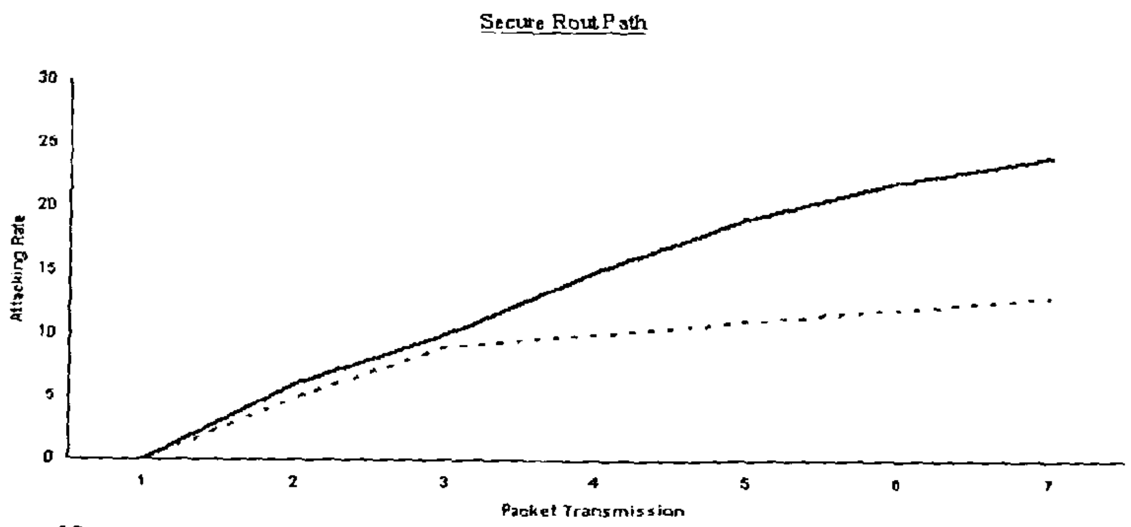
In next graph, figure 5.9, we have shown the performance of Secure Data Transmission between source and destination. In this graph we observe a turn in both lines. The turn in dark line shows the transmission of the signature and public key field in the header, after this transmission the line again moves up shows the speeding up in transmission. There is a little twist in the dotted line which shows that the reply packet only transfers the signature field.



SRREQ ――――――
SRREP ------------

Figure 5.9: Secure Data Transmission Graph.

In the following graph we have shown the statistics of attacking rate on packets transmission in the secure and non secure route path. As shown in Figure 5.10 the attacking rate on non secure route in very high while on the secure route path the attacking rate is lowered. In graph the lines moves at some time in same sequence that means that secure route first the route path is formation is non secure then the authenticity is given to source and destination of data transmission.



Non secure ――――――
Secure ------------

Figure 5.10: Secure path graph.

5.3 Conclusion

This project is a research project and we have extended the work done by the Charles E. Perkins and Elizabeth M.Royer in 1991 on Ad hoc On-demand Distance Vector Routing Protocol, and also get some idea of secure route path from Manel Guerreo Zapata, work done on Secure Ad hoc On-demand Distance Vector Routing Protocol. We have introduced an idea about the secure rout path, in which the route path is secure, and hacker can not able to get through the source or became a destination; for getting any information.

In the previous system that route path is not secure and hacker can easily inter in the route path and get any information or able to change address to source or destination for illegal use. But now in secure route path, hacker is not able to get the source and destination addresses or may change any information about source and destination. By these changes in the packet header the route path is secure and difficult to hack.

The only danger is that the hacker has already knows the destination address and is present in between the route path of source and destination. And also know the way to communicate with the source. Other wise it is very difficult to him to get to source or destination.

The work is done only on the unicast route path, not on the multicast route path formation. In future the research work is also being done on the multicast routing. This idea is also not fully developed and not tested in real environment, only simulation are made to test it, so, enhancements are possible in this idea. Ad hoc networks is a vast field of research work because very little work is done in this field and those algorithms which are working in practical field are not quiet accurate according for network traffic rules. This protocol works on the demand of source to connect with destination, so the path is not defined for communication.

Appendix A
User Manual

Overview

This is a program developed for security purpose and can be used in buildings, offices and sensitive areas in order to avoid the unauthorized person's access to particular area. This program uses an easy to use interface and accurate.

This software is developed in Visual C++. This software uses the TCP/IP sockets for communication between different applications. This software shows the route table maintained every time the transfer of data between source and destination.

This Program is developed by: Software Development Team

Main Window (Application)

This is what the main program window looks like:

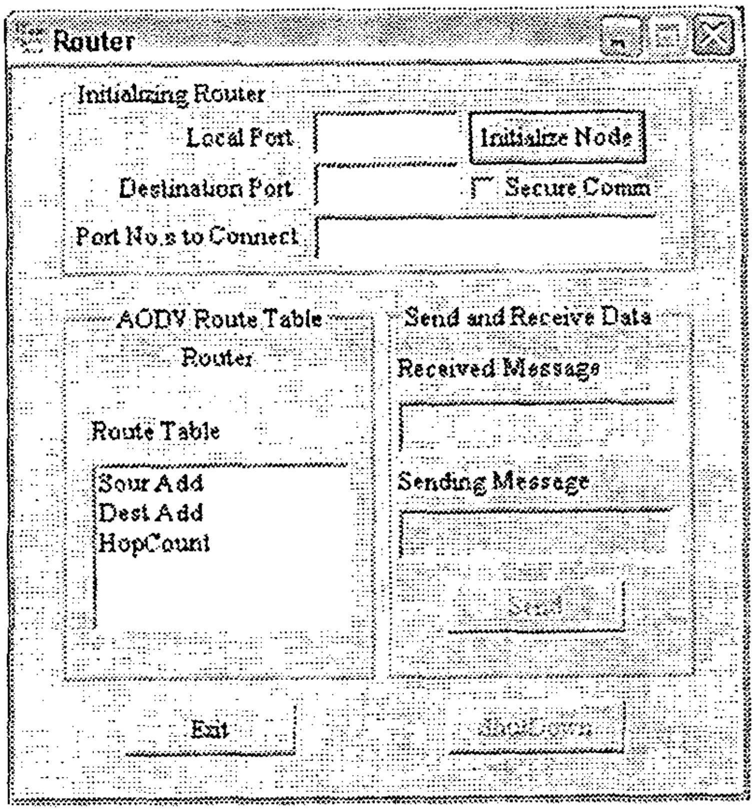


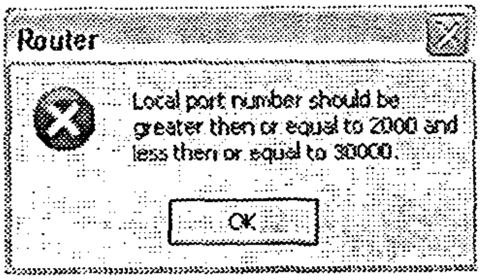
Figure 1. Main Window

Initializing Router

This Group Box is use to initialize the application. This group box gets the local port to connect with other application, when the port number is entered and initialize node button is pressed, it will startup the socket, create the socket, bind the socket with socket family, and listen the connecting socket. After entering in listening state the thread accept socket is created and wait for other socket to connect. As the socket is connected the thread exits and closes the listening socket. In this application, the shortest path is defined first because of the use of TCP/IP sockets, which are connection oriented.

Local Port

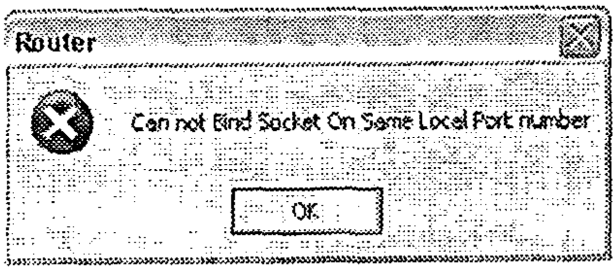
This edit box gets the local port number of local host, and creates the socket for connection or acceptance. The port number range from 2000 to 30,000, if it is ignored then error dialog box shown in figure 2 is displayed. Because the lower number then 2000 is reserved for windows, and above then 30,000 is not allowed by the integer value. This edit box gets only digits, not any other entries. The short cut key for this edit box is ALT+L.



Local port number should be greater then or equal to 2000 and less then or equal to 30000.

Figure 2. Error report, exceeding port number from range.

The errors displayed for destination and port no.s to connect edit box, but change of "Local port" to "Destination port" and to "Connecting port". When the same local port number is assigned to other node then it socket can not be bind on same port. This is handled by the following message shown in figure 3.

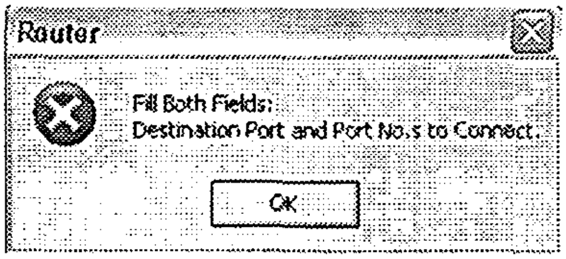


Can not Bind Socket On Same Local Port number

Figure 3. Cannot bind on same port number.

Destination Port

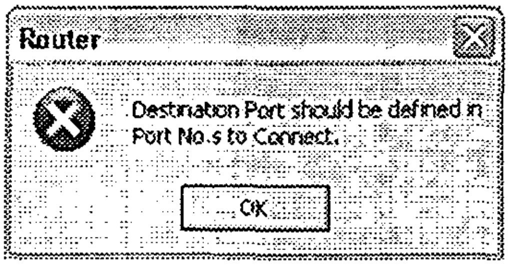
This edit box gets the destination port number of local host, to which the source will communicate. The number after the destination port number in Port No. to Connect will not make connecting because when destination is reached other port numbers are discarded. This edit box gets only digits, not any other entries. The short cut key for this edit box is ALT+D. If we enter the port numbers for connection and not the destination or enter the destination port number not the connection port number the error is reported as shown in figure 4.



Fill Both Fields:
Destination Port and Port No.s to Connect.

Figure 4. Error report of missing entry in edit box.

In the Port No.s to Connect, if destination is not entered it will report for the following error (as shown in figure 5.).



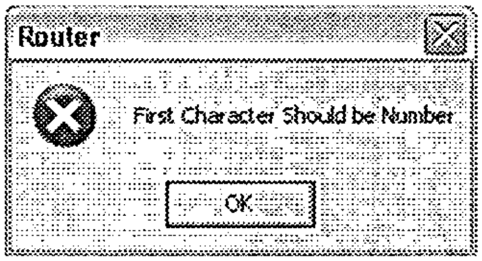
Destination Port should be defined in Port No.s to Connect.

Figure 5. Destination missing in connecting port numbers.

Port No.s to Connect

This edit box gets the port numbers, separated by comma, for making connection till the destination is reached. The entries of this edit box connect the socket with the next enter of router local port number with the accepting socket. This edit box gets only numbers and commas.

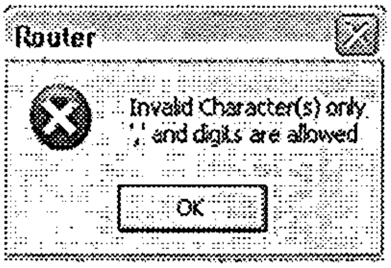
In this edit box first entry should be number, other wise it will generates the following error.


Router

First Character Should be Number

Figure 6. Error report of invalid first character.

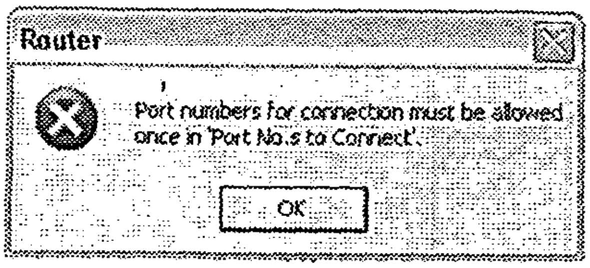
If any single entry is other then comma and digit. Then it is handled as shown in figure 7.


Router

Invalid Character(s) only ',' and digits are allowed

Figure 7. Error report of invalid entry.

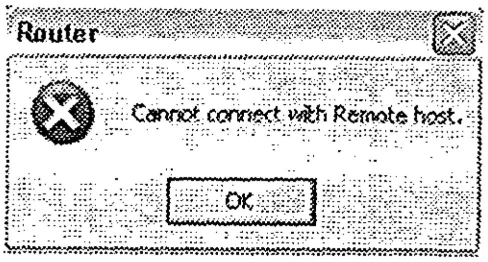
If there is repetition of port number in the edit box then this exception is displayed as shown is figure 8. This is handled because the connection with one node is made once, if the making connection is repeated then the connection will be failed and the application is aborted.


Router

Port numbers for connection must be allowed once in 'Port No.s to Connect'.

Figure 8. Error report of repetition of number.

If a port number is entered, which is not initialized or initialized and then shutdown or not present online. Then the connection error is reported as shown in figure 9 and stop making connection with other nodes, and shutdown (i.e. reset) it's self.


Router

Cannot connect with Remote host.

Figure 9. Error report of connecting failed.

Initialize Node

When this button is pressed, it will initialize the node by starting the socket, creating socket, binding the socket on given port number, listen the connecting socket and created thread of accepting the connection of other socket (when it is intermediate node or destination node). If the node or router is intermediated or destination then the dialog box appears as shown in figure 10.

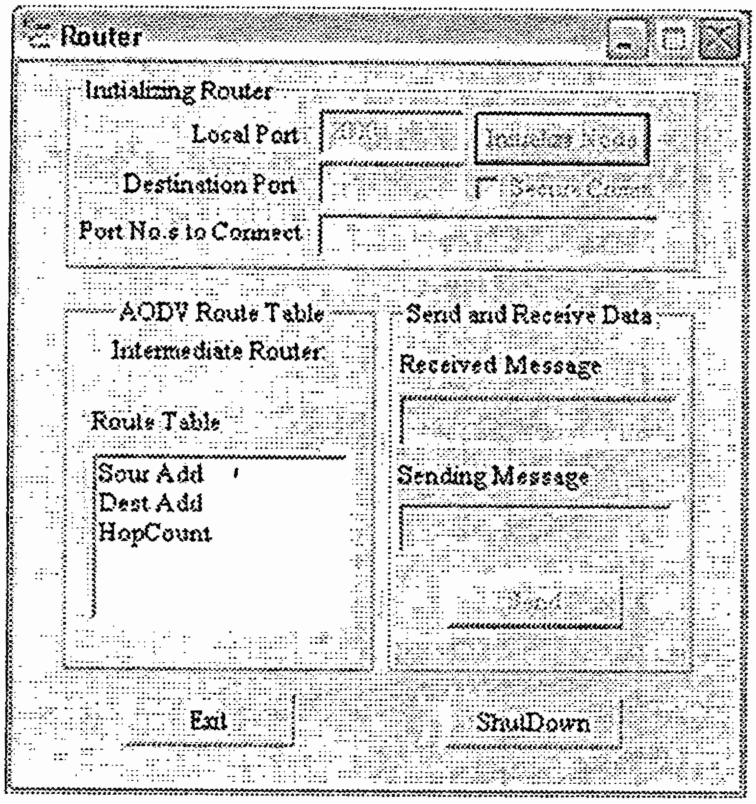
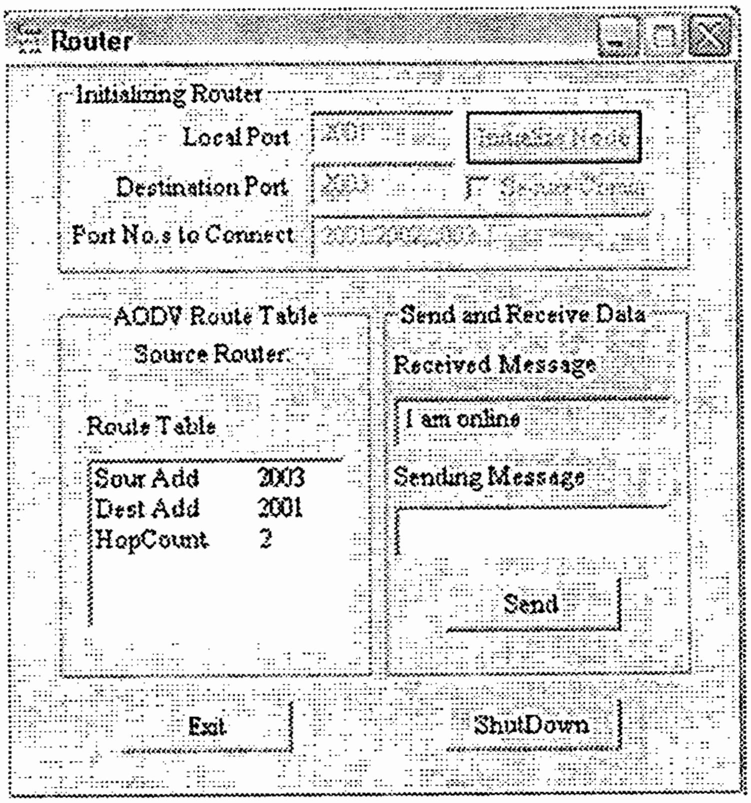


Figure 10. Intermediate node initialized.

This node or router is now in the accept state. The information display in the group box of AODV Route Table shows that it is the destination, intermediate or source node. The route table is update when the communication between source and destination is started. If the source node or router is initialized, it is displayed as shown in figure 11.


Router

Initialising Router

Local Port [illegible]

Destination Port [illegible]

Port No.s to Connect [illegible]

AODV Route Table

Source Router:

Route Table

```
Sour Add     2003
Dest Add     2001
HopCount     2
```

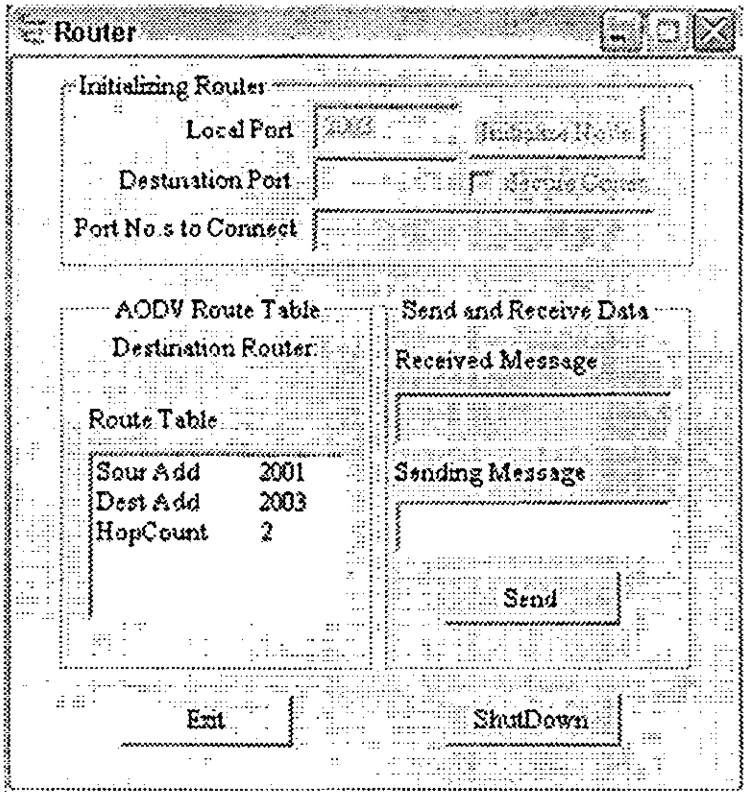
Send and Receive Data

Received Message: I am online

Sending Message

Figure 11. Source node initialized.

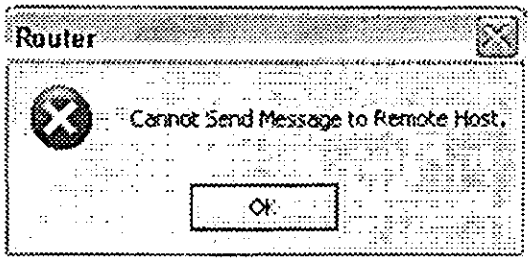
When this node is initialized, it will make connection with following node. That node will send message that "I am online" after that it will pass the connecting nodes port number for further connection, till the destination is reached. When destination is reached the window of destination is shown in figure 12.



Sour Add 2001
Dest Add 2003
HopCount 2

Figure 12. Destination node reached.

As like the source the destination send button and sending message edit box is enable for sending message to source. If the intermediate node is shut down and the message is send to source the message will be lost, but here it is controlled by displaying the error message shown in figure 13.



Cannot Send Message to Remote Host.

Figure 13. Error report of sending failed.

Secure Comm

This check box is used for secure or non secure, communication between source and destination. If the check box is checked , as shown in figure 14, then the communication between source and destination is secure i.e. the route path is secure and intermediate node does not know the source node from where the communication is started, it just send the data to destination.

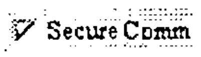


Figure 14. Secure Communication check.

If the check box is check at intermediate node and initialize node button is pressed it just uncheck the box and continue the process. The check box is checked at source node only.

Send and Receive Data

This Group box is use to send the data from source and destination, or receive data from neighboring node and send it forward. Only source and destination can send data through intermediate nodes. The send edit box and button are enabled only in these two nodes, while the intermediate nodes send edit box and button is disable (i.e. they can not send data to other nodes). The receive edit box only shows the data which is passed to intermediate nodes or received by destination node.

This group box is use to show that if the terminal, next to source or destination node, has send data (it may be header information or data requested) to destination.

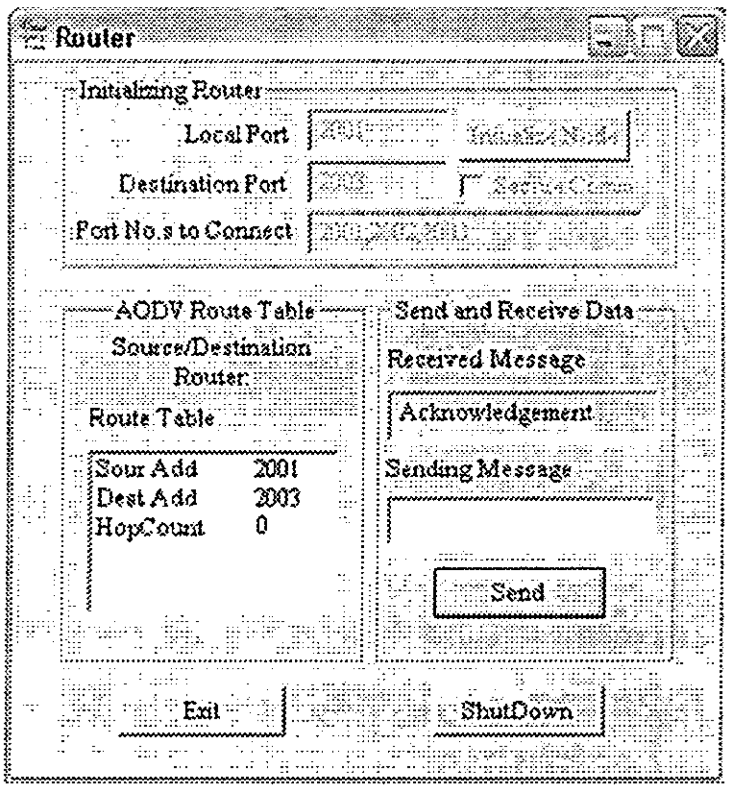
AODV Route Table

This Group box as shown in figure 1, is use to show the updating route table every time the packet is passed to it and also it shows the router or node status (i.e. it intermediate, source or destination) every time, when it online.

When the route is secured requested the source and destination group box name is changed from "AODV Route Table" to "SAODV Route Table". This shows that the source and destination only know the route is secure while the intermediate nodes do not know it.

Non Secure Communication

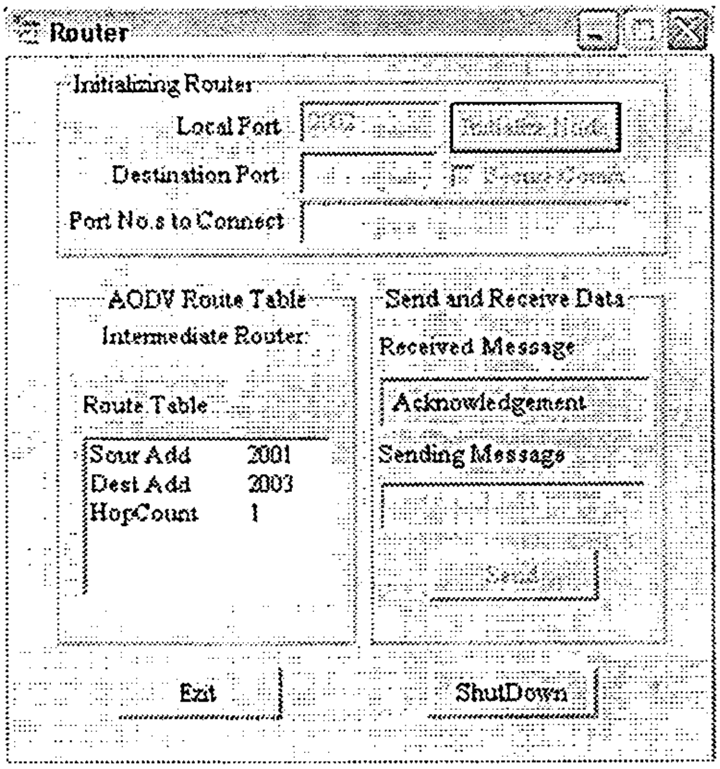
In this type of communication all node know the actual hop counts and the original source node and destination node. They communicate with each other on the basis of actual entries in route table. Some snap shorts are shown in figure 15, 16, 17.



```
Sour Add    2001
Dest Add    2003
HopCount    0
```

Received Message: Acknowledgement

Figure 15. Non Secure Source node.


Router

Initializing Router

Local Port

Destination Port

Port No.s to Connect

AODV Route Table

Intermediate Router:

Route Table

```
Sour Add    2001
Dest Add    2003
HopCount    1
```

Send and Receive Data

Received Message

Acknowledgement

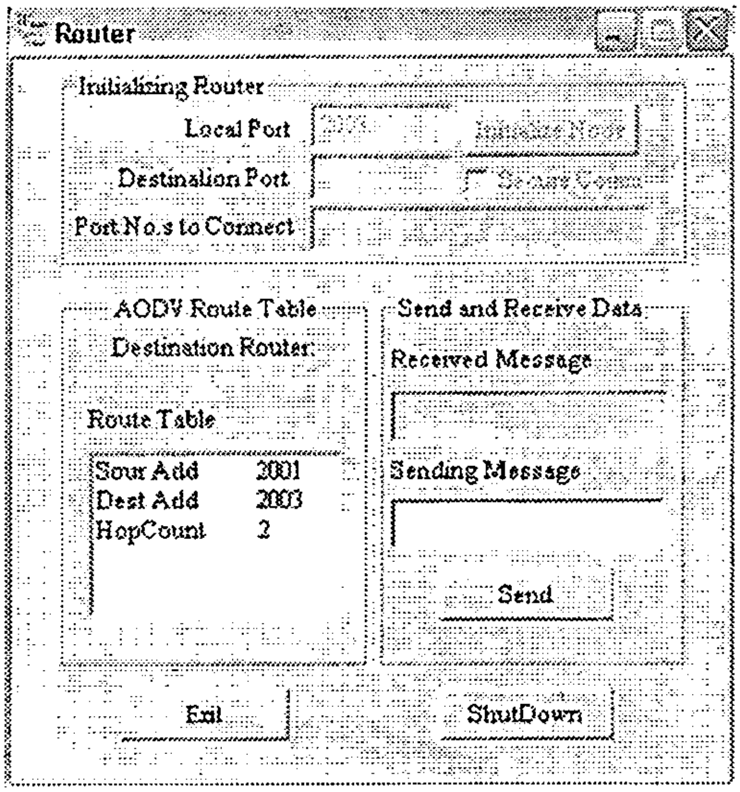
Sending Message

Send

Exit

ShutDown

Figure 16. Non Secure Intermediate node.


Router

Initializing Router

Local Port

Destination Port

Port No.s to Connect

AODV Route Table

Destination Router:

Route Table

```
Sour Add    2001
Dest Add    2003
HopCount    2
```

Send and Receive Data

Received Message

Sending Message

Send

Exit

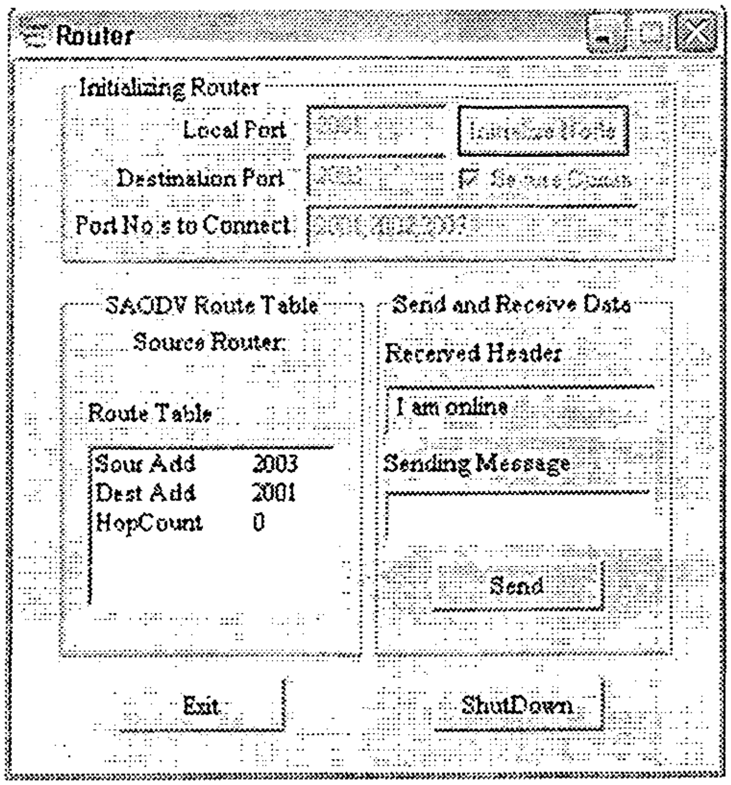
ShutDown

Figure 17. Non Secure Destination node.

When the destination is reached, "Acknowledge" is send to source, which means that the destination is reached and the route is established for communication. After receiving the "Acknowledge" message the route tables are updated with the current entries of source and destination.

Secure Communication

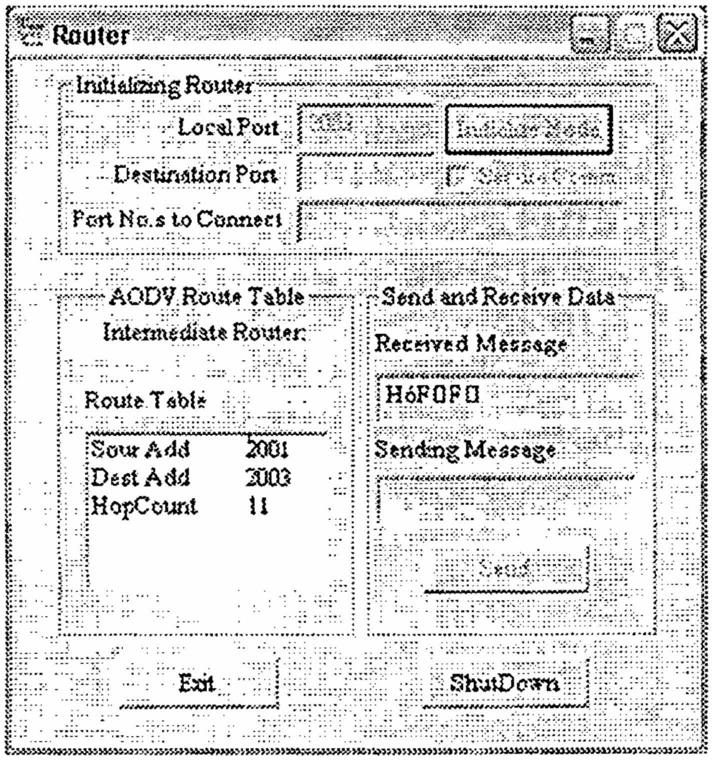
In this type of communication the intermediate node does not know the actual source address and actual hop counts. But it shows the source address of previous node and hop counts are the offset hop count which is given by the original source and source address field is filled by the previous node address. The fields of route table are field as shown is following figure 18, 19, and 20.



Sour Add	2003
Dest Add	2001
HopCount	0

I am online

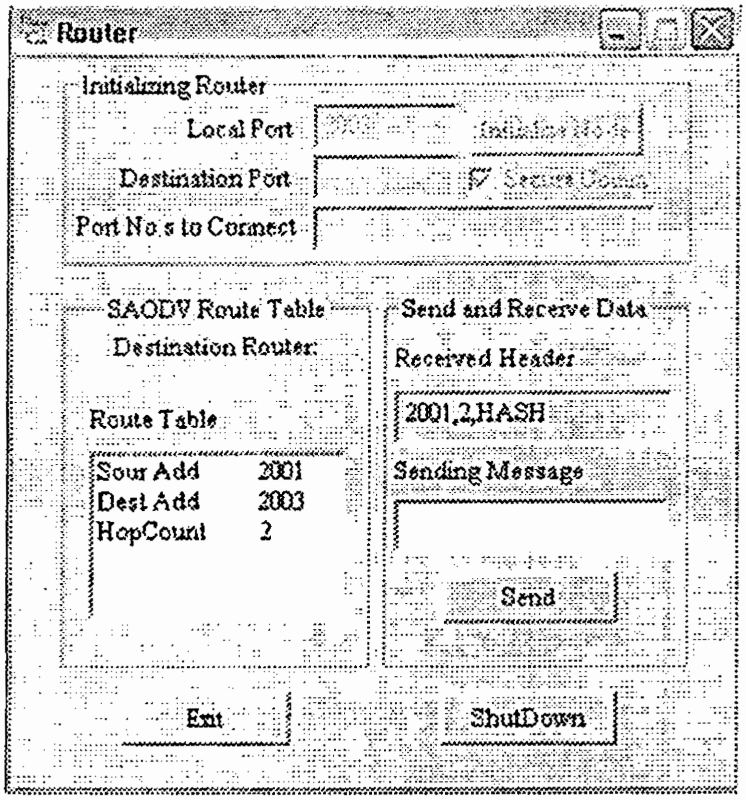
Figure 18. Secure Source node.



```
Sour Add     2001
Dest Add     2003
HopCount     11
```

Received Message: HóFOFO

Figure 19. Secure Intermediate node.



```
Sour Add     2001
Dest Add     2003
HopCount     2
```

Received Header: 2001,2,HASH

Figure 20. Secure Destination node.

When the "Acknowledgement" from destination is reached to source then it will send message to destination that the ready to receive secure data. The destination will acknowledge the source for transmission of secure data, and then source will encrypt its own address and offset hop count, also append the public key of encrypted data (this public key is shifted to destination only once in communication) and send it to the intermediate node. The offset hop count is also set as actual hop count for intermediate node. The intermediate node will transmit the packet header to next node till the destination is reached. The destination know how to decrypt that portion, it gets the key and store it for this communication. If the link breaks then the key is changed and again transmit to the destination. As like the destination will shift its own public key after receiving the source key but that key is also encrypted in with other fields. By this way the transmission is secure and hacker can not hack the source or destination for getting the benefits from them.

Software Development Team

Software Engineer:

Muhammad Asfand-e-yar (yar_n_yar@yahoo.com)

Supervisors:

Muhammad Sher (msher313@yahoo.com)

Contact Me

Muhammad Asfand-e-yar

Email: yar_n_yar@yahoo.com

Mobile: 0300-5921740.

Postal Address:

House Number 502, Street 60, Sector D-2,
Phase 1, Hayat Abad Town,
Peshawar.

Bibliography and References

Bibliography and References

Research Papers

1. Route Life time Assessment Based Routing Protocol for Mobile Ad Hoc Networks: by Ashich Ahuja, Jatinder Pal Singh, Rajeev Shorey and Sulabh Agarwal.

2. An On Demand Secure Routing Protocol Resilient to Byzantine Failures: by Baruch Awerbuch, Cristina Nita Rotaru, David Holmer and Hervert Rubens.

3. Performance Comparision of Two On demand Routing Protocols for Ad Hoc Networks: by Charles E. Perkins, Elizabeth M.Royer and Samir R.Das.

4. AODVSTAT: Intrusion Detection in AODV: by Elizabeth, Giovanni Vigna, Richard A.Kemmerer and Sumit Gwalani.

5. On-demand Multipath Distance Vector Routing in Ad Hoc Networks: by Mahesh K.Marina and Samir R.Das.

6. An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks: by Rajendra V.Boppana and Satyadeva P Kondum.

7. Manel Guerreo Zapata, "Secure Ad hoc On-demand Distance Vector Routing", IETF internet Draft draft-guerrero-manet-saodv-00.txt (work in progress), 2001.

Books

1. Ad Hoc Networking by Charles E.Perkins and Elizabeth M.Royer Pages 173-219.

2. Software Engineering A Practitioner's Approach (Fourth Edition) by Roger S. Pressman 1992, McGraw-Hill Companies Inc.

3. Network and Internetwork Security by William Stallings.

Research Paper

Secure Route Path Formation in Ad hoc On-demand Distance Vector Routing Protocol

Muhammad Asfand-e-yar and Muhammad Sher
Department of Computer Science
International Islamic University, Islamabad

Abstract: This study describes the secure route path formation in the Ad hoc On-demand Distance Vector routing protocol. For secure route path formation in Ad hoc On-demand Distance Vector Routing Protocol RSA for encryption is used. The Secure Ad hoc On-demand Distance Vector is an extension in Ad hoc On-demand Distance Vector routing protocol that is used to protect the route path from integrity and authentication attacks. Ad hoc network maintains a routing table giving distance from itself to all possible destinations. In Ad hoc On-demand Distance Vector routing protocol the route discovery is done by route request and reply packets. After establishing a route the communication between source and destination takes place. Ad hoc On-demand Distance Vector routing protocol solves the loop formation. It works on wired as well wireless networks.

Key words: Ad hoc on-demand distance vector algorithm, route discovery, secure route, security attacks.

Introduction Ad hoc wireless networks are self organizing multi hop wireless networks where all hosts take part in the process of forwarding packets. Ad hoc networks can easily be developed since they do not require any fixed infrastructure, such as base stations or routers (B.Awerbuch, 1998). Ad hoc network is a collection of mobile nodes that communicate with each other with out any centralized administration. Ad hoc networks use different types of protocols like Distance Vector, Distance Sequence Distance Vector, Dynamic Source Routing, On-demand Distance Vector Routing Protocols.

In DV routing, each router maintains a routing table giving the distance from itself to all possible destinations. In route path formation loops are formed between source and destination. RIP (Routing Information Protocol) handles these routing loops. DSDV solves the looping problem in DV routing by attaching Sequence number to routing entries. Node increments its current sequence number and includes it in the updates originated at that node. This method in DSDV increase routing overhead (Charles E.Perkins, 2000), (R.V.Boppana, 2001). Ad hoc On-demand Distance Vector routing protocol (AODV) is introduced which solves the looping and routing overhead problems.

The AODV routing protocol provides quick and efficient route establishment between nodes desiring communication. AODV is designed specifically for ad hoc wireless networks; it provides communication between mobile nodes with minimal control overhead (Charles E.Perkins, 1999). When a path is formed by the route request and route reply packets in AODV routing protocol, then the communication starts between source and destination. If any hacker is sitting in the network and wants to steal data from source and destination then it can easily change the route path, by adding its own address in the route and increasing the hop count. By this hacker can easily get information from source and destination. To get rid of this attack we make the route authentic and secure. For authentication and security we use the Revist Shamir and Adleman (RSA) algorithm which encrypt the source and destination address and make difficult for the hacker to know the source or destination node.

Ad hoc On-demand Distance Vector Algorithm In AODV algorithm the nodes do not have to discover and maintain a route until the two nodes, need to communicate and the former node is offering its service as an intermediate forwarding station. The primary objectives of algorithm are:

- To broad cast discovery packets only when necessary.
- To distinguish between local connectivity management and general topology maintenance.
- To disseminate information about changes in local connectivity to those neighboring mobile nodes that is likely to need the information.

Instead of source routing, AODV relies on dynamically establishing route table entries at intermediate nodes (Charles E.Perkins, 1999).

Route Discovery In AODV the route discovery is purely on demand and follows a route request/reply discovery cycle. The route requests are sent using the Route Request message (RREQ) and the reply is sent back by the Route Reply message (RREP). When a node wants to make a route to destination then it sends packet. Before sending a packet it checks the Route table whether it has a current route to that node. If so, it forwards the packet else it initiates a route discovery process. This packet contains the following instructions, (Fig.1).

Type 8bits	J 1bit	R 1bit	Reserved 14 bits	Hop Count 8 bits	Broad cast ID 32 bits
Destination IP Address 32 bits	Destination Sequence No 32 bits	Source IP Address 32 bits	Source Sequence No 32 bits		

Fig. 1: Route request packet

After receiving the packet then it sets the reverse route entry form the source node in its route table. The route reply packet contains the following information, (Fig. 2).

To response the RREQ, the node must have the unexpired entry for the destination in its route table. Also the sequence number associated with that destination must be latest that indicated in the RREQ packet. This prevents the formation of routing loops.

Type 8bits	R 1bit	Reserved 10 bits	Pfx Length 5 bits	Hop Count 8 bits	Destination IP Address 32 bits
Destination Sequence No 32 bits	Source IP Address 32 bits	Source Sequence No. 32 bits	Life Time 32 bits		

Fig. 2: Route reply packet

It must be also sure that the route returned is never old enough that points to a previous intermediate node. If the node is able to satisfy these two conditions then it responds by unicasting a route reply RREP back to the source, otherwise the RREQ hop count is incremented and broadcast the packet to its neighbor (Charles E.Perkins, 1999).

Loop Freedom AODV play a key role in ensuring loop freedom. Every node maintains a single increase sequence number for itself, and also maintains highest sequence number for each destination in routing table (i.e. "destination sequence number"). This increase of sequence number along a valid route prevents routing loops.

```
**If** (($seqnum^d_i < seqnum^d_j$) or (($seqnum^d_i = seqnum^d_j$) and ($hopcount^d_i > hopcount^d_j$))
**then**
        $seqnum^d_i := seqnum^d_j$;
        $hopcount^d_i := hopcount^d_j + 1$;
        $nexthop^d_i := j$;
**endif.**
```

Fig. 3: AODV route update rule

A node can receive a routing update via a RREQ or RREP packet either forming or updating a reverse or forward path. The update rule (Fig.3) is invoked upon receiving a route request or reply packet. It is easy to see why loops cannot be formed if this rule is followed. Consider the tuple ($-seqnum^d_i$, $hopcount^d_i$) where $seqnum^d_i$ represent the sequence number at node i for the destination d. Similarly, $hopcount^d_i$ represents the hopcount to the destination d from node i. For any two successive nodes i and j on a valid path to the destination, j being the next hop from i to d, the route update rule in (Fig. 3) on forces that

$$(-seqnum^d_i, hopcount^d_i) > (-seqnum^d_j, hopcount^d_j)$$

Where the comparison is in the lexicographic sense. Thus, the tuples ($-seqnum^d_i$, $hopcount^d_i$) *along any valid* route are in a lexicographic total order, which in turn implies loop freedom (M.K.Marina, 2001).

Secure Routing In Ad hoc networks there must be two security systems, one to protect the data transmission and other to make the routing secure (M.G.Zapata, 2001). Here we are concerned with the secure route communication not with the protection of data transmission. First it should be described that what security measures are needed for AODV routing protocol. It is needed to have the authentic route. It is also desired to avoid tampering attacks like reusing of packet; this can be solved by using the sequence number in AODV. In the Ad hoc network it is *impossible* to prevent denial of service attacks (E.G.Vigna, 2000).

AODV protocol is under consideration of Internet Engineering Task Force (IETF), therefore it is not standardized for the internet or intranet communication (M.K.Marina, 2001). We made changes in the routing packet header of AODV for secure route communication. In the request and reply packets we introduced the authentication and encryption algorithm.

In SAODV protocol the route is *made on-demand of source node.* Therefore, the RREQ packet is send to the neighboring node with Source Address, Destination Address and with other instruction to the neighboring node, (Fig. 1). Neighboring node transmits the RREQ packet to other nodes, till the destination is reached. Then the destination replies to source by route reply packet, (Fig. 2). The source node discards all other entries of neighboring node when the destination reply is reached. After defining route path the source node encrypts the source address, *destination address* and hop count by RSA algorithm (W.Stalling, 1993), and then embed the public key and encrypted data with in the packet header. This security is done so, that any other node may not easily configure the source address, destination address and the hop counts to the destination node.

The secure request packet (SREQ) (Fig. 4):

Type 8bits	J 1bit	R 1bit	Reserved 14 bits	Offset Hop Count 8 bits	Broad cast ID 32 bits
Destination IP Address 32 bits	Destination Sequence No 32 bits	Get Source IP Address 32 bits	Source Sequence No 32 bits		
Signatures (Encrypted Source Add and Hop Count) 80 bits	Public Key 8 bits				

Fig. 4: Secure request packet

Then source node transmit SREQ packet to destination. When SREQ packet is received by the destination it performs the check sum of packet and then decrypts the packet by public key sent by source node. On second time when the source packet needs to transmit SREQ packet it done not need to attach the public key of RSA algorithm, (Fig. 5). This will make less load on routing traffic. When the destination receives the SREQ packet it replies to source by the following secure reply packet (SREP), (Fig. 6).

Type 8bits	J 1bit	R 1bit	Reserved 14 bits	Offset Hop Count 8 bits	Broad cast ID 32 bits
Destination IP Address 32 bits	Destination Sequence No. 32 bits	Get Source IP Address 32 bits	Source Sequence No. 32 bits		
Signatures (Encrypted: Source Add and Hop Count) 80 bits					

Fig. 5: Secure request packet 2

The security can be increased by increasing the number of bits in RSA algorithm. This will make more difficult for the hacker to decrypt the signature field in the SREQ or SREP. The secure communication takes place between source and destination, (Fig. 7), after the route establishment.

The route path is maintained by the hello packets. These hello packets will make the route active till the source needs.

Type 8bits	R 1bit	Reserved 10 bits	Pfx Length 5 bits	Offset HopCount 8 bits	Destination IP Address 32 bits
Destination Sequence No 32 bits	Signature (Encrypted Source Add and HopCount) 80 bits	Source Sequence No 32 bits	Life Time 32 bits		

Fig. 6: Secure reply packet

The hello packets and route error packets are same as the AODV routing protocol use. The only extension in the SAODV is the addition of signature field in the packet header. In SAODV routing communication the hacker can not get the source and destination addresses due to signatures and authentication make the route path authentic, by which any intruder can't enter in route path.

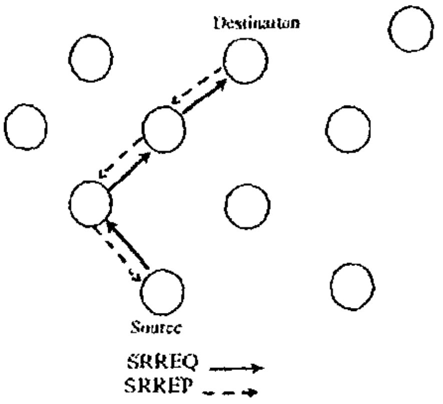


Fig 7: Route path of SAODV

Conclusion SAODV is a secure route path formation protocol, of AODV. It is used for the authentic route formation and signature prevents attacks of integrity and non-repudiation. Hacker can not easily get the source and destination address from SAODV packet header, which could be done in AODV routing packet.

SAODV routing protocol simulation works only on unicast routing path. This simulation is the secure route communication between the source and destination. Further work is to done on the multicast routing path formation. In which one source or multi source nodes communicate with multi destination nodes.

Reference

B.Awerbuch, C.N.Rotaru, D.Holmer and H.Rubens, 1998, An On Demand Secure Routing Protocol Resilient to Byzantine Failures, ACM Press. New York, NY, USA.

Charles E.Perkins and Elizabeth M.Rouer, 1999, Ad hoc On-Demand Distance Vector Routing, In Proceeding of the 2nd IEEE workshop on Mobile Computing Systems and Applications, New or leans LA, 11-12 Feb.

Charles E.Perkins and Elizabeth M.Royer, 1999, Ad Hoc Networking, Prentice Hall of India Private Limited New Delhi, pp: 173-219.

Charles E. Perkins, Elizabeth M.Royer and Samir R.Das, 2000, A Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks, In Proceeding of the IEEE Conference On Computer Communication (INOCOM), Tel Aviv, Israel, pp: 3-12, Mar.

Elizabeth, G.Vigna, R.A.Kemmerer and S.Gwalani, 2000, AODVSTAT: Intrusion Detection in AODV, In Proceedings of the 23rd National Information System Security Conference, Oct.

M.K.Marina and S.R.Das, 2001, On demand Multipath Distance Vector Routing in Ad Hoc Networks, In Proceeding of IEEE International Conference on Network Protocols (ICNP), pp 14-23.

M.G.Zapata, 2001, Secure Ad hoc On-demand Distance Vector Routing, IETF internet Draft draft-guerrero-manet-saodv-00.txt (work in progress).

R.V.Boppana and S.P.Kondum, 2001, An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks, IEEE Infocom.

W.Stallings, 1993, Network and Internet work Security, The McGraw Hill Companies, Inc.