ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY
ELECTRICAL AND COMPUTER ENGINEERING

DEPARTMENT

SECURITY IMPROVEMENT FOR MOBILE IP COMMUNICATION

By
Girma Kassa

A thesis submitted to the school of Graduate studies of Addis Ababa University in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Engineering

August 2007

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER

ENGINEERING

SECURITY IMPROVEMENT FOR MOBILE IP COMMUNICATION

By

Girma Kassa

Advisor

Abyot Asalefew

.

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

SECURITY IMPROVEMENT FOR MOBILE IP COMMUNICATION

By

Girma Kassa

FACULTY OF TECHNOLOGY

APPROVAL BY BOARD OF EXAMINERS

_____	_____
Chairman Dept. of Graduate Committee	Signature
Abyot Asalefew	_____
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Name: Girma Kassa Signature: _____

Place: Addis Ababa

Date of submission: August 17, 2007

This thesis has been submitted for examination with my approval as a university advisor.

Abyot Asalefew Signature: _____

Advisor's Name

Acknowledgment

First I would like to take this opportunity to thank my supervisor Abyot Asalefew for his continuous support and follow up, and advices that he gave me through out my thesis work. Then my sincere gratitude goes to my co-advisor Dr. Kumudha Raimond who guided me in this work especially in title selection and clarification of the problem. I am also very thankful for all my colleagues who gave me materials and helpful ideas.

I am also very pleased to thank my wife who has been helping and encouraging me, through out the course of this work. Finally I give thank to the almighty God as, with out His help one can finish nothing.

Tables of Content

List of tables

List of Figures

List of Appendixes

List of Abbreviations

AH	Authentication Header
BU	Binding Update
CN	Correspondent Agent
CoT	Care of Test
CoTI	Care of Test Initialization
DHCP	Dynamic Host Configurable Protocol
DoS	Denial of Service
ESP	Encapsulation Security Header
FA	Foreign Agent
FMN	Fake Mobile Node
HA	Home Agent
HMAC	Hashed Message Authentication Code
HoT	Home Test
HoTI	Home Test Initialization
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
MAC	Message Authentication Code
MD	Message Digest
MIPv6	Mobile IP Version 6
MN	Mobile Node
NS2	Network Simulator 2
SA	Security Association

SecMIP ---------- Secured Mobile IP

SGW ---------- Secured Gateway

SHA ---------- Secure Hash Algorithm

TTL ---------- Time to Leave

Abstract

Route optimization is an enhancement to mobile IP which improves routes for the communication of mobile node with its correspondent nodes. Despite its improvement for the performance of this communication, it brought additional security requirements for the Mobile IP communications since it needs securing the new binding update messages. If the binding update toward the home or correspondent nodes is attacked, future communication between the mobile node and the node to which the binding update was intended to reach will be highly in threat. For this reason the IETF designed solution to secure binding update toward the home agent and the correspondent nodes. The binding update toward the home agent uses the security architecture for Internet protocol and no attacks or flaws have been observed. However the binding update toward correspondent nodes is secured with return routable protocol and there are some flaws that are seen on it which can be analyzed from the design. This thesis presents such flaws on the protocol and describes a solution designed to avoid these security problems. In the solution the return routable protocol is modified to decrease the number of message transfers to send binding update and integrated to work with IPSec for securing it. The design is implemented for simulation using ns-2.29 with C++ and tested for its security and performance. Analysis made from the test results showed that security requirements of binding update to the correspondent nodes in Mobile IP communication system are satisfied as the correspondent node is able to identify false binding updates. In addition the system's performance is equivalent to the performance of base Mobile IP; the average packet processing time obtained from simulation is 0.75 sec and the average network delay is 12ms.

CHAPTER 1: BACKGROUND

1.1 Problem Statement

Since the introduction of the Mobile IP Protocol, there have been a number of researches done in different areas. The first area of research focuses on making Mobile IP communication system efficient. This area of research proposed enhancements on mobile IP for improving the performance. The researches include methods of avoiding triangular routing, using smooth handoff techniques [18] and recently route optimization system [13], [19]. Most of the researches became realistic due to improvements they brought to the Mobile IP communication. The other groups of researches were aimed at making the Mobile IP communication system secure. On this respect, a number of researches have been done to secure each part of communication including registration messages and binding update messages of Mobile IP [1], [26].

Route optimization is a new development in Mobile IP that adds a new message transfer between the mobile node and correspondent nodes. To do this, binding update messages are periodically sent from the mobile node to the correspondent nodes. The binding update message has effect on future communication between the mobile node and correspondent nodes. Any impersonation of this message could bring critical problem. This shows that securing binding update message is a must as it could avoid serious security flaw that could occur otherwise.

The IETF has designed security mechanisms to avoid possible attacks that could occur on the binding update message. But due to unpredictable nature of attackers, there are still noticeable attacks that could occur on route optimized mobile IP systems protected by IETF's security protocol. So it is very important to circumvent security problems in it.

1.2 Scope of the work

This research focuses on enhancing the security of Mobile IP. Particularly securing the new addition to Mobile IP, route optimization for Mobile IP, was selected to be the core of this research. The work extends:

1. Modification of the current route optimization protocol
2. Securing the modified route optimization protocol

To achieve the main goal of this work, securing mobile IP route optimization, it was important to modify the existing protocol. The protocol modification was done to decrease the possible attacks that could occur on binding update. Security design of the protocol includes adapting part of IPSec with Mobile IP route optimization security. Pertinent part of IPSec that satisfies the security requirements was selected and working conditions for it were described. Details of the protocol and the security are described and some issues and assumptions for using the protocol and security designed are explained. Some simulations are performed to test the whether the design objectives are satisfied.

1.3 Related works

1. SecMIP: Secure Mobile IP (SecMIP) is a protocol that provides Mobile IP users secure access to their company's firewall protected network. It is proposed by Torsten Braun and Marc Danzeisen [26]. It doesn't introduce new protocol to have a secure system rather it requires the modification of end systems to adapt to the mobile IP and internet security protocol (IPSec).

In [26] they described the five steps that are needed in order to have the secure communication between the two entities. It is implemented and tested for performance. Accordingly, the paper presented that a reasonable security and performance results were obtained. The disadvantage seen on this system is every communication made to/from the mobile node is made through the home agent. Due to this, this design will overload the home agent and packets will traverse unnecessarily long route.

2. Secure Mobile IP Protocol: This is a protocol proposed by Atsushi Inoue, Masahiro Ishiyama, Atsushi Fukumoto and Toshio Okamoto [1]. A firewall called Secured Gateway (SGW) with IPSec processing capability is placed on networks to be secured and the mobile node also needs to process IPSec. In order to communicate with nodes inside a network, the mobile node should know at least an address of a SGW. A dynamic means of identifying the nearest secure gateway is designed for the mobile node and the mobile node uses this service to have communication with a network. Implementation status and performance of the system was reported in the above paper. A problem observed with this protocol is that if the mobile node doesn't know the address of SGW, its communication with the internal network will be limited or denied which defeats the purpose of mobile IP.

3. Using IPSec between Mobile Node (MN) and Home Agent (HA): This is a research for securing Mobile IP communication between a home agent and mobile node. To do this, the research first specified the detailed security requirements to be achieved. Then packet format for each of the messages to be transferred is illustrated.

Under its requirements section, the research presented security requirements of the path (the control traffic between the mobile node and the home agent requires message authentication, integrity, correct ordering and anti-replay protection.) and a number of attacks that could occur if the path is not protected. In order to avoid attacks, the paper specified the use of IPSec Encapsulating Security Payload (ESP) [13]. The traffic to be secured between the nodes is of three types: Binding Update and Acknowledgement messages, Return Routable Protocol Home Test messages, ICMPv6 messages exchanged between the mobile node and the home agent for the purposes of prefix discovery and optionally the payload packet.

Format of parameters like security association databases are discussed. Manual configuration and dynamic keying of the parameters for each of the packet types is illustrated. Finally an implementation case is stated with possibility of using other formats.

4. Return routable protocol: It is designed by IETF as security mechanism for route optimization in IPv6 [19]. There are some steps for calculating a common key for securing the binding update from the mobile node to the correspondent node. Once a common key is calculated, a message authentication code (MAC) will be calculated and the binding update will be sent by securing it with the MAC. The major problem with this design is that it doesn't secure all the paths of communication that participate in calculating the shared key which opens a hole for attackers. The details of this protocol are discussed in chapter 4.

5. Cryptographically Generated Addresses: This is another means of securing the binding update by using a one way hash value generated by the MN [27]. A mobile node wishing to send binding update generates a hash output from its public key and auxiliary parameters and takes the first 64 bits of the output . It uses this output as an interface identifier part of the source of the message. The mobile node attaches its public key value and auxiliary parameters with the binding update message and sends the message. The receiver will recalculate the one way hash from the public key and auxiliary parameters and compares the output with the interface identifier of the source of the message.

This approach is welcomed as it avoids the use of public key infrastructure and produce one-to-one value for each input. However it has the following drawbacks:

1. It doesn't set any means of authentication for MN.
2. The protocol requires that the MN, which has low computation capacity, to produce a hash value for each binding update (BU) to be sent.

Since there is no identification or authentication of a node to be a mobile node, an attacker can produce a binding update message and generate a hash output from its public key and auxiliary values and make the output as its interface identifier. The receiver using the normal step of checking will obtain a positive check which could lead to attack.

1.4 Objective

The objective of this thesis is to make an improvement on security of Mobile IP communication particularly security of the Route optimization part of Mobile IP. Specifically this thesis:

1. analyzes the flaws in the security of route optimization of Mobile IP
2. proposes a design to solve the flaws identified and implement the design using simulating language NS2

1.5 Methodology

The methodology followed in this thesis is a case study followed by selection. That is Mobile IP is first studied as a whole and focus is made on route optimization part of it. In studying the route optimization, attention was given to its security to point out different cases of threats and the possible flaws are clarified. Once the flaws are pointed out, a design was made to mitigate the flaws observed. In proposing the solution, different possible ways of solving the problem are investigated and the one which is straightforwardly to satisfy our objective is selected, designed and implemented. To arrive at conclusion test cases are stated and tests are conducted.

1.6 Outline of the thesis

This thesis consists of seven chapters which are organized as follows. In the first chapter an introduction on the work is described under the sub topics of problem of statement, scope of the work, related works, objectives of the thesis, methodology and its report outline. In the next two chapters subject matters related to the protocol and security over the internet are discussed. Chapter two gives an overview of Mobile IP, part of which is going to be secured and chapter three explains the means selected for securing the route optimization part of Mobile IP. In the fourth chapter security problems in Mobile IP are explored and illustrated in detail with specific examples. Chapter five contains the design developed to mitigate the problems identified in chapter four; here both the design and implementation were described. In chapter six the simulation set up for different scenarios are set and the results obtained are presented and discussion is made on the results. Finally, conclusion of the work and future recommendations are presented in chapter seven.

CHAPTER 2: REVIEW OF MOBILE IP

2.1. Introduction

Mobile IP is an open standard defined by the Internet Engineering Task Force (IETF) in 1996 with RFC 2002 as an enhancement on the existing internet protocol. Mobile IP allows users to keep the same IP address and stay connected while they are moving over networks. It is scalable to the Internet as it is based on IP [3].

In Mobile IP a mobile node will have two addresses that are named home address and care of address. In the base Mobile IP, mobile node will be identified with its home address irrespective of the network it is connected to. Any communication that is going to be made between the mobile node and any other node will identify the mobile node with its home address. When the mobile node is away from its network, it assigns a representative in the network that will receive packets destined to it. So any packet whose destination address is the home address of the mobile node will be intercepted by the representative and the representative will send it to the mobile node.

Mobile IP is made up of different components and functional units. In one respect it comprises of different sub protocols and on the other hand there are major entities in mobile IP. The combination of all these, makes up the complete and functional mobile IP.

2.2. Importance of mobile IP

The requirement for mobile IP is necessary because of the limitations that are observed over the traditional IP. In traditional IP system, a node will always have one address that uniquely identifies it. And this address will have two components, Network Identifier and Host identifier. The principle of network and host portion of an IP address would lead us to the following points:

1. If a node changes its point of attachment it is a must for it to change its IP address. This does mean that there are no places (nodes) over the internet with the same host and network identifier.

2. If a node changes its address, other nodes will not know its new address unless they are informed about it

For these reasons a node with traditional IP that moves to another network will not be able to communicate at the new network with out additional IP configuration. There are situations which highly demand the node's communication not to be disconnected from its base network. In addition reconfiguration may not be simple for average users. For all these reasons it would be better for users to be connected to remote networks on which they work and enjoy as if they are in their home network which can be done using mobile IP.

2.3. Architecture of Mobile IP

The emergency of mobile IP introduced new architectural entities and terminologies [2]. Most important entities and terminologies in mobile IP are the following:

1. ***Mobile Node (MN):*** it is a node that can change a point of connection with out changing its IP address.
2. ***Home Agent (HA):*** a system in the mobile node's home network which registers the location of the mobile node receives packets destined for the mobile node at the home network and tunnels the packets to the care of address of the mobile node.
3. ***Correspondent Agent (CN):*** any node that communicates with the mobile node.
4. ***Foreign Agent (FA):*** it is a router at the foreign network that assists a locally reachable mobile node in delivering packets between the mobile node and the home agent. It also assists the mobile node in getting the care of address.
5. ***Home address:*** the IP address of the mobile node at the home network. This address remains the same irrespective of the attachment place of the mobile node.
6. ***Care of Address:*** Care-of-address is an IP address that identifies the current location of a mobile node when the node is not attached to its home network. It will be obtained from the foreign network and will be registered to home agent
7. ***Home Network:*** The network in which the mobile node exists for most of the time.
8. ***Foreign Network:*** a network which the mobile node visits.

9. ***Binding Update:*** The association of a home address with a care-of address, along with the remaining lifetime of that association.

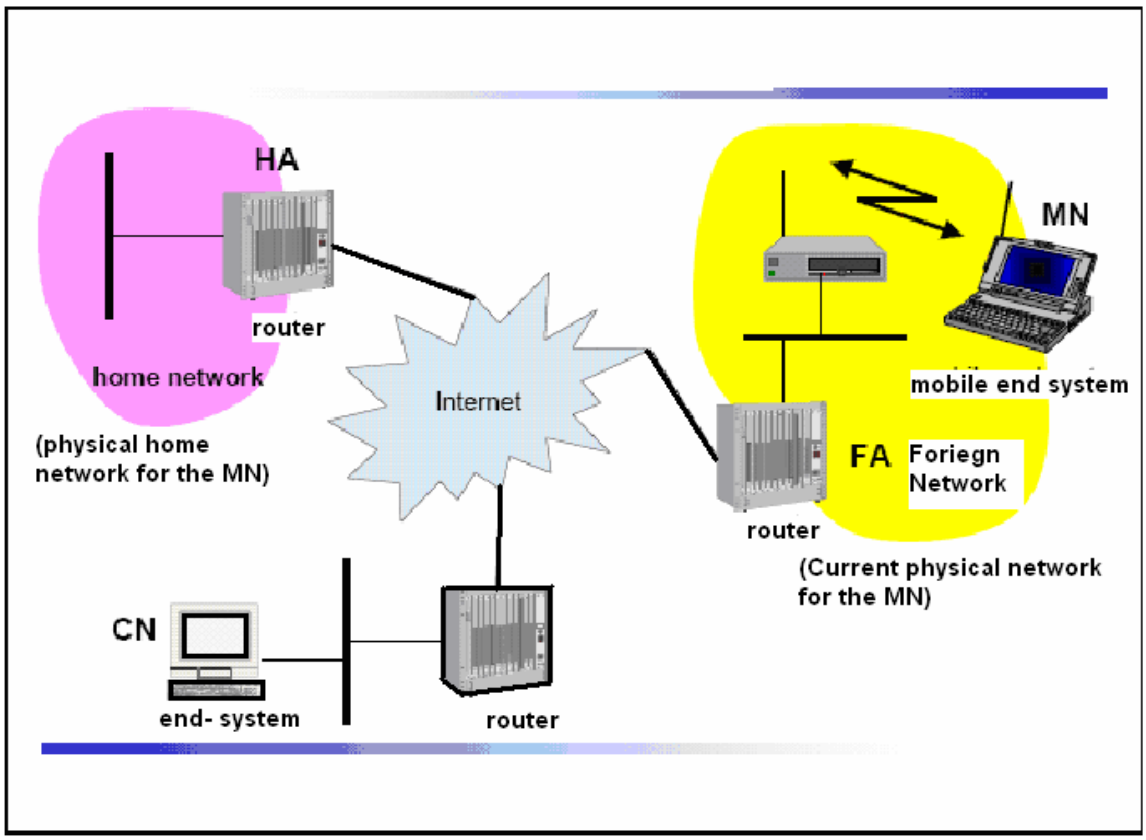


Fig 2.1 Architecture of Mobile IP Communication [15]

2.4 Protocols in Mobile IP

Mobile IP introduced the use of two IP addresses for a particular mobile node. The first is the home address while the other one is the care of address as stated above. The home address is given from any IP service provider, statically from administrator or may be dynamically assigned from DHCP server in its home network. The care of address will be given from an IP provider in the foreign network with the permission of home agent. Usually the home address is static and remains for long period with the mobile node while the care of address, which

specifies the network in which the mobile node currently exists, changes as per the location of the mobile node.

The Mobile IP protocol is the combination of the following sub protocols. The cooperation of these protocols provides the fully functional mobile IP [2].

1. Agent Discovery: the home agent and foreign agent advertise their presence and the mobile agent will discover them in this phase.
2. Registration: The mobile node will register the care of address obtained from foreign network with home agent in this phase.
3. Tunneling: A tunnel is set up to route packets from the home agent to the foreign agent and finally to the mobile node.
4. Route Optimization: this is a means by which correspondent nodes know the care of address of mobile node and reduce future packet's traversal between mobile node and correspondent node.

2.4.1 Agent Discovery

The foreign agent and the home agent periodically advertise their presence by sending a broadcast message. All advertisements are within a sub network and will not be routed outside this network. This is achieved by sending a broadcast message with its TTL value set to one. A mobile node is also allowed to send ICMP Router Solicitation messages in order to elicit mobility agent advertisement [4].

A mobile node entering a new network listens to advertisements, checks the value of the network component of the source address of the advertised message. The mobile node will use this value to decide the network with which it is connected. If the value is the same as network identifier value of its IP address, it will know that it has connected to its home network; otherwise it will understand that it is connected to the foreign network.

If the mobile node understands that it is in its home network, it will deregister itself so that it will have only one IP address which is its home address. On the other hand, if the mobile node is connected to a foreign network, it will decide to ask for care of address. It will acquire the care of address in one of the following two forms:

1. Care of address obtained from foreign agent
2. Collocated care of address

A Foreign Agent care-of address is an IP address of a Foreign Agent that has an interface on the foreign network being visited by a Mobile Node. A Mobile Node that acquires this type of care-of address can share the address with other Mobile Nodes. A collocated care-of address is an IP address temporarily assigned to the interface of the Mobile Node itself. A collocated care-of address represents the current position of the Mobile Node on the foreign network and can be used by only one Mobile Node at a time [2].

2.4.2 Registration

Once the mobile node obtains a care of address, it sends a registration message to its home agent. The registration process is almost the same whether the mobile node has obtained its care-of address from a foreign agent, or alternatively has acquired it from another independent service such as DHCP [4]. In the former case, the mobile node basically sends the request to the foreign agent which will send it to the home agent. In the latter case, the mobile node sends its request directly to the home agent, using its collocated care-of address as the source IP address of the request.

The foreign agent inspects the validity of the registration request, adds the request to its pending list and forwards this request to the home agent to ascertain if the request is valid. The foreign agent responds with an appropriate error code if the request is invalid. A request may be invalid if the value specified in the registration lifetime field is too high.

The home agent authenticates the mobile node and checks the validity of the registration request. If the request is valid, the home agent associates the mobile node with its care-of-address and

creates a tunnel in order to forward packets to the foreign agent. The home agent may send a registration reply with appropriate error code if the registration request was found to be invalid.

The foreign agent then inspects the validity of the registration reply. If the reply is valid, the foreign agent adds the mobile node to its visitor list and creates a tunnel to the home agent. Finally this reply is forwarded to the mobile node. The mobile node checks the validity of the registration reply, and the presence of a valid registration specifies that the mobile agents are aware of its roaming.

The registration of the care of address will have the following procedure [2]:

When the registration is through the foreign agent

1. The mobile node sends the registration request to the foreign agent.
2. The Foreign agent processes the registration request, gets the address of the home agent from the registration request message and forwards it to the home agent.
3. The home agent sends the registration reply to the foreign agent accepting or denying the request.
4. The Foreign agent processes the registration reply and forwards it to the Mobile node.

When the registration is directly from mobile node to home agent

1. The Mobile node sends a registration request to the home agent.
2. The home agent sends the registration reply to the mobile node accepting or denying the request.

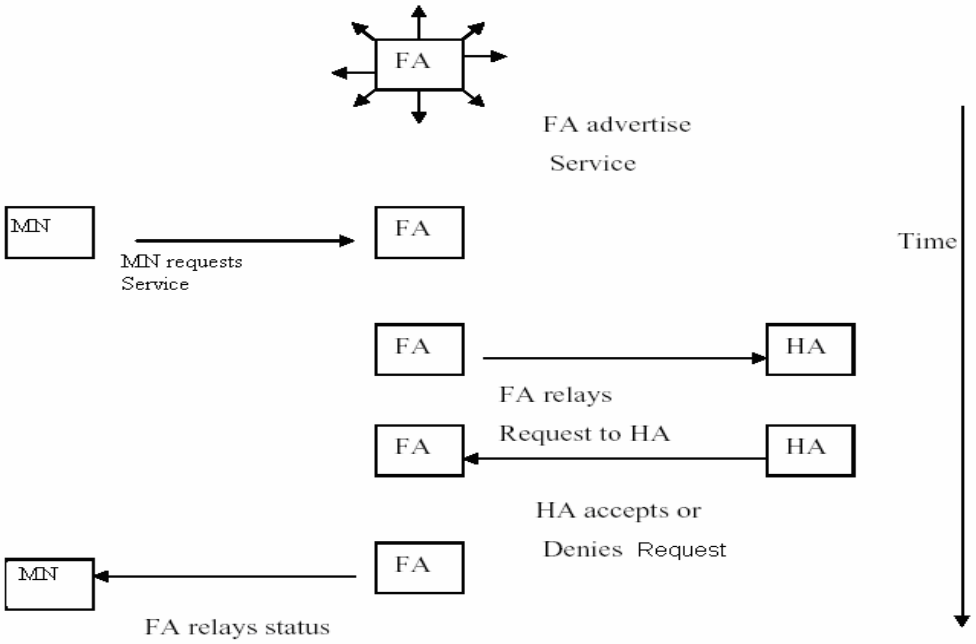


Fig. 2.2 Agent Discovery and Registration Process

2.4.3 Tunneling

Once the process of registration is completed, the mobile node can establish a seamless communication irrespective of changing its point of attachment across the internet. The home agent, after a successful registration, will begin to attract datagrams destined for the mobile node and tunnel each one to the mobile node at its care-of address. This tunneling is done by the IP-in-IP encapsulation [16].

With IP-in-IP encapsulation, an outer header is inserted before the datagram's existing IP header. There are spaces between the outer header and inner header for including other headers when security is required to protect the original payload during tunneling. In the case of Mobile IP, the values of the fields in the new header are selected naturally, with the care-of address used as the destination IP address in the tunnel header. The encapsulating IP header indicates the presence of the encapsulated IP datagram by using the value in the outer protocol field. The inner header is not modified except to decrement the TTL by 1.

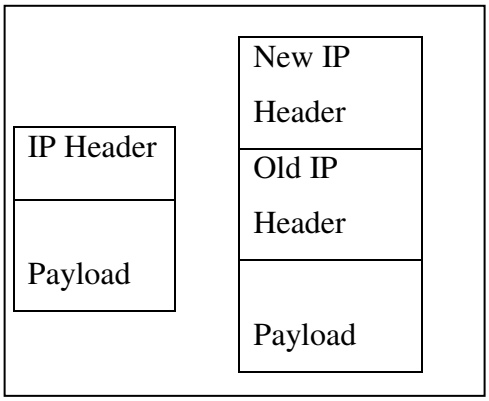


Fig. 2.3 IP-in-IP Encapsulation [16]

2.4.4 Mobile IP Route Optimization

In the original mobile IP, any communication towards the mobile node was through the home agent that has the recent care of address of the mobile node [4]. This is an asymmetric routing and is called triangle routing, and it is generally far from optimal, especially in cases when the correspondent node is very close to the mobile node.

This triangular routing has a major problem when the correspondent node is very close to the mobile node. Consider a case that a correspondent node wants to send a packet to a mobile node that is on the same network. If the sender node happens to be the far side of the Internet, far away from the mobile node's home network, the packet has to route a long distance and has to pass lots of routers to reach the home agent. Once it reaches the home agent, it has to return back to the mobile node.

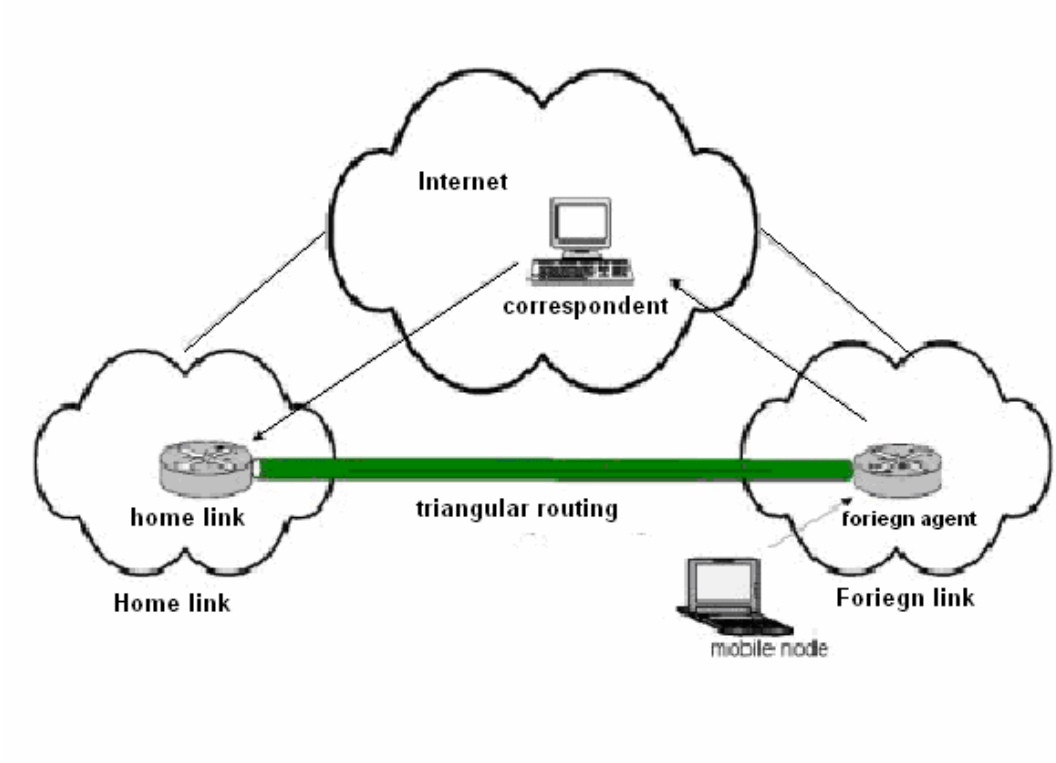


Fig. 2.4 Triangular Routing [29]

As can be seen from the figure 2.4,

1. packets traverse unnecessary long route
2. performance is degraded due to overhead on home agent

To alleviate the performance penalty, Mobile IPv6 includes a mode of operation that allows the mobile node and its peer, the correspondent node (CN), exchange packets directly, bypassing the home agent completely after the initial setup phase. This mode of operation is called route optimization (RO). When route optimization is used, the mobile node sends its current care-of address to the correspondent node, using binding update message. The correspondent node stores the binding between the home address and care-of address into its Binding Cache and uses the care of address of mobile node for future communication with MN [19]. The possibility of tunneling through the home agent is also kept as option in route optimized Mobile IP.

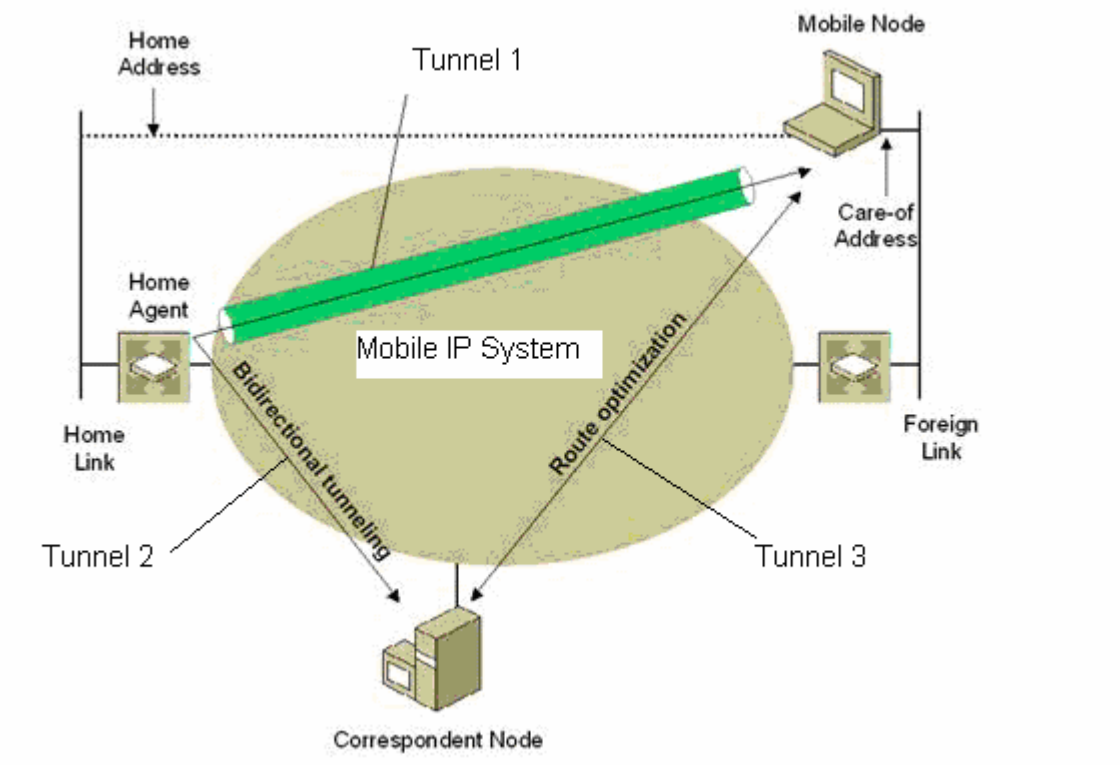


Fig. 2.5 Typical Route optimization

Figure 2.5 above shows typical topology for route optimized Mobile IP communication. As shown in the figure, the mobile node and correspondent nodes can make communication directly using tunnel 3 with complete by passing of the home agent. However before direct communication is made between the two entities, the correspondent node needs to obtain the binding update message using some communications through home agent using tunnel 1 and tunnel 2 communication links.

CHAPTER 3

BACKGROUND ON IP SECURITY AND HASHED MESSAGE AUTHENTICATION CODE

Information security can be defined as the process of protecting information from unauthorized access, use, disclosure, destruction, modification, or disruption. It is also defined in terms of insuring some security services like confidentiality, integrity, availability and others. The way of satisfying security needs of a system depends on the nature of the system and the possible attacks that could occur on system. For these reason the security needs of communication systems vary depending on the nature of the system.

3.1 Security Assurance

Full security assurance is not an easy task. This is due to the unpredictable nature of attacks that could be made by attackers over the internet. However many researches show that data communication over an IP could be secured if it satisfies authentication, integrity, confidentiality and non-repudiation security services [21], [9], [28]. Some illustration on these services is given below.

Authentication

The property of knowing that the sender of the data is the actual sender is authentication. It is used to verify the identity of a user, device or other entity in a computer system. With authentication we can verify the user, device or any other entity in a system and it is often used as a prerequisite for allowing access to resources in a system.

Integrity

Data integrity is defined as the property of ensuring that data is transmitted from source to destination without undetected alteration.

Confidentiality

The property of communication such that the intended recipients know what was being sent but unintended parties cannot determine what was sent is confidentiality of data. It is also defined as the way of holding sensitive data in confidence, limited to an appropriate set of individuals or organizations.

Non-repudiation

The property of a receiver being able to prove that the sender of some data did in fact send the data even though the sender might later desire to deny ever having sent that data.

3.2 Internet Protocol Security (IPSec)

3.2.1 Introduction

The Internet Engineering Task Force defined a protocol called IPSec for securing communication over the Internet. IPSec creates a boundary between unprotected and protected interfaces for a host or network. It can be used to protect one or more paths between a pair of hosts, between a pair of security gateways or between a security gateway and a host [21].

IPSec provides three major security services. These are confidentiality, authentication and key management. In order to provide these functionalities IPSec uses two IP headers. These are the Authentication Header (AH) and Encapsulation Security Payload (ESP). Both headers are available for both versions of IP, Ipv4 and Ipv6.

3.2.2 Architecture of IPSec

The objective of the architecture of IPSec is to provide security services for traffic at IP layer [12]. The fundamental components of IPSec are described in the figure below:

1. Security protocols: include the authentication header (AH) and encapsulation security payload (ESP)

2. Security association: states how the protocols work, managed and the associated processing
3. Key management: it is the how keys are produced which could be automatic or manual. In IPSec it is internet key exchange (IKE) that is used for key management.
4. The algorithms used for encryption and authentication of message.

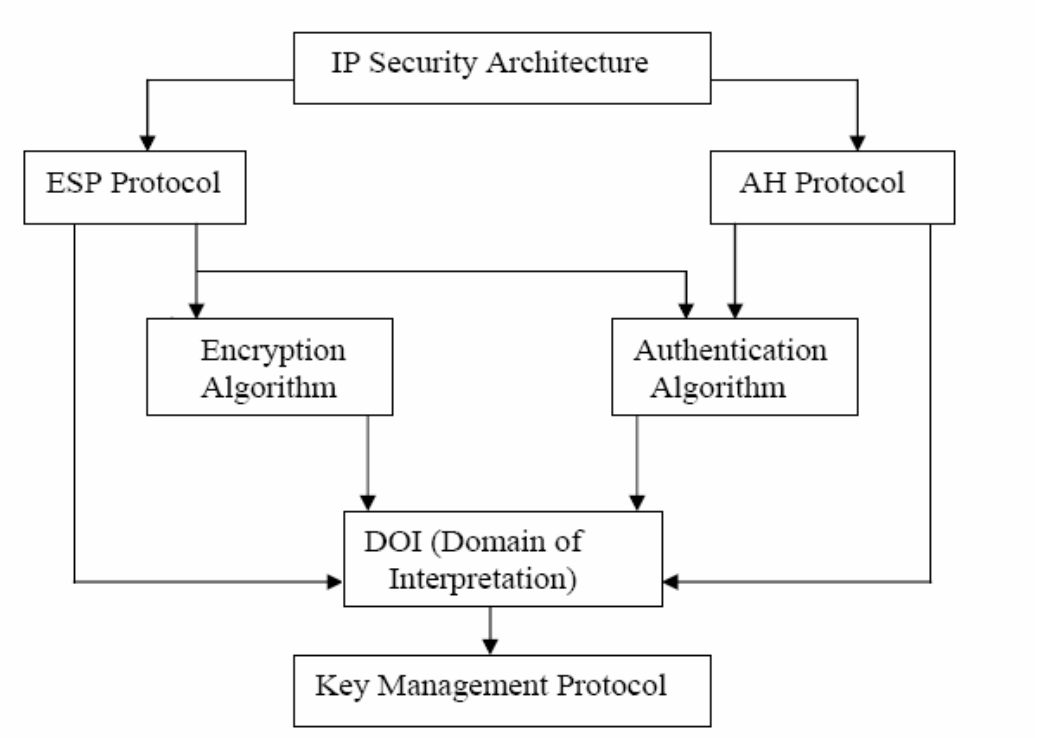


Fig. 3.1 Architecture of IPSec [12]

The Authentication Header is used to provide the services of original data authentication and integrity with optional anti-replay feature. The Encapsulation Security Payload provides the same sets of services that are available in AH. In addition to services provided by AH it also offers the data confidentiality service. Below is the detail discussion of these headers.

3.2.3 The Authentication Header

The AH's authentication header verifies the source's claimed identity. The Integrity part of AH ensures that datagram is not altered by malicious node before reaching the destination. And the anti replay protection ensures users not to receive packets that are intentionally captured and replayed by malicious users.

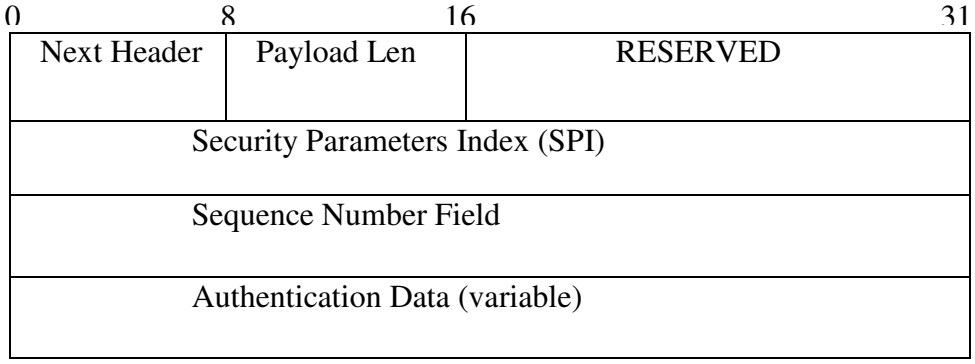


Fig. 3.2 AH header [23]

The 8-bit Next Header field identifies the type of the next payload after the Authentication Header. The payload length field is also an 8-bit field used to specify the length of the payload in 32 bits word. The 16 bit reserved part is reserved for future use. The security Parameter Index is used to uniquely identify the Security Association. The sequence number field is a monotonically increasing field that protects anti replay problems. The variable length Authentication data field contains the Integrity Check Value (ICV) or Message Authentication Code (MAC) for this packet. AH can be used alone or in combination with ESP. Both AH and ESP operate in two modes of operation, the Transport mode and Tunnel mode.

Transport Mode AH

The transport mode is available for host implementations and is used to protect the upper layer protocols and some selected IP headers. In this mode, AH is inserted after the IP header and before an upper layer protocol or before any other IPsec headers that have already been inserted. In IPv6 the destination option header can be inserted either before or after the AH header.

Though the exact mechanism used for placing the header into the datagram and for linking the headers together varies, transport Mode AH is available for both Ipv4 and Ipv6. The AH is inserted into the IP header as an extension. The placement of AH in data gram is shown below in the diagrams.

Original IP Header	AH	TCP	Data

Fig. 3.3 Authenticated Transport IPv4 Packet

Original IP Header	Hope-by-hope, Dest, Routing, Fragment	AH	Dest	TCP	Data

Fig. 3.4 Authenticated Transport Ipv6 Packet

Tunnel Mode AH

The tunnel mode could be implemented either in hosts or gateways and is mandatory for securing gateways using AH. In this mode AH provides security for the entire IP datagram. The inner IP headers identify the ultimate source and destinations of the datagram while the outer headers contain some address in between like security gateways. The position of the AH header in this mode is shown below.

New IP Header | AH | Orginal IP Header | TCP | Data

Fig 3.5 Authenticated Tunnel IPv4 packet

New IP Header | Ext Header | AH | Orgin IP Header | Ext Header | TCP | Data

Fig. 3.6Authenticated Tunnel IPv6 packet

3.2.4 The Encapsulation Security Payload

The ESP is used to provide confidentiality, authentication, connectionless integrity and anti replay services as stated above. It has the advantage of having the confidentiality service over the AH. Set of security services can be selected while a security association is made [24]. The ESP is designed to work with different authenticating and encryption algorithms but the common encryption algorithm in ESP is data encryption service (DES).

<table>
<tr><td colspan="3">Security Parameter Index (SPI)</td></tr>
<tr><td colspan="3">Sequence Number</td></tr>
<tr><td colspan="3">Payload Data (variable)</td></tr>
<tr><td>Padding</td><td>Pad Length</td><td>Next Header</td></tr>
<tr><td colspan="3">Authentication Data (variable)</td></tr>
</table>

Fig. 3.7 IP Encapsulating Security Payload Header [12]

The SPI is an arbitrary 32-bit value that with the combination of destination IP address and security protocol identifies the security association. The sequence number field also has the same functionality as in AH header. Payload Data is a variable-length field containing data described by the Next Header field. Padding is an optional field which is needed for making the plain text multiple of 4 bytes. The pad length indicates the number of bytes the pad has. And the 8-bit next header indicates the type of data contained in the payload data field. Authentication data field is a variable length field containing an integrity check value (ICV) computed over the ESP packet minus the Authentication Data. The length of the field is specified by the authentication function selected [24]. ESP operates in tunnel mode and transport mode as AH does.

ESP Transport Mode

In transport mode, ESP is inserted after an IP Header and before the upper layer protocols or before any IPSec header that has been inserted. ESP authentication data field is placed after the ESP trailer when authenticating after the ESP trailer is needed. The entire ESP segment and ESP trailer are encrypted. The location of ESP is as shown below for both versions of IP.

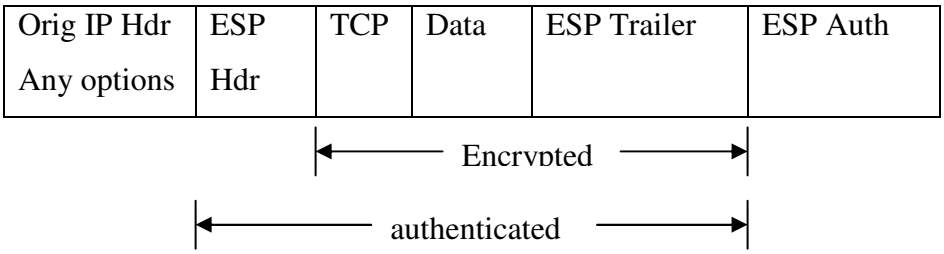
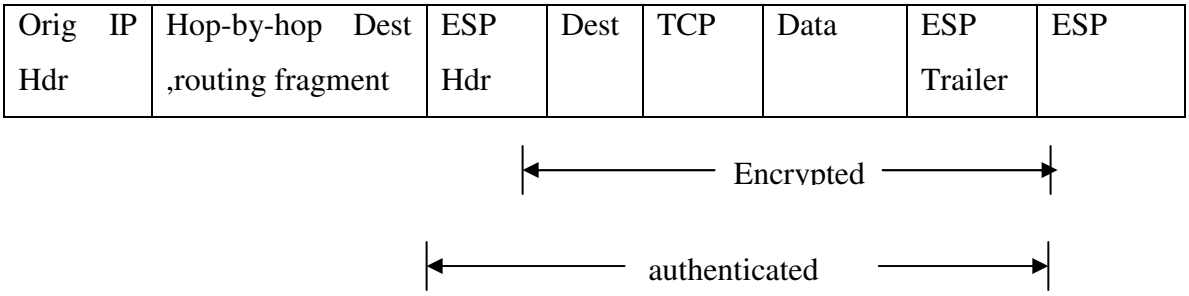


Fig. 3.8 ESP Header in Transport Mode IPv4 Packet



3.9 ESP Header in Transport Mode IPv6 Packet

ESP Tunnel Mode

In this mode of operation the entire IP packet will be encrypted so that the protection will be for both IP header and packet. And ESP in this mode may be implemented in either host or gateways. The outer and Inner IP address in this mode have similar identifications like the AH tunnel mode.

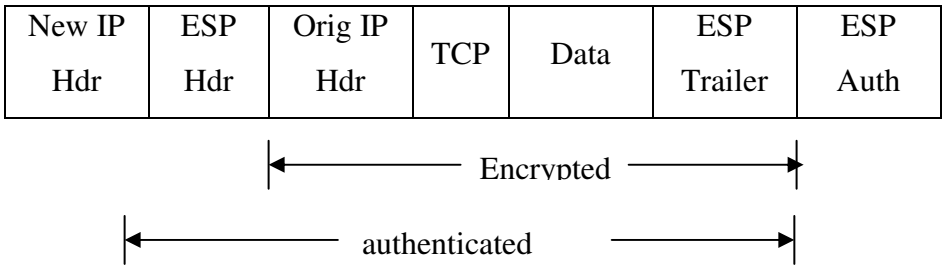


Fig 3.10 ESP Header in Tunnel Mode IPv4 Packet

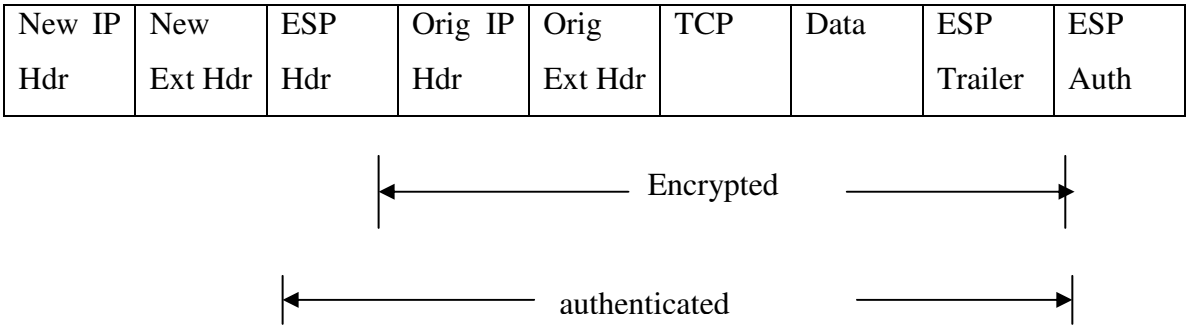


Fig 3.11 ESP Header in Tunnel Mode IPv6 Packet

3.2.5 Key Management for IPSec

Basics of keys: A key can be defined as a secret code or number that is required to read, modify, or verify secured data. Keys are used in algorithms for securing a given data. Since Key management is one of the aspects of IPSec for its secure operation, the internet society gave attention to it and designed a suitable key management.

Keys could have variable or fixed length. In IPSec in case of variable length keys, it is possible to select integrity and encryption algorithms that provide a range of key lengths. If the length of a key is increased by one bit, the number of possible keys doubles, which will make it exponentially more difficult to determine the key. Shorter key lengths are processed more quickly with less resource than longer key lengths. Longer keys are selected for higher levels of security.

Key Management: Key management is a function for systems to create IPSec tunnels without requiring administrators to manually program keys. Using key management of IPSec, two devices that want to send information securely encode and decode it using a piece of information that only they know. And it will not be possible for any other party to read the information (in case ESP is used) or temper it while not possible to detect it (in case AH is used) for IPSec. The primarily used protocol for this purpose in IPSec is called *Internet Key Exchange (IKE)*. IKE is used to negotiate a key exchange between two IPSec enabled nodes so that both devices can share data across an IPSec tunnel [22].

Internet Key Exchange (IKE)

In IPSec protected communication, a security agreement should be established between two communicating parties and the parties need to agree on how to exchange before the ability for exchanging secure data and protect their information transfer between them [4]. The protocol that enables the parties to make security association and authentication is IKE. IKE is a combination of two earlier security protocols (Oakley and SKEME), within an ISAKMP (Internet Security Association and Key Management Protocol) [4]. This is why it is also called a hybrid protocol.

Phases of IKE

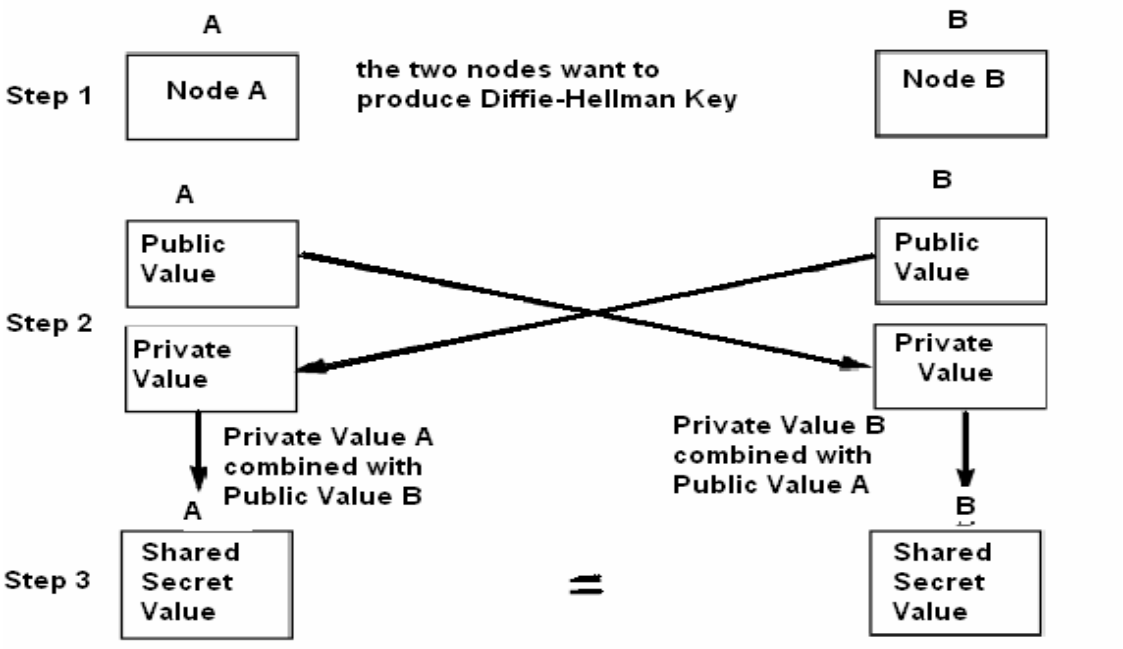
IKE operates in two phases which are the main mode (first phase) and quick mode (second phase).

In the first phase of IKE, a secured channel between the two peers that use IKE will be made by establishing a security association between the peers. The security association makes up secured communication channel, keys with their life time, encryption methods and other parameters. IKE is based on diffie-hellman key generation. In this phase, the two systems generate a diffie-hellman key value and further IKE communication is encrypted and authenticated using this symmetric key [4].

In the second phase, using the SA established in the first phase, negotiatiation of secure channels for IPSec (AH and ESP) will be done.

Diffie – Hellman Algorithm

Diffie-Hellman key exchange (also called key agreement or key negotiation) is a cryptographic protocol invented in 1976 by Whitfield Diffie and Martin Hellman [10]. It is a protocol that allows two parties to agree on a secret shared key that they can do encryption of a data over an insecure communication channel and make the secrete key to be unavailable to eavesdroppers. The Diffie- Hellman key agreement requires the two peers to have key pairs of which one is private and the other public. By combining one's private key and the other party's public key, both parties can compute the same shared secret number. The figure and equations below show how the two parts will produce the shared key using same mathematical function.



.

Fig.3.12 Diffie-Hellman Key Exchange

The shared secret key is mathematically defined as: [10]

$ZZ = g \wedge (x_b * x_a) \bmod p$ **Equation 1**

the actual operation for each of the peers is:

$ZZ = ZZ = (y_b \wedge x_a) \bmod p = (y_a \wedge x_b) \bmod p$ **Equation 2**

where ^ denotes exponentiation

y_a is public key for a; $y_a = g \wedge x_a \bmod p$

y_b is public key for b; $y_b = g \wedge x_b \bmod p$

x_a and x_b are private keys for a and b respectively

p and q are large primes

g = h^ {(p-1)/q} mod p, where

h is any integer with 1 < h < p-1 such that h{(p- 1)/q} mod p > 1

g has order q mod p; i.e. g^q mod p = 1 if g!=1

j a large integer such that p=qj + 1

The Diffie-Hellman may be attacked if a third party can intercept messages between the two parties. Because of this, Diffie-Hellman should be used with some form of authentication to ensure that symmetric keys are established between correct parties. For this certificate authorization (CA) techniques like public key infrastructures (PKI) are used [10].

3.3 Hashed Message Authentication Code

3.3.1 Introduction

In the previous section we have seen that we can employ IPSec authentication header for securing communication when authentication is our security requirement. The authentication and integrity of a message over an insecure environment can be achieved by the use of Hashed Message Authentication Code (HMAC). HMAC is defined in RFC 2104 [22] and it has two basic components. These are the key part (K) and the cryptographic hash function (H) part.

HMAC uses Message Authentication Code (MAC) to authenticate the source of a message and its integrity without the use of any additional mechanisms. HMAC uses two functional entities, the secrete key shared by the message originator and intended receiver and input message. Once the key agreement is made they will have a shared secret key. Then HMAC will use this key and the hash algorithm for authenticating message transfer.

The hash function takes a variable length input message and produces a fixed length output called Message Authentication Code (MAC) for the input. The message originator will send the MAC value with the message to the receiver. The receiver will use the key and message received to calculate its own MAC value. This MAC will be compared with the one sent from the source of the message. In a secure transmission of a message the two MACs must match each other. Otherwise the receiver will assume that the message is altered.

3.3.2 Steps in Operation of HMAC

There are certain steps for calculating MAC value of HMAC [22]. The function used to compute a MAC over the data '*text*' using the HMAC function is:

$$MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus opad) \| H((K_0 \oplus ipad) \| text))_t \quad [8]$$

The variables in this algorithm are:

- K is the key to be used in HMAC
- ipad is the byte 0x32 repeated B times
- opad is the byte 0X5C repeated B times
- H is the hash function to be used
- B block length of the hash function
- || is symbol for appending
- $\oplus$ is an XOR operation

Next we will describe the above function in a more detail manner.

1. Key length Calculation: The first stage of the algorithm is to convert the key to be exactly B bytes long. If the key length is less than B bytes, this is done by adding zero bytes to the end of the key, to form K_0 of exactly B bytes.
2. The key will be XORed with ipad.
3. The message to be hashed will be appended to the result of 2.
4. Hash algorithm will be applied to the above result (result of 3).
5. The key will be XORed with opad.
6. The result from 4 will be appended to the result of 5.
7. The hash algorithm will be applied to the result of 6.
8. The output will be used for signing the data to be sent

3.3.3 Cryptographic Hash algorithms

A cryptographic hash algorithm is also known as message digest or one-way hash takes a variable length input message and produces a fixed length output for a given input. The output is the digest or finger print of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest.

The above supposition is based on the following two properties for the cryptographic hashing:

1. It must be resistant to collision, i.e. any message's output should not collide with any other message's output.
2. It is very difficult to obtain the input from output. A small change in the input will bring easily noticeable change in the output.

There are two common varieties of cryptographic hashing algorithms. These are the Message Digest (MD) and the Secure Hash Algorithm (SHA). SHA 1 from SHA group and MD5 from MD are the currently widely used cryptographic hash algorithms.

A. MD5 Algorithm

MD5 is an acronym for message digest 5 and is an extension for MD4. The MD5 was designed to make improvement on the security of MD4. MD4 was designed to be very fast and it is susceptible to risks of successful cryptanalytic attack.

The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" as an input [20]. It is intended to use this algorithm for digital signature applications where a large file must be signed before being encrypted with private key under public key crypto systems.

B. Secure Hash Algorithm 1

SHA-1 is a cryptographic hashing standard defined in April 1995 by the National Institute of Standards and Technology in the FIPS 180-1 paper. SHA-1 is nonreversible, collision-resistant, and has a good avalanche effect. There is no successful attack reported on it till now.

Principally, SHA-1 is similar in operation with the MD5 algorithm except for the message digest size outputted from it. So we will only see the illustration of SHA-1 algorithm. It is computationally vast compared with MD5 but it is more secure. SHA-1 is an iterated hash function which takes a string bits of any size less than 2^64 bits and produces a message digest of 160 bits. This digest is computed by logical operations on the bit strings of the input.

Steps of operation in SHA-1

Message digest for both data file and a message can be produced with SHA-1. The input is to be considered in bit string and the length of the input is the number of bits. The message can be represented with hex value for compactness purpose. SHA-1 sequentially processes blocks of 512 bits when computing the message digest.

Once the message is arranged in entry format there are some operations that will be made in order to get the message digest [7].

1. Message padding: The purpose of message padding is to make the total length of a padded message a multiple of 512.

- A 1 is always appended immediately to the message
- Then n number of 0s will be appended where n is the difference between 448 and the length of the message appended with 1.
- The final step in message padding is appending the message length. The message length will be represented with in 64 bit format and will be appended after the 0s. In this stage the padded message will contain 16 * n words (32 bits of string) for some number n. The padded message is regarded as a sequence of n blocks M(1), M(2), first characters (or bits) of the message.

2. Functions and constants used: A sequence of logical functions and constant words are used in the SHA-1.

Logical functions f (0), f (1),..., f (79) are used in SHA-1. Where each f (t), 0 <= t <= 79, operates on three 32-bit words B, C, D and produces a 32-bit word as output.

f (t; B,C,D) is defined as follows: for words B, C, D,

f (t;B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19)

f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79).

Constant words K (0), K (1), ..., K (79) represented by hex as below are used

K(t) = 5A827999 (0 <= t <= 19)

K(t) = 6ED9EBA1 (20 <= t <= 39)

K(t) = 8F1BBCDC (40 <= t <= 59)

K(t) = CA62C1D6 (60 <= t <= 79).

3.3.4. Computing the Message Digest

Once the message padding is done and constants are defined, the message digest will be computed in the following way [though there is another method, it increases execution time so this method is preferred] [7]. The computation is described using two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labeled A, B, C, D, E. The words of the second 5-word buffer are labeled H0, H1, H2, H3, H4. The words of the 80-word sequence are labeled W (0), W (1),..., W(79). A single word buffer TEMP is also used.

To generate the message digest, the 16-word blocks M (1), M(2),..., M (n) above is processed in order. The processing of each M (i) involves 80 steps.

Before processing any block, the H's are initialized as follows: in hex,

H0 = 67452301

H1 = EFCDAB89

H2 = 98BADCFE

H3 = 10325476

H4 = C3D2E1F0.

Now M(1), M(2), ... , M(n) are processed. To process M(i), we proceed as follows:

- Divide M (i) into 16 words W(0), W(1), ... , W(15), where W(0) is the left-most word

- For t = 16 to 79 let

W (t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).

- Let A = H0, B = H1, C = H2, D = H3, E = H4.

- For t = 0 to 79 do

TEMP = S^5(A) + f (t; B, C, D) + E + W (t) + K (t);

E = D; D = C; C = S^30(B); B = A; A = TEMP;

- Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E

After processing M(n), the message digest is the 160-bit string represented by the 5 words H0 H1 H2 H3 H4.

CHAPTER 4

SECURITY OF AND ATTACKS ON ROUTE OPTIMIZATION

4.1 Security Requirements of Mobile IP

IP mobility support or Mobile IP enables a mobile node to change its attachment point on the Internet while maintaining its IP address as well as its network connectivity using this IP address. Due to this unique nature of Mobile IP protocol, there are additional security issues to be considered.

The general security goals for Mobile IP are: [16]

1. When a Mobile Node visits a foreign network, it deserves the same connectivity and safety as it is in the home network.
2. When a mobile node visits a foreign network, the foreign and home networks should be protected from both active and passive attacks.

To satisfy the above security goals, the following specific components (communication parts) need be secured.

1. The Mobile IP registration and location update message must meet the following security parameters: data integrity, data origin authentication and anti-replay protection.
2. When a MN visits a foreign network, access control mechanism must be provided to access resources of the foreign network.
3. The IP packet-redirecting tunnel must meet the following security parameters: data integrity, data origin authentication and data confidentiality.

Fulfilling the above security requirements will help us to avoid the following main attacks.

1. The possibility for an adverse node to spoof the identity of a mobile node and redirect the packets destined for the mobile node to other network locations,

2. The risks for potentially hostile nodes (coming from different network administrative domains) to launch passive/active attacks against one another when they use common network resources and services offered by a mobility supporting subnet

4.2 Attacks on Route Optimization

Since the mobile node has to constantly send binding update message to nodes communicating with it, route optimization creates additional vulnerability to attack. If we don't authenticate this message, malicious binding updates can be done which will open the door for intruders to perform different attacks.

The goal of the attacker can be to corrupt the correspondent node's binding cache and to cause packets to be delivered to a wrong address. This can compromise secrecy and integrity of communication and cause denial-of-service (DoS) both at the communicating parties and at the address that receives the unwanted packets. Some of the major attacks on route optimization are listed as below: [19]

1. Address Stealing: this is the most obvious danger in Mobile IPv6. The attacker illegitimately claims to be a give node at a given address and tries to steal traffic destined to that address. There are some variants of this attack.
    a. Basic address stealing: an attacker fabricates and sends spoofed binding update from anywhere in the internet. All correspondent nodes would become unwitting accomplices to this attack. The attacker can select a care of address to be either its own current address or another address. If the attacker selected a local care of address allowing it to receive the packets, it would be able to send replies to the correspondent nodes.

       The binding update authorization mechanism used in the Return Routable protocol is primarily intended to mitigate this threat and limit the location of attacker to the path between the correspondent node and home agent.

b. Address stealing of Stationary Nodes: This is where an attacker steals the address of a well known address like servers and sends traffic using the stolen address. The security design must take reasonable measures to prevent the creation of fraudulent binding cache entries in the first place.

c. Future Address Stealing: If an attacker knows an address that a node is likely to select in the future, it can launch a future address stealing attack. The attacker creates a Binding Cache Entry with the home address that it anticipates the target node will use. If the home agent allows dynamic home addresses, the attacker may be able to do this legitimately. That is, if the attacker is a client of the home agent and is able to acquire the home address temporarily, it may be able to do so and then return the home address to the home agent once the binding cache entry is in place

2. Attacks against Secrecy and Integrity: By spoofing binding updates, an attacker could redirect all packets between two IP nodes to itself. If A wants to send binding update for B. An attacker, by sending a spoofed binding update to A, could capture the data intended to B. That is, it could pretend to be B and highjack A's connections with B, or it could establish new spoofed connections. The attacker could also send spoofed binding updates to both A and B and insert itself in the middle of all connections between them (man-in-the-middle attack). Consequently, the attacker would be able to see and modify the packets sent between A and B.

   Strong end-to-end encryption and integrity protection, such as authenticated IPSec, can prevent all the attacks against data secrecy and integrity. When the data is cryptographically protected, spoofed binding updates could result in denial of service but not in disclosure or corruption of sensitive data beyond revealing the existence of the traffic flows.
3. By spoofing binding update message, an attacker could redirect all the packets sent between two IP nodes to a random address. The mitigation mechanism proposed is the use of return routability.

4. Replaying binding update: Here, an attacker may be able to replay recently authenticated binding updates to the correspondent nodes and, consequently, direct packets to the mobile node's previous location. In this case an attacker can create both denial of service attack and can capture information exchange.

   Though the above are the major once, there are some additional types of attack like flooding attack, inducing unnecessary binding update, refelection and amplification [19].

4.3 Securing Route Optimization

The IETF has designed a security mechanism for MIPV6 binding update. Since binding update is to be made from mobile node to home agent and correspondent nodes, securing binding updates both to the home agent and to the correspondent node will be considered.

The design made for securing binding update toward the home agent is IPSec that gives integrity and authentication service for binding update and acknowledgement packets. This has satisfied the security needs for binding update toward the home agent [13]. The IETF has designed another protocol called Return Routable protocol for securing binding update toward the correspondent agent [6].

As IPSec is considered to be secure enough and avoids most of the attacks over the internet, this thesis relied on its security. We will next analyze the return routable protocol designed to secure the binding update toward the correspondent node.

4.3.1 Return Routable Protocol

Return routable (RR) is a protocol designed to secure route optimization deployed in mobile IPv6 [19]. The basic idea on Return Routable Protocol is that the mobile node sending the binding update should be able to verify that there is a correspondent node that is able to respond to packets sent to a given address. It also requires that the responding node is the node to which the binding update is sent.

A. Operation of the RR Protocol

In RR protocol messages are exchanged between the mobile node and correspondent nodes in order to get a session key which will help in securing the system. The integrity and authenticity of the Binding Updates messages to correspondent nodes is protected by using a keyed-hash algorithm. The binding management key, Kbm, is established using return routable procedure by exchanging messages and it is used in key the hash algorithm for securing the binding update. The data exchange is accomplished by use of node keys, nonces, cookies, tokens, and certain cryptographic functions. [6]

Node keys: Each correspondent node has a secret key, Kcn, called the "node key", which it uses to produce the keygen tokens sent to the mobile nodes.

Cookies: The "home init cookie" and "care-of init cookie" are values sent to the correspondent node from the mobile node, and later returned to the mobile node. The home init cookie is sent in the Home Test Init message, and returned in the Home Test message. The care-of init cookie is sent in the Care-of Test Init message, and returned in the Care-of Test message.

Tokens: The "home keygen token" and "care-of keygen token" are values sent by the correspondent node to the mobile node via the home agent (via the Home Test message) and the care-of address (by the Care-of Test message), respectively.

When a mobile node wants to send a binding update, it sends home test initialization (HoTI) and care of test initialization (CoTI) messages to the correspondent node. The first message will be send through the home agent while the second directly to the correspondent node and messages will be sent simultaneously by the mobile node.

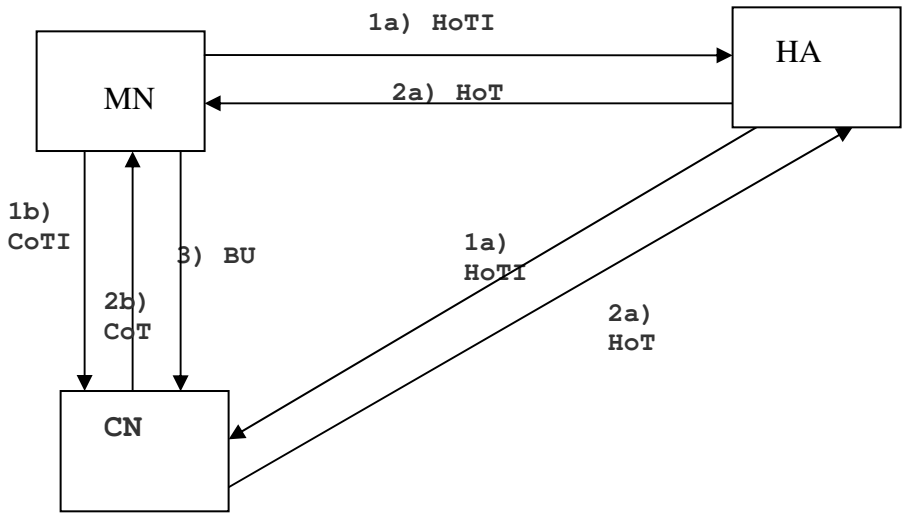


Fig 4.1 Return Routable Operation [19]

The content and function of each message is as described below:

HoTI = {*HoA, CN, Ch*} and CoTI = {*CoA, CN, Cc*}

Ch and Cc are cookies for matching responses initializations

The correspondent node will produce home keygen and care of keygen tokens

Home keygen token: =First (64, HMAC_SHA1 (Kcn, (home address | nonce | 0)))

and

Care-of keygen token: = First (64, HMAC_SHA1 (Kcn, (care-of address | nonce | 1)))

Once the keygen tokens are generated the correspondent node will produce the Home Test and Care Test messages.

HoT= {*HoA, CN, Ch, Hkn} and CoT*= {*CoA, CN, Ch, Ckn}*

where Hkn and Ckn are the home and care of tokens

The correspondent node sends these test messages to the mobile node. The CoT will be sent directly to mobile node while HoT will be sent through home agent. Once the mobile node receives the test messages, the return routable procedure is finished. To authorize a Binding Update, the mobile node creates a binding management key, Kbm, from the keygen tokens.

Kbm = SHA1 (Hkn|Ckn)

And produce the binding update message BU consisting of

BU= {CoA, CN, HoA, Seq#, i, j, MAC_{BU}}

Where - Seq# is sequence number within binding update

- i and j are home and care of nonce values
- MAC_{BU} is the binding update message authenticating code calculated as

MAC_{BU}= First (96, HMAC_SHA1 (Kbm, (CoA|correspondent| BU)))

Once the authentication code is produced the binding message will be send directly from the mobile node to the correspondent node.

B. Analysis of the Protocol

The binding update message is protected by the binding update key generated at the mobile node hashing the two test values. So the security of binding update message depends on the secrecy of the binding key. The secrecy of the binding key also depends on the home keygen token and care of keygen token. The IETF group tested and verified the security of binding update message assuming the two tokens are secured.

The motivation of the design of MIPv6 was to give sufficient security for mobile IP with out creating additional burdens on it. The working group clearly stated that the aim doesn't include protecting the path from CN to HA. The return routable protocol doesn't avoid attack made by intruders who can monitor the CN –HA path.

The working group stated that the CN-HA path is available even if the mobile node is at its home network. It is true that this path is available always as far as there is a communication between

the correspondent node and mobile node. But attacks which are to be protected using the return routable protocol can happen unless this path is protected while the mobile node is away from its home network. And using return routable security protocol will not satisfy our security requirement for the binding update and will not secure the communication between the correspondent agent and the mobile node.

Most of the message transfers in the RR protocol are not secured as seen in the table below.

Message	**Path**	**Protection**
HoTI	MN-HA	IPSec
HoTI	HA-CN	No
CoTI	MN-CN	No
HoT	CN-HA	No
HoT	HA-MN	IPSec
CoT	CN-MN	No

Table 1 Paths/Messages protected and not protected

Two of the messages i.e. exchanges between the home agent and the mobile node (HoT and HoTI) are protected using IPSec that exist between the two entities.

The paths from and to the correspondent node are free and there is no security mechanisms stated. So the messages HoTI and HoT (HA-CN) and CoTI and CoT (CN-MN) do not have protection mechanisms.

An intruder who wants to make attack can constantly listen to the path from HA to CN and monitor it. And this intruder can take the HoT from the Ipv6 with out much difficulty. The attacker can collaborate with another body called FMN (fake mobile node) or can make the attack alone. In case there is FMN, the attacker who listens to HA-CN path takes out HoT and

extracts the Ch and sends it to FMN. The FMN can produce CoTI and send to CN. CN will respond with CoT to FMN. Once FMN obtains the CoT, he can follow the valid procedure to send the binding update message.

Since there is no security considered for both paths (MN-CN and HA-CN) before the generation of the binding key, it is possible to get both the home test and care of test message. If one can get these two messages, the binding key could be easily calculated and further attacks will be very simple.

Another approach of securing the binding update of mobile IP is designed by IETF that is called Cryptographically Generated Address. Its working principle and limitations under it are discussed in related works of this paper.

CHAPTER 5

THE PROPOSED SYSTEM DESIGN AND IMPLEMENTATION

The main objective of the thesis being improving the security of Mobile IP communication by avoiding the flaws on security of route optimization we saw in the previous chapter, the thesis has gone through a number of design assumptions and issues. The security requirements of route optimization are restated and a design to satisfy the requirements is made and implemented for simulation.

5.1 Design assumptions and issues

Assumption: The assumptions that we took for our system are:

1. There is a security association between the home agent and the mobile node. They know and trust the other. Due to the trust relationship they have, home agent will not modify any packet from/to the mobile node illegally. This assumption is valid as it is based on ITEF's specification of Mobile IP.
2. It is possible to use public key infrastructure (PKI) for two communicating parties to get the public key of the other with out intervention. Nowadays PKI is used in different security systems. The security mechanisms designed for securing binding update toward the home agent [16] uses PKI. So we can take this assumption with out much concern.

Issues: There are some major issues that are considered while designing the protocol and a security enhancement in our system. These are the security, performance and scalability of the system.

1. ***Performance*:** The overall system (i.e. the protocol and its security) should work in a reasonable performance value i.e. the performance of Mobil IP communication should not decline due to adding our enhancement. We shouldn't put much burden on mobile nodes like PDA with relatively less resource. Mobile nodes usually have:
    a. less powerful CPU, less memory and disk space and etc.
    b. less physical security

c. limited battery power

In our case, since we are using IPSec for securing the path between the home agent and the mobile node, we should devise a very convenient means to avoid computational intensive operations on the mobile node.

2. ***Security:*** the other issue for our system design is security. The designed security mechanism should satisfy security requirements of the system. The system must be able to provide protection against most of the known attacks and reduce vulnerability.
3. ***Scalability:*** scalability and easier integration with other security modules is the other issue considered in the design phase presented on this thesis. This issue is especially important since the system designed is part of another system and if there are possibilities for modification or extension of a system in the future.

5.2 Security Requirements of Binding Update Message

Our aim is to have secured binding update toward the correspondent node. In our design as stated below the path to be secured is the HA-CN. And the important information that is going to be transmitted through this path is the binding update. So our security requirement can be stated with respect to the binding update message like "**The binding update should be current from real source with out modification"**

The implication of this is that the authentication and integrity of binding update message need to be kept and anti replay attacks have to be provided.

5.3 The Proposed System

This work modified the RR binding update protocol designed by IETF so that possible attacks on it could be reduced. Additional design is included to secure this modified protocol. In other words, the work in this thesis could be divided in to two parts; protocol modification and its security enhancement. Below is the detailed discussion of these tasks.

5.3.1 The protocol

There is no change made on the list of nodes that the MN will send binding update to or when the binding update will be sent. The modification made on this protocol is on the path used to forward the binding update message to the correspondent nodes and the security association that is made between the home agent and correspondent node.

A mobile node wishing to send binding update message will send the home agent and the correspondent node simultaneously. The binding update message could be sent in one of the following ways:

1. Sending the binding update to the correspondent node with one message: In this case binding update to the home and correspondent agents will be sent in one message by only inserting the correspondent nodes' address in the binding update message to the home agent. The home agent receiving the message and checking the field that contains the address of the correspondent node will forward the binding update to the correspondent node for which the binding update is intended.
2. Sending the binding update to correspondent nodes alone: in this case the binding update message toward the correspondent nodes will be sent alone through the home agent. The home agent receiving the binding update message will send it to the destination correspondent node.

Though the first way decreases the computation at the mobile node, it requires the modification of the existing binding update toward the home agent which is going to be very vast for implementation in this thesis. For this reason this work follows the second method in the implementation and testing to decrease the complexity of the implementation.

The protocol will use the IPSec protected path from mobile node to the home agent. As we stated in our security assumption, the home agent and mobile node are trusted and the path is secured using IPSec as stated earlier [16].

Once the mobile node sent the binding update message to the correspondent node, the home agent receiving the packets buffers update messages to correspondent node and sends it to the intended destination. The binding update message transfer is shown in the figure below.

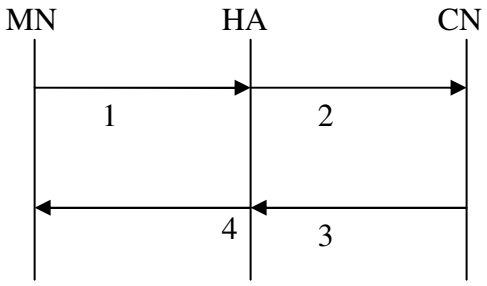


Fig. 5.1 Modified binding updates protocol toward CN

The protocol is depicted in Fig 5.1 above. The mobile node will send binding update to the home agent (message 1). The home agent will identify the type of packet to be binding update to the correspondent node and send it to correspondent node (message 2). When the correspondent node receives the binding update message, it will acknowledge the home agent by sending an acknowledgement message (message 3). The home agent will forward this message to the mobile node indicating the address of the correspondent node (message 4).

The mobile node and the home agent will have a reasonable time value to wait for acknowledgment. If acknowledgment message doesn't reach with in the expected period of time, they are obliged to send the binding update again as in other IP communication. Each packet transfer will have a sequence number field that increases by one which is used to identify the recent packet number transferred.

Packet structure:

The binding update message sent to home agent will contain a number of fields that are relevant for our message. The structure of the binding update packet when it is sent in different packet from the binding update to home agent is

Packet= {ha, coa, sadd, newcoa, cn, type, seq_no, packet_len, bl}

Where:

- ha is the home address
- coa is care of address
- sadd is source address of the message
- newcoa is new care of address
- cn is the correspondent node
- type is the type of message in the protocol
- seq_no is the sequence no of the message
- packet_len is length of the packet
- bl is binding life time

and the acknowledgement packet will have the following fields:

Packet= {cn, dadd, type, seq_no, packet_len}

Where:

- the dadd is the destination address of the acknowledgment message, which is care of address of mobile node
- cn is the source of the message (correspondent node)
- type is the type of message
- seq_no is sequence number of the response message
- packet_len is the length of the packet

5.3.2 Security of the Protocol

We use IP security (IPSec) to secure our binding update toward the correspondent node. We selected to use authentication header and transport mode of operation which is discussed in chapter three of this thesis document. Referring to our earlier discussion on Authentication Header of IPSec, we recall that it is possible to satisfy security requirements for the binding update messages. Using AH, we can authenticate that the binding update is from the valid source

and is not modified in the middle of transmission with the possibility to add anti-reply security options.

As is stated in our previous discussions on AH, it employs message authentication codes for achieving authentication and integrity of the message. In our system we used the hashed message authentication code (HMAC) and the hash algorithm selected is SHA1. We selected to use SHA-1 for its security over the MD5 which is computationally low. For key management part of the protocol we selected to use IKE which intern uses Diffie-Hellman algorithm. Using the Diffie-Hellman algorithm we can produce a shared key that will help us later for our message digest. The public keys of the home and correspondent agents are to be obtained from valid public key infrastructure however there is also a possibility to manually configure the keys [13].

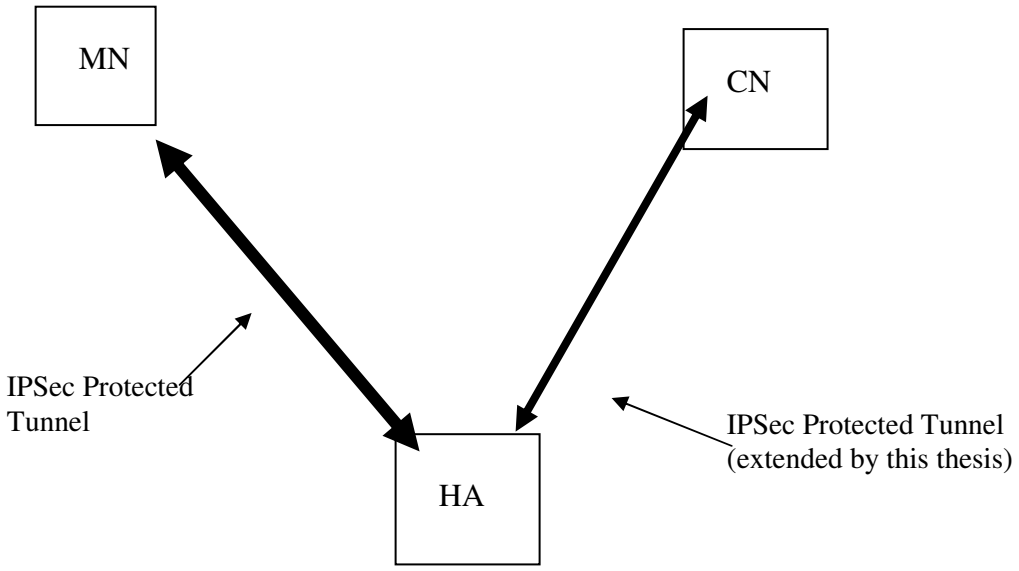


Fig.5.2 Security of the Binding Update Designed

5.3.3 Analysis of the Protocol

The protocol scheme is such that it traverses in one direction from the mobile node to the correspondent node through the home agent. In order to make binding update the mobile node,

once it creates a secured channel with the home agent, will send the binding update messages to the home agent. The home agent will have two secured channels one with the mobile node and the other with the correspondent node. Using the secured channel between the home agent and correspondent node, the home agent will send the binding update toward the correspondent node.

Based on this, it is possible to see that we didn't add any burden of computation on the mobile node. So we satisfied our performance issue related to the mobile node. In addition we can say that we satisfied the scalability issue in our design since we used IPSec for our security and IPSec is a widely known and used security for most systems over the internet.

The security of the system is described as follows. Using AH of IPSec we authenticate the sender of the message and check the integrity of the message sent. Once the shared key is calculated, we can use HMAC and calculate a hash value to make digital signature of the message. The signature will be sent with the authenticated message for verification at the recipient. The receiving node calculates a hash value for the message it received from the source. If the two signatures match, the receiver will understand that the message is from valid source. Other wise the message will be ignored.

The two digests will be equal if and only if the following two cases are satisfied:

1. there is shared key between the two ends
2. the message is not changed in transit

With these two points we can see that we satisfied the authentication and integrity of the binding update message. The protection against replay attacks is straight forward using packet sequence numbers in binding update messages.

The following table (table 2) shows the security and performance comparisons between the new protocol and the return routable protocol that are analyzed from the two designs.

Protocol	Security	Performance
Return Routable	- unprotected paths - attacks could occur	- 4 message exchanges before obtaining binding key - MN participates in key generation
New Protocol	No unprotected paths	- only 2 messages before key generation - the low resource MN is free from key calculation

Table 2: comparison between the new protocol and RR

5.4 Implementation of the System

In order to carry out some tests on the designed system, we have to implement it on a suitable language and platform. So we make a choice of the implementation language and implement it in a way that is convenient for simulation.

5.4.1. Implementation Tool

The simulation of the protocol and its security is done with Network Simulator 2 (NS2). Since NS2 is easily available, we have some previous experiences on it and suitable for our work and we choose it to use for our work. The work on NS2 consists of two major parts. The first is an enhancement of the protocol i.e. implementation of a new agent and the second is simulating the protocol.

NS2 uses an object oriented language that is used to simulate different network protocols. Its backend is written in C++ and its front end is OTCL. With otcl it is possible to configure different network parameters for a system. In order to add our protocol on NS2, we first explore some theoretical concepts of the NS simulator itself. Then specific concepts related to our system which includes adding new protocol and agent and how to integrate the work with NS2 were studied. For each work done we have seen different resources helpful for our work. The

following are the most valuable resources that are referred for implementing and simulating our system.

The NS Manual

The NS Manual is the full document for the ns2 [17]. It comprises of 46 chapters grouped in different categories. In the beginning chapters it describes the ns2 structure. It describes how ns2 works, interpretation of C++ the configurations done with OTCL. It also describes the importance of each of programming components in ns2.

The second part of this document comprises a wide variety subjects titled basics of simulator. It describes from the simple node configuration to the implementation of new protocol. It also describes some specific network types and their simulation in ns2. Third part of the document consists of a support for using ns2. The support includes debugging ns, mathematical support, tracing and monitors, ns code styles and others.

There are other additional documentations included in it like routing methods, different applications, emulation services, network animations.

Implementing New MANET Protocol in NS2

This is another resource for ns2 that is written in order to develop new protocol in ns2. It is specifically written for people who need developing their own routing protocol in ns2 [11]. In the beginning of the document, it describes the basics of ns2 and it begins on how to write a new protocol. It describes the steps in creating new agent giving a specific example throughout the explanation. It gives the important changes that are needed to be made on different files in ns2.

NS By Example

This resource is a different way of approach to teach ns2 simulator [14]. It discusses an issue on ns2 and gives examples on each of the points of discussion. It begins with simple simulation setup and discusses also post simulation analysis like trace analysis formats. The document also

describes extending ns2. It gives some references and guides to resources in its part titled where to find what.

5.4.2 The C++ Development of the Protocol

The C++ programming language is used in implementing the backend of NS2. Our implementation in C++ consists of different modules for the intended operation that is made in chapter 6. We can classify the implementation in to two basic parts. The first is the implementation of the protocol itself and the other is the implementation of the security part of the protocol. In the following we describe both of the implementations.

5.4.2.1 Implementation of the protocol

The protocol implementation has two parts. The first part of the implementation is the implementation of the packet structure. It basically consists of different variables and elements of the protocol that are used in the agent. It is implemented in a struct type like

```
hdr_mipbu {
        int ha;         //home address
        int coa;        // care of address
        –
        –
}
```

There are additional variables like packet type, offset, security components that are used for fulfilling the protocols functionalities.

The second part of the protocol implementation is the agent class. A class called **Mipbu()** is derived from the Agent base class. It consists of different functions that are needed for full operation of the protocol. The major functions in it include the following:

void recv (Packet *p, Handler *h)

It is a function that is inherited from the base class Agent (). The function basically guides the protocol in taking appropriate action when an agent receives packets. Specifically this function will be invoked when one of the nodes (mobile node, home agent or correspondent node)

receives any of the messages. Once the message is received, the agent checks the type of message and responds appropriately.

void send_bu ():

This function is used to send the binding update message to the home agent. Before it sends the packet, it will fill the necessary field in the packet. This function will be invoked when the binding time expires. The source address of the packet sent with this function will be the address of the mobile node and the destination address will be the home agent's address.

void send_bu_toCN ():

As discussed in the recv() function, when an agent receives a message, it checks the type of packet taking the packet type field from the packet structure. It will also check the source and destination address. If the packet type is BU_SOL (binding update solicitation), it will also check the source is the mobile node and the destination is towards the receiver (home agent). If both the requests match, the home agent will take the binding update information to itself. It will also replace the source and destination addresses of the packet, see the list of addresses to which the binding update message should be sent, set the packet type; and finally sends the packet toward the correspondent agent. This packet will be produced and sent by the **send_bu_toCN** () function**.**

void send_ack_toHA():

When the correspondent agent receives a binding update message, it will check the type of packet and source and destination addresses. If it found the packet to be binding update (BU_to_CN), the agent will produce an acknowledgment packet and set appropriate fields and reply to the home agent.

void send_ack_toMN ():

When the home agent obtains the acknowledgment from the correspondent agent, it will produce an acknowledgment to the mobile node. In the acknowledgment packet, it will show the list of correspondent nodes from which the acknowledgment is found. This will give information to the mobile node that which node obtained the binding update. For this simulation purpose we did only for one correspondent node. The packet will be sent to the mobile node. This is done with **send_ack_toMN ()** function**.**

Static class MIPBUHeaderClass: public PacketHeaderClass ()
This is a class that is used to bind the header packet header with other NS2. The implementation of this function with some modifications at the packet.h file in ns will enable the ns to know our packet header.
Static class MipbuClass: public TclClass ()
This class is used to bind our agent with NS2. Once this is implemented and some modifications are done on some of the tcl files [as in appendix II] we can get our Agent.

In both the binding update and acknowledgement packets we used a sequence number field to differentiate each message.

5.4.2.2 Implementation of the security

In the implementation of the security, we tried to avoid complication by simplifying some inputs without any effect on the security of the system. Generally the HMAC-SHA-96 is implemented based on its operation. The implementation consists of the base class called HMAC () with many functions and variables in it.

The implementation of the security is for the path between the home agent and the correspondent node. The home agent and the correspondent agent will produce a shared key when they want to make communication. When the home agent receives the binding update message from the mobile node, it will call the **Result(unsigned char data)** function**.** Data is part of header that is used to produce the signature being computed with the shared key.

Since HMAC operation has two appends and hashes, for its inner and outer hash operations, we implemented **innerhash(), outerhash(), append1(), append2()**. Each of the two functions perform the same operation except their input is different. For the SHA1 part of the security, **PadMessage(), ProcessMessageBlock () and CircularShift()** are implemented.

The home agent after producing the appropriate key value will apply the above functions and authenticate the binding update message to be transferred and will generate a signature for it. When the correspondent agent receives a binding update message, it will call **Result (unsigned char data)** and verify the signature.

In addition there are changes made on some files for the protocol to be known on the ns2 environment. The change is made both on codes written with C++ and TCL. All source codes including the changes on files are given on the Appendix II part of this paper.

CHAPTER 6: SIMULATION AND ANALYSIS OF RESULTS

After the implementation is done the system was tested for its performance and some of the security threats. Based on the tests made, the results are obtained in trace files and manipulated accordingly to calculate the required parameters.

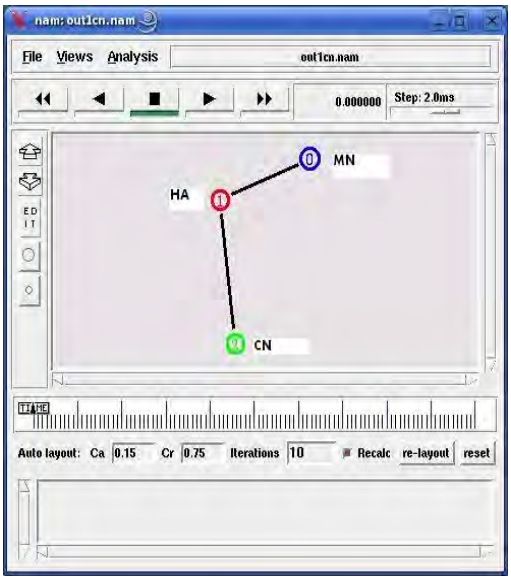
Results made from our tests were used for making analysis of our system. The analysis is made for both performance of the system and security.

6.1 Simulation Setup and Results

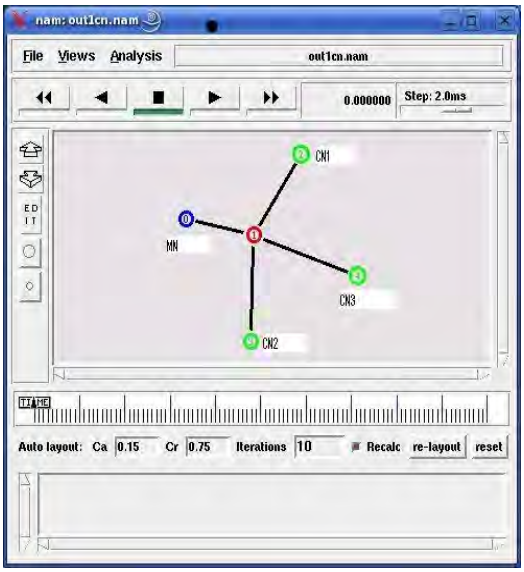
The performance parameters of our system were first selected. The basic performance parameters that we want to measure for our system are the sender to receiver packet delays and the packet processing time at each node. These values were calculated for both binding update and acknowledgment packets.

In order to test our system we set our basic simulation setup and simulation parameters. The setup consists of the basic components of our system which are the mobile node, the home agent and correspondent nodes which are interconnected based on a way that represents real environment. Once the setup is created simulation parameters were set and the number of correspondent nodes was varied as it could occur in real world situation.

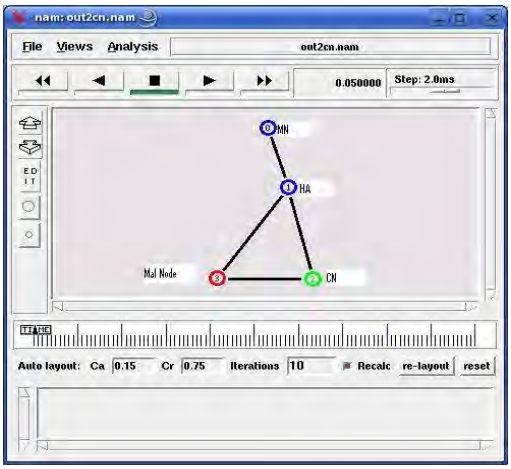
The simulation period was set to 240 seconds and the number of packet exchanges was 176. The packet size for binding update packets was set to 600 bytes and the acknowledgment packet size was set to 200 bytes on the original tcl code written to simulate the system. The number of correspondent nodes was made to one, two and three. The packet sizes were also changed and the average end to end delay is shown in figure 6.4. The figures below show the network topologies created for different test cases.



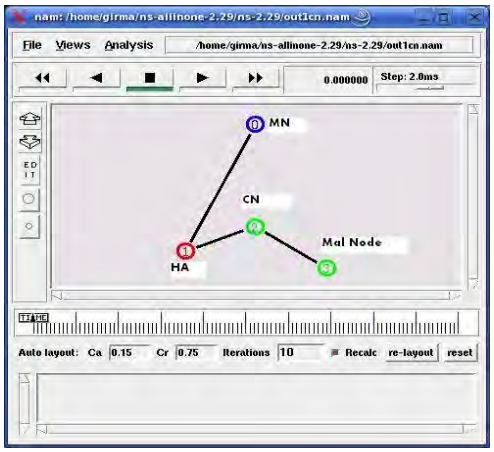
a)



b)



c)



d)

Fig 6.1 Simulation setups a) basic simulation CN=1

b) CN=3

c) When malicious node spoof, change and send BU to CN

d) Malicious node producing and sending BU to HA

The Fig in 6.1c shows when an attacker spoofs a BU message from HA and modifies and sends to CN and Fig in 6.1d shows when an attacker itself produces and sends BU directly to CN.

TCL code [appendix III) was written and modified for each of the cases and was made to run for the simulation period. The trace files that are generated after simulation were analyzed to obtain the required performance parameters. The results obtained for performance parameters are shown in the tables below. In the results we saw that a case where the output is divisive in the average delay decreases when the number of nodes increase, but since the change is very small we can pass it as doesn't affect our overall conclusion.

	Binding	**Acknowledging**
Nodes	3	3
Packet size	600 bytes	200 bytes
Average Delay (MN-HA)	0.0101 sec	0.0101 sec
Average Delay (HA-CN)	0.0137 sec	0.0116 sec
Average process at HA	0.75 sec	0.75 sec
Average process at CN	0.748	-

Table 3 Result of simulation for CN=1

	Binding	**Acknowledging**
Nodes	4	4
Packet size	600 bytes	200 bytes
Average Delay (MN-HA)	0.0101 sec	0.0086 sec
Average Delay (HA-CN)	0.0116 sec	0.0116 sec
Average process on HA	0.753 sec	0.75 sec
Average process at CN	0.75 sec	-

Table 4 Result of simulation for CN=2

	Binding	**Acknowledging**
Nodes	5	5
Packet size	600 bytes	200 bytes
Average Delay (MN-HA)	0.0101 sec	0.0134 sec
Average Delay (HA-CN)	0.0137 sec	0.0116 sec
Average process on HA	0.75 sec	0.751 sec
Average process at CN	0.749 sec	-

Table 5 Result of simulation for CN=3

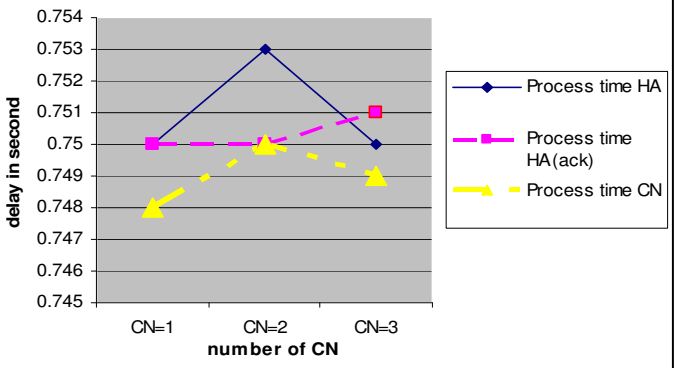
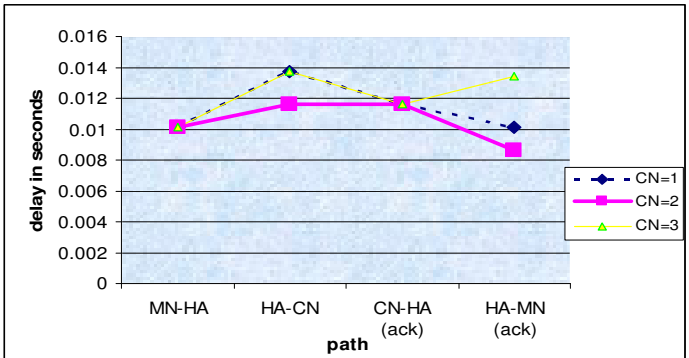


Fig 6.3 Average Packet Processing Delay



6.2 Average packet delay for varying CN

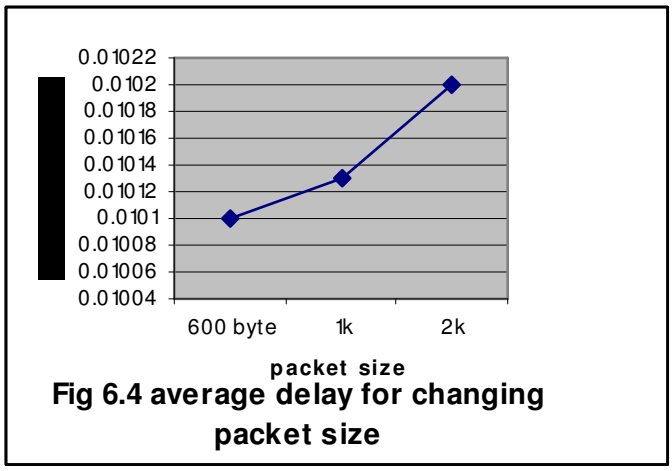


Fig 6.4 average delay for changing packet size

To test the security of the protocol designed, the above simulation parameters were taken and the topology of the simulation was changed as shown in figure 6.1 c. The setup represents a case were attacker spoof traffic from the home agent to the correspondent node. The attacker after modifying some of the fields that it needs, sends the modified binding update message to the correspondent node. In test case 176 BU packets were sent modified in the middle and this test is conducted for four different modifications. A simulation case where an attacker itself produces a fake binding update message and sends directly to the correspondent node is also made as in Fig 6.1 d and results found are the same as in Fig 6.1c.

The result that was observed in simulating the security of the system was the number of acknowledgement packets from the correspondent node to the source of the packet. All binding update messages from the home agent to the correspondent nodes were all acknowledged. There was no acknowledgement for all the packets that were sent from malicious node to the correspondent nodes. Numerically the 176 binding update packets from home agent to the correspondent nodes were fully acknowledged while there were no acknowledgement messages for binding update messages sent from malicious nodes.

Test type	**No of BU packets sent**	**No of ack. packets**
BU from HA	176	176
BU from Malicious Node	176	0

Table 6 Number of acknowledgement packets replied

6.2 Analysis of the Results

Verification for system security and evaluation of system performance are done based on the results from the simulation.

In order to verify the security of the protocol, acknowledgment packets were examined to check the integrity and authentication of binding update messages. When there were no malicious nodes inserted into the system, all binding update packets were acknowledged by correspondent nodes. For each valid binding update message sent from home agent, the correspondent node acknowledges the reception of such message. But when malicious node sends binding update message to the correspondent node, there were no acknowledgments from the correspondent nodes. This is because the correspondent node knows that either the message is not from authenticated source or the message is modified in transit. For this reason we can deduce that our design achieved security requirements of our system.

In order to evaluate the performance of our system, we take the nam output obtained from our simulation as discussed above. The average end to end packet delay on the system designed was found to be 0.0122 seconds. And the packet processing time at the nodes is 0.75 seconds.

For comparison purpose we took the average packet delay over the network and average processing time of packets at nodes for base mobile IP network. The values obtained are 0.014 sec for delay and 0.55 sec for packet processing at nodes. The average end to end delay of our system was found to be less than the end to end delay for the base Mobile IP while the packet processing time on nodes is less in the base Mobile IP system than in the current one. The delay in processing time at each node in our system could come because we use IPSec for producing message digest for securing our system. The figure below shows the comparisons graphically.

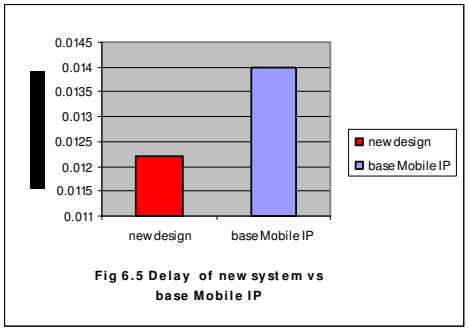


Fig 6.5 Delay of new system vs base Mobile IP

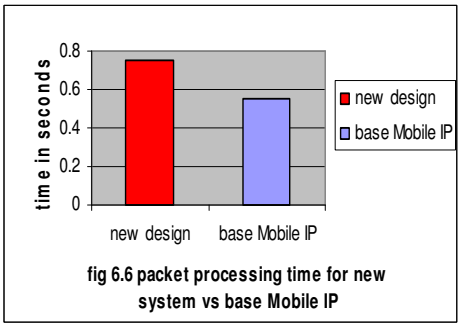


fig 6.6 packet processing time for new system vs base Mobile IP

CHAPTER 7: CONCLUSION AND FUTURE WORKS

7.1 Conclusion

In our analysis we have observed that there were no acknowledgments packets seen for packets which are unauthenticated and whose integrity is not realized. In addition in our simulation the packet delay over the network and processing time at the nodes were close to the time taken by when simulating equivalent systems.

The security flaw that was observed in the design of return routable protocol is avoided by the new design made. We satisfied our security requirement we set for mobile IP binding update to the correspondent node. With these all issues we understand that we improved security of mobile IP particularly security of binding update and our design's performance is equitable when compared with other part of Mobile IP operation.

7.2 Future Work

This work has implementation for simulation purpose. The simulation is done very well but it lacks integration with the base Mobile IP. So to simulate all the features of mobile IP including the binding update sub protocol at a time, it is necessary to integrate it with mobile IP base network on ns2. Therefore the recommended feature work here is integrating this work with the base Mobile IP.

Reference:

1. Atsushi Inoue, Masahiro Ishiyama, Atsushi Fukumoto and Toshio Okamoto, “Secure Mobile IP Using IP Security Primitives”, Communication and Information Systems Research Laboratories, Toshiba Corporation, R&D Center, 1997
2. C. Perkins Ed., "IP Mobility Support for IPv4", RFC 2002, IBM, October 1996
3. C. Perkins Ed., "IP Mobility Support for IPv4", RFC 3344, Nokia Research Center, August 2002
4. C. Kaufman, Ed., "The Internet Key Exchange (IKE)", RFC 4306, Microsoft, December 2005
5. Charles E. Perkins, "Mobile IP", IEEE Communications Magazine, May 1997 , pp 84-99
6. D. Johnson, C. Perkins and J. Arkko," Mobility Support in IPv6", RFC 3775, Rice University, Nokia Research Center and Ericsson , June 2004
7. D.Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, Motorola and Cisco, September 2001
8. Donald L. Evans, Philip J. Bond and Arden L. Bement , "The Keyed-Hash Message Authentication Code", FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, Information Technology Laboratory National Institute of Standards and Technology, March 2002
9. Dr. Walter L. McKnight, What is information Assurance? , Journal of Defense Engineering, June 2002
10. E. Rescorla ,"Diffie-Hellman Key Agreement Method", RFC 2631, RTFM Inc, 1999
11. Francisco J. Ros and Pedro M. Ruiz, "Implementing a New Manet Unicast Routing Protocol in NS2", Dept. of Information and Communications Engineering University of Murcia, Report, December 2004
12. Gloria Tuquerres, Marcos Rogério Salvador and Ron Sprenkels, "Mobile IP: Security and Applications", University of Twente, Report, December 1st,1999

13. J. Arkko and V. Devarapalli, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents", RFC 3776, June 2004
14. Jae Chung and Mark Claypool,"NS by Example", Worcester Polytechnic Institute, 2004
15. Jochen H. Schiller, "Mobile Communication", University of Karlsruhe Institute of Telematics, 1999
16. John Zao, Joshua Gahm, Gregory Troxel, Matthew Condell, Pam Helinek, Nina Yuan, Isidro Castineyra, Stephen Kent, " A public- key based secure mobile IP", Wireless Networks, Volume 5 Issue 5, October 1999
17. Kevin Fall and Kannan Varadhan (Editors), "The NS Manual", UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 2007
18. Kuang-Yeh Wang and C. Perkins, “Optimized Smooth Handoffs in Mobile IP”, Sun Microsystems Inc & University of Maryland, 1999
19. P. Nikander, J Arkko, T. Aura, G. Montenegro and E. Nordmark , " Mobile IP Version 6 Route Optimization Security Design Background", RFC 4225, December 2005
20. R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321, April 1992
21. R. Atkinson, “Security Architecture for the Internet Protocol”, RFC 1825, Naval Research Laboratory, August 1995
22. S. Kent and R. Atkinson ," Security Architecture for the Internet Protocol", RFC 2401, November 1998
23. S. Kent and R. Atkinson," IP Authentication Header", RFC 2402, November 1998
24. S. Kent and R. Atkinson," Encapsulation Security Payload", RFC 2406, November 1998
25. Salem Itani ,”Use of IP Sec in Mobile IP”, Department of Electrical and Computer Engineering, the American University of Beirut, Term Paper, May 21, 2001.
26. Torsten Barun and Marc Danzeisen,"Access to Mobile IP Users to Firewall Protected VPNs", Institute of Computer Science and Applied Mathematics, Report, University of Bern, Report, 2001

27. T. Aura," Cryptographically Generated Addresses (CGA)", RFC 3972, Microsoft Research, March 2005

28. W. Victor Maconachy, Corey D. Schou, Daniel Ragsdale and Don Welch, "A Model for Information Assurance: An Integrated Approach", United States Military Academy, June 2001

29. How Mobile IP works, http://www.gigaport.nl/netwerk/access/ta/mip/en_mobileip.html, website visited on June 2007.

Appendix I Source code for our system

```
/******************   mipbu.h   *************/

// Done for testing IPSec integration on Mobile IP route optimization
// It is for security the binding update message sent from mobile node to
//Corresponedent nodes
// Permission to use, copy, modify, and distribute this software and
//by Girma Kassa 2006/2007

#define JITTER                                          1.5

#ifndef ns_mipbu_h
#define ns_mipbu_h
#include "/home/girma/ns-allinone-2.29/ns-2.29/common/packet.h"
#include <assert.h>
#include "/home/girma/ns-allinone-2.29/ns-2.29/common/agent.h"
#include "/home/girma/ns-allinone-2.29/ns-2.29/classifier/classifier-addr.h"
#include "/home/girma/ns-allinone-2.29/ns-2.29/classifier/classifier-port.h"
#include "/home/girma/ns-allinone-2.29/ns-2.29/common/timer-handler.h"
#include "/home/girma/ns-allinone-2.29/ns-2.29/tools/random.h"
#include "/home/girma/ns-allinone-2.29/nam-1.11/address.h"
#include "/home/girma/ns-allinone-2.29/ns-2.29/trace/trace.h"

#define MIP_BU_TIMER          1
#define MIP_TIMER_AGTLIST     1

/*  cache ADS until MH receives REG REPLY -> new COA */
#define AD_CACHE_SIZE 10 /* TS_JCW */
#define BU_PACKET_SIZE 600
#define BU_ACKPACK_SIZE 200

typedef enum{
BU_SOL,BU_to_CN,ACK_HA_FromCN,ACK_MN_fromHA
}BUMesType; /// the binding update message type

struct hdr_mipbu {
      //int haddr_;
      int ha;           //home address
      int coa;          //care of address
      int s_add;        //source address of the msg
      int newcoa; //new care of address
      int cn;           // when acknowledgement shows from which cn , when bu
shows to which cn
      BUMesType type;   // type of the msg
      int seq_no;
      int packet_len;
      unsigned char hash[3];
      int testhash;  //testing
```

```
// Header access methods
      static int offset_; // required by PacketHeaderManager
      inline static int& offset() { return offset_; }
      inline static hdr_mipbu* access(const Packet* p) {
            return (hdr_mipbu*) p->access(offset_);
      }
};

class Mipbu;
class BUTimer : public TimerHandler {
 public:
        BUTimer(Mipbu *a) : TimerHandler() { a_ = a; }
 protected:
        virtual void expire(Event * e);
        Mipbu *a_;
};

class Mipbu:public Agent{
public:

      friend class BUTimer;
      nsaddr_t ra_addr_;
       Mipbu();
      ~Mipbu();
      int seqnumber;
      void reset_mipbu_bu_timer();
      void send_bu();
      unsigned char hash1[3];
      int command(int argc, const char*const* argv);
      void recv(Packet *p,Handler *h);
      void send_bu_toCN(int aadd);
      void send_ack_toHA(int acksaddr);
      void send_ack_toMN();

protected:
      void timeout(int);
      inline nsaddr_t& ra_addr() { return ra_addr_; }
      int m_node;
      int cn_agent;
      int ha_agent;
      int co_address;
      int new_coa;
      int bu_no;
      int ack_no;
      int ackmn_no;
      int cnaddress;
      int haaddress,coaaddress;
      unsigned char newcoaaddress;
      BUTimer bu_timer;
      void get_digest(unsigned char input);
```

```
      bool check_integrity(unsigned char hash1[3],unsigned char hash2[3]);
      Trace* logtarget; //for trace file generation
      //int PortClassifier* dmux_;  // for passing packets up to agents
      int get_dd(int k);
      int check(int y);
};

#endif

/******************   mipbu.cc   *************/

////////////////////////////////////////////////////////////////
///binding update done for testing the design proposed by /////
///the thesis improving security of mobile IP communication ////
/// by Girma Kassa 2006/2007                                  ////
////////////////////////////////////////////////////////////////

#include "mipbu_sec"
#include "mipbu.h"
#include <float.h>
#include <iostream>

int hdr_mipbu::offset_;
static class MIPBUHeaderClass : public PacketHeaderClass {
public:
      MIPBUHeaderClass() : PacketHeaderClass("PacketHeader/MIPBU",
                                  sizeof(hdr_mipbu)) {
            bind_offset(&hdr_mipbu::offset_);
      }
} class_miphdr;
 static class  MipbuClass  : public TclClass {
 public:
     MipbuClass() : TclClass("Agent/Mipbu") {}
     TclObject* create(int argc, const char*const* argv) {
          return (new Mipbu());

     }
 } class_mipbu;


Mipbu::Mipbu():Agent(PT_MIPBU),m_node(0),ha_agent(1),cn_agent(2),co_address(3
),new_coa(4),bu_timer(this)
{
      //ra_addr_ = id;
}
Mipbu::~Mipbu()
{
}

int Mipbu::command(int argc,const char*const* argv){
      if (argc==2)
      {
```

```
            if(strcasecmp(argv[1],"start")==0){
                  timeout(0);

            return TCL_OK;
            }

      }
      return (Agent::command(argc, argv));
}
/*
**********************************************************
      differnet implementations related to time
**********************************************************
*/

void Mipbu::timeout(int)
{
        send_bu();
        bu_timer.resched(0.05);
}


void BUTimer::expire(Event* e)
{

      a_->timeout(0);
}
/*
**********************************************************
      differnet operations and functions that
     are going to implemented by the Mobile Agent
**********************************************************
*/
void Mipbu::recv(Packet* p,Handler*) //when MN recieves message
{
      hdr_mipbu *miph=hdr_mipbu::access(p);           //set mip header
      hdr_ip *ih=hdr_ip::access(p);
      hdr_cmn *th = hdr_cmn::access(p);
      assert(ih->sport() == RT_PORT);
      assert(ih->dport() == RT_PORT);
      HMAC te;
      switch (miph->type){
      //cout<<"Recieving"<<endl;
      case BU_SOL:

            if (ih->daddr()==1 && ih->saddr()==0)
            {
            cnaddress=miph->cn;
            haaddress=miph->ha;
            coaaddress=miph->coa;
            newcoaaddress=miph->newcoa;
```

```
            bu_no=miph->seq_no;
            send_bu_toCN(2);
            //send_bu_toCN(3);
            //send_bu_toCN(4);
            }
            break;

      case BU_to_CN:          //binding update from HA using IPSec

            int ll;
            ll=ih->daddr();
            if (ih->saddr()==1)
            {
            newcoaaddress=(unsigned char)miph->newcoa;
            //get_digest(newcoaaddress);
            unsigned char hashcn[3];
            te.Result(newcoaaddress); // get digest at the recieving end
            for (int i=0;i<3;i++)
            {
                  hashcn[i]=(unsigned char)te.message_digest_array[i];
            }
            if (check_integrity(hashcn,miph->hash)) //check if integrity is
achieved
            {

                  send_ack_toHA(ll);
            }


            }
            break;
      case ACK_HA_FromCN:                 //Acknowledgement from CN in IPSec
case

            if (ih->daddr()==1 && ih->saddr()==2)
            {
            ackmn_no=miph->seq_no;
            cnaddress=miph->cn;
            send_ack_toMN();              //acknowledge MN
            }
            if (ih->daddr()==1 && ih->saddr()==3)
            {
            ackmn_no=miph->seq_no;
            cnaddress=miph->cn;
            send_ack_toMN();
            }
            break;
      case ACK_MN_fromHA:     //if message is acknowledgement do nothing
      //    do_nothing();
      //cout<<8<<endl;
            break;
      default:
```

```
            Packet::free(p);
            break;
      }
}

void Mipbu::send_bu()
{
      Packet* p=allocpkt();         //alocket packet
      hdr_ip *iph=HDR_IP(p);  //header
      hdr_cmn* hdrc=HDR_CMN(p);
      hdr_mipbu *miph=hdr_mipbu::access(p);           //set mip header

      //set common header
      hdrc->ptype()=PT_MIPBU;
      hdrc->size()=BU_PACKET_SIZE;
      hdrc->addr_type()=NS_AF_INET;
      hdrc->direction()=hdr_cmn::DOWN;
      hdrc->next_hop()=IP_BROADCAST;

      //set IP header
      iph->saddr()=0;
      iph->daddr()=1;
      iph->sport()=port()
      iph->dport()=port()

      // set MIPBU header
      miph->type=BU_SOL;
      miph->ha=ha_agent;                              //set home address
      miph->coa=co_address;                                    //set care
of address
      miph->newcoa=new_coa;                                 //set new care of
address
      miph->cn=cn_agent;
      miph->seq_no=seqnumber;

      seqnumber++;
       Scheduler::instance().schedule(target_, p, JITTER);
}
/*
*********************************************************
            differnet operations and functions that
            are going to implemented by the home agent
*********************************************************
*/

void Mipbu::send_ack_toMN()
{
      Packet* p=allocpkt();
      hdr_ip *iph=HDR_IP(p);  //header
      hdr_cmn* hdrc=HDR_CMN(p);
      hdr_mipbu *miph=hdr_mipbu::access(p);           //set mip header
```

```
	iph->saddr()=1;
	iph->daddr()=0;                      //set destination later
	iph->sport()=port();
	iph->dport()=port();

	hdrc->size()=BU_ACKPACK_SIZE;
	hdrc->ptype()=PT_MIPBU;

	miph->type=ACK_MN_fromHA;
	miph->cn=cnaddress;
	miph->seq_no=ackmn_no;

	ackmn_no++;
	Scheduler::instance().schedule(target_, p, JITTER);

}

void Mipbu::send_bu_toCN(int aadd)
{
	HMAC tt;
	Packet *p=allocpkt();          //allocate new packet
	hdr_cmn *hdrc=HDR_CMN(p);
	hdr_ip *iph=HDR_IP(p);
	hdr_mipbu *miph=hdr_mipbu::access(p);

	hdrc->ptype()=PT_MIPBU;
	hdrc->size()=BU_PACKET_SIZE;
	hdrc->addr_type()=NS_AF_INET;
	hdrc->direction()=hdr_cmn::DOWN;
	hdrc->next_hop()=IP_BROADCAST;

	iph->saddr()=1;
	iph->daddr()=aadd;
	iph->sport()=port();
	iph->dport()=port();

	miph->type=BU_to_CN;
	miph->ha=haaddress;                              //set homke address
	miph->coa=coaaddress;                                    //set care
of address
	miph->newcoa=newcoaaddress;                              //set new
care of address
	miph->cn=cnaddress;
	miph->seq_no=bu_no;

	tt.Result(newcoaaddress);

	for (int i=0;i<3;i++)
	{
		miph->hash[i]=(unsigned char)tt.message_digest_array[i];
```

```
      }

      bu_no++;
      Scheduler::instance().schedule(target_, p, JITTER);
}

/*
**********************************************************
      differnet operations and functions that
      are going to implemented by the Corresponednt agent
**********************************************************
*/
void Mipbu::send_ack_toHA(int acc)
{

      Packet* p=allocpkt();
      hdr_ip *iph=HDR_IP(p);  //header
      hdr_mipbu *miph=hdr_mipbu::access(p);           //set mip header
      hdr_cmn* hdrc=HDR_CMN(p);

      iph->saddr()=2;
      iph->daddr()=1;
      iph->sport()=port();
      iph->dport()=port();

      miph->type=ACK_HA_FromCN;
      hdrc->size()=BU_ACKPACK_SIZE;
      hdrc->ptype()=PT_MIPBU;
      miph->seq_no=ack_no;

      ack_no++;
      Scheduler::instance().schedule(target_, p, JITTER);
}

/*
**********************************************************
      Obtain the message digest for sender & receiver
**********************************************************
*/
void Mipbu::get_digest(unsigned char input)
{
      HMAC  ipsec;
      ipsec.Result(input);
      for(int i=0;i<3;i++)
      {
            hash1[i]=(unsigned char)(ipsec.message_digest_array[i]);
            cout<<hash1[i]<<endl;
      }
//    return hash1;
}
/*
**********************************************************
```

```
The function checks the integrity, authentication of the message
*********************************************************
*/
bool Mipbu::check_integrity(unsigned char hasha[3],unsigned char hashb[3])
{
      bool equality;
      int check[3];
      for (int i=0;i<3;i++)
      {
            if (hasha[i]==hashb[i])
            {
                  check[i]=1;
            }
            else
            {
                  check[i]=0;
            }
      }
      if(check[0]==1 && check[1]==1 && check[2]==1)
      {
            equality=true;
      }
      else
      {
            equality=false;
      }
      return equality;

}
////////////////////////////////////////////////
//
//              for test
////////////////////////////////////////////////
int Mipbu::get_dd(int k)
{
      return k*4;
}

int get_dd(int k)

{
      if (k==9){
            return 1;
      }
      return 0;
}

/*****************  mipbu_sec *************/

////////////////////////////////////////////////////////////////////
///
///  IPSec implementation header
```

```
//////////////////////////////////////////////////////////////////

//MAC(text)t = HMAC(K, text)t = H((K0 Ã… opad )|| H((K0 Ã… ipad) || text))

/*
produce opad and ipad
ipad= '36' repeated B times
opad= '5c' repeated B times
B is input size to the hash function
1. produce K0
2.K0 XOR with ipad
3.append result 2 with text to be hashed
4. hash result 3
5. K0 XOR with opad  // symbol for XOR is ^
6. append result 4 to result 5
7. hash result 6
8. output the result
*/

class HMAC
{
public:
      HMAC();
      ~HMAC();
      unsigned char appended1[65];        //inner append
      unsigned char appended2[69];        //outer append
      int inner;
      int outer;
      void Reset();
      void Result(unsigned char data);
      unsigned  message_digest_array[5];
      unsigned char data;
protected:
      unsigned int k_ipad[64];    /* inner padding - * key XORd with ipad */
//consider size
      unsigned int k_opad[64];    /* outer padding -* key XORd with opad
*///consider size
      void set_values();

      unsigned int inner_hash_value[5];  // data type to be corrected
      unsigned int outer_hash_value[5];  // data type to be corrected

      void append1();//unsigned ,unsigned );
      void append2();//unsigned ,unsigned );
      void innerhash();//unsigned char);
      void outerhash();//unsigned char);
      void ProcessMessageBlock();

      void PadMessage(unsigned char message[],int length);
      inline unsigned CircularShift(int bits, unsigned word);
      unsigned H[5];                                  // Message digest
buffers
```

```
      unsigned char Message_Block[64];    // 512-bit message blocks
      int Message_Block_Index;      // Index into message block array
      unsigned char message_length[8];
      unsigned char inner_hash_input[15]; // length of input to the inner
hash
      unsigned char outer_hash_input[23]; // length of input to the outer
hash
      int inner_length;
      int outer_length;
      int hash_input_length; //parameter two of padmessage

};

HMAC::HMAC()
{
      Reset();
      set_values();
}
HMAC::~HMAC()
{
}

//************************************************
//          start out by storing key in pads
//************************************************
void HMAC::set_values()
{

        char
key[64]="988845CDEBABBCDE51CB26D87E942F14988845CDEBABBCDE51CB26D87E942F1";

        memset( k_ipad,0,sizeof k_ipad);
        memset( k_opad,0,sizeof k_opad);
        memcpy( k_ipad, key, 64);
        memcpy( k_opad, key, 64);

    // XOR key with ipad and opad values
        for (int i=0; i<64; i++) {
                k_ipad[i] ^= 0x36;
                k_opad[i] ^= 0x5c;
        }

}

void HMAC::Result(unsigned char data)
{
      HMAC::outerhash();
}


//************************************************
//    functions in inner hash
```

```
//************************************************
void HMAC::append1() //message padding for the first hash

{
      int i,j;

      for (i=0;i<64;i++)
      {
            appended1[i]=(unsigned char)k_ipad[i]; //copy inner pad
      }

            appended1[i]=data;      // append text

      // message input to hash in our case let to be a 90 bits or 15 byte

      for (j=0;j<15;j++)
      {
            inner_hash_input[j]=appended1[i];
            i=i-1;

      }
      inner_length=15;
}
void HMAC::innerhash()
{

      HMAC::append1();
      HMAC::PadMessage(inner_hash_input,inner_length);

      for(int i = 0; i < 5; i++)
      {
            inner_hash_value[i] = H[i];
      }
}

//************************************************
//    functions in outer hash
//************************************************

void HMAC::append2()
{
      HMAC::innerhash();
      int i,j;
      for (i=0;i<64;i++)
      {
            appended2[i]=(unsigned char)k_opad[i];
      }
      int inerhash_index=0;
      for (i=64;i<69;i++)
      {
            appended2[i]=(unsigned char)inner_hash_value[inerhash_index];
            inerhash_index=inerhash_index + 1;
```

```
      }
      // we take only 184 bits (23 bytes )of the appended message begning
from outer
      for (j=0;j<23;j++)
      {
            outer_hash_input[j]=appended2[i];
            i=i-1;
      }
            outer_length=23;
}

void HMAC::outerhash()
{

      HMAC::append2();
      HMAC::PadMessage(outer_hash_input,outer_length);
 //   return outer_hash;
            for(int i = 0; i < 5; i++)
      {
            message_digest_array[i] = H[i];
      }

}
//************************************************
//    some functions and constants in hash algorithm
//************************************************

void HMAC::Reset()
{
      Message_Block_Index     = 0;

      H[0]        = 0x67452301;
      H[1]        = 0xEFCDAB89;
      H[2]        = 0x98BADCFE;
      H[3]        = 0x10325476;
      H[4]        = 0xC3D2E1F0;

}


void HMAC::ProcessMessageBlock()
{
      const unsigned K[] =    {                     // Constants defined
for SHA-1
                                                0x5A827999,
                                                0x6ED9EBA1,
                                                0x8F1BBCDC,
                                                0xCA62C1D6
                                          };
      int         t;                                  // Loop counter
      unsigned    temp;                         // Temporary word value
      unsigned    W[80];                              // Word sequence
```

```
        unsigned    A, B, C, D, E;                      // Word buffers

        /*
         *    Initialize the first 16 words in the array W
         */
        for(t = 0; t < 16; t++)
        {
              W[t] = ((unsigned) Message_Block[t * 4]) << 24;
              W[t] |= ((unsigned) Message_Block[t * 4 + 1]) << 16;
              W[t] |= ((unsigned) Message_Block[t * 4 + 2]) << 8;
              W[t] |= ((unsigned) Message_Block[t * 4 + 3]);
        }

        for(t = 16; t < 80; t++)
        {
           W[t] = CircularShift(1,W[t-3] ^ W[t-8] ^ W[t-14] ^ W[t-16]);
        }

        A = H[0];
        B = H[1];
        C = H[2];
        D = H[3];
        E = H[4];

        for(t = 0; t < 20; t++)
        {
              temp = CircularShift(5,A) + ((B & C) | ((~B) & D)) + E + W[t] +
K[0];
              temp &= 0xFFFFFFFF;
              E = D;
              D = C;
              C = CircularShift(30,B);
              B = A;
              A = temp;
        }

        for(t = 20; t < 40; t++)
        {
              temp = CircularShift(5,A) + (B ^ C ^ D) + E + W[t] + K[1];
              temp &= 0xFFFFFFFF;
              E = D;
              D = C;
              C = CircularShift(30,B);
              B = A;
              A = temp;
        }

        for(t = 40; t < 60; t++)
        {
              temp = CircularShift(5,A) +
                        ((B & C) | (B & D) | (C & D)) + E + W[t] + K[2];
              temp &= 0xFFFFFFFF;
```

```
            E = D;
            D = C;      Message_Block[Message_Block_Index++] = 0x80;
            C = CircularShift(30,B);
            B = A;
            A = temp;
      }

      for(t = 60; t < 80; t++)
      {
            temp = CircularShift(5,A) + (B ^ C ^ D) + E + W[t] + K[3];
            temp &= 0xFFFFFFFF;
            E = D;
            D = C;
            C = CircularShift(30,B);
            B = A;
            A = temp;
      }

      H[0] = (H[0] + A) & 0xFFFFFFFF;
      H[1] = (H[1] + B) & 0xFFFFFFFF;
      H[2] = (H[2] + C) & 0xFFFFFFFF;
      H[3] = (H[3] + D) & 0xFFFFFFFF;
      H[4] = (H[4] + E) & 0xFFFFFFFF;

      Message_Block_Index = 0;
}

//*************************************************
//    pad message to make block size
//*************************************************

void HMAC::PadMessage(unsigned char message[],int hash_input_length)
{
      Message_Block_Index=0;

      int i=0;
      for (int
Message_Block_Index=0;Message_Block_Index<hash_input_length;Message_Block_Ind
ex++)
      {
            Message_Block[Message_Block_Index] = message[i];

            i++;
      }
      //s=hash_input_length;
      char hexstring[2];
      itoa(hash_input_length,hexstring,16);//convert length to hex value */
//input length to the values to the array
      message_length[7]=0;
      message_length[6]=0;
      message_length[5]=0;
      message_length[4]=0;
```

```
      message_length[3]=0;
      message_length[2]=0;
      message_length[1]=0;
      message_length[0]= unsigned char)hexstring;

//pad message with 1
      Message_Block[Message_Block_Index] = 0x80;
//pad with 0s
      while(Message_Block_Index < 56)
      {
            Message_Block[Message_Block_Index++] = 0;
      }
//pad with the message len
      Message_Block[56] = message_length[7];
      Message_Block[57] = message_length[6];
      Message_Block[58] = message_length[5];
      Message_Block[59] = message_length[4];
      Message_Block[60] = message_length[3];
      Message_Block[61] = message_length[2];
      Message_Block[62] = message_length[1];
      Message_Block[63] = message_length[0];

      ProcessMessageBlock();
}


unsigned HMAC::CircularShift(int bits, unsigned word)
{
      return ((word << bits) & 0xFFFFFFFF) | ((word & 0xFFFFFFFF) >> (32-
bits));
}
```

Appendix II

```
# changes on different files

#file packet.h
enum packet_t {
      PT_TCP,
         -
         -
         -
      PT_HDLC,

      PT_MIPBU, //mobile ip bu
      PT_NTYPE // This MUST be the LAST one
}
class p_info {
public:
      p_info() {
            name_[PT_TCP]= "tcp";
            name_[PT_UDP]= "udp";
                  -
                  -
                  -
            name_[PT_XCP]="xcp";
            //binding update
            name_[PT_MIPBU]="MIPBU";
}
// trace file, cmu-trace.h

class CMUTrace : public Trace {
public:
      CMUTrace(const char *s, char t);
      void  recv(Packet *p, Handler *h);
            -
            -
            -
      void    format_aodv(Packet *p, int offset);
      void  format_mipbu(Packet *p, int offset);
}

//cmu-trace.cc

void CMUTrace::format_mipbu(Packet *p, int offset)
       {
             struct hdr_mipbu * ph = HDR_MIPBU(p);
            //hdr_mipbu *miph=hdr_mipbu::access(p);

             if (pt_->tagged()) {
                   sprintf(pt_->buffer() + offset,
                   "-mipbu:s %d -mipbu:l %d ",
                   ph->seq_no,
                   ph->packet_len);
                  // ph->pkt_len());
```

```
                }
                else if (newtrace_) {
                      sprintf(pt_->buffer() + offset,
                      "-P mipbu -Ps %d -Pl %d ",
                      ph->seq_no,
                      ph->packet_len);
                     // ph->pkt_len());
                }
                else {
                      sprintf(pt_->buffer() + offset,
                      "[mipbu %d %d ] ",
                      ph->seq_no,
                      ph->packet_len);
               //     ph->pkt_len());
                }
        }
void CMUTrace::format(Packet* p, const char *why)
{

       switch(ch->ptype()) {
             case PT_MAC:
                 :
                 .
             case PT_PING:
                   break;
             case PT_MIPBU:
                   format_mipbu(p,offset);
                   break;
}
}
#ns-agent.tcl
Agent/MIPBU instproc init args {

         $self next $args
}
#ns-packet.tcl
foreach prot {
# Common:
       Common
       Flags
         -
         -
         -
#Mobile IP ,Binding update:
       MIPBU # mobile IP binding update
}
```

Appendix III

#NS2 code for testing the simulation

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf


proc finish {} {
        global ns nf
        $ns flush-trace
       #Close the trace file
        close $nf
       #Execute nam on the trace file
        exec nam out.nam &
        exit 0
}

#Create two nodes
set Mobile_Node  [$ns node]
set Home_Agent   [$ns node]
set Corres_Agent [$ns node]

#Set color for the nodes
$Mobile_Node color Blue
$Home_Agent color Red
$Corres_Agent color Green

#Create a duplex link between the nodes
$ns duplex-link $Mobile_Node $Home_Agent 1Mb 10ms DropTail orient right
$ns duplex-link $Home_Agent $Corres_Agent 1Mb 10ms DropTail orient down


#Create a
MIPBU agent and attach it to node n0
set mipbu0 [new Agent/Mipbu]
$ns attach-agent $Mobile_Node $mipbu0
#Create a MIPBU agent and attach it to node n1
set mipbu1 [new Agent/Mipbu]
$ns attach-agent $Home_Agent $mipbu1
#Create a MIPBU agent and attach it to node n2
set mipbu2 [new Agent/Mipbu]
$ns attach-agent $Corres_Agent $mipbu2

#attach agents
$ns connect $mipbu0 $mipbu1
$ns connect $mipbu1 $mipbu2

$ns at 0.0 "$mipbu0 start"

$ns at 9.5 "$mipbu0 stop"
```

```
#Call the finish procedure after 5 seconds of simulation time
$ns at 10.0 "finish"

#Run the simulation
$ns run
```