ENHANCING SECURITY IN MOBILE IPv6 WITH PRIVATE KEY-BASED BINDING UPDATE PROTOCOL

HERO MODARES

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE & INFORMATION
TECHNOLOGY - UNIVERSITY OF MALAYA
KUALA LUMPUR

2013

Abstract

In the Mobile IPv6 (MIPv6) protocol, a Mobile Node (*MN*) is a mobile device with a permanent Home Address (*HoA*) on its home link. The *MN* will acquire a Care-of Address (*CoA*) when it roams into a foreign link. It then sends a Binding Update (*BU*) message to the Home Agent (*HA*) and the Correspondent Node (*CN*) to inform them of its current *CoA* so that future data packets destined for its *HoA* will be forwarded to the *CoA*. The *BU* message, however, is vulnerable to different types of security attacks such as: the Man-In-The-Middle (MITM) attack, session hijacking attack, Denial-of-Service (DoS) attack. The current security protocols in MIPv6 are not able to effectively protect the *BU* message against these attacks. The Private Key-based Binding Update (PKBU) protocol is proposed in this thesis to overcome the shortcomings of some existing MIPv6 protocols. The proposed PKBU protocol incorporates: (a) a method to assert the address ownership of the *MN* by creating a 128-bit MIPv6 address based on the *MN*'s private key, and computing one-way hash function in order to authenticate the *MN*s authority, thus, allowing the *CN* to validate that the *MN* is not a malicious node. The results obtained show that it addresses the security requirements as well as it able to check the address ownership of the *MN*; and (b) a method to verify the reachability of the *MN* by sending packets from the *MN* to the *CN*, directly, and through the *HA*. The *CN* will then use both packets to verify the reachability of the *MN*. The results show that the *CN* is able to correctly validate the *HoA* and the *CoA*, and also address the security requirements such as authentication, confidentiality and integrity. The PKBU protocol also offers protection for the *MN* against false binding-update attacks, in which an attacker attempts to spoof two different messages, which are sent to the *CN* in two different paths. Thus, the PKBU protocol has addressed the important security requirements of mobile communication such as address ownership, reachability, and device authentication. The PKBU protocol was subjected to formal security verification to identify any security flaws, by using the Protocol Composition Logic (PCL). The verification results show that the PKBU protocol meets all the security requirements and is able to successfully defend the *BU* message against common attacks. The INETMANET-2.0 framework in the OMNeT++ network simulator was used to model the working of the PKBU protocol against common attacks and for attack detection, and attack mitigation. The results of the formal security verification and the network simulation modelling of the PKBU protocol show that the proposed protocol can improve the level of security in the MPv6 protocols, especially, the security of the *BU* messages transmitted between the *MN* and the *CN*.

Abstrak

Di dalam protokol Mudah Alih IPv6 (MIPv6), sebuah Nod Bergerak (*MN*) adalah satu peranti yang mempunyai Alamat Rumah Tetap (*HoA*) pada pautan asalnya. *MN* akan memperoleh Care-of Address (*CoA*) apabila ia berkeliaran di dalam pautan asing. Kemudian ia akan menghantar mesej Kemaskini Ikatan (*BU*) kepada Ejen Rumah (*HA*) dan Nod Koresponden (*CN*) untuk memaklumkan tentang *CoA* semasa supaya paket seterusnya yang ditujukan kepada HoA akan disalurkan kepada *CoA*. Namun begitu, mesej *BU* terdedah kepada pelbagai jenis serangan keselamatan seperti: serangan orang tengah (MITM), serangan rampasan sesi, serangan penafian servis (DoS). Protokol keselamatan MIPv6 pada masa ini masih tidak dapat melindungi mesej *BU* dari serangan-serangan ini dengan berkesan. Protokol Kemaskini Ikatan Berasaskan Kunci Persendirian (PKBU) telah dicadangkan di dalam tesis ini untuk mengatasi kelemahan protokol MIPv6 sedia ada. Protokol PKBU yang dicadangkan menggabungkan: (a) kaedah untuk menegaskan pemilikan alamat *MN* dengan mewujudkan satu alamat 128-bit MIPv6 yang dihasilkan dari kunci persendirian *MN*, dan mengira fungsi cincang sehala untuk mengesahkan autoriti *MN*, sekali gus, membolehkan *CN* untuk mengesahkan bahawa *MN* bukan satu nod yang berniat jahat. Keputusan yang diperolehi menunjukkan bahawa ia dapat memenuhi keperluan keselamatan serta memeriksa pemilikan alamat *MN*; dan (b) kaedah untuk mengesahkan kebolehcapaian *MN* dengan menghantar paket dari *MN* kepada *CN* secara langsung, dan melalui *HA*. *CN* kemudian akan menggunakan kedua-dua paket untuk mengesahkan kebolehcapaian kepada *MN*. Keputusan menunjukkan bahawa *CN* dapat mengesahkan *HoA* dan *CoA*, dan juga memenuhi keperluan keselamatan. Protokol PKBU juga menawarkan perlindungan kepada *MN* terhadap serangan kemaskini-ikatan palsu, di mana penyerang cuba untuk menipu dua mesej yang berbeza, yang dihantar kepada *CN* melalui dua laluan yang berbeza iaitu secara terus dan melalui *HA*. Oleh yang demikian, protokol PKBU telah menangani keperluan keselamatan penting dalam komunikasi mudah alih seperti pemilikan alamat, kebolehcapaian, dan pengesahan peranti. Protokol PKBU telah disahkan dengan menggunakan keselamatan formal untuk mengenal pasti apa-apa kelemahan keselamatan, iaitu dengan menggunakan Protokol Komposisi Logik (PCL). Keputusan pengesahan menunjukkan bahawa protokol PKBU memenuhi semua keperluan keselamatan, dan berjaya mempertahankan mesej *BU* terhadap serangan. Rangka kerja INETMANET-2.0 di dalam rangkaian simulator OMNeT++ telah digunakan untuk melaksanakan PKBU sebagai model terhadap serangan biasa, dan untuk mengesan serangan, dan juga untuk pengurangan serangan. Keputusan pengesahan keselamatan formal dan model simulasi rangkaian daripada protocol PKBU menunjukkan bahawa protokol yang dicadangkan itu boleh meningkatkan tahap keselamatan dalam protokol MIPv6, terutamanya, keselamatan mesej *BU* yang dihantar di antara *MN* dan *CN*. Pembangunan protokol PKBU oleh it telah berjaya menangani beberapa kelemahan keselamatan protokol rangkaian yang sedia ada.

Acknowledgement

First and foremost, I would like to record my sincere gratitude to my supervisor, Dr. Rosli Salleh, for his valuable advice and guidance from the very early stage of this research, as well as sharing his extensive knowledge and experiences with me. Above all and the most needed, he provided me unflinching encouragement and support in various ways, which truly inspire and enrich my growth as a student, and what a researcher wants to be.

Words fail me in expressing my true feeling and my gratitude to my family whose love, unwavering support and encouragement have motivated me all these years in my studies.

I also want to thank Dr. Por Lip Yee, Mr. Amirhossein Moravejosharieh and Mr. Majid Talebi for all their help, support, interest and valuable advice and Mr. K. H. Teh for his help in proofreading this thesis.

This work was supported in part by the University of Malaya, Kuala Lumpur Malaysia under UMRG Grant (RG080/11ICT). I would like to express my gratitude to all those who gave me the opportunity to complete this research. I want to thank the Faculty of Computer Science and Information Technology, University of Malaya (UM) for giving me permission to commence this thesis and for the necessary facilities for me to carry out my research work.

List of Publications

Published Journal Articles

1- **H. Modares**, A. Moravejosharieh, J. Lloret, and R. Salleh, “A Survey of Secure Protocols in Mobile IPv6”, 2013, Journl of Network and Computer Applications, Elsevier, ISSI: 1084-8045, (*ISI-Cited-Published*).

2- **H. Modares**, A. Moravejosharieh, J. Lloret, and R. Salleh, 2013, “A Survey on Proxy Mobile IPv6 Handover”, 2013, IEEE Systems Journal, ISSN: 1932-8184, (*ISI-Cited-Accepted*).

3- **H. Modares**, A. Moravejosharieh, R. Salleh and J. Lloret, “Securing Binding Update in Mobile IPv6 Using Private Key-based Binding Update Protocol”, 2013 , Electronics and Telecommunications Research Institute (ETRI) Journal, ISSN: 1225-6463, (*ISI-Cited-Accepted*).

4- A. Moravejosharieh and **H. Modares**, “A Proxy MIPv6 Handover Scheme for Vehicular Ad-hoc Networks”, Wireless Personal Communications, 2013, Springer, ISSN: 0929-6212, (*ISI-Cited-Published*).

5- **H. Modares**, A. Moravejosharieh, R. Salleh, and J. Lloret “Security Overview of Wireless Sensor Network”, Life Science Journal, volume 10, issue 2, May 2013, ISSN: 1097-8135, (*ISI-Cited-Published*).

6- **H. Modares**, A. Moravejosharieh, and R. Salleh "Secure Connection in Mobile IPv6", Life Science Journal, volume 10, issue 2, May 2013, ISSN: 1097-8135, (*ISI-Cited-Published*).

7- A. Moravejosharieh, **H. Modares** & R. Salleh, and E. Mostajeran "Performance Analysis of AODV, AOMDV, DSR, DSDV Routing Protocols in Vehicular Ad Hoc Network", Feb. 2013, Research Journal of Recent Sciences, ISSN: 2277-2502, *(ISI-Thomson Reuters-Web of Knowledge).*

8- **H. Modares**, M. T. Shahgoli, H. Keshavarz, A. Moravejosharieh, and R. Salleh, 2012, "Make a Secure Connection Using Elliptic Curve Digital Signature", International Journal of Scientific & Engineering Research, Volume 3, Issue 8, Sep.2012, ISSN: I069432, (Non-ISI/Non-SCOPUS Cited Publication).

Journal Articles under Review

1- **H. Modares**, A. Moravejosharieh, and R. Salleh, 2012, "Infrastructure–less Protocols in Mobile IPv6", Jan 2013, Maejo International Journal of Science and Technology, ISSN: 0020-7160, *(ISI-Cited Publication).*

2- A. Moravejosharieh, **H.Modares,** and R. Salleh, "Proxy Mobile IPv6 with Location Assisted Approach in Wireless Networks", Dec. 2012, Maejo International Journal of Science and Technology, ISSN: 0020-7160, *(ISI-Cited Publication).*

Conferences Papers

1- **H. Modares**, A. Moravejosharieh, H. Keshavarz, and R. Salleh, 2012, “Protection of Binding Update Message in Mobile IPv6”, ISI 2012: 1st International Conference on Intelligent Systems and Informatics, Nov. 19, 2012 - Nov. 21, 2012, Bandung, Indonesia.

2- **H. Modares**, A. Moravejosharieh, and R. Salleh, 2012, “Overview of Mobile IPv6 Security”, Third International Conference on Intelligent Systems, Modelling and Simulation, IEEE ISMS2012, Kota Kinabalu, Malaysia, 8–10 February 2012. ISBN 978-0-7695-4668-1.

3- A. Moravejosharieh, R. Salleh, and **H. Modares,** 2012, “ Overview of Handover Latency and Resource Consumption in Hierarchical Mobile IPv6 Protocol and GPS Approaches”, Third International Conference on Intelligent Systems, Modelling and Simulation, IEEE ISMS2012, Kota Kinabalu, Malaysia, 8–10 February 2012. ISBN 978-0-7695-4668-1.

4- **H. Modares**, A. Moravejosharieh, and R. Salleh, 2011, “Wireless Network Security Using Elliptic Curve Cryptography”, First International Conference on Informatics and Computational Intelligence, IEEE ICI2011, Department of Informatics - Parahyangan Catholic University, Bandung, Indonesia, 12 - 14 December 2011.

5- A. Moravejosharieh, R. Salleh, and **H. Modares**, 2011, " Overview of Latest Approaches in Heterogeneous and Mobile IPv6 Homogeneous Wireless Networks Based on Location-Assisted Information", First International Conference on Informatics and Computational Intelligence, IEEE ICI2011, Department of Informatics - Parahyangan Catholic University, Bandung, Indonesia, 12 - 14 December 2011.

6- **H. Modares**, R. Salleh, and A. Moravejosharieh, 2011, "Overview of Security Issues in Wireless Sensor Networks", Third International Conference on Computaional Intelligence Modelling and Simulation, IEEE, Langkawi, Malaysia, 20-22 September 2011.

7- A. Moravejosharieh, R. Salleh, and **H. Modares**, 2011, "A Novel Approach For Efficient Resource Consumption In GPS-Based Mobile IPv6 Wireless LAN", Third International Conference on Computaional Intelligence Modelling and Simulation, IEEE, Langkawi, Malaysia, 20-22 September 2011.

8- **H. Modares**, M. T. Shahgoli, R. Salleh, and Y. Salem, 2011, "Realizing Public key in WSN", UK-Malaysia-Ireland Engineering Science Conference 2011 (UMIES 2011), 12th-14th July 2011, Faculty of Economics & Administration.

9- Y. Salem , **H. Modares**, M. R. Hosseiny, and R. Salleh, "An Overview of Elliptic Curve Cryptography in constrained Applications", 3rd International Conference on Informatics and Technology, 2009 (Informatics '09), 27th-28th October, 2009, Kuala Lumpur, Malaysia, pp. 158-163.

10- **H. Modares**, M. R. H. Fatemi, and R. Salleh, 2009, "Elliptic Curve Cryptography: The Next Generation Encryption Technology for Secure Data Communication", E-CASE 2009 conference-Singapore.

11- M. T. Shahgoli, **H. Modares**, and Y. Salem, 2009, "Elliptic Curve Cryptography in E-commerce and Smart Card", E-CASE 2009 Conference-Singapore.

12- **Modares H**., Y. Salem, M. R. H. Fatemi, and R. Salleh, 2009, "Application of Elliptic Curve Cryptography", 3rd International Conference on Informatics and Technology 2009, Kuala Lumpur, Malaysia.

Chapter Book

1- **Modares H.**, Lloret J., Moravejosharieh A., and Salleh R., "Security in Mobile Cloud Computing," Mobile Computing over Cloud: Technologies, Services, and Applications, IGI Global, 2013.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AAA	Authentication, Authorization, and Accounting
BA	Binding Acknowledgement
BC	Binding Cash
BU	Binding Update
CA	Certification Authority
CBU	Certificate-based Binding Update Protocol
CGA-OMIPv6	CGAs to Optimize Mobile IPv6 Protocol
CN	Correspondent Node
CN_{PUK}	Correspondent Node’ Public Key
CoA	Care-of Address
CoTI	Care-of Test Init
CoT	Care-of Test
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
DH	Diffie-Hellman
DoS	Denial of Service

DSA	Digital Signature Algorithms
EAP	Extensible Authentication Protocol
EBU	Early Binding Update
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ERO-MIPv6	Enhanced Route Optimization for Mobile IPv6
ESP	Encapsulating Security Payload
HA	Home Agent
HCBU	Hierarchical Certificate-based Binding Update
HoA	Home Address
HoTI	Home Test Init
HoT	Home-of Test
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
IKE	Internet Key Exchange
ISAKMP	Internet Security Association and Key Management Protocol

MD4, MD5	Message-Digest Algorithm
MIPv6	Mobile Internet Protocol version 6
MITM	Man-In-The-Middle attack
MN	Mobile Node
MN_{PRK}	MN's Private Key
MN_{PUK}	MN's Public Key
ND	Neighbour Discovery
OMIPv6	Optimizing Mobile Internet Protocol version 6
PAK-based	Password-based Authenticated Key Exchange
PBK	Purpose-Built Key
PCL	Protocol Composition Logic
PKBU	Private Key-based Binding Update Protocol
PUK	Public Key
PKC	Public key cryptography
PKI	Public Key Infrastructure
PRK	Private Key
RR	Return Routability
RO	Route Optimisation

SA	Security Associations
SAD	Security Association Databases
SPD	Security Policy Databases
SPI	Security Parameters Index
TBU	Ticket-based Binding Update
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDHBU	Unauthenticated Diffie-Hellman Binding Update

Chapter 1 : INTRODUCTION

1.1 Overview

Over the last decade, the mobile Internet has been widely used by millions of people all over the world due to its availability. Mobile communication is now leading the way in today’s Internet communications, and users expect to be able to roam the Interent from anywhere in the world through the different networks. The mobility support protocol for IPv4 (Mobile IP) [RFC 5944] (Perkins, 2010) dates back to 1993 when the first IPv4 mobility support protocol was proposed. In 1996, Mobile IPv4 was standardized as RFC 2002 (Johnson, Perkins, & Arkko, 1996), followed by the latest revision, RFC 3344 (Perkins, 2002). The late 1990s witnessed the beginning of the Internet age, at which time the first pioneers began attempting to commercialize various Internet connectivity services. Companies and universities offered their services and information via the Internet. Shortly afterwards, individuals also started using the Internet for similar purposes. It has now become the largest global information network, and the Mobile IP plays a crucial role in keeping the network connected.

In a mobile Internet environment where mobile devices move frequently and get attached to other networks, they need to obtain new IP addresses to continue communicating with their correspondent nodes. As a result of the unpredictable growth in mobile Internet usage over the years, however, a larger address space is inevitably necessary as nearly all IPv4 addresses will be used up. In addition, network security is now an issue of concern with the ever increasing number of people using the mobile devices. The use of mobile devices were greatly affected by a wide range of attacks on the MIPv4 protocol, as it was not sufficiently secure to shield the nodes from the attacks. MIPv6 was developed to address the weaknesses of the IPv4 protocol.

MIPv6 uses a 128-bit address as opposed to IPv4 which uses a 32-bit address, and MIPv6 has other advantages, as well, such as auto-configuration and better security features. In the mobile Internet, IP routing is based on the information set up in the IP headers so that data can be sent correctly to the appropriate nodes. In this method, old IP links get destroyed and new links are constructed as the data is transmitted from one location to another. MIPv6 offers a way to mitigate this problem without the need to make major adjustments to the nodes or routers in a network. However, not all security and privacy issues in MIPv6 have been eliminated. Many of the MIPv6 protocols developed to overcome security vulnerabilities are defined in the standards and drafts of the Internet Engineering Task Force (IETF). On December 1998, the IPv6 protocol was included in the RFC 2460 standard, and many of its features were incorporated into other standards, as well. Table 1.1 shows a list of these standards. The MIPv6 standard remains as a draft or proposal form because of many unresolved issues.

Table 1.1: IPv6 and MIPv6 IETF RFC

IPv6 and MIPv6 IETF RFCs	Name
Internet Protocol Version 6	RFC 2460
Neighbour Discovery for IP Version 6 (IPv6)	RFC 4861
Privacy Extensions for Stateless Address Auto-configuration in IPv6	RFC 4941
Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification	RFC 2463
Recommendations for IPv6 in Third Generation Partnership Project (3GPP) Standards	RFC 3314
Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents	RFC 3776
Mobility Support in IPv6	RFC 6275

1.2 Operation of IP Mobility Protocol

IP mobility is a protocol for mobile networking and computing, and results from the convergence of the Internet and mobile communication technologies. IP mobility is designed to allow a Mobile Node (*MN*) to move from one network to another during communication, even though the *MN*'s point of attachment to the network has physically changed (Perkins, 1997). When a mobile device is disconnected from the present network and gets reconnected to another network, portability is achieved. The *MN* achieves portability by having two IP addresses:

- Home Address (*HoA*): a static IP address that resides in the home network.
- Care-of Address (*CoA*): the *MN*’s address in a foreign network.

Whenever a Correspondent Node (*CN*) attempts to communicate with the *MN* located in a foreign network, it sends a packet to the Home Agent (*HA*). The *HA* will use IP encapsulation, whereby an outer header is added to the data packet along with the new address, and it then proceeds to tunnel the packet to the *MN*'s *CoA*, at which point the packet will be encapsulated (Figure 1.1). This happened where in the IP Mobility, Binding Update (*BU*) message is sent from the *MN* to the *HA* and *CN*. Binding updates are a very important optimisation within Mobile IPv6, which allows the *CN* to bypass the *HA* router and communicate directly with the *MN*. *BU* message contains the Home Address (*HoA*) and Care-of Address (*CoA*) of *MN*.

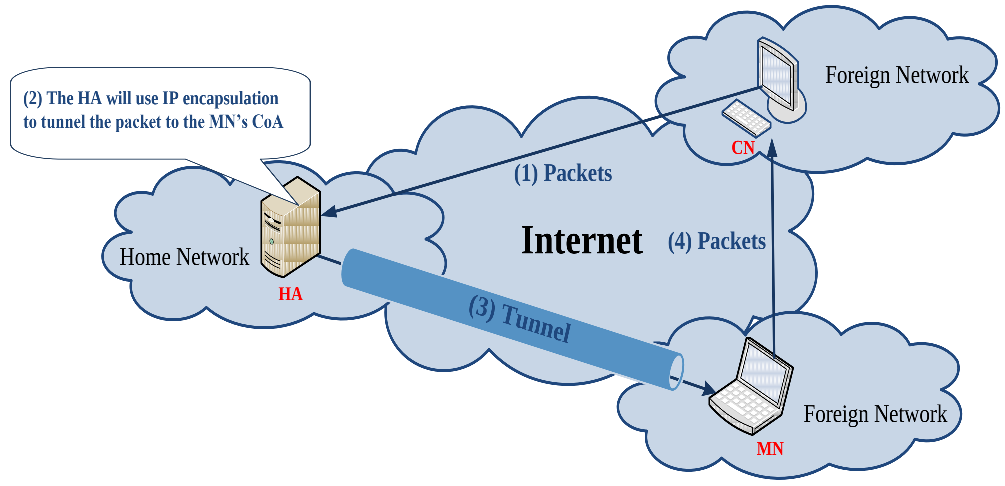


Figure 1.1: Packet Routing in IP Mobility Protocol

There are several security vulnerability issues with *BU* messages, and these include: data packet interception where an attacker is able to eavesdrop on the content, thus, causing a breach in confidentiality; modification of the transmitted packets to suit the malicious purpose of the attacker; Men-In-The-Middle (MITM) attacks; session hijacking attacks; Denial of Service (DoS) attacks, and Return to Home spoofing attacks (Soliman, 2004). In order for the attacks to happen, the attacker must know the IPv6 addresses of the *MN*'s *HoA* and the *CN*. The vulnerability to all these attacks clearly shows the weaknesses of the current Mobile IP architecture. It is very crucial that appropriate security measures must be taken to overcome or to mitigate these vulnerabilities. Further research on these weaknesses might even reveal that a single or common vulnerability could be the cause of many different attacks.

1.3 Motivation

The increasing use of mobile devices for communication, today, has raised great concern about the security of the data or information being transmitted. As discussed above, in the MIPv6 protocol, the *BU* message is vulnerable to a range of attacks. The research effort undertaken and reported in the thesis, was motivated by the need to protect the *BU* messages which are sent from the *MN* to the *CN*. The reasons for protecting *BU* messages are:

- *BU* messages contain sensitive data such as the *MN's HoA* and *CoA*, and such information must not be revealed to potential attacks.
- Many security measures introduced to protect *BU* messages, in the past, had various drawbacks because they could not conceal the location data from the attackers, i.e., they do not have location privacy feature (Yan, et al. 2012).
- *BU* messages are used for redirecting addresses among the network nodes. Attackers exploit the security vulnerability of *BU* messages by intercepting the *BU* message and sending a false binding update message, instead, to the destination nodes. In a false binding update attack, the attacker sends a spoofed message

pretending to be a valid *MN*. It redirects the traffic that is intended for the original node into itself (Soliman, 2004).

- Another type of attack that involves information theft is eavesdropping (Ying & Feng, 2010), which can either be active or passive. In active eavesdropping, the attacker will initiate an independent connection with the victim node and send messages to it. This will make the victim node believe that there is direct communication among the nodes using a private connection, but in actual fact, it is under the attacker's control. In passive eavesdropping, however, the attacker gets the needed information by gathering the session data that travels between the *HA* and its mobile device, by listening to the communication traffic. All messages that travel between the two nodes are intercepted, and new messages are inserted by the attacker.
- Attackers can mount a Denial of Service (DoS) attack in the mobile network. This DoS attack prevents communication between the nodes by sending false *BU* message. This type of attack involves several security components such as authentication, confidentiality, and integrity. The largest amount of the victims' bandwidth is consumed during a DoS attack. This leads to overloading of computational resources at the victim's host location, causing loss of connectivity, and disruption of service to legitimate users.

In this research, the above issues were studied in more detail and a method is proposed to make the connection between the *MN* and the *CN* more secure, and in particular, to protect *BU* messages against Man-In-The-Middle attack, session hijacking attack, and Denial of Service attack.

1.4 Problem Statements

The MIPv6 has a number of security vulnerabilities, particularly, the vulnerability of the *BU* message to various types of attacks. In the MIPv6, the *MN* needs to authenticate itself every time it moves to a foreign network. Without a secure authentication process, the data packets sent from one node to another node might be intercepted by a malicious node, causing the data packets to be redirected to its location or to an arbitrary IP address. This will deny service for the intended legitimate receiver node. This happens because current protocols do not have an effective authentication method to check the validity of the users, or to conceal the *HoA* and *CoA* of the *MN* (Georgiades, 2011). It is safe to conclude that most protocols such as Return Routability, Early Binding Update (EBU), Optimizing Mobile IPv6 (OMIPv6) and etc, (Chapter 3), face security vulnerabilities because they do not have an effective way of:

- authenticating a user's identity, hence, there is no way for the *CN* to know whether a *MN* is a valid node or a malicious one;
- hiding the Care-of Address (*CoA*) in the current MIPv6 security protocols, hence, there is no way for the *CN* to validate the owner of the address, especially, *CoA*s with spoofed or modified address.

In view of the weaknesses of the protocols, mentioned above, effective measures must be taken to provide a secure connection between the *MN* and the *CN* to protect the *BU* messages from common attacks on the MIPv6 protocol. These attacks include: Denial of Service attacks, where the attacker impersonates and denies services intended for the target node by directing network traffic to itself or another node (false node); Men-In-The-Middle attacks; and session hijacking attacks.

1.5 Research Aim

Several security protocols have been introduced to overcome the vulnerabilities of the *BU* message to attacks, which could disrupt normal network operations or breach the confidentiality of users in the network. Most existing protocols, however, are not able to protect the *BU* message from the different types of attacks, effectively. The aim of this research is to develop a way to secure the *BU* message that is sent from *MN* to the *CN*.

1.6 Research Objectives

To produce the best security solution, the following objectives must be achieved:

- To propose a method to assert the address ownership of the *MN*. This technique allows the *CN* to validate the *MN* to ensure that it is not a malicious node by checking the address ownership.
- To propose a method to verify the reachability of the *MN*. This technique allows the *CN* to verify reachability of *MN* by checking *CoA* of *MN*.
- To propose a method that is protecting the *BU* messages against MITM, session hijacking, and DoS attacks, and to ensure a secure connection between the *MN* and the *CN*.
- To evaluate the security requirement of our proposed methods using Protocol Composition Logic (PCL) tool.

1.7 Research Scope

In the MIPv6 network, any node is vulnerable to different types of security attacks. This research, however, focus on the security of the *BU* messages. This research design a framework based on the *MN*'s private key to ensure a secure connection between the *MN* and the *CN*. To achieve the research objectives, the proposed protocol is evaluated using informal and formal security evaluation methods, and simulated in the INETMANET-2.0 framework using the OMNeT++ simulator.

1.8 Thesis Outlines

This research work is structured as follows. Chapter 2, provides a background of the research, and this is followed by an overview of the MIPv6 protocol, the security threats to the *BU* message, analysis of security services and architectures for *BU*, authentication mechanisms, and the architecture of the MIPv6 protocol. The chapter also includes discussions of the cryptographic systems and the IPSec protocol.

Chapter 3 presents a security analysis of existing binding update protocols. In this chapter, existing protocols used to safeguard *BUs* are reviewed with the aim of identifying their strengths and weaknesses vis-a-vis in defending the *BU* message against various security threats. These protocols fall under two categories, namely, Infrastructure-less and Infrastructure-based protocols. Each of the existing protocols is explained with respect to their category. The chapter then focuses on all earlier protocols and highlight their advantages and disadvantages.

The proposed framework appears in Chapter 4. The solution to the security problems with respect to the *BU*s is identified after an extensive review of the literature.

This is further refined by taking into consideration input from existing works. This chapter initially provides an explanation of the requirements, assumptions, and principles of our proposed method. The components, operation and justification of the proposed security solution introduced are then explained. Chapter 4 also discusses the verification of the proposed system's security methods. The chapter concludes with the informal analysis as a way of validating our proposed solution. The analysis serves to examine the security requirements, ownership and reachability as well as the risks to the proposed protocol, as identified during the literature review.

Chapter 5 elaborates on the formal security analysis performed on the protocol in Chapter 4. This examination was carried out as a rigid and methodological form of testing the precision of the proposed solution. In this respect, formal analysis was used, and the Protocol Composition Logic (PCL) framework was selected (Cremers, 2008; Datta, Derek, Mitchell, & Roy, 2007; Durgin, Mitchell, & Pavlovic, 2001). Chapter 5 contains the design, implementation and results of the formal analysis of PKBU protocol simulation. The performance of the proposed protocol's was then evaluated. The INETMANET-2.0 framework in OMNeT ++ simulation environment was used to carry out the simulation with the aim of verifying the ideas and concepts proposed in this thesis. Verification is based on how the proposed protocol reacts to various types of attacks, while observation is based on how successful is the protocol in preventing or mitigating attacks.

Chapter 6 presents the conclusions and recommendations for future work. It is also a brief introduction to the basic encryption methods used where a secure method was designed to protect *BU* messages sent between *CN* and *MN*.

1.9 Summary

Binding Update (*BU)* is one of most important message in MIPv6. It contains information that the attacker can use them to attack the *MNs* or *CNs*. In order to protect the *BU* against these threats we need to find a secure way for authenticating the authority of the users, and at the same time, hide the data of the *CoA*. In addition, we need to find a secure technique for the *CN* to certify that the *MN* is valid and not a malicious node. In this research, we tried to establish a secure connection between the *MNs* and the *CNs* by protecting *BU* message which contains sensitive data.

Chapter 2 : THE MOBILE INTERNET PROTOCOL VERSION 6 (MOBILE IPv6)

2.1 Introduction

This chapter gives an insight into the current specifications of the MIPv6 protocol. Section 2.2 presents the basic operation of the MIPv6 protocol, and the three possible modes of communication between a Mobile Node (*MN*) and a Correspondent Node (*CN*). Section 2.3 provides an overview of the Binding Update (*BU)* messages in MIPv6, and details the security threats that could be launched against the protocol. It also presents the security services and performance requirements needed to overcome the attacks. Section 2.4 gives an overview of the Return Routability (RR) procedure. It also identifies the security and performance limitations of the procedure against the requirements specified in Section 2.3. Section 2.5 presents the problem formulation. Finally, Section 2.6 summarises the chapter.

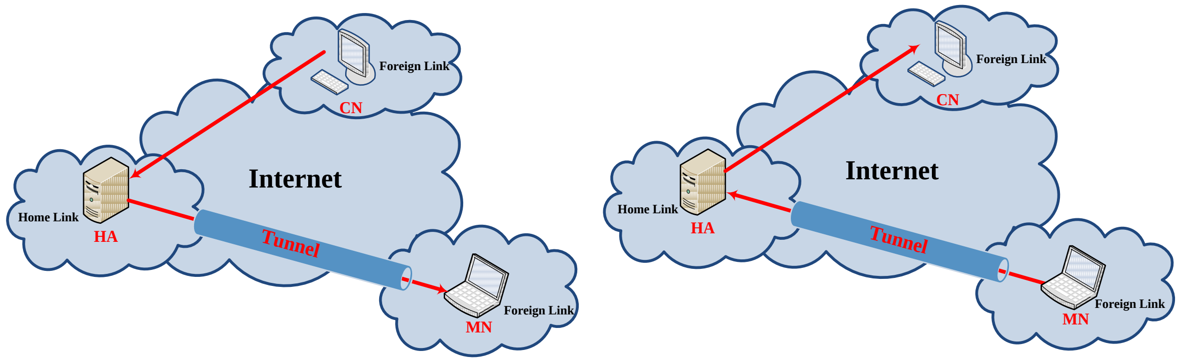
2.2 Basic Operations in Mobile IPv6

The Mobile IPv6 (MIPv6) protocol is one of the most representative efforts being made toward achieving the next-generation all-IP networks. Unlike MIPv4, there is no foreign agent functionality in MIPv6 because a *MN* obtains a new IPv6 address within a foreign network by address auto-configuration (Belding, Elizabeth, Sun, & Perkins, 2001). The *MN* is a mobile device that has a Home Address (*HoA*) on its home link. The MIPv6 protocol provides the Mobile Node (*MN*) with a unique, permanent identifier (IPv6 address) independently of the attached network. MIPv6 consists of three entities: the Mobile Node (*MN*), Home Agent (*HA*), and the Correspondent Node (*CN*), which is the peer that communicates with the *MN*. When a *MN* roams into a foreign link, it will acquire a new Care-of Address (*CoA*). The *MN* must register its current *CoA* with the *HA*. As a result, when the *MN* is located away from home, by using this binding method, the *HA* can intercept the message destined for the *HoA* and forward it to the binded *CoA*. This can be achieved by sending a Binding Update (*BU*) message in which the *MN* will initialise the home registration process. At the end, the *HA* will store the binding of the *MN's HoA* and the *CoA* in its cache (Johnson, Perkins, & Arkko, 1996; Robert, 2003).

MN and the *CN* can communicate in three ways. All traffic between the *MN* and the *CN* will be routed by the *HoA* using the standard IPv6 routing mechanism without any special procedures, whenever the *MN* is at its home link (Deering & Hinden, 2006). However, when the *MN* is at a foreign link, there are three possible communication modes:

1. Bi-directional tunnelling
2. Triangle routing
3. Route Optimization (RO)

In the Bi-directional tunnelling mode, all traffic is routed indirectly to the *MN's* home link. This means that all *CN's* packets destined to the *MN* are routed to the home link of the *MN*, and are intercepted by the *HA* before they are forwarded to the *MN's CoA* via a tunnel. Similarly, packets sent from the *MN* to the *CN* are tunnelled to the *HA* ("reverse tunnelling"). The packets will then be routed back normally to the *CN* from the home link. Usually, the IPv6 encapsulation method is used to perform the tunnelling between the *HAs* and *MNs* (Conta & Deering, 1998). Figure 2.1 shows the Bi-directional tunnelling mode.



(a) **(b)**

Figure 2.1: Bi-directional Tunnelling: (a) Traffic from *CN* to *MN*

(b) Traffic from *MN* to *CN*

In the Triangle routing mode, a *MN* is able to send packets directly to the *CNs*. Figure 2.2 shows how the *MN* is able to deliver packets directly to the *CN*. However, the *CN* delivers packets to the *MN's* home address (*HoA*), and the *HA* routes them to the *MN*. The weakness of this type of routing is that the packet travels a longer path from the *CN* to the *MN*, when compared to the use of the direct path (Zuleger, 2005). Route Optimization (RO) is standard procedure in MIPv6 to eliminate inefficient triangle routing. RO routes packets between the *MN* and the *CN* using the shortest possible path (Anari & Mehdizadeh, 2008; Nikander, Arkko, Aura, & Montenegro, 2003; Ren et al., 2006).

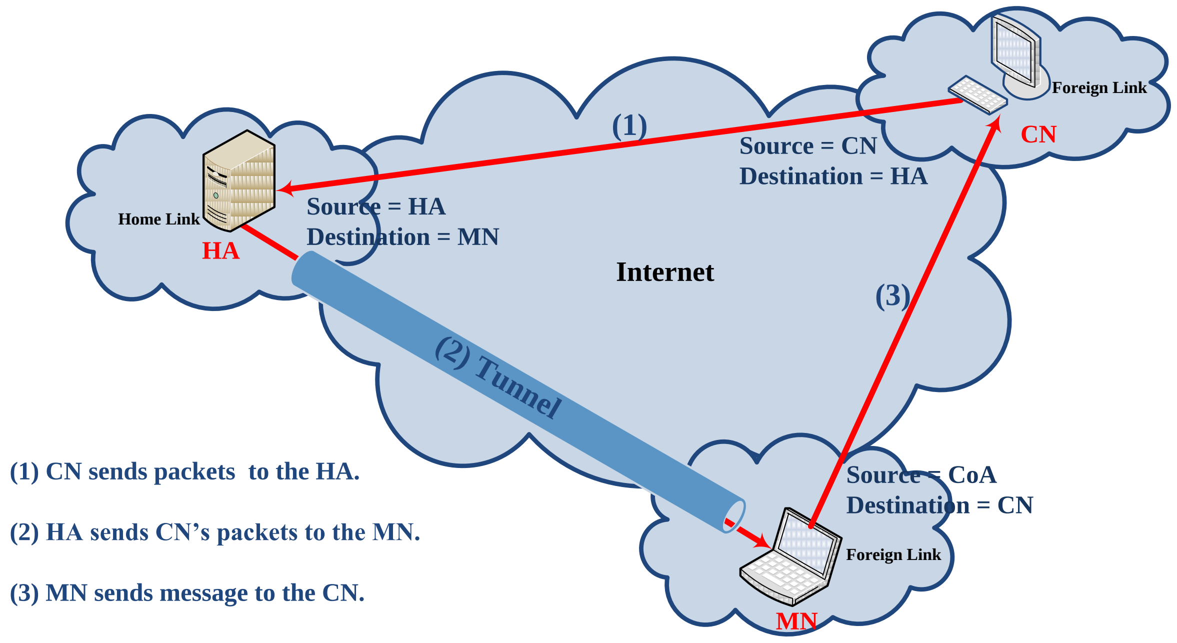
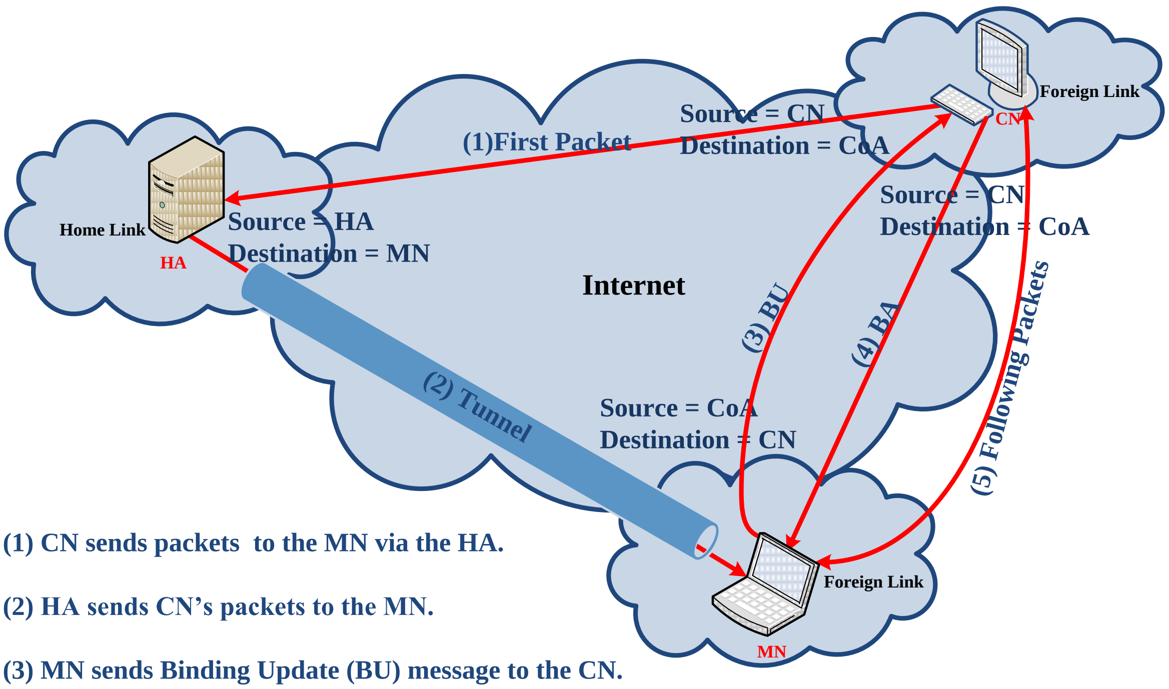


Figure 2.2: Triangle Routing Packet Transmission

In Route Optimisation (RO) mode, the *CNs* deal directly with the *MNs* because they are allowed to skip the *HA* router. This method is shown in Figure 2.3. The *MN* must register its current location with the *CN*. As a result, a binding cache will be created and stored in the *CN*. It contains the binding information between the *MN's HoA* and the *CoA*. As such, all packets sent to the *MN* from the *CN* will be sent directly to the *CoA* rather than through the *HoA*. On the other hand, the *HA* holds the current addresses of the *MNs* and will send packets to the *MNs* whenever a *CN* does not know the address and sends it to the *HA*. However, the speed of delivery increases if the *HA* is bypassed using the *BU* route optimisation method. In the RO mode, a presumably shorter path is used between the *MN* and the *CN*, thus, the traffic volume at the *HA* as well as the home link is minimised (Ren et al., 2006).

The security for RO that handles the *BU*s, however, is still a major security concern because attackers can use the *BU* to launch attacks such as MITM attacks, session hijacking attacks and DoS attacks (See Section 2.3.1). A few solutions have been proposed to make RO secure. One of the solutions is the Return Routability (RR) security protocol based on RFC 3775 and adopted by the Internet Engineering Task Force (IETF) (See Section 2.4). However, RR has a number of security vulnerabilities related to the *BU* messages (See Section 2.3.1).



(1) CN sends packets to the MN via the HA.

(2) HA sends CN's packets to the MN.

(3) MN sends Binding Update (BU) message to the CN.

(4) CN sends Binding Acknowledgement (BA) to the MN.

(5) MN and CN deal directly.

Figure 2.3: Standard Route Optimization (RO) Model

This section segment continues by discussing the *BU* message which is one of the most important messages in Mobile IP, and its vulnerability to various malicious attacks (Ying, & Feng, 2010).

2.3 Binding Updates and its Security Threats in Mobile IPv6

The MIPv6 protocol allows a *MN* to maintain its network connection during attachment transfers (Johnson, Perkins, & Arkko, 2004). A *MN* may be reached at its Home Address (*HoA*) anytime, including instances when the home network is not at the main physical location - in which case, the *MN* receives a Care-of Address (*CoA*) from the local router through stateless or stateful auto-configuration. Home registration subsequently takes place when the *MN* forwards its current *CoA* in a Binding Update (*BU*) message to its associated *HA*. The *HA* then redirects and tunnels the packets directly to the *MN*'s *CoA*. If there is no binding between the *CN* and the *MN*, meaning that registration is in progress, or MIPv6 is not supported by the *CN*, then, bidirectional tunnelling can be used. In this case, an *HA* acts as a medium through which a foreign location *MN* communicates with a *CN* (a stationary or mobile peer communicating with a *MN*).

For MIPv6 to run properly, it is important to protect the *BUs*. Various security loopholes and weaknesses exist in the connections between the nodes. These weaknesses could result in data packet interceptions, thus, allowing malicious attackers to eavesdrop on the content. This could also result in breach of user privacy and modification of data packets. Other weaknesses include address spoofing, redirection, and Denial of Service (DoS) attacks (Aura, 2004; Aura, Roe, & Arkko, 2002; Deng, Zhou, & Bao, 2002; Ehmke et al., 2009; Georgiades, 2011; Mankin et al., 2001; Nikander et al., 2003; Nikander et al. 2005; Sudanthi, 2003). The section below will describe the security risks and attacks, followed by Section 2.3.2, which will discuss the security services to prevent such attacks, as well as the performance requirements for such security services.

2.3.1 Security and Attacks in Binding Update

In order to design a protocol that can mitigate or avoid the associated security risks, it is crucial to identify all possible threats to the protocol. Many attacks that exploit weakness in the MIPv6's binding features, cause inappropriate bindings to happen. For example, the *MN's* location may be misinformed by other entities such as the *HA* and the *CN*, and as a result, the intended traffic is directed to the wrong destination, instead of the *MN*. A brief description of such attacks is given, below (Nikander et al., 2005; Perkins, Johnson, & Arkko, 2011):

- **Malicious MNs flooding attacks**

In malicious *MN* flooding attacks, the *HA* and/or the *CN* receives a spoofed message from the attacker. The spoofed message contains the *HoA* of the fake *MN*'s own *HoA* along with the *CoA* of the victim's address. It falsely claims that it has shifted to the victim's node. The *HA* and the *CN* would then proceed to send all packets that are meant for the *MN* to the victim's address, flooding the victim's node with redundant data. Figure 2.4 shows an example of such attack (Soliman, 2004).

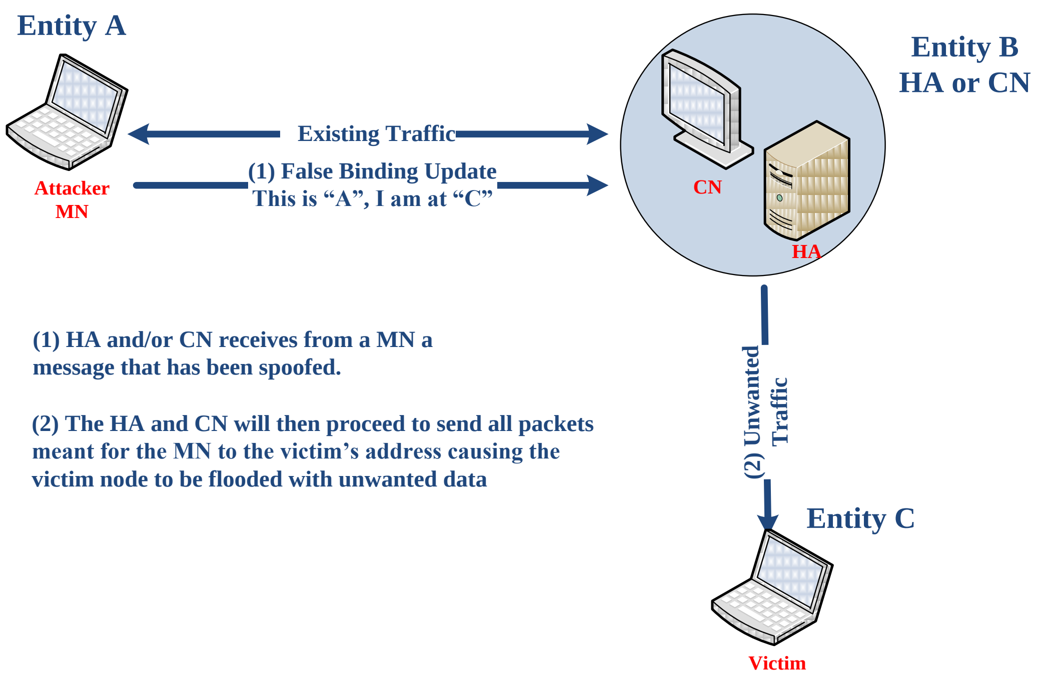

Entity A
Attacker MN
Existing Traffic
(1) False Binding Update
This is "A", I am at "C"
Entity B
HA or CN
CN
HA
(1) HA and/or CN receives from a MN a message that has been spoofed.
(2) The HA and CN will then proceed to send all packets meant for the MN to the victim's address causing the victim node to be flooded with unwanted data
(2) Unwanted Traffic
Entity C
Victim

Figure 2.4: Malicious *MN*s Flooding Attacks

- **Session hijacking attacks**

In the session hijacking attack, illustrated in Figure 2.5, it is assumed that *MN1* is communicating with the *CN*. A false *BU* message is sent (or an old *BU* message is replayed) to the *CN*, claiming that the *MN1* has moved to a new *CoA* that belongs to *MN2* (the *HoA* is set to the *HoA* of a victim *MN*, and the CoA is set to the attacker's address (*MN2*)). If the *CN* accepts this false binding update message, it will begin to communicate with *MN2* instead of *MN1*. Such an attack may result in information leakage, *MN* impersonation, or *MN2* flooding.

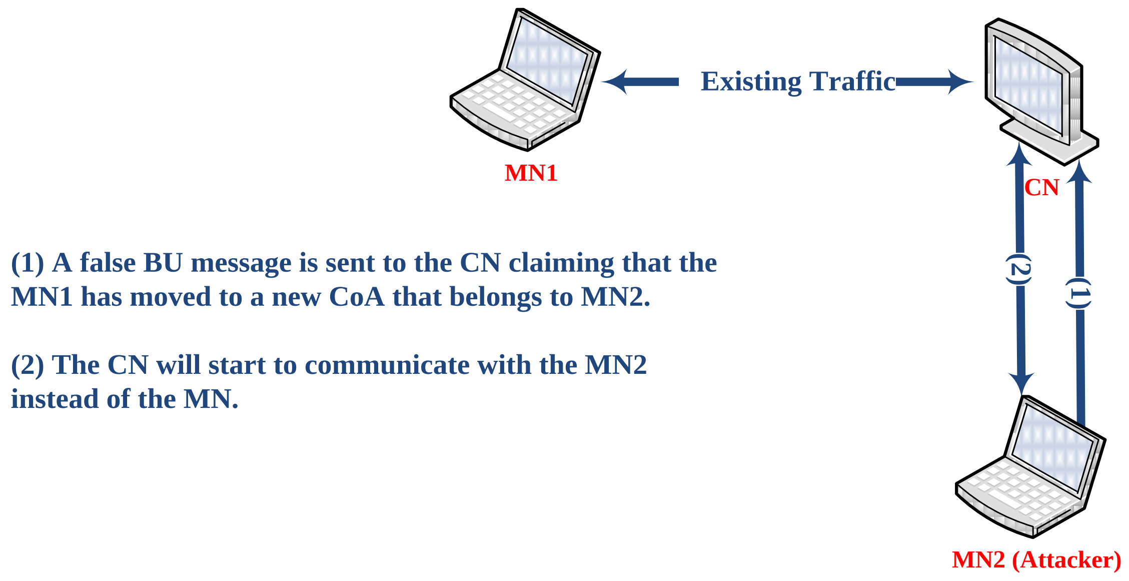

Existing Traffic
MN1
CN
(1) A false BU message is sent to the CN claiming that the MN1 has moved to a new CoA that belongs to MN2.
(2) The CN will start to communicate with the MN2 instead of the MN.
(2)
(1)
MN2 (Attacker)

Figure 2.5: Session Hijacking Attacks

- **Denial-of-Service (DoS) attacks**

In a DoS type attack, the assailant uses a *BU* message to stop any service from being transmitted from the *CN* to the *MN's CoA*. A false binding update message may be sent requesting a *CN* to forward packets meant for the *MN* to a fake address that is not the real *MN's CoA*. By using such a spoofed *BU*, an attacker can cause a large amount of unwanted traffic to overwhelm the resources of a single node or network. Initially, the attacker locates a website with streaming video or a different heavy data stream, and connects to it. It then sends a *BU* to the *CN* requesting traffic redirection to the attackers' new location (Figure 2.6). This node will, therefore, be bombarded constantly with unwanted traffic. Attackers may also use a target network's prefix to pass on a spoofed *BU* message, thus, redirecting a great deal of streaming traffic to the intended network and flooding it with unwanted data (Ren et al., 2006).

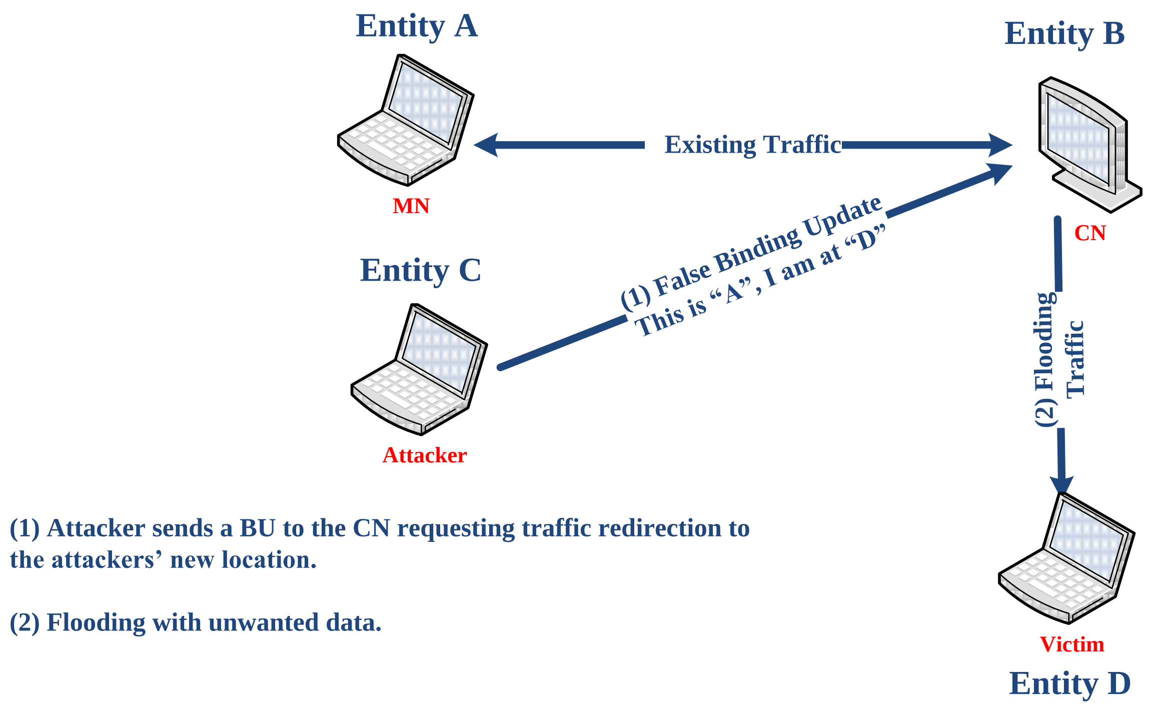


Figure 2.6: Denial-of-Service Attacks

- **Man-In-The-Middle (MITM) attacks**

In Man-In-The-Middle attacks, the *MN* and the *CN* receive spoofed BUs from the attacker, and thus, deem the attacker as the middleman between the *MN* and the *CN* (Figure 2.7). When the attacker is located on the path between the *MN* and the *CN*, it can modify the *BU* message, potentially hijack ongoing connections or create a reflection attack between the *MN* and the *CN* (Elshakankiry, 2010; Soliman, 2004).

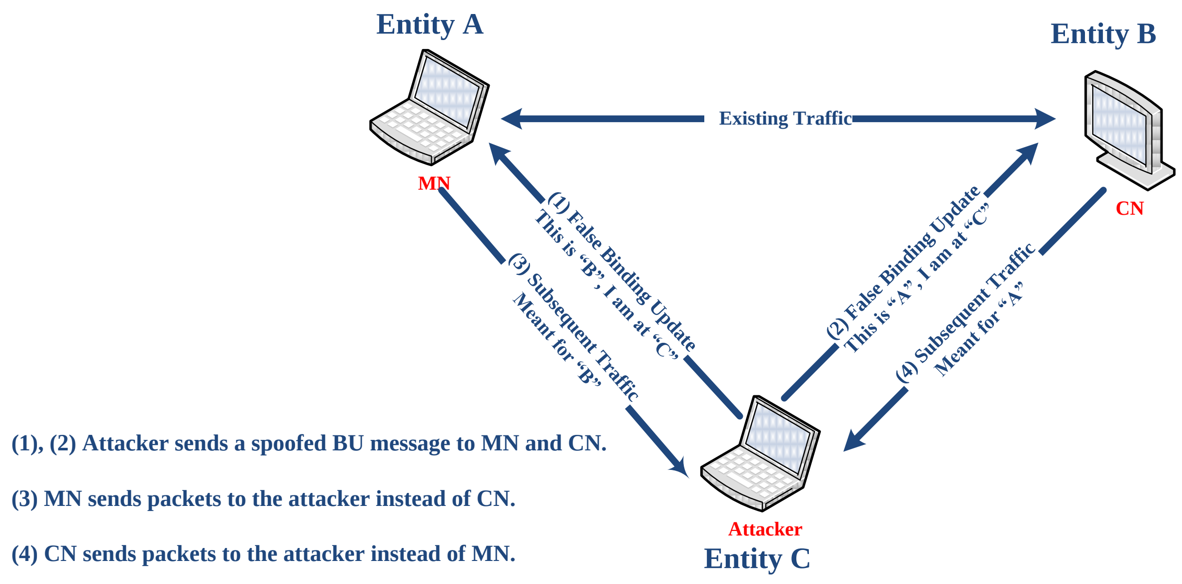


Figure 2.7: Man-In-The-Middle Attack

- **Return-to-home spoofing attacks**

In return-to-home spoofing attacks, an attacker will pretend to be a *MN* that is away from its home link. It will communicate with a *CN* and send a spoofed *BU* which contains the *HoA* of a victim's *MN*, while the attacker's address is set as the *CoA*. Figure 2.8 shows the attacker requesting heavy data download e.g, video streaming, from the *CN*. Another spoofed *BU* is sent to the *CN*, informing it that the attacker has returned to its home link. As a result, the *CN* will continuously direct data traffic to the victim node, eventually flooding the node with excessive amount of unwanted data (Elshakankiry, 2010; Johnson et al., 2004; Nikander et al., 2005; Ren et al., 2006; Soliman, 2004).

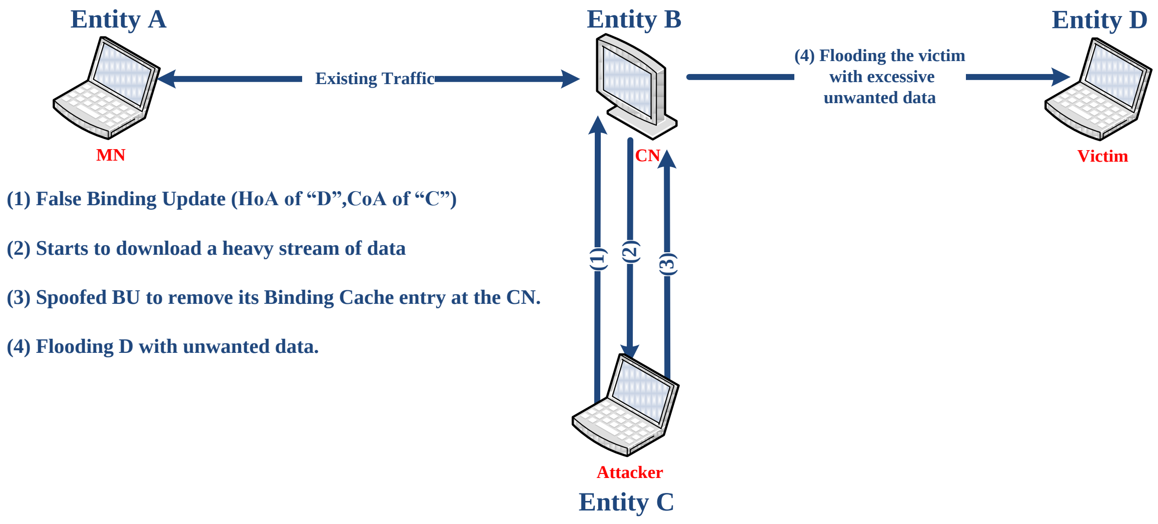

Entity A
Entity B
Entity D
Existing Traffic
(4) Flooding the victim with excessive unwanted data
MN
CN
Victim
(1) False Binding Update (HoA of "D",CoA of "C")
(2) Starts to download a heavy stream of data
(3) Spoofed BU to remove its Binding Cache entry at the CN.
(4) Flooding D with unwanted data.
(1)
(2)
(3)
Attacker
Entity C

Figure 2.8: Return-to-home Spoofing Attack

- **Replaying Binding Updates**

In this type of attack, an attacker replays a recently authenticated *BU* to the *CN*. Here, the *CN* will redirect data packets to the *MN's* previous location. Similar to spoofed *BU*s, replay attacks can be used to capture data packets or initiate DoS attacks. An attacker can impersonate a *MN* by capturing the data packets if it succeeds in obtaining the *MN's* previous address after it has moved away from the network. The attacker would then redirect the packets to the former location by replaying the previous *BUs*. In addition, the attacker can block the *BU* of the *MN* new location, for example, by using a radio jammer or by initiating a flooding attack. Following that, the attacker would use the old location to take over the mobile connection. Any packets forwarded to the *MN* will also be captured by the attacker. This process would continue until the entry of the correspondent's Binding cache expires.

In summary, all *BU* exchanges between a *MN* and its *HA*, as well as between a *MN* and its *CNs*, must be protected. When security is not adequate, attacks such as DoS attacks, MITM attacks, and session hijacking are possible. Data secrecy and integrity can be safeguarded with the use of strong encryption and integrity protection techniques. However, DoS attacks may still occur with spoofed *BUs* even if the data had been cryptographically protected, but this would not result in any disclosure or corruption of sensitive data in the traffic flow. The two nodes can also refuse any *BU* from another node to protect the communication path (Aura & Arkko, 2002; Nikander et al., 2005). The next section discusses the security services for overcoming the above-mentioned attacks.

2.3.2 Security Services

In order to address the security risks and attacks discussed above, the following security services should be provided. Many security requirements have been suggested by Stoneburner (2001), Riedel (2003), Elshakankiry (2010), Georgiades (2011).

- **Authentication**

The process of verifying the identity of a user or a device for the purpose of communication is called the authentication process. Authentication provides security assurance to all *BU* messages coming from nodes with original *HAs* and *CoAs*. Authenticating the *CoA* ensures that the entity is indeed located at that address. Today, a variety of authentication schemes are available, such as the Kerberos(Neuman, Yu, Hartman, & Raeburn, 2005; Yu & Astrand, 2012), X.509 standard (Cooper et al., 2008), and Public-Key Infrastructure (PKI) trust model (Perlman, 1999). Most of these, however, refer to a central, Infrastructure-based authentication database for user verification (Eronen & Arkko, 2006). An Infrastructure-less or decentralised authentication system may provide much better security for a Mobile IP. However, the drawback is that a large amount of resources is consumed, or possible exposure of vital address data can lead to attacks. Commonality of features in crucial in producing an ideal solution, and the relevant details can be obtained by understanding all the fundamental components of the many different types of authentication systems. The commonality features include the components for hashing (Arkko, Nikander, & Montenegro, 2002), digital signatures (Tanenbaum & Van Steen, 2003), address-based keys (Kempf, Gentry, & Silverberg, 2002), and Cryptographically Generated Addresses (CGA) (Aura, 2005).

Other authentication systems such as IPSec (Arkko, Devarapalli, & Dupont, 2004), and AAA (Authentication, Authorisation and Accounting) have also been investigated (Rigney, Willens, Livingston, Merit, & Simpson, 2000) (Appendix B). From the analysis of these, authentication systems, it has highlighted the importance of implementing a good security architecture. An additional consideration concerns the resources needed to run these architectures beyond what is required by the mobile devices.

- **Integrity**

The integrity of the data is another important security cousideration when dealing with attacks and other security risks. This service ensures that any *BU* messages sent out will contain the same binding data upon arrival. The hash algorithm is adopted to ensure data integrity. By default, the Mobile IP provides support for the MD5 (Message-Digest) algorithm (RFC 1321) - an algorithm that can be used for integrity verification and secret-key authentication (Rivest, 1992). Currently, an IPSec security association is applied between the *HA* and the *MN* as a form of integrity protection with *BU*s and acknowledgement authentication (Kroeselberg, Patil, Tschofenig, & Korhonen, 2012). The Encapsulation Security Payload (ESP) (Kent, 2005b) header must be supported by both the *MNs* and the *HA*. It must be used during the transport mode and feature a non-null payload authentication algorithm (Kent, 2005a).

- **Confidentiality**

Any data passing through the Internet can be intercepted and be falsified. With the *BUs*, some security breaches could potentially happen. These breaches include data packet interception, resulting in an intruder eavesdropping on the content being transmitted, thus, violating a user's confidentiality; or the modification of transmitted packets for malicious purposes.

2.3.3 Cryptography

Communication between nodes is kept secret through cryptography even if third-party interception occurs. Mathematical algorithms that scramble the data serve this purpose. Data with the relevant information is placed into strings by the algorithm for the purpose of data encryption. In cryptography, the transmitted data is scrambled into a message with packets undecipherable by the attacker. Only the parties possessing the relevant keys have the ability to decrypt the message.

There are two types of algorithm in cryptography: symmetric, and asymmetric algorithms (Appendix A). In symmetric algorithms, the encryption and decryption processes use the same key. It is easy to implement cryptography, but the difficulty is in maintaining data security and secrecy during data exchange. Many security solutions such as the Diffie-Hellman algorithm and the traditional RSA public key, use cryptography to provide security for *BU* messages (Xiao, 2003). The Diffie-Hellman algorithm communicates using a key (Boyko, MacKenzie, & Patel, 2000). A major drawback, however, is that the Diffie-Hellman algorithm cannot determine the validity of the keys sent by the sender or the receiver (Werapun & Unakul, 2004).

Asymmetric cryptography helps in mitigating the problems encountered with the symmetric algorithm. In the asymmetric algorithms, two keys are used - a private key and a public key. The private key is only available to the owner, while the public key is available to everyone. This type of algorithm runs complex mathematical formulas with exponentiation and large prime numbers (Rivest, Shamir, & Adleman, 1978). In order for someone to send a message to another person, he must first obtain the receiver's public key to encrypt the message. The encrypted message is then forwarded to the recipient where it has to be decrypted with the recipient's private key to obtain the plaintext. Traditional public key cryptography is applied in the RSA method. This security model is popular in various types of networks, including MIPv6. The model, however, consumes a large amount of bandwidth because of the recommended RSA key length of 1024 bits to improve security. The security levels of an Elliptic Curve Cryptography (ECC)-based system with a 160-bit prime order is similar to the 1024-bit modulus *p* DSA model, as well as the 1024-bit modulus *n* RSA model (Boneh & Franklin, 2003; Menezes, 1993). In this research, the Elliptic Curve Cryptographic system (Miller, 1986) and the Elliptic Curve Digital Signature Algorithm (ECDSA) (Werapun, & Unakul, 2004) (Appendix A) are adopted to develop the protocol to provide protection for the *BU* messages as they have same security level with less number of bit compare other algorithm.

2.4 Return Routability

Return Routability (RR) is a security protocol based on RFC 3775 and adopted by the Internet Engineering Task Force (IETF). The RR protocol is used to conduct *HoA* and *CoA* checks. Such checks are necessary for a node to confirm the presence of another node that answers to the packets traveling to a certain address (Hasan & Hassan 2013). A positive reply confirms the existence of a node at the given address (Figure 2.9). Return routability checks include message pairs such as (Home Test, BU) and (Care-of Test, BU). To activate the test packets, Home Test Init (*HoTI*) and Care-of Test Init (*CoTI*) are needed, while the *BU* message plays the role of a combined routability response for both tests (Johnson et al., 2004; Perkins et al., 2011; Ren et al., 2006).

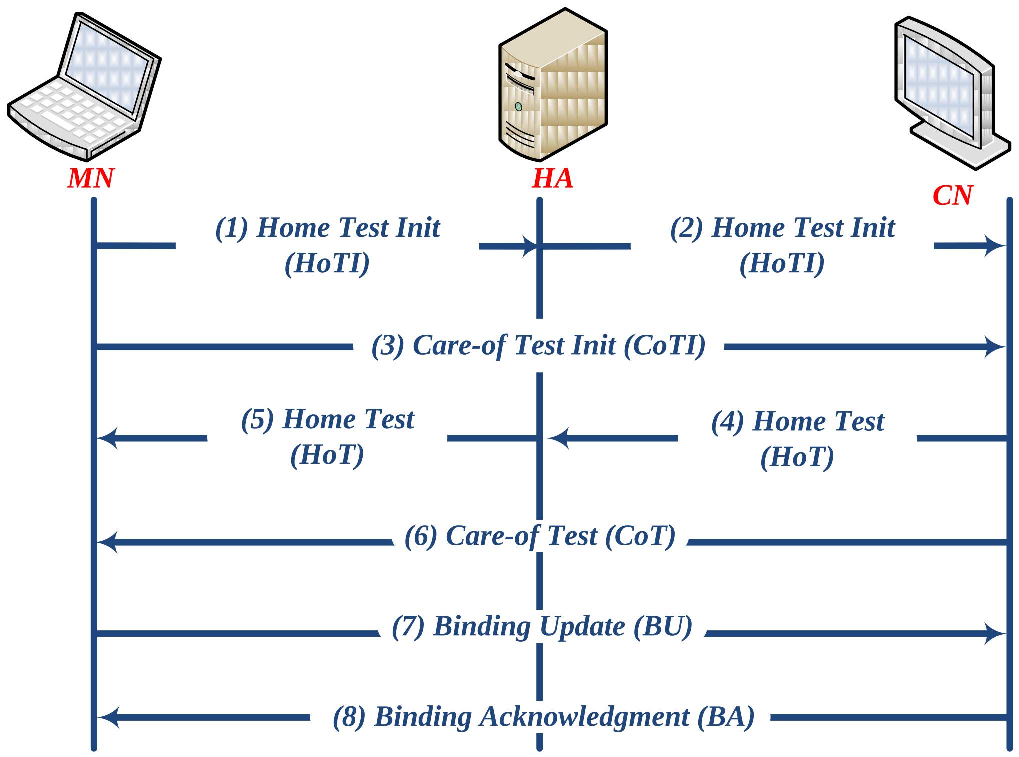

MN
HA
CN
(1) Home Test Init (HoTI)
(2) Home Test Init (HoTI)
(3) Care-of Test Init (CoTI)
(5) Home Test (HoT)
(4) Home Test (HoT)
(6) Care-of Test (CoT)
(7) Binding Update (BU)
(8) Binding Acknowledgment (BA)

Figure 2.9: Return Routability Protocol

Home Test Init

The *CN* will receive a Home Test Init (*HoTI*) message from the *MN* requesting the Home Keygen token through the *HA*. The message content can be summarised and presented accordingly:

Source Address =	home address
Destination Address =	correspondent address
Parameters:	home init cookie

The *CN* receives the *HoTI* message which contains the *HA* of the *MN*. The *CN* also receives the Home Init Cookie from the *MN* which it needs to reply later on.

The reverse tunnelled process is done to the *HoTI* message via the *HA*. The values of the cookies are 'remembered' by the *MN* thus giving it the assurance that the *CN* is processing the protocol messages.

Care-of Test Init

The *CN* will receive a Care-of Test Init (*CoTI*) message from the *MN* requesting the Care-of Keygen token directly, without going through the *HA*. The message content can be summarised and presented accordingly:

Source Address =	care-of address
Destination Address =	correspondent address
Parameters:	care-of init cookie

The *CN* receives the *CoTI* message which contains the *CoA* of the *MN*. The Care-of Init cookie is also sent along by the *MN* which must be returned by the *CN*. The *CN* receives the *CoTI* Message directly from the *MN*.

Home Test

To reply the Home Test (*HoT*) message, the *HoTI* message is sent back through the *HA*. The message contains:

Source Address =	correspondent address
Destination Address =	home address
Parameters:	home init cookie
	home keygen token
	home nonce index

Once the *HoTI* message has been received by the *CN*, the Home Keygen token is generated with the following information:

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

Where | represents concatenation. The last '0' of the HMAC_SHA1 functions represents the single zero octets used to differentiate the Home cookies and the Care-of cookies. And the first 64-bit of the MAC is used to construct the Home Keygen token. This Home Keygen token ensures that the messages sent to the *MN's HA* can be received. In order for the *CN* to verify the Home and Care-of nonces that it generated, the production of the Home Keygen token will use *Kcn*. It is used so that the *CN* won't be forced to recall the list of all token that has been handed out.

The home network is used by the *MN* to send the *HoT* message. It is presumed that the message will be tunnelled by the *HA* to the *MN*. This means that the *BU* should already be sent to the *HA* by the *MN* in order for the *HA* to receive and authorised the *MN's CoA* just before the RR procedure.

The home init cookie of the *MN* is returned via the *HoT* message. It is used to ensure that the message actually comes from a node on the route existing between the *CN* and the *HA*.

The *MN* receives the home nonce index which allows the *CN* to find the nonce value used to create the home keygen token more efficiently.

Care-of Test

The reply to the Care-of test (*CoT*) message is sent directly to the *MN* and not through the *HA*. This message contains the following:

Source Address =	correspondent address
Destination Address =	care-of address
Parameters:	care-of init cookie
	care-of keygen token
	care-of nonce index

Once the *CoTI* message arrives at the *CN*, the Care-of Keygen token is generate with the following details:

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

The final '1' inside the HMAC_SHA1 represents a single octet which contains the hex value of 0x01. This value will be used to discern the home cookie and care-of cookie from each other. This token is created using the first 64-bit of the MAC and is sent as the *CoA* to the *MN*. And to ensure that the message actually comes from the node on the route the *CN*, the Care-of Init message from the *CoTI* message is sent back.

In order to identify the nonce used for the Care-of Keygen token, the Care-of nonce is used. The indices of the Home and Care-of nonces may be similar or it may be different to the Home and Care-of Test messages.

The RR procedure is complete once the *HoT* and *CoT* messages are received by the *MN*. Now the *MN* has the necessary data to send a *BU* to the *CN*. A 20 octet binding key K_{bm} is generated from the hashed token by the mobile node. It contains the following details:

$$\mathbf{K_{bm}} = \text{SHA1 (home keygen token | care-of keygen token)}$$

Until the *CN* receives the *BU* from the *MN*, it will not create any state specific for that *MN*. The binding management key (K_{bm}) is not maintained by the *CN* and is created only when a new nonce indices along with the *MN* address has been given. Once the K_{bm} has been created by the *MN*, the verifiable *BU* can now be sent to the *CN*.

Binding Update

The binding management key (K_{bm}) is created by the *MN* using the keygen tokens mentioned in the previous section. This is done in order to authorise a *BU*. The *BU* content has the following details:

Source Address =	care-of address
Destination Address =	correspondent address
Parameters:	home address (within the Home Address destination option if different from the Source Address)

sequence number (within the Binding Update message header)

home nonce index (within the Nonce Indices option)

care-of nonce index (within the Nonce Indices option)

First (96, HMAC_SHA1 (Kbm, (care-of address | correspondent | BU)))

The nonce indices option is contained in the *BU*. It indicates which home and care-of nonces can be use by the *CN* to recomputed the K_{bm}. Once the MAC has been verified by the *CN*, the Binding Cache entry can now be created for the *MN*.

Binding Acknowledgement

In certain cases, the *BU* is acknowledge and accepted by the *CN*. The message of the content is as shown below:

Source Address =	correspondent address
Destination Address =	care-of address
Parameters:	sequence number (within the Binding Update message header)
	First (96, HMAC_SHA1 (Kbm, (care-of address |correspondent | BA)))

The Binding Acknowledgement (*BA*) and *BU* contain the same sequence number. The source address field of the IPv6 contains the value of the *BU* and is normally used for the *CoA* for the binding procedure. However, an Alternate *CoA* mobility option in the *BU* may be included to create a different *CoA*. If such message was received by the *CN* and the authorisation method used was the RR procedure, the *CoTI* message and the Care-of message must be performed according to the address in the alternate *CoA* option and not the source address. The information gathered in this test must be used for the nonce indices and MAC value.

- **Security Analyse of Return Routability**

In the RR protocol, the two token exchanges confirm that the *MN* is active at its corresponding *HoA* and *CoA*. The *MN* activity at both the *HoA* and the *CoA* is important. The RR protocol security is also important because it clearly depends on the *MN* secretly sharing K_{bm} with *CN*, which is, in turn, is dependent on, at least, the secrecy of one token. The RR protocol security, however, is particularly fragile because anyone with access to the *CoT* and the *HoT* messages can obtain the tokens. Although the RR protocol was originally designed with sufficient Mobile IP support in mind, and with the hope that it would not incur new major security problems, protection against attacks, even prior to the introduction of IP mobility protocols, was somewhat overlooked (Aura, 2004; Johnson et al., 2004; Nikander et al., 2003). The protocol does not defend against intruders observing the *CN–HA* path because such intruders would be able to actively attack an *MN* at its home location, anytime (Deng et al., 2002). Attacks on MIPv6 are usually easier to carry out than attacks on IPv6. For example, for an intruder to carry out a false binding update attack, he should always be on the *CN-HA* path on a base IPv6 without mobility (equivalent to an *MN* at its home link in MIPv6). To redirect the *CN* traffic to attach the *MN* to a malicious node, the intruder must gain control of a router or a switch on the *CN–HA* path. If the malicious node (disguised as an *MN*) wishes to continue its session with the *CN* after taking over the session from the *MN*, it must continue to collaborate with the router. In such a case, the router would tunnel the *CN* traffic to the malicious node, and vice versa. Another example, assume that the *MN_1* and the *CN* are communicating and the intruder wishes to redirect the *CN* traffic to his associated *MN_2* (Figure 2.10). In this case, the intruder would monitor the *CN–HA* path (i.e., anywhere from the *MN_1*'s home network to the *CN*'s network) to obtain the *HoT*, after which it would extract the Home token and send it to *MN_2*. When the *MN_2* receives the Home token, it would forward a *CoTI* to the *CN*, and the *CN* would reply with a Care-of token (*CC*). The *MN_2* can then hash the two cookies for a valid session key to use to send a *BU* message to the *CN* on behalf of the *MN_1*. The *CN* would accept the *BU*, and the *CN* would finally direct data traffic to *MN_2*.

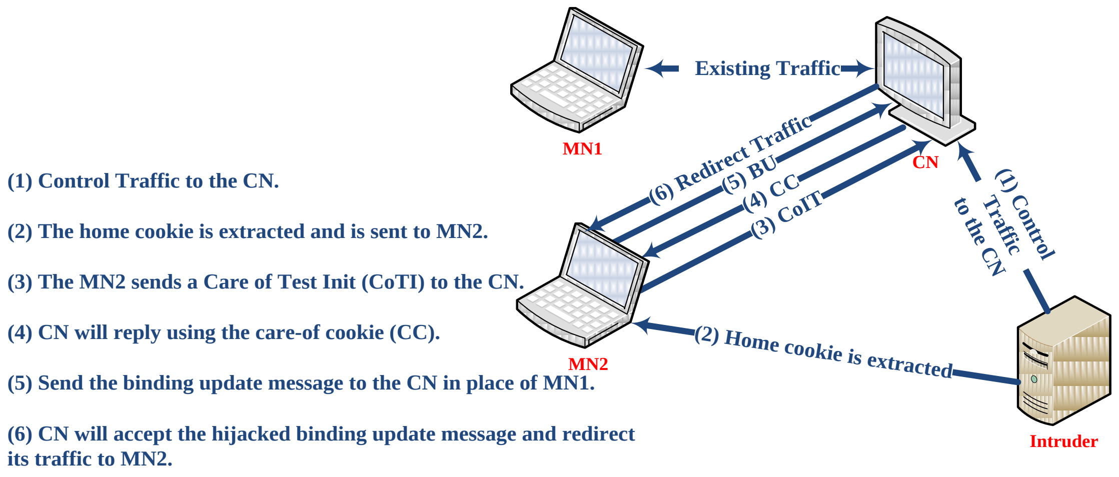

MN1
Existing Traffic
CN
(6) Redirect Traffic
(5) BU
(4) CC
(3) CoIT
(1) Control Traffic to the CN
MN2
(2) Home cookie is extracted
Intruder
(1) Control Traffic to the CN.
(2) The home cookie is extracted and is sent to MN2.
(3) The MN2 sends a Care of Test Init (CoTI) to the CN.
(4) CN will reply using the care-of cookie (CC).
(5) Send the binding update message to the CN in place of MN1.
(6) CN will accept the hijacked binding update message and redirect its traffic to MN2.

Figure 2.10: Break the RR Protocol by Attacker

A DoS attack is a good example of an attack in a base IPv6 without mobility, where many compromised systems attack one target. A malicious *MN* flooding attack on MIPv6 can be carried out in many ways on a node or network. For example, a *MN* could begin communicating intensively with the *CN*s and shift to the victim's network or its exterior boundary. The malicious node would then send *BU* messages via the RR protocol to redirect traffic from the *CN*s to the victim's network. No special software or network setup is needed for this attack, thus, making it easy for an intruder to disrupt the operations through the RR protocol. In another scenario, if a *CN* offers on-line services to many mobile clients, an intruder could effortlessly eavesdrop on the RR protocol messages and gather cookies from its location between the *CN* and the Internet. The intruder could then redirect the traffic to any mobile client by randomly hashing the cookie pairs to create session keys, and deliver the *BU* messages to the *CN*. The result would be detrimental to the *CN*'s services.

The Early Binding Updates protocol (EBU) was developed with the aim of improving the RR protocol (Vogt, Bless, Doll, & Kuefner, 2005). This protocol is used to send message fractions prior to handover. Both protocols, however, have similar security architectures, and this makes the EBU to be prone to the same types of attacks, mentioned above. The Dual Identity Return Routability protocol may be used to enhance the RR by having a second identity in the same device (Georgiades, Luo, Lasebae, & Comley, 2006). The second identity helps to transport half of the security tokens required to generate a *BU* key. It sends data to both identities, each of which has their own *HA*, and the *MN* amalgamates them to generate the key. In this method, the *MN* needs two separate IP addresses, as a single address will not make it efficient enough.

2.5 Problem Formulation

Binding updates (*BUs)* have several security weaknesses such as: data packet interception which infringes on the user's confidentiality as attackers eavesdrop on packet content; and modification of transmitted packets to suit the attackers' malicious aims. On the other hand, Mobile IP security vulnerabilities include address spoofing, denial of service attacks, and data traffic redirection.

Certain protocols like IPSec and RR are specifically designed to improve *BU* security. With IPSec, *BU* messages are protected between the *HA* and its *MNs*, while RR relies on securing the communication path between the *CN* and the *MN*. However, both protocols have similar security vulnerabilities, as mentioned in the Section 2.3. Based on our review of the researches on *BU* security, many protocols have been developed to address the issue. These protocols include: Cryptographically Generated Address (CGA) (Aura, 2005); Early Binding Update (EBU) (Vogt et al., 2005); Purpose-Built Key (PBK) (Mankin, Bradner, & Schiller, 2003); CAM-Child-Proof Authentication (Lowe, 1997); Unauthenticated Diffe-Hellman-based Binding Update (Le & Faccin, 2001); Enhanced Route Optimization (Arkko, Haddad, & Vogt, 2007); Certificate-based Binding Update (Deng et al., 2002); shared key (Dupont & Combes, 2006; Perkins, 2006); Ticket-based (Koo, Koo, & Lee, 2006) and the BAKE/2 protocol (Roe et al., 2002). Their main aim is to defend *BU* messages against eavesdropping, modification, and DOS attacks. These protocols can be classified into Infrastructure-less, and Infrastructure-based protocols. The earlier security solutions will be reviewed in the next chapter.

2.6 Summary

This chapter presents an overview of the MIPv6 protocol. Using the security feature found in MIPv6, the potential attacks and security threats were identified along with security services and requirements needed to minimise such problems. The chapter also discusses Return Routability (RR), the standard framework for ensuring security in IP communication.

Chapter 3 : ANALYSIS OF BINDING UPDATE PROTOCOLS IN MOBILE IPv6

3.1 Introduction

The main aim of protecting *BU* message in the MIPv6 protocol is to ensure a secure connection between the *MN* and the *CN*. This chapter presents the existing protocols that are used to protect Binding Update (*BU*) messages, and it also highlights the strengths and weaknesses of these protocols vis-a-vis the security threats to *BU* message and the security services to address the threat, covered in Chapter 2. In the rest of this chapter, each existing protocol under the Infrastructure-less and Infrastructure-based categories will be reviewed. Figure 3.1 illustrates this categorisation of the protocols. Before discussing the existing protocols, we will present briefly the cryptographically-generated IPv6 addresses (See Section 3.2) as they are essential elements of some of the Infrastructure-less protocols. Section 3.3 discusses the Infrastructure-less protocols, while Section 3.4 discusses the Infrastructure-based protocol. Section 3.5 is the chapter summary.

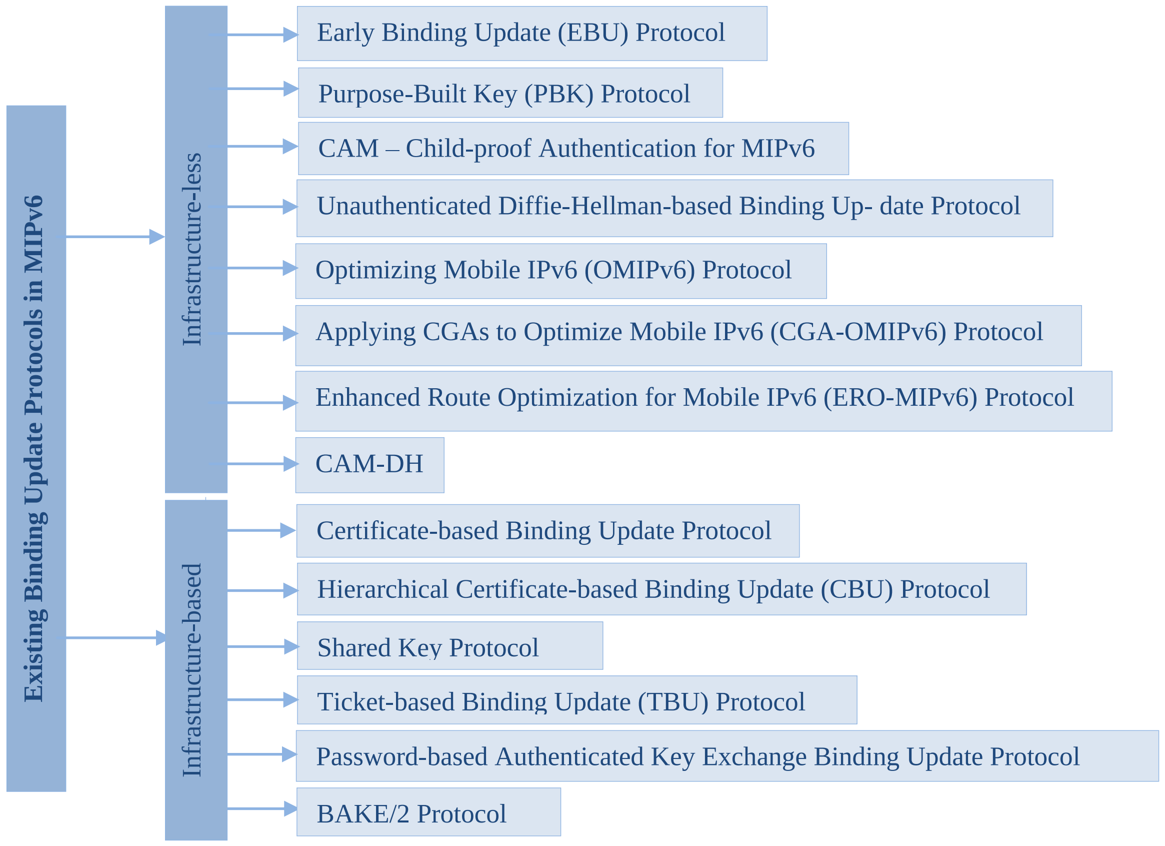
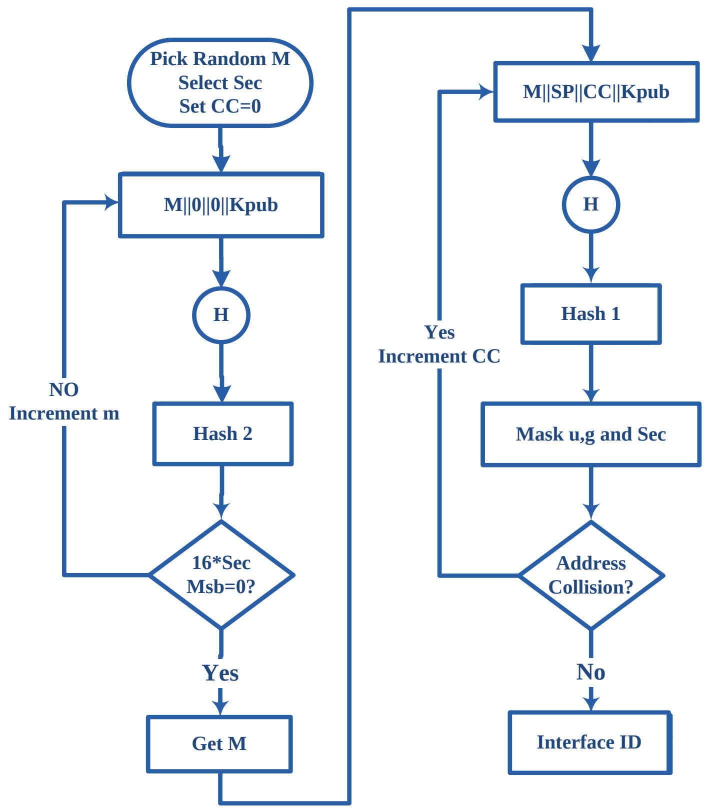

Existing Binding Update Protocols in MIPv6
Infrastructure-less
Infrastructure-based
Early Binding Update (EBU) Protocol
Purpose-Built Key (PBK) Protocol
CAM – Child-proof Authentication for MIPv6
Unauthenticated Diffie-Hellman-based Binding Up- date Protocol
Optimizing Mobile IPv6 (OMIPv6) Protocol
Applying CGAs to Optimize Mobile IPv6 (CGA-OMIPv6) Protocol
Enhanced Route Optimization for Mobile IPv6 (ERO-MIPv6) Protocol
CAM-DH
Certificate-based Binding Update Protocol
Hierarchical Certificate-based Binding Update (CBU) Protocol
Shared Key Protocol
Ticket-based Binding Update (TBU) Protocol
Password-based Authenticated Key Exchange Binding Update Protocol
BAKE/2 Protocol

Figure 3.1: Protocols under Infrastructure-less and Infrastructure-based Categories

3.2 Cryptographically Generated Addresses (CGA)

The Cryptographically Generated Addresses (CGA) is an essential element of some of the Infrastructure-less protocols. It includes the IPv6 address – a 64-bit address generated by hashing the address owner's public key (Aura, 2005). CGA-based authentication techniques that are used to protect the IP-layer signalling protocols include the neighbour discovery and mobility protocols. The CGA is based on the RFC 3972 standard for securing the neighbour discovery protocol (Bos, Özen, & Hubaux, 2009; Tuomas, 2005). To authenticate that a message originates from a specific CGA address, the address owner signs the message with his private key. The message, the public key, and the signature from the CGA address, are then sent to the verifier. The verifier uses the public key to verify the signature on the message. If the signature is valid, the verifier is assured that the message was sent by the owner of the specific address.

In those techniques that use the CGA, the public key used for authentication is already provided in the IPv6 addresses. Thus, there is no necessity to use trusted third parties of PKI to prove that the IPv6 address belongs to the owner of the public key. However, this CGA-based method has several weaknesses (Cao, Deng, Ma, & Hu, 2007): (i) there is no guarantee that the owner is reachable at the stated address, for example, a malicious hacker can simply use a cryptographically-generated non-used address of its own public key together with the victim’s network as its own subnet prefix; (ii) even though it can effectively prevent impersonation of a valid IPv6 address by any attackers, it cannot prevent attacks on the entire network via data redirection to illegal address; (iii) there are limitations in the technique, in view of the heavy reliance on the computer to perform signature key calculations and verification of public-key operations, particularly, the digital signatures; this could lead to denial-of-service attacks, especially on devices that have lower computation power such as mobile devices, or when the participant has to perform simultaneous digital signature verification for a large number of peers; (iv) the public key and the signature of the address owner are carried by a messenger for generating an address cryptographically, thus, consuming a certain amount of bandwidth. The procedure for generating an IPv6 address using CGA is illustrated in Figure 3.2 (Bos, Özen, & Hubaux, 2009).



Data	Notation
IPv6 Address	IPv6
Subnet Prefix	SP
Interface Identifier	Interface ID
Public Key	K_{pub}
Private Key	K_{priv}
Digital Signature	Sign
Modifier	M
Collision Count	CC
Security Parameter	Sec
u , g flags	u , g

Figure 3.2: Data flow of the Address Generation in Cryptography Generated Address(Bos, et al., 2009)

1. Set the modifier to a random 128-bit value. Select the security parameter Sec and the collision count to zero.

2. Concatenate the modifier, 64+8 zero bits, and the encoded public key. Execute the H algorithm on the concatenation. The leftmost 112 bits of the result are Hash2.

3. Compare the 16*Sec leftmost bits of Hash2 with zero. If they are all zero (or if Sec=0), continue with Step (4). Otherwise, increment the modifier and go back to Step (2).

4. Concatenate the modifier, subnet prefix, collision count and encoded public key.Execute the H algorithm on the concatenation. The leftmost 64 bits of the result are Hash1.

5. Form an interface identifier by setting both reserved bits u and g in Hash1 to 1 and the three leftmost bits to Sec.

6. Concatenate the subnet prefix and interface identifier to form a 128-bit IPv6 address.

7. If an address collision with another node within the same subnet is detected, increment the collision count and go back to step (4). However, after three collisions stop and report the error.

Address ownership verification is accomplished by executing the following steps given the IPv6 address, collision count and the modifier (Bos, et al., 2009):

1. Check that the collision count is 0, 1 or 2 and that the subnet prefix is equal to the subnet prefix of the address. CGA verification fails if either check fails.

2. Concatenate the modifier, subnet prefix, collision count and the public-key. Execute the H algorithm on the concatenation. The 64 leftmost bits of the result are Hash1.

3. Compare Hash1 with the interface identifier of the address. The differences in the two reserved bits u and g and in the three leftmost bits are ignored. If the 64-bit values differ (other than in the five ignored bits) then CGA verification fails.

4. Concatenate the modifier 64 + 8 zero bits and the public key. Execute the H algorithm on the concatenation. The leftmost 112 bits of the result are Hash2.

Read the security parameter Sec from the three leftmost bits of the address' interface identifier. Compare the 16*Sec leftmost bits of Hash2 with zero. If any one of these bits is not zero, CGA verification fails; otherwise verification succeeds. If Sec = 0 verification never fails at this step.

These weaknesses could be overcome by combining the CGA, RR and authentication verification methods to create a security solution - the Distributed Authentication Protocol (Georgiades, Luo, Lasebae, & Comley, 2006). In this proposed technique, however, the *MNs* require more processing power, thus, rendering the mobile devices to be less efficient (Rossi, et al. 2012). CGA can also be combined with the Optimizing Mobile IPv6 (OMIPv6) protocol to produce CGA-OPMIPv6, a new protocol which will be discussed in Section 3.3.6.

3.3 Infrastructure-less Binding Update Protocols

In the Infrastructure-less Binding Update protocols, the *MNs* and the *CNs* belong to different administrative domains. These types of protocols are widely applied, and are for general usage on the Internet. The Infrastructure-less protocols are used across the world because they allow authentication between the *MNs* and the *CNs* without the need for any security infrastructure.

3.3.1 Early Binding Update (EBU) Protocol

Early Binding Update (EBU) was proposed by Vogt et al. in 2005 to enhance the RR procedure, as discussed in Chapter 2. EBU reduces the long registration delay caused by the RR, by shifting the order of the *HoA* and the *CoA* tests. The *HoA* test is executed prior to handover, while the *MN* uses the old *CoA*. After handover, the *CoA* test is carried out in parallel with data transfer. EBU uses the Credit-Based Authorisation technique (Vogt, 2005; Vogt et al., 2005) to limit the amount of data that a *CN* sends to the *MN* and its unconfirmed *CoA*, and it runs the *CoA* test concurrently. Figure 3.3 shows the exchange of messages between the nodes in the EBU protocol (Modares, et al.2013).

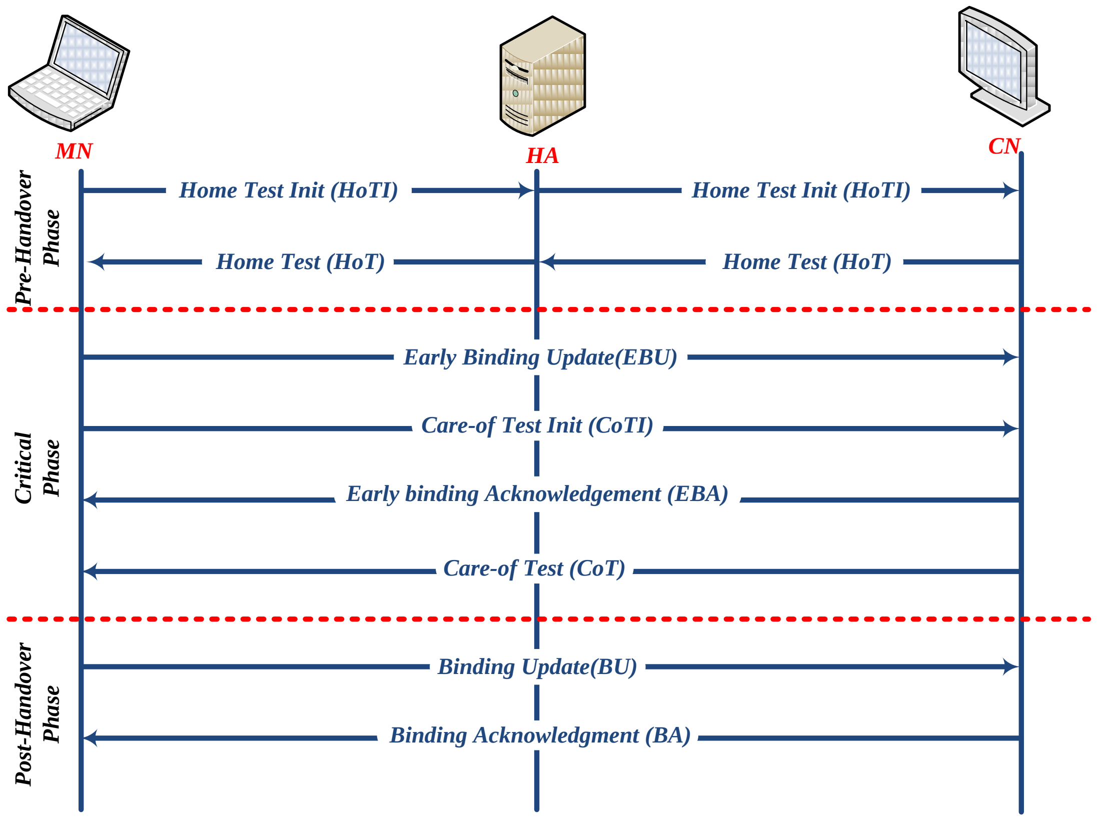


Figure 3.3: Early Binding Update (EBU) Protocol (Elshakankiry, 2010)

EBU enhances the RR and reduces the correspondent registration delay. However, one to two additional messages are needed, and if the *MN* needs to run the *HoA* test periodically, the signalling overhead increases. Implementing Credit-Based Authorisation in the *CN* raises complexity and EBU will still be vulnerable to attacks applicable to the RR (Elshakankiry, 2010).

3.3.2 Purpose-Built Key (PBK) Protocol

The Purpose-Built Key (PBK) protocol was developed to authenticate and verify the network communication initiator (Mankin, Bradner, & Schiller, 2003). Here, the data packets must arrive continuously from the same source in order for the protocol to run, but the initiator identity is not crucial. As such, the protocol is suitable in confirming that a *CN* has to acknowledge that the correspondent's registration details originate from the *MN* that started the communication. During the registration of any correspondent, the PBK protocol needs four messages.

Compared to the RR procedure, this method, in fact, reduces the signalling overhead. Registration delay, however, remains because of the 1.5 round-trip between the *MN* and the *CN*. Also, there is higher vulnerability to DoS attacks, leading to an overconsumption of resources because the process requires verification of two-digital signatures during protocol execution. An additional weakness is its vulnerability to MITM attacks in the initialisation process. The hash key or public key can be intercepted during the initialisation process if an attacker listens in on the transmission path, and then transmits a different key. In the PBK protocol, the authentication of the *CoA* is based on the address reachability test, but it does not authenticate the *HoA* (Elshakankiry, 2010).

3.3.3 Child-proof Authentication for MIPv6 (CAM)

CAM is a security system that authenticates the *BUs* in MIPv6 (O'shea & Roe, 2001). It functions by incorporating a one-way hash function of its public key into the *MN*'s chosen *HA*. Demonstrating knowledge of the corresponding private key validates the address ownership. Therefore, if a key pair must correspond to the hash function, it is difficult to falsify (Bidan, 2013). A CAM node creates a key pair which is stored locally upon initialization, after which a *HA* has to be chosen. A 64-bit routing prefix is obtained by listening for the local router advertisements. In most cases, the interface ID is obtained from the media access control address (MAC address) because it is economical, and globally unique. However, the CAM protocol alone cannot ensure node reliability, thus, other protocols are needed to compensate for this deficiency. Moreover, in a situation where an attacker jams a mobile's *BU* but delivers a binding acknowledgement that cuts the *MN* off from the *HA* and the correspondent, the protection alternatives would be IPSec or some variants of CAM (Lowe, 1997). The advantage of CAM is that it reduces both the signalling overhead and the registration delays. In CAM protocol, only one message is required between the *MN* and the *CN*. It checks the authenticity of the *HoA* by using the CGA-based address and the reachability test. However, a drawback is that the protocol will then be open to *MN* flooding attacks because it is unable to verify the authenticity of the *CoA*. The CAM protocol can detect any replay *BUs* using time stamps. In addition, both the *MN* and the *CN* must possess a synchronised clock, and this gives rise to the same security problem as for the CGA-based technique, mentioned above.

3.3.4 Unauthenticated Diffie-Hellman-based Binding Up- date (UDHBU) Protocol

The Diffie-Hellman key exchange protocol is used in the Unauthenticated Diffie-Hellman Binding Update (UDHBU) protocol (Le & Faccin, 2001). It provides a session key that is shared by the *MN* and the *CN*. The two nodes will use this session key for authentication during correspondent registration. The UDHBU protocol is applied at the start of correspondent registration, as well as at subsequent registrations. It requires four messages in the first registration, but only two for the subsequent correspondent registrations. This reduces delays because only a one-way registration is necessary for subsequent correspondences between the *MN* and the *CN*. Moreover, the *CN* only performs two exponential calculations in the initial registration phase. The protocol also allows the long-term use of the shared key between the *MN* and the *CN* because redundant message signalling is reduced – usually, due to the RR procedure, which constantly generates new shared keys.

The UDHBU protocol is particularly prone to flooding attacks at the *MN* due to its inability to authenticate the *CoA*. MITM attacks and false binding update attacks can also adversely affect this protocol because of the unauthenticated DH key. For example, an attacker might plant itself on the path and intercept the DH public value, and relay its own version during the first correspondent registration. It is also possible for the attacker, instead of a legitimate *MN*, to initialise the protocol. If this happens, the attacker has an opportunity to capture the session key, and subsequently use it to transmit false binding updates.

3.3.5 Optimize Mobile IPv6 (OMIPv6) Protocol

The combination of the RR method with the Diffie-Hellman key exchange protocol (Diffie & Hellman, 1976) produces the Optimize Mobile IPv6 (OMIPv6) protocol (Haddad & Krishnan, 2004). The two techniques combine to produce a shared key for the sessions between the *MN* and the *CN* via a similar process to the above mentioned UDHBU protocol. OMIPv6 was developed with the aim of including the initial registration phase with the subsequent registration of the correspondents. Its course begins in the initialisation phase of the correspondent registration. Both the *MN* and the *CN* will execute the RR protocol to generate a shared secret key. The two nodes will then exchange the DH key that has been verified by the shared secret key during the previous RR process, for long-term shared key session. OMIPv6 uses exactly the same subsequent correspondent registration as that used by the UDHBU protocol. It takes eight messages and 2.5 round trips between the *MN* and the *CN* for OMIPv6 to register the correspondent. Subsequent registrations only require one-way trips between the *MN* and the *CN* and two messages. This protocol, therefore, gives better protection against DoS attacks compared to that provided by UDHBU. Generally, both OMIPv6 and UDHBU share similar features and have their advantages and disadvantages. The disadvantages include: an attacker is able to intercept the DH public value; and OMIPv6 is prone to MITM attacks due to its inability to authenticate the *CoA*.

3.3.6 CGAs - Optimize Mobile IPv6 (CGA-OMIPv6) Protocol

The CGA-OMIPv6 protocol is a combination of the CGA-based techniques and the RR procedure (Haddad, Dupont, Madour, & Arkko, 2005). The optimisation steps of the MIPv6 protocol are as follows: (1) Public/private keys are self-generated by the *MN*; (2) the *HoA* is configured by the *MN* according to the CGA concept, described in Section 3.2; (3) the *MN* exchanges CoTI and CoT messages with the *CN* to obtain the *CoA* reachability proof when a new *CoA* is requested; (4) following the initial correspondent registration with the *CN*, the *MN* provides the reachability proof along with *HoA* ownership. These steps are carried out by signing the first *BU* message with its own private key, after which the public key is sent with the signed message. The proof can be obtained during the exchange between the CoTI and the CoT.

The CGA-OMIPv6 protocol has two phases: the initial registration of a correspondent and the subsequent registration. In the initial correspondent registration phase, the *CN* authenticates the *MN*’s *HoA* along with the reachability of the *MN*’s *HoA* and the *CoA*. These two nodes can communicate by exchanging secret session keys. Subsequent registration takes place when there is a new *CoA* at the *CN* that needs to be registered by the *MN*. The *CN* can verify the *MN*’s reachability at the new *CoA*. Five messages are sent during the initial phase of the correspondent registration, and four messages are sent during the subsequent registration are passed to the CGA-OMIPv6 protocol. The correspondent registration, between the *MN* and the *CN* requires 1.5 round trips. When a new *CoA* is claimed, a *CoA* reachability test is performed to ensure partial protection against DoS attacks on third parties. The *CN* will use RR to establish the K_{BM}, which is used to verify *BU* message authenticity before any CGA-based address verification algorithm is run. In the initial correspondent registration phase, both the *MN* and the *CN* are also required to carry out two public key operations.

3.3.7 Enhanced Route Optimization for Mobile IPv6 (ERO-MIPv6) Protocol

The functions of EBU and CGA-OMIPv6 (See Sections 3.3.1 and 3.3.6) are combined to produce the ERO-MIPv6 protocol (Arkko et al., 2007). The protocol uses both CGA and address reachability to authenticate the *HoA*. In this protocol, six messages are sent during the first correspondent registration, and four messages are sent in the subsequent registration. For any correspondent registration, the delay time between the *MN* and the *CN* has been optimised to a one-way route by this protocol. In addition to the delay time optimisation function, this protocol has the same advantages and disadvantages as CGA-OMIPv6, as elaborated in Section 3.3.6. The implementation of the Credit-based Authorisation technique in the *CN* is becoming increasingly complex, as in EBU (Elshakankiry, 2010).

3.3.8 CAM-DH Protocol

CGA provides a basis for CAM-DH to secure the *BUs* (Roe et al., 2002). CAM-DH contains a digital signature system in which every *MN* possesses a public key and a private key pair, P_{MN} and S_{MN}. The *MN's HoA* is CGA generated from P_{MN}. CGA presents CAM-DH with a way to bind the *MN's* P_{MN} to its IPv6 addresses. However, CAM-DH has some limitations, one of which is that CAM-DH does not authenticate the *CoA*. Furthermore, an attacker intercepting packets meant for the *CoA* can run the protocol and become the source of *CoA* data overload, even if the real *CoA* owner-host does not wish to partake in the protocol. An alternative to *CoA* authentication is to develop the *CoA* and the *HoA* using the node's public key. This may not succeed, however, if the subnet enforces constraints on the *CoA*. Also, CGA generation involves high amount of computation. The CAM-DH protocol requires large processing power to handle asymmetric cryptography and CGA and it is vulnerable to the problems of scalability and flexibility as well as spoofing, and replies attacks. Moreover, the *MN* incurs a high computation load for each *BU* message that needs a signature to be produced by the *MN*, in addition to signature validation by the *CN* (Kavitha, et al. 2010).

3.4 Infrastructure-Based Binding Update Protocols

Infrastructure-based Binding Update protocols generally require support from some security infrastructure in order to provide protection for the *BU*. This type of protocol provides assurance to the *CN* that the *MN's HoA* is valid and correct, which means that the *MN's HoA* is related to the secret key or the public key of the private/public pair. Hence, it is not necessary to test the *HoA*, leading to reduce delays during registration, and less overhead in signalling.

3.4.1 Public-Key based Protocols

3.4.1.1 Certificate-based Binding Update (CBU) Protocol

A CBU protocol was proposed to provide secure connection in MIPv6 (Deng et al., 2002; Feng, et al., 2005). The CBU protocol authenticates a *MN* and its *HoA* using a certificate. However, the CBU does not address the *HoA* certificate management concerns. It is built around disjointed authentication infrastructures existing within individual or various domains. The Certificate Authority (*CA*) provides each home link subnet prefix with a certificate, and this is not very practical. Flexibility is clearly another issue in this flat-structured trust management, hence, it is impossible to implement it across domains with various infrastructures on a global scale. The *CN* is not responsible for tracking complex individual subnet prefix changes of the home links. To minimize and better manage these shortcomings, a divide-and-conquer tactic is recommended. Another disadvantage of the CBU protocol is that the *CN* cannot ensure that the *MN* is live on its claimed *CoA* in the *BU* message. Fake *BUs* within a *MN’s CoA*, can be transmitted by malicious nodes disguised as *MNs*, could be potential intense attacks (Kavitha, et al. 2010).

3.4.1.2 Hierarchical Certificate-based Binding Update (HCBU) Protocol

The HCBU protocol (Ren et al., 2006) provides additional improvements to CBU in that it asserts the *MN's* ownership of a claimed *CoA*. The signature of a foreign link is obtained with the binding of the *HoA* and the *CoA* whenever the *MN* roams to a foreign link with a newly-configured *CoA*. The *HA's* role covers not only all home-linked *MNs* but other roaming *MNs* from different links, all of which are the nodes in their home links. The home link proves to the *CN* that both the *HoA* and the *CoA* are owned by the *MNs*. All claimed *CoAs* are authenticated, resulting in added security and protection against malicious *MN* flooding attacks on third party nodes. In addition, a trusted third-party or foreign link is required to verify the *CoAs* of the *MNs*. This service is available if an authentication infrastructure is established. The role of the *HA* then becomes more complex because it also covers the roaming *MNs* from other links. Another matter to note is that foreign *HAs* are needed to assist in computing the roaming *MN's* signature, and this would reduce the foreign network throughput. The HCBU protocol has pre-handover phase which will increases the signalling overhead (Yeh, Lo-Yao, et al. 2013).

3.4.2 Secret-Key based Protocols

3.4.2.1 Shared Key Protocol

The Shared key protocol (Dupont & Combes, 2006; Perkins, 2006) is used for authenticating the *BUs* between a *MN* and a *CN* where both nodes share a symmetric key. A *CN* randomly generates a number (a nonce) at regular intervals, then uses the same key to nonce all *MN*s it communicates with. The node, therefore, does not have to generate or store new ones for fresh mobile communication. The Correspondents can differentiate between the nonces because their values have a subscript, and if *'+1'* is added; the values can be verified and compared with the previous messages. *CNs* store current and old nonces, but can dispose of older values, while messages with old nonces are abandoned as replays. The keys can maintain fixed values or get repeatedly updated. Updating both a key and a nonce can be done simultaneously, in which case, the subscript would signify the key and the nonce, together. The Shared key protocol requires only two messages, and this reduces signalling overhead when compared to the RR procedure. However, the Share key protocol is unable to authenticate the claimed *CoA*, and it is vulnerable to flooding attacks by malicious *MN*s. The *MN* is required to store different confidential information of each *CN*, and in turn, the *CN* is required to do the same for each *MN*. In order to protect itself against replay attacks, the *CN* has to keep track of the latest value of sequence numbers in the *BU* message sent from the *MN*. If the value received from the *MN* is not properly maintained by the *CN*, then the chances of it being fooled into accepting a replay *BU* message is very high (Suresh, et al. 2013).

3.4.2.2 Ticket-based Binding Update (TBU) Protocol

In the TBU protocol (Koo, Koo, & Lee, 2006), communication between the *MN* and the *CN* is protected using the IPSec ESP tunnel. Other requirements include extending the *HA's* role to verify the *MN's HoA* validity, establishing a ticket as well as a secret key between the *MN* and the *CN*, and providing a mutual authentication system for both nodes.

The TBU protocol is divided into two parts: the initial correspondent registration, and the subsequent registration. Only two messages going one-way are required by the TBU protocol for subsequent correspondent registration between the *MN* and the *CN*. In this way, both signalling overhead and registration delay are reduced to a minimum. Security is further enhanced because *HoA* ownership is ensured. For example, return-to-home spoofing attacks are greatly reduced, and this too causes a reduction in redundant signalling messages of frequently-generated shared keys during the RR procedure, as both the *MN* and the *CN* are provided with a long-term secret key and a ticket. During the initial registration of the correspondents, the TBU protocol requires four messages and one-way communication between the *MN* and the *CN*. A one-way time between the *CN* and the *HA* is included. The TBU protocol, the *MN*, *HA* and *CN* clock must be fully synchronised as it relies on timestamps to recognise replay messages. In the TBU, there is greater complexity of the *MN* correspondent registration at the *HA*, nevertheless, this protocol is still open to *MN* flooding attacks due to its inability to authenticate the requesting *CoAs* (Koo & Lee, 2007).

3.4.2.3 Password-based Authenticated Key Exchange (PAK-based) Binding Update Protocol

For the *BU* protocol based on PAK (Yoon, Kim, Hong, & Youm, 2006), both the *MN* and the *CN* are required to share the same password. The *MN* and the *CN* can authenticate each other using an optimised PAK scheme, allowing future correspondent registration one-time binding of the established management key, as well as authenticating the validity of the *MN's HoA*. In the PAK-based *BU* protocol, four messages are exchanged between the *MN* and the *CN* during the correspondent registration (Lee et al. 2013). The optimised PAK scheme is in the first two messages, whereas, the Diffie-Hellman based password authentication key exchange scheme is used by both the *MN* and the *CN* (Boyko et al., 2000). The remaining two messages are the standard *BU* and *BA* messages that are protected by the PAK scheme's optimised key. Four messages are needed by the PAK-based *BU* protocol, as well as about 1.5 round trips. Verification of the *MN's* reachability at the *CoA* is carried out as protection against malicious flooding attacks on the *MN*, and other third parties. Security is further strengthened as the *HA* ownership is verified, and because the password is set according to the *MN's HoA*. However, both the *MN* and the *CN* must each store different passwords. The two nodes must also compute two different exponential calculations for every correspondent registration.

3.4.2.4 BAKE/2 Protocol

The BAKE/2 protocol (Roe et al., 2002) facilitates the dynamic establishment of a shared secret key by expanding the shared key protocol. BAKE/2 is appropriate for communication between nodes which have been secured against eavesdropping, or secured by IPSec - something not offered by this protocol. A weakness of BAKE/2 is that it can be breached by attacks somewhere between the *HA* and the *CN* (Gunderson, 2008). The protocol does not offer protection against an attacker who can monitor the *HA* to *CN* route. However, it can protect the *CN* against denial of service attacks. This conserves resources as there is no need for large amount of processing power to authenticate the messages. This protocol facilitates communication between a *MN* and a non-mobile server, but may not be suitable for communication with a mobile server. This protocol provides protection against DoS attacks, in which the attacker uses the victim’s *CoA* to redirect high bandwidth traffic to it.

3.5 Summary

This chapter highlights several existing protocols used to protect *BU* messages in MIPv6. The protocols are divided into two different categories: Infrastructure-less and Infrastructure-based protocols. A comparison of these two categories shows that the Infrastructure-less protocols are more secure than the Infrastructure-based protocols due to centralised nature of the information. This is a big disadvantage and not secure, if a third-party directory has been successfully breached, then all the node information is available to the attackers. As a result, we decided to work on Infrastructure-less protocols and to protect *BU* between *MN* and *CN*. Moreover, in the review of previous works, the security threats, and the need to authenticate the *HoA* and *CoA* have highlighted as a vulnerability of the *BU* messages.

In general, the results of the analyses indicate that the protocol for *BU* authentication should be based on the fact that the location information in IPv6 must be authenticated. The *MNs* should authenticate themselves every time they move to foreign networks. In the absence of a proper authentication process, the packet flow from one node to another might be intercepted by a malicious node that could redirect the flow to its own location or to another IP address. As a consequence, service meant for the intended legitimate receiver will be denied. These security problems arise because the current protocols do not have effective authentication methods to verify the user's validity or conceal the *HoA* and the *CoA* location data. These weaknesses would, inevitably, make the protocol vulnerable to malicious attacks. Hence, the main objectives of this study are to propose a method that is able to authenticate the *HoA* and the *CoA* by checking address ownership and the reachability of the *MN*, and to overcome the vulnerabilities of earlier protocols in protecting the *BU* message against the MITM attacks, session hijacking attacks, and DoS attacks, and making the connection between the *MN* and the *CN* secure. The next chapter presents our proposed protocol for securing the *BU* message as it is transmitted between the *MN* and the *CN*. It will also present the framework of the PKBU protocol, and evaluate the ownership and reachability methods to check the correctness of the *MN*'s IP addresses.

Chapter 4 : PRIVATE KEY-BASED BINDING UPDATE (PKBU) PROTOCOL

4.1 Introduction

This chapter presents the Private Key-based Binding Update (PKBU) protocol which provided the motivation for the proposed method to assert the address ownership of the *MN*, and also to verify the reachability of the *MN*. The PKBU protocol offers protection against false binding update attacks, in which an attacker attempts to spoof two different messages, which are sent to the *CN* from two different paths. The PKBU protocol requirement will be explained in Section 4.2, while the protocol design concepts will be outlined in Section 4.3 and Section 4.4. Section 4.5 will discuss the PKBU protocol, in detail. Evaluation of the proposed methods will be presented in Section 4.6.

4.2 Proposed Protocol's Requirement

The main aim of developing the proposed protocol is to check the ownership and reachability of *MN's HoA* and *CoA*. The proposed protocol must fulfill the following security requirements:

- assure the *CN* that the *BU* request comes from an entity that actually owns the *HoA* by verifying the authenticity of the claimed *HoA*;
- assure the *CN* that the entity that sent the *BU* message is located at the *CoA* by verifying the authenticity of the claimed *CoA*.

- detect any unauthorised modification of the binding data, thus, improving the integrity of the binding request.
- ensure that the *BU* message is protected against session hijacking, MITM attacks, and DOS attacks.
- ensure that the number, and the length of the messages sent to or received from the *MN* is kept to the minimum.

There must also be less dependence on third-party nodes for the proposed protocol to carry out its functions.

Hash functions should be included in the binding update protocol as part of data integrity checks. Also, to ensure that the node is in the location it claims to be, there should be a method of verifying the location authenticity. A sound, while complementary solution can be produced with the combination of a cryptographic system, digital signature, hash function, and IP address creation based on the private and public key of the user.

4.3 Proposed Protocol's Assumptions

The design of the protocol was based on several assumptions:

- The Bi-directional security association for the communication that is encrypted and authenticated is preconfigured at the *MN* and its *HA*. The IPSec ESP protocol is used as a method for protecting the mobility-related message that is exchanged between the *MN* and the *HA*.
- The validity period is agreed between the *MN* and its *HA*, and the *HA* will reject any mobility-related message that arrives after the validity period.

- One trusted entity is the *MN's HA*, and both the *MN* and the *CN* believe that the *HA* will behave itself if left on its own.
- The *HA* is aware that the *CoA* belongs to the *MN* because it has been registered with the *HA* by the *MN*.

4.4 Proposed Protocol’s Principles

In the design of our proposed protocol, the following measures have been taken in order to fulfill all the security requirements stated in Section 4.2, above:

Measure 1: Both the *MN* and *CN* use the PKBU protocol. The *MN* uses it to register a new *CoA* whenever it roams away from its original home link, while the *CN* uses it to verify that the *CoA* belongs to the *MN*.

Measure 2: The PKBU assures that the address of the user actually belongs to the actual user and is not a spoofed address. It also prevents spoofing by authenticating the location of the communicating device and ensuring that the IP address is correct. The *CN* receives a hash of the authentication data from the *MN* through the *HA*. The data is stored as hash by the *HA*, and cannot be read by attackers who might try to intercept it during transmission. The *MN* will sign the cipher text and send it to the *CN*. The *CN* can verify that it is using the *MN's* public key and decipher the text using the *CN*’s private key. The *CN* then proceeds with calculating the hash of the text, and if they match, then the authentication process is successful.

Measure 3: By using the PKBU, the centralised authority will be removed and a decentralised authentication system is used, instead. The *HA*, which is maintained and managed by the Internet Service Provider (ISP), stores the security data of the *MN* such as its *HoA* and *CoA*.

4.5 Private Key-based Binding Update (PKBU) Protocol

Our proposed method asserts the address ownership of the *MN* by creating a 128-bit MIPv6 address based on the *MN*'s private key and computing one-way hash function to provide a way to authenticate the *MN*s authority. This technique allows the *CN* to validate the *MN* and ensure that it is not a malicious node by checking the address ownership. The PKI is not needed anymore with this binding ownership of the *MN*'s *HoA*. The *MN* retrieves its private and public key pair, while the user's ID is used to obtain the node's private and public key pairs. In the proposed method of checking the ownership of *MN*'s IP address, the *HoA* of the *MN* is certified using the secure interface ID. It is based on the *MN's* private key and a valid subnet prefix.

The second method proposed in PKBU involves verification of the reachability of the *MN*. In this method, the hash value of the *MN's HoA*, the public key of *MN*, and request for the *CN's* public key are sent from the *MN* to the *CN* through the *HA*. After the *MN* receives the *CN's* public key, it will send another message directly to the *CN*. In this message, the *MN* encrypts the *MN's CoA* and *HoA* using the *CN's* public key. The encrypted message will be signed with the *MN's* private key. When the *CN* receives these messages from the *MN*, the *CN* uses the *MN's* public key to verify the signature, then the *CN* decrypts the message to obtain the *MN's CoA* and *HoA*. The *CN* then calculates the hash value of *HoA* and compares it with the hash value from the message received from the *MN* via the *HA*. If one of the signature verification, decryption or comparison of hash value processes produces a negative result, the message will be rejected. If the results of the checking and the validation process are positive, the *CN* is assured that the *MN's HoA* and *CoA* are correct and that the *MN* can be reached at this *CoA*. This method will be described in detail.

4.5.1 Private Key-based Binding Update (PKBU) Protocol’s Phases

Figure 4.1 shows the three phases involved in the exchange of messages between the nodes in the PKBU protocol. Phase 1 consists of three steps in our proposed method to assert the ownership of the *MN*’s IP address. In phase 2, we describe our proposed method to check the reachability of *MN*. Phase 3 consists of four steps pertaining to the validation process. It is important to note that the *MN* assures the ownership and reachability of both *HoA* and *CoA* to the *CN* using the validation process.

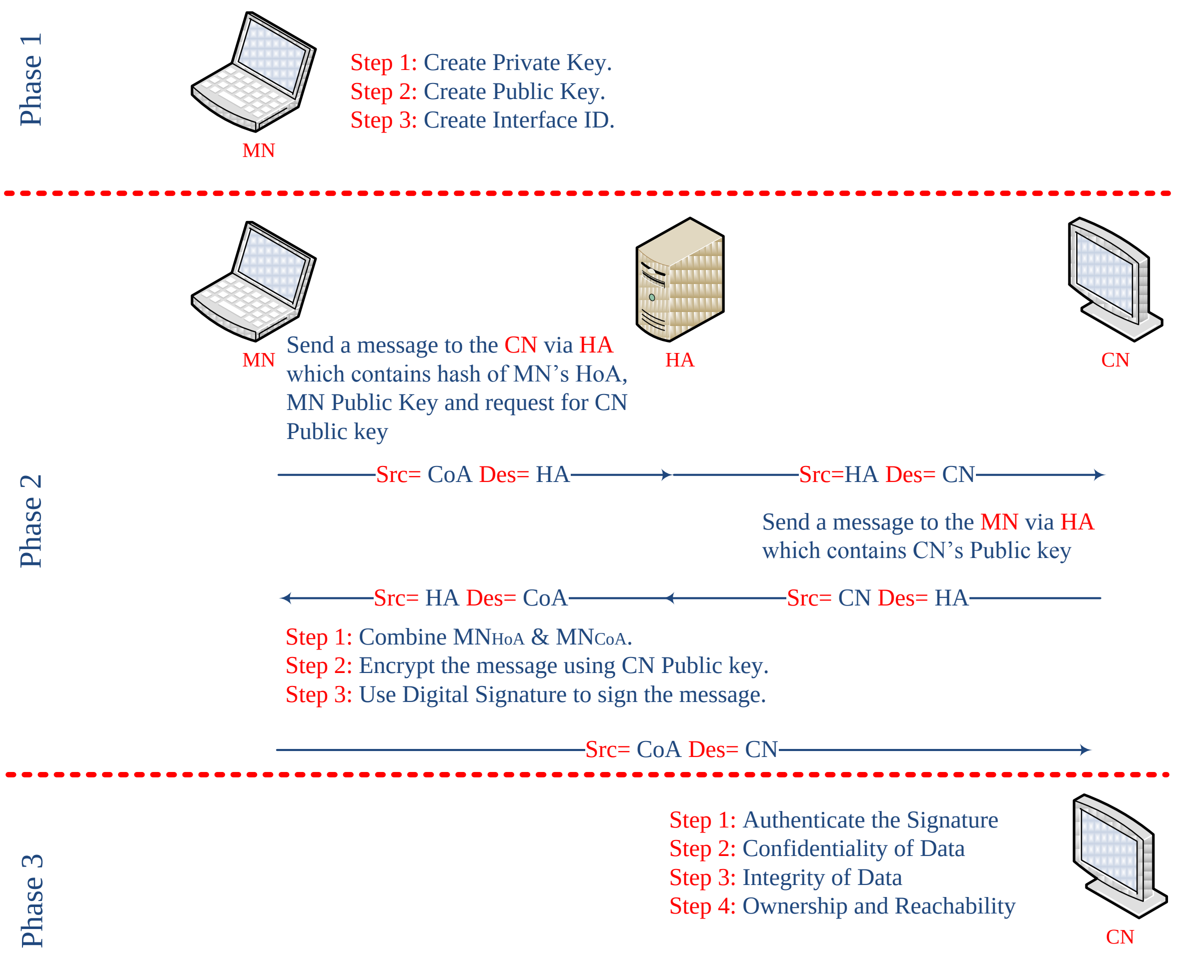

Phase 1
MN
Step 1: Create Private Key.
Step 2: Create Public Key.
Step 3: Create Interface ID.
Phase 2
MN
HA
CN
Send a message to the CN via HA which contains hash of MN's HoA, MN Public Key and request for CN Public key
Src= CoA Des= HA
Src=HA Des= CN
Send a message to the MN via HA which contains CN's Public key
Src= HA Des= CoA
Src= CN Des= HA
Step 1: Combine MNHoA & MNCoA.
Step 2: Encrypt the message using CN Public key.
Step 3: Use Digital Signature to sign the message.
Src= CoA Des= CN
Phase 3
Step 1: Authenticate the Signature
Step 2: Confidentiality of Data
Step 3: Integrity of Data
Step 4: Ownership and Reachability
CN

Figure 4.1: Message Exchanged between the *MN*, *CN* and *HA*

➢ **Phase 1:**

This phase consists of the three steps involved in generating the *MN*'s private and public keys, and creating the *MN*'s interface ID.

○ **First Step (Create Private Key):**

In this step, the *MN* creates its own private key, which can be acquired depending on the user ID (Figure 4.2). To obtain the private key, a hash functions *Hash (User ID) * i* is used, where '*i*' is a random integer in the range [1, n-1]. If the private key of user's *MN* is an integer, MN_{PRK}, then:

$$MN_{PRK} = Hash\ (User\ ID) * i$$

The private key is a number based on the user *ID* hash value and random number, meaning that it cannot be predicted, and it is, thus, secure.

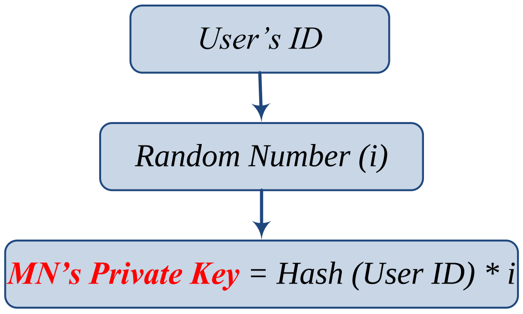


Figure 4.2: Generating the Private Key in PKBU

- **Second Step (Create Public Key):**

In this step, the *MN* creates its public key. This protocol uses Elliptic Curve Cryptography (*ECC*) for creating the public key. The sender uses the receiver's public key to encrypt the message, then, signs it using its private key. The receiver then decrypts the ciphered message using its own private key and uses the sender's public key for verification. Thus, one of *ECC* advantages over other asymmetric algorithms is that it offers the same level of security, but uses smaller size keys. *ECC* implementation is also much more efficient as it consumes less power and computes faster. Less memory and bandwidth are required due to the shorter bit length of the key (De Dormale, Bulens, & Quisquater, 2004). Such attributes are particularly attractive in security applications with restricted computation power and integrated circuit space (Modares, Salem, Salleh, & Shahgoli, 2010). *ECC* is a public key cryptography, and every user or device involved in the communication normally has a pair of keys - a public key, and a private key - as well as a set of operations associated with the keys for performing cryptographic operations.

Therefore, only the legitimate or valid user knows the private key, while all other users in the communication receive the public key. *ECC* mathematical processes are characterized by the elliptic curve $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$. Each *'a'* and *'b'* value produces a different elliptic curve, and all points (x, y) that satisfy the above equation, including a point at infinity, lie on the elliptic curve. The public key is a point on the curve obtained by multiplying the private key with the generator point *G* in the curve. The *ECC* parameters are $T = \{a, b, G, n\}$, where '*a*' and '*b*' are parameters of the elliptic curve $E: y^2 = x^3 + ax + b$, *'G'* is the base point on the curve and '*n*' is the elliptic curve order. If the private key of *MN* is an integer MN_{PRK}, then *MN*'s public key (MN_{PUK}) is '$MN_{PRK}.G$', which is also a point on '*E*' (Appendix A). Hence, a public key is a point on the curve generated from a private key (Figure 4.3). Thus, the *MN* has its own public and private keys which will be used to check the ownership of the IP address of the *MN*.

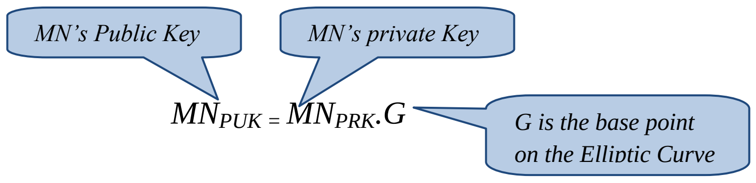


Figure 4.3: Creating Public Key for PKBU

- **Third Step (Create Interface ID):**

In this sub step, the *MN* creates the final 128-bit IPv6 address. The IPv6 has a 128-bit address, together with a given 64-bit subnet prefix and a 64-bit interface identifier, derived from a *MN's* private key hash value. This new method creates a solid cryptographic binding between the *MN's* interface identifier and the *MN* owning the private key. The binding proves *MN* ownership of *HoA* without a PKI (Figure 4.4).

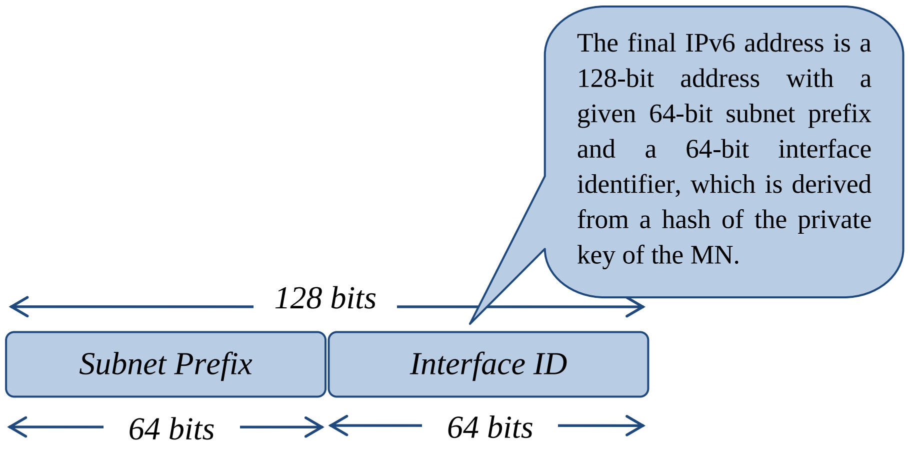

The final IPv6 address is a 128-bit address with a given 64-bit subnet prefix and a 64-bit interface identifier, which is derived from a hash of the private key of the MN.
128 bits
Subnet Prefix
Interface ID
64 bits
64 bits

Figure 4.4: 128 bits IP Address (Subnet Prefix, Interface ID)

- **Phase 2:**

The *MN* sends the *CoA* to the *HA* via IPSec. Every time a *MN* enters a new network, it will be configured with a new *CoA*. The *MN* then must register its new *CoA* and other operations with its *HA* before the new *CoA* can be used (Johnson et al., 2004).

- **First Step (Send Message 1):**

Message 1 will be sent through the *HA* all the requirements for routing optimisation to the *CN*. These include the *MN's HoA* hash value, and the public key of the *MN*, obtained in the first phase. The *MN* will also request for the *CN's* public key *(*$ReqCN_{PUK}$*)*. The message contains the *CN's* address to indicate the first destination of the message (Figure 4.5).

- **Source: MN(CoA)→Destination: HA** *Hash (MN (HoA))*, MN_{PUK}, $ReqCN_{PUK}$

  (It is sent to the *HA* via a pre-established secure tunnel of IPv6)

- **Source: HA → Destination: CN** *Hash (MN (HoA))*, MN_{PUK}, $ReqCN_{PUK}$

- **Second Step (Send Message 2):**

In Message 2, *BU* message preparation is completed and the *CN* sends its own public key to the *MN* through the *HA*, and the *CN* stores the hash value of the *MN's HoA* and *MN's* public key.

- **Source: CN→ Destination: HA** (CN_{PUK}) (*MN* receives CN_{PUK} from the *HA*, and prepares message 3, which has the *BU*).

- **Source: HA→ Destination: MN** (CN_{PUK})

- **Third Step (Send Message 3):**

In Message 3, the *MN* encrypts the *MN's CoA* and *HoA* using the *CN's* public key (the one sent by the *CN* to the *MN* via the *HA*). The encrypted message will be signed with the *MN's* private key.

- **Source: MN(CoA)→ Destination: CN**

  $Sign_{MN(PRK)}\,(Enc_{CN(PUK)}(\,MN(CoA),\, MN\,(HoA)))$

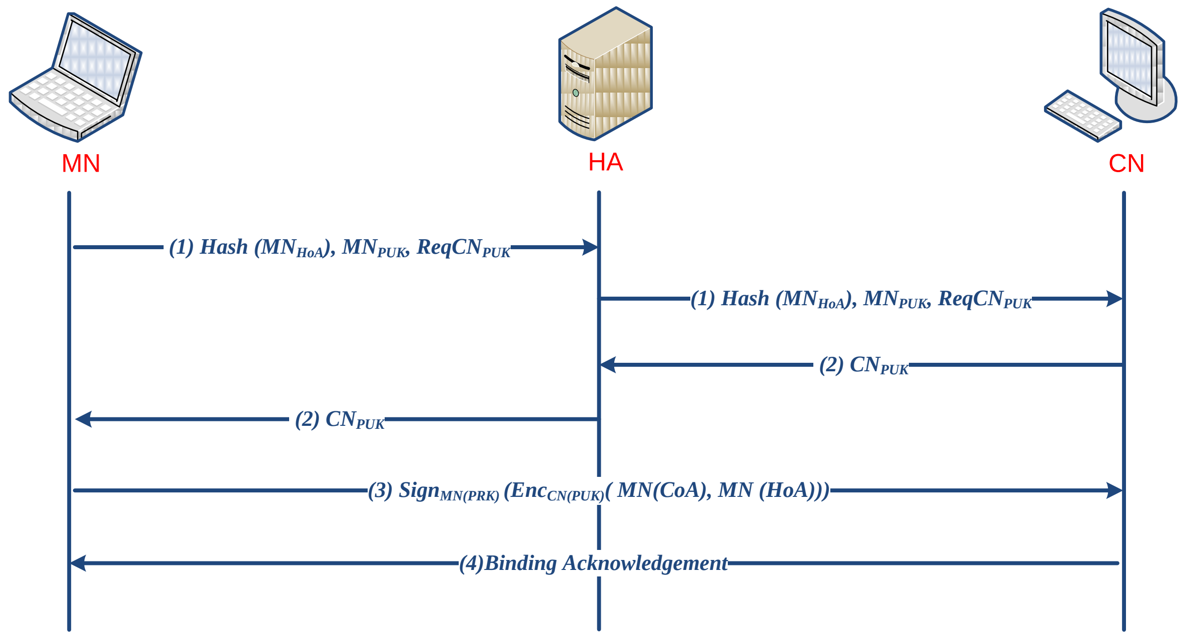

MN
HA
CN
(1) Hash (MN_{HoA}), MN_{PUK}, $ReqCN_{PUK}$
(1) Hash (MN_{HoA}), MN_{PUK}, $ReqCN_{PUK}$
(2) CN_{PUK}
(2) CN_{PUK}
(3) $Sign_{MN(PRK)}$ ($Enc_{CN(PUK)}$(MN(CoA), MN (HoA)))
(4)Binding Acknowledgement

Figure 4.5: Message Exchange in Phase two of PKBU

➢ **Phase 3:**

This phase consists of four steps, and is used to evaluate the security requirements, ownership, and reachability of the *MN*'s IP addresses (Figure 4.6).

○ **First Step (Authentication):**

To authenticate the *MN's* signature '*Sign* (*MN_{PRK}*)', the *CN* will use the *MN's* public key. If the message is not signed by the *MN*'s private key *(MN_{PRK})*, then the *CN* will not be able to verify it and the process will end. But if the *CN* can verify the *MN*'s signature, it means that the *MN* is authenticated. The *CN* then checks the confidentiality of the message via the second step (Figure 4.6).

○ **Second Step (Confidentiality):**

In this step, the *CN* will use its own private key to decrypt the message. After decryption, the *CN* can obtain the *MN's CoA* and *HoA*. If the *CN* is be able to decrypt the message using its own private key that means the data of the *MN's CoA* and *HoA* were secured.

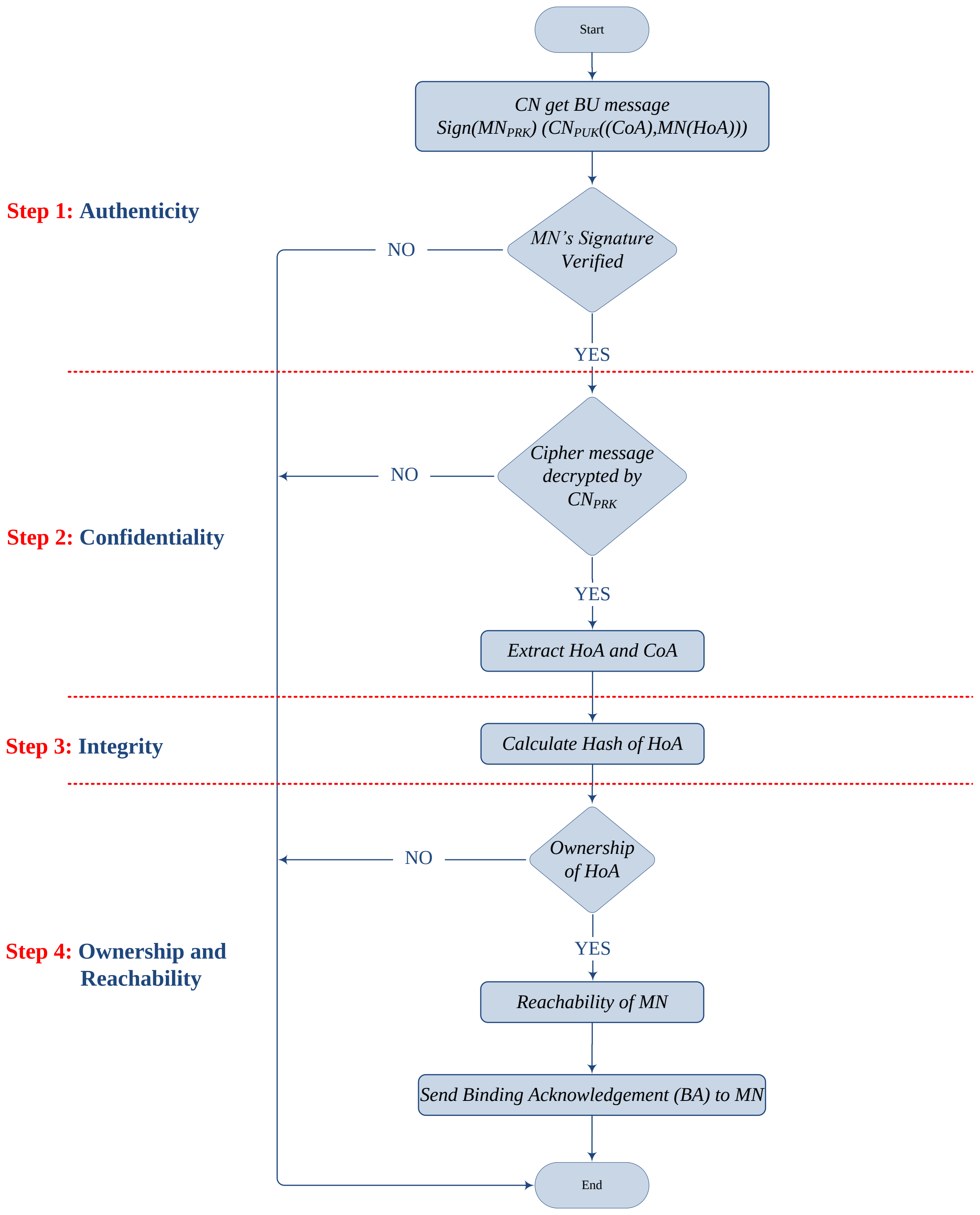


Figure 4.6: Process to check the Authenticity, Confidentiality, Integrity, Ownership, and Reachability in the PKBU.

- **Third Step (Integrity):**

To assert the integrity of the *HoA*, the *CN* needs to calculate the hash value of the *HoA*. The *CN* will compare the calculated hash value with the hash value of the *HoA* which was sent from the *MN* to the *CN* via the *HA* in phase 2 to validate the *HoA*. If either one of the values is negative, the message will be rejected.

- **Fourth Step (Reachability and Ownership):**

If the *CN* can validate the correctness of the *HoA* in the third step, it means that the ownership is proven, and the *MN* is reachable if the *CoA* is known. Then the *CN* will send message 4 to the *MN* as the Binding Acknowledgement (*BA*).

- **Source: CN→ Destination: MN(CoA)** *Binding Acknowledgement (BA)*

Figure 4.7 shows a framework of the proposed PKBU scheme. The main objective of the PKBU protocol is protecting the *BU* message against malicious attacks by checking the ownership, and reachability of the *MN*’s *HoA* and *CoA*, and scrutinizing the security requirements. In the next section, we will evaluate the PKBU protocol to show its efficiency and its strengths in defending *BU* messages against malicious attacks.

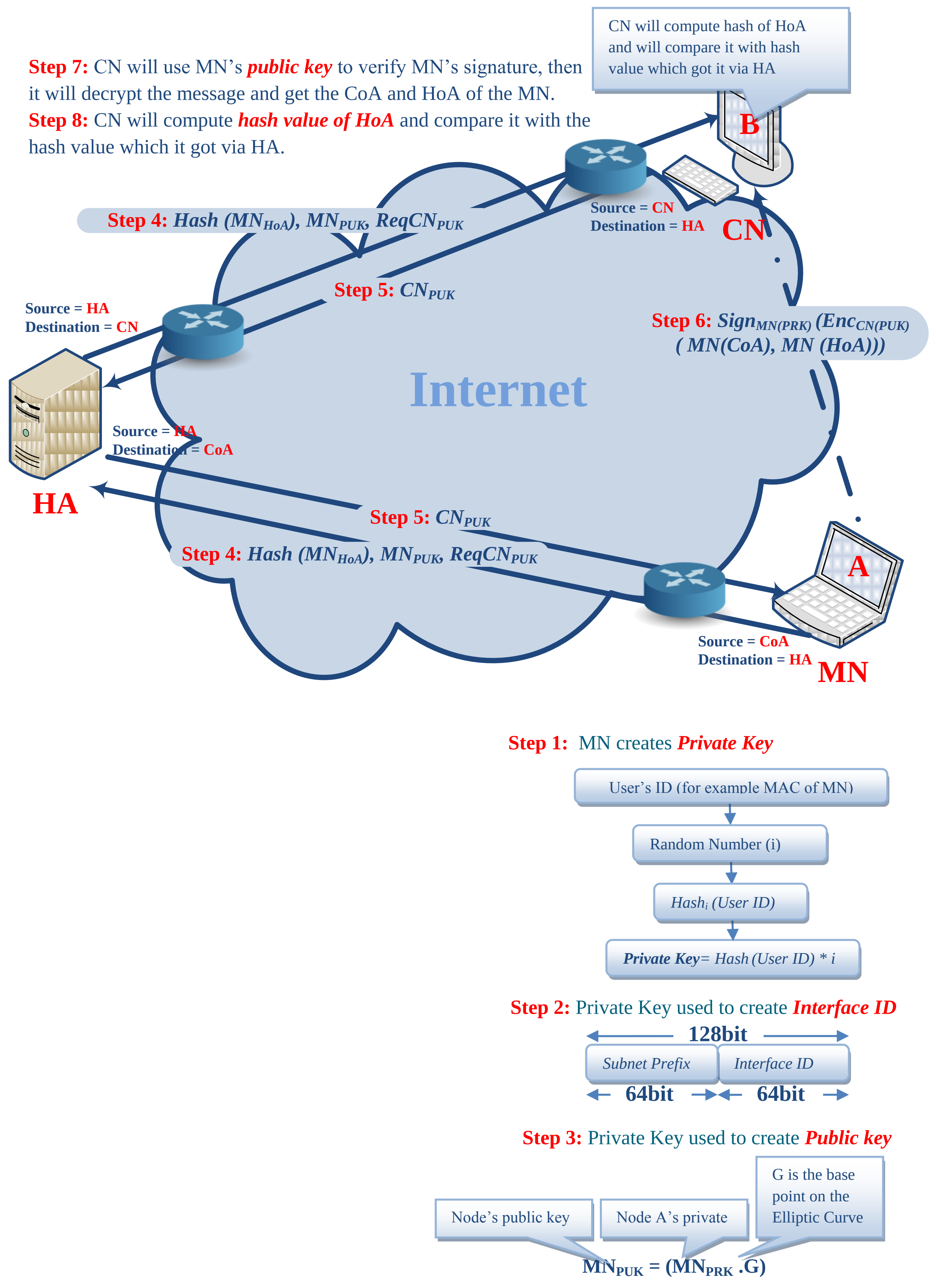

CN will compute hash of HoA and will compare it with hash value which got it via HA
Step 7: CN will use MN's public key to verify MN's signature, then it will decrypt the message and get the CoA and HoA of the MN.
Step 8: CN will compute hash value of HoA and compare it with the hash value which it got via HA.
B
Step 4: Hash (MN_{HoA}), MN_{PUK}, $ReqCN_{PUK}$
Source = CN
Destination = HA
CN
Step 5: CN_{PUK}
Source = HA
Destination = CN
Step 6: $Sign_{MN(PRK)}$ ($Enc_{CN(PUK)}$ (MN(CoA), MN (HoA)))
Internet
Source = HA
Destination = CoA
HA
Step 5: CN_{PUK}
Step 4: Hash (MN_{HoA}), MN_{PUK}, $ReqCN_{PUK}$
A
Source = CoA
Destination = HA
MN
Step 1: MN creates Private Key
User's ID (for example MAC of MN)
Random Number (i)
$Hash_i$ (User ID)
Private Key= Hash (User ID) * i
Step 2: Private Key used to create Interface ID
128bit
Subnet Prefix
Interface ID
64bit
64bit
Step 3: Private Key used to create Public key
Node's public key
Node A's private
G is the base point on the Elliptic Curve
MN_{PUK} = (MN_{PRK} .G)

Figure 4.7: Framework of Proposed PKBU Schemes

4.6 Ownership and Reachability Test

The *BU* message is vulnerable to different types of attacks such as: data packet interceptions that potentially allow attackers to eavesdrop on contents, thus, violating the user’s confidentiality, or altering transmitted packets, for the attacker’s own malicious purposes; address spoofing attacks; and denial of service or redirection attacks. This research will propose solutions for preventing *BU* message against these attacks, and thus, the focus will be on four major areas related to the security of the *BU* message - cryptography, authentication, ownership of the *MN*’s *HoA*, and reachability of the *MN* at the *CoA*. Cryptography allows the transmitted data to be scrambled to render it undecipherable. In this way, intercepted packets are completely unreadable, and only those possessing the appropriate key can decrypt the data to make it readable. Authentication is the process of verifying the authenticity of the nodes involved in a communication, while ownership and reachability check the correctness of the *MN*’s *HoA* and *CoA*. The exchange of messages between the *MN* and the *CN* provides strong evidence to *CN* regarding the ownership of the *MN*'s *HoA*. This is done using a one-way hash function to create IPv6 addresses along with the *BU* message that is signed using the *MN* own private key. In addition, in the proposed method, the hash value of the *MN's HoA* is sent in Message 1 (See section 4.5.1) to the *CN* through the *HA*. The *MN* then sends the *HoA* to the *CN* in Message 3 (See Section 4.5.1), the *CN* will calculate the *HoA* hash value and compare it with the hash value in Message 1 received via the *HA*. Thus, the reachability of the *CoA* is checked. In the rest of this section, Session hijacking, MITM attacks, and DoS attacks are simulated in order to evaluate the security requirements, ownership of the *HoA*, and reachability of the *CoA*.

4.6.1 Session hijacking Attack

The proposed methods can prevent this attack because an attacker does not have the *MN's* private key to sign the message containing the *CoA* (see Section 4.5.1). Also, the *CN* will use the *MN's* public key to verify that the signature does not match the attacker*'s* private key. If the authentication process fails, it means that the *HoA* and the *CoA* do not belong to the *MN*, which in turn, means that the ownership and the reachability checks have failed, and the *CN* will not direct the data traffic to the attacker. As a result, the *BU* message will be protected against session hijacking attacks (Figure 4.8).

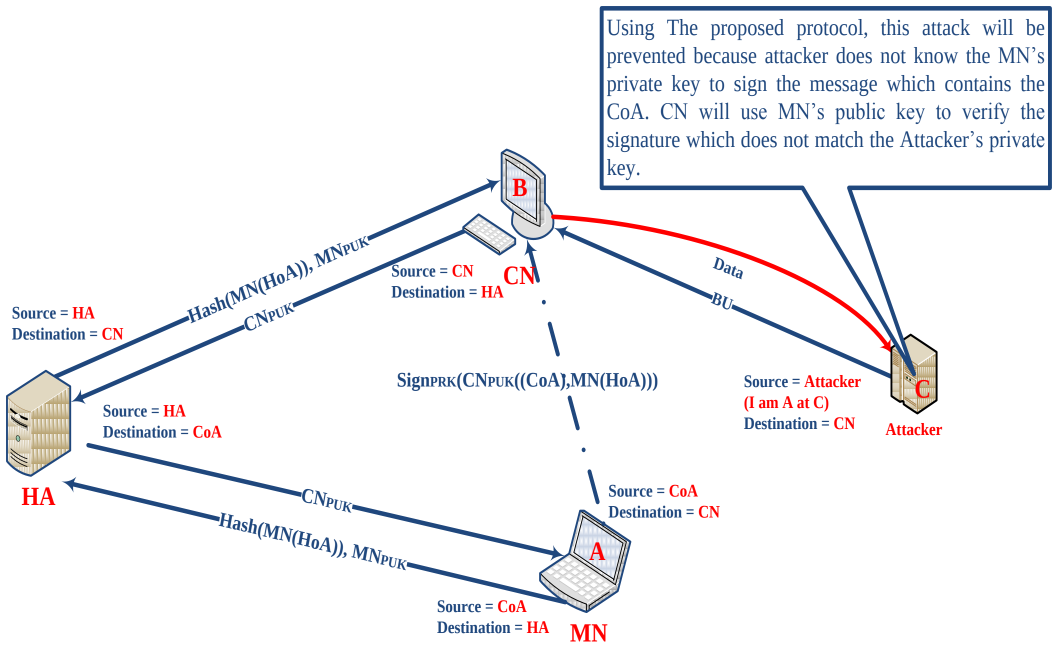

Using The proposed protocol, this attack will be prevented because attacker does not know the MN's private key to sign the message which contains the CoA. CN will use MN's public key to verify the signature which does not match the Attacker's private key.
B
CN
Source = CN
Destination = HA
Hash(MN(HoA)), MNPUK
CNPUK
Source = HA
Destination = CN
Data
BU
Source = Attacker
(I am A at C)
Destination = CN
C
Attacker
SignPRK(CNPUK((CoA),MN(HoA)))
Source = HA
Destination = CoA
HA
CNPUK
Hash(MN(HoA)), MNPUK
Source = CoA
Destination = CN
A
Source = CoA
Destination = HA
MN

Figure 4.8: Protection against Session Hijacking Attack

4.6.2 Man-in-the-Middle Attack

This attack can be prevented by using the proposed methods because of the node's signature, hash function, and cryptography algorithm. Figure 4.9 shows that if an attacker sends a *BU* message to the *CN*, the *CN* will check the ownership and the reachability of the IP addresses. In the proposed method, the ownership of the *MN*'s IP address is checked by using the *MN's* private key to create a 128-bit IP address which is secure, and cannot be spoofed by the attacker. In addition, the proposed method also checks the reachability of the IP address. To do this, the *MN* will send its *HoA* hash value to the *CN* via the *HA*, and the *CN* will then compute the *HoA's* hash value and compare it to the previous value. The attacker does not have the *MN's* private key to sign the message containing the *CoA*. Thus, by checking the authenticity, ownership, and reachability of the IP address, the *CN* can ensure that the *BU* message is from a valid *MN* and not from an attacker. An attack would only happen if the attacker has the *MN*'s public and private key, the *CN*'s public and *MN's HoA*. The proposed methods make it very difficult to get the node's private keys, to mount a MITM attack. In addition, in an MITM attack, it is not possible for the attacker to eavesdrop on the users' message and compromise the user's confidentiality because the message is encrypted using cryptography algorithms.

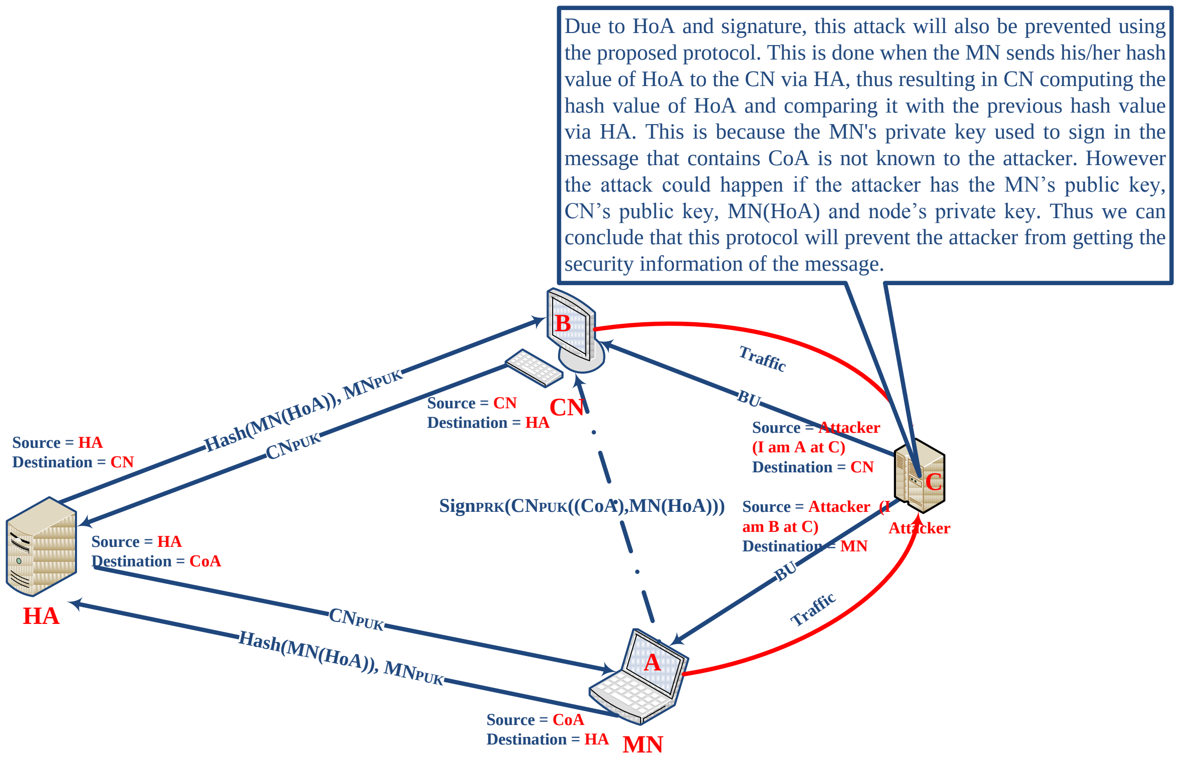

Due to HoA and signature, this attack will also be prevented using the proposed protocol. This is done when the MN sends his/her hash value of HoA to the CN via HA, thus resulting in CN computing the hash value of HoA and comparing it with the previous hash value via HA. This is because the MN's private key used to sign in the message that contains CoA is not known to the attacker. However the attack could happen if the attacker has the MN's public key, CN's public key, MN(HoA) and node's private key. Thus we can conclude that this protocol will prevent the attacker from getting the security information of the message.

Figure 4.9: Protect Against Man-In-The-Middle Attack

4.6.3 Denial-of-Service Attack

The proposed methods can also prevent DoS attacks. The *CN* will not accept a new *CoA* if a new message carries an unverified *CoA*. To verify the *CoA*, the *CN* must first verify the authenticity of the *MN*. The *CN* uses the *MN's* public key to verify the signature which is created based on the *MN*'s private key. The signature cannot be verified if the attacker does not know the *MN's* private key that was generated based on the proposed method. The *CN* then checks the confidentiality of the message by decrypting the message using its own private key, and subsequently, the *CN* can get the *HoA* and the *CoA*, and check the ownership of the *MN*'s *HoA*, and the reachability of the *CoA* (Figure 4.10).

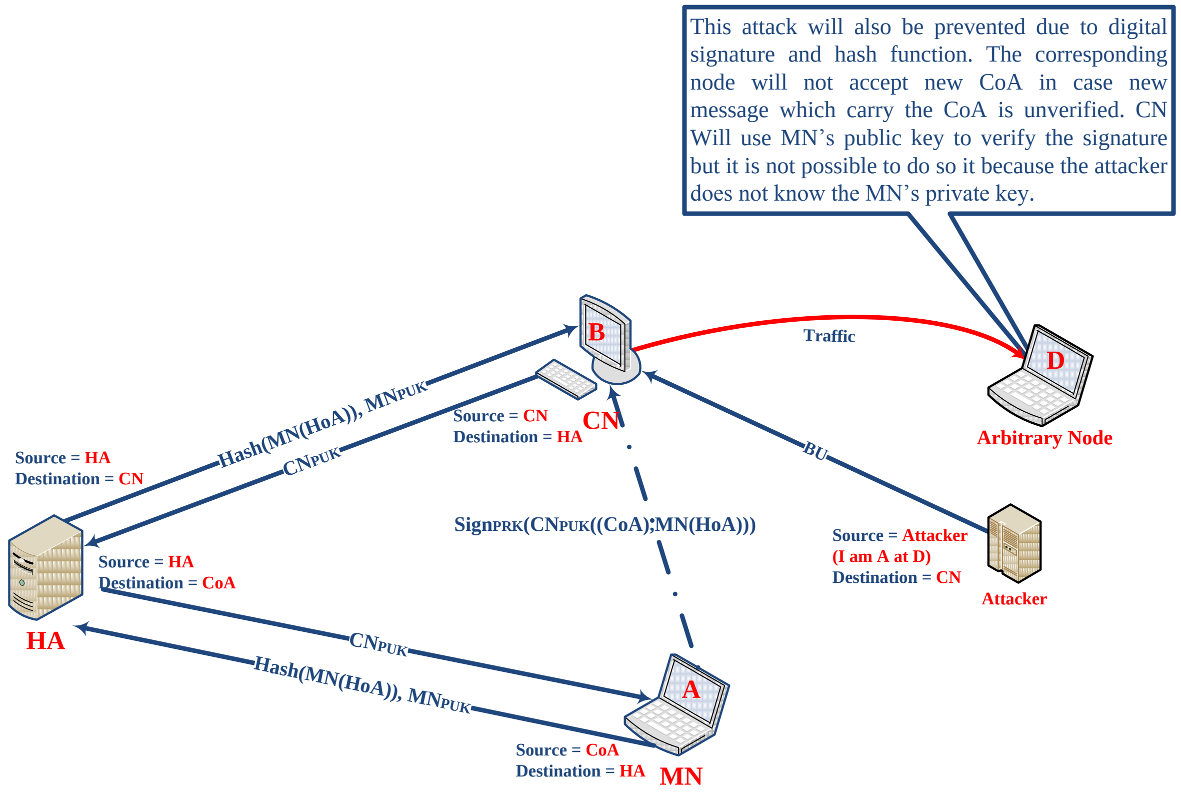

This attack will also be prevented due to digital signature and hash function. The corresponding node will not accept new CoA in case new message which carry the CoA is unverified. CN Will use MN's public key to verify the signature but it is not possible to do so it because the attacker does not know the MN's private key.
B
CN
Traffic
D
Arbitrary Node
Source = CN
Destination = HA
Hash(MN(HoA)), MNPUK
CNPUK
Source = HA
Destination = CN
BU
SignPRK(CNPUK((CoA);MN(HoA)))
Source = Attacker
(I am A at D)
Destination = CN
Attacker
Source = HA
Destination = CoA
HA
CNPUK
Hash(MN(HoA)), MNPUK
A
Source = CoA
Destination = HA
MN

Figure 4.10: Protection against DoS Attack

4.7 Summary

The framework for PKBU protocol was discussed in this chapter. This protocol can ensure a secure connection between the *CNs* and the *MNs*, and protect the *BU* message against the MITM attacks, session hijacking attacks, and DoS attacks. These objectives have been achieved through the use of various methods and strategies as follows:

1. Using an Infrastructure-less algorithm instead of an Infrastructure-based algorithm, and using asymmetric cryptography instead of symmetric cryptography.
2. Development of a method to check the ownership of the *MN*'s *HoA* because the *MN* must ensure the *CN* that:
    a. the *MN* owns the *HoA*
    b. the actual *BU* request belongs the *MN*.
3. Development of a method to verify that the *MN* is reachable at the claimed *CoAs.*

Based on our objectives, PKBU uses the *MN's* private key to create a secure IP address, and then uses it to check the ownership of the *MN*'s *HoA* and to check the reachability of the *MN* at the claimed *CoA*. The PKBU was evaluated using different attack scenarios, and the results show that the protocol can be used to protect the *BU* message against security threats such as MITM attacks, DoS attacks, and session hijacking attacks. PKBU also aids all the security services by protecting data integrity, confidentiality, and node authentication. A formal evaluation and simulation modelling of the proposed protocol is presented in the following chapter.

Chapter 5 : Formal Analysis and Simulation of the Private Key-based Binding Update Protocol

5.1 Introduction

This chapter discusses the formal security verification and simulation of the PKBU protocol. Section 5.2 presents the formal analyses of the PKBU protocol, while Section 5.3 presents the simulation modelling. Finally, section 5.4 is the chapter summary.

5.2 Formal Analysis of the Private Key-based Binding Update (PKBU) Protocol

Formal verification of a security protocol is useful for identifying security flaws that are often subtle, and difficult to find (Datta et al., 2007). In this section, the protocol security features are discussed and the Protocol Composition Logic (PCL) framework is used with the formal verification methods. In the PCL framework, logical methods and procedures will be used to prove the correctness of the protocol. It is useful in verifying the authenticity, secrecy, and other protocol security features (Datta, Derek, Mitchell, & Pavlovic, 2003a, 2003b, 2004, 2005; Datta et al., 2007; Datta, Derek, Mitchell, & Warinschi, 2006; Roy, Datta, Derek, Mitchell, & Seifert, 2006). Therefore, the PCL method was chosen to test and verify the PKBU protocol.

5.2.1 Protocol Composition Logic (PCL)

The Protocol Composition Logic (PCL) is a tool for testing and proving the security properties of a particular network protocol, especially, one that uses symmetric and public key cryptography. The logic behind this method is a process calculus that has different methods of testing the protocols that involve random number generation, message sending and receiving, performing decryption, and verifying digital signatures. This proven system consists of several steps that also contain information about the individual protocol actions, and inference rules that produce the assertion result of the protocol. Even though the assertion result is produced by following the protocol steps, sound logic is also used by having each provable assertion containing a sequence of actions to be executed by the protocol, and with each action and the following arbitrary actions to be executed by a malicious attacker. The compositional reasoning for complex security is supported by PCL, and this is being applied in numerous industry standards such as SSL/TLS, IEEE, 802.11i, and Kerberos V5.

The PCL's basic assertion is similar to the Hoare logic (Hoare, 1969), and the dynamic logic (Harel, Kozen, & Tiuryn, 2001). By using the formulae $\theta[P]_X \varphi$, once the actions *P* are executed in thread *X*, the resulting state of formulae φ is true when the starting state of formula θ is true. While actions of *P* in thread *X* is only mentioned in this formula, the state of *P* after *X* could be the result of actions along with the additional actions performed by other threads. This may include the arbitrary actions of attackers. PCL has several predicated actions such as *Send (X; t)*, *Receive (X; t)*, *New (X; t)*, *Decrypt (X; t)*, and *Verify(X; t)*. It asserts that specific actions have been performed by the named thread. For example, if thread *X* sends the term *t* message, then *Send(X; t)* will hold a run. There is one class of secrecy property that can be specified by using the predicate *Has (X; t)*. This means that *t* is generated from the elements from *X*, which is either from generating (using a new action) or receiving a constituent that does not hide under a key encryption from *X*. The Honest ($\bar{X}$) is a predicator that is unique to PCL. It asserts that the protocol prescribes all the actions of $\bar{X}$. When one party follows the protocol's prescribed steps, then it is assumed that Honest is used. For example, if a *CN* receives a transaction initiation from the *MN* and specifies that only the *CN* will have knowledge of the data being sent, it will explicitly assume that the *CN* is honest. The *CN* may make known its private key to attackers if it is not honest, thus, allowing attackers to decrypt the messages it intercepted.

There are several categories of the PCL axioms and inference rules. A simple yet necessary axiom class states that the indicated thread has performed the action once the action is done. Another axiom class specifies the cryptographic operation properties. For example, when an axiom reflects the un-forgeability property of a particular digital signal, it states that the agent must have generated a signature when it has been verified as an honest agent. The message will have the generated signature, and is sent out in earlier messages.

5.2.2 Modelling of Private Key-based Binding Update (PKBU) Protocol

The protocols sections must first be set as mathematical objects and define their execution before we can formally state and prove the security properties of the protocol. One of the important parts in analysing security capabilities is to understand how an honest principal in a protocol responds to messages from a malicious attacker. Next, there will be explanation on how the PKBU protocol is represented. The definition below shows the role of the *MN*. The actions taken by the *MN* after creating the 128-bit IP address, are based on the private and public keys, and the subnet prefix are as follows:

1. Create a new message that has the *HoA* hash value
2. Send the *HoA* hash value via the *HA* to the *CN*
3. Send a secure message directly to the *CN*.

$$
\begin{aligned}
&MobileNode \equiv [\\
&\qquad New\ m_{InitCR} := HASH(MN_{HoA});\\
&\qquad Send\ \hat{MN}, \hat{HA}, m_{InitCR;}\\
&\qquad \mathrm{Re}ceive\ \hat{HA}, \hat{MN}, Ack;\\
&\qquad New\ m_{BU} := SIG_{MN_{SK}}\{|\, ENC_{CN_{PK}}\{MN_{CoA}, MN_{HoA}\}\,|\};\\
&\qquad Send\ \hat{MN}, \hat{CN}, m_{BU};\\
&\qquad \mathrm{Re}ceive\ \hat{CN}, \hat{MN}, Ack;\\
&]MN()
\end{aligned}
$$

The definition below shows the roles undertaken by the *HA* where the *HA* sends messages from the *MN* to the *CN*. This is with the assumption that the message from *MN* has already been verified by the *HA* before it is sent to the *CN*.

$$
\begin{aligned}
&HomeAgent \equiv [\\
&\qquad \mathrm{Re}ceive\ \hat{HA}, \hat{MN}, m_{InitCR};\\
&\qquad Send\ \hat{MN}, \hat{HA}, Ack;\\
&\qquad Send\ \hat{CN}, \hat{HA}, m_{InitCR};\\
&\qquad \mathrm{Re}ceive\ \hat{HA}, \hat{CN}, Ack;\\
&]HA()
\end{aligned}
$$

The definition below shows the role of the *CN*. The *CN* will first receive a message which contains a hash value from the *MN* through the *HA*. The *CN* will also receive another message sent directly from the *MN*.

$$
\begin{aligned}
&CorrespondingNode \equiv [\\
&\qquad \mathrm{Re}ceive\ \hat{H}\hat{A}, \hat{C}\hat{N}, m_{InitCR};\\
&\qquad \mathrm{Re}ceive\ \hat{M}\hat{N}, \hat{C}\hat{N}, m_{BU};\\
&\qquad Verify\ (m_{BU}, MN_{K_{MN-CN}});\\
&\qquad DEC\ (m_{BU},\ CN_{\bar{K}_{MN-CN}});\\
&\qquad HASH\ \ m_{\mathrm{Re}\,spCR} := (MN_{HoA});\\
&\qquad match(m_{\mathrm{Re}\,spCR,}\, m_{InitCR});\\
&]CN()
\end{aligned}
$$

The session authentication information is stored in the security property. The concept of matching conversation forms the basis used in the authentication property in PCL. The idea behind this concept is that after the *CN* has performed its role, the existing role intended for the *HA* and *MN* is proven via the corresponding view during interactions. For $\hat{C}\hat{N}$ communicating with $\hat{M}\hat{N}$ and $\hat{H}\hat{A}$, matching conversations are formulated ϕ_{PKBU} defined, below.

$$
\begin{aligned}
&\phi_{PKBU} ::= Honest(\hat{M}\hat{N}) \wedge Honest(\hat{H}\hat{A}) \wedge Honest(\hat{C}\hat{N}) \supset\\
&\exists(\hat{H}\hat{A} \wedge \hat{M}\hat{N}).action(\\
&Send(MN, \hat{M}\hat{N}, \hat{H}\hat{A}, m_{InitCR}),\\
&\mathrm{Re}ceive(HA, \hat{M}\hat{N}, \hat{H}\hat{A}, m_{InitCR}),\\
&Send(HA, \hat{H}\hat{A}, \hat{C}\hat{N}, m_{InitCR}),\\
&\mathrm{Re}ceive(CN, \hat{H}\hat{A}, \hat{C}\hat{N}, m_{InitCR}),\\
&Send(MN, \hat{M}\hat{N}, \hat{C}\hat{N}, m_{BU}),\\
&\mathrm{Re}ceive(CN, \hat{M}\hat{N}, \hat{C}\hat{N}, m_{BU}))
\end{aligned}
$$

The above definition shows that if the *MN*, *HA* and *CN* are honest, they will execute the actions expressed in the ϕ_{PKBU} formulae. This will confirm the authenticity of the authenticators. The *CN* receives the message in the PKBU protocol, therefore, if it authenticates the *MN*, both the *MN* and the *CN* are now mutually authenticated. By using this definition, we can conclude the following theorem:

$$\Gamma_{PKBU1} \wedge \Gamma_{PKBU2} \wedge \theta_{PKBU1} \wedge \theta_{PKBU2} \wedge \theta_{PKBU3} \vdash$$
$$Honest(\hat{MN}) \wedge Honest(\hat{HA}) \wedge Honest(\hat{CN}) \wedge$$
$$\hat{MN} \neq \hat{HA} \neq \hat{CN} \supset \phi_{PKBU}$$

where:

θ_{PKBU1}, θ_{PKBU2} and θ_{PKBU3} state the PKBU protocol's pre-condition. This means that the *MN*, *HA* and *CN* will share the K_{MN-HA}, K_{MN-CN} and K_{CN-HA} before the protocol is executed. Both Γ_{PKBU1} and Γ_{PKBU2} specify that the *MN*, the *HA* and the *CN* drive the private key, as shown in the definition below.

$$\theta_{PKBU1} := (Honest(\hat{MN}) \wedge Honest(\hat{HA}) \wedge Has(\hat{X}, K_{MN-HA})) \supset \hat{X} = \hat{MN} \vee \hat{X} = \hat{HA}$$
$$\theta_{PKBU2} := (Honest(\hat{MN}) \wedge Honest(\hat{CN}) \wedge Has(\hat{X}, K_{MN-CN})) \supset \hat{X} = \hat{MN} \vee \hat{X} = \hat{CN}$$
$$\theta_{PKBU3} := (Honest(\hat{HA}) \wedge Honest(\hat{CN}) \wedge Has(\hat{X}, K_{HA-CN})) \supset \hat{X} = \hat{HA} \vee \hat{X} = \hat{CN}$$
$$\Gamma_{PKBU1} := Computes(\hat{X}, (SIG_{MN_{SK}}\{|ENC_{K_{MN-CN}}\{HoA \| CoA\}|\})) \supset \neg(Send(\hat{X}, m) \wedge$$
$$Contains(m, SIG_{MN_{SK}}\{|ENC_{K_{MN-CN}}\{HoA \| CoA\}|\}))$$
$$\Gamma_{PKBU2} := Computes(\hat{X}, HASH_{K_{HA-CN}}(\hat{X}, HoA) \supset \neg(Send(\hat{X}, m) \wedge Contains$$
$$(m, HASH_{K_{HA-CN}}(\hat{X}, HoA)))$$

Table 5.1 shows the encryption, and signature axioms for PKBU. It will be used as the proof, below, of the proposed PKBU. The explanation is presented below:

Table 5.1 : Axioms for protocol actions and encryption and signature axioms for PKBU

Extended axioms	Explanations
AA1 $\phi[a]_X \ \langle\rangle a$	If *X* has completed an action in a role, then the matching predicate states that the action that had occurred in the past is true.
AA4 $\phi[a_1,\ldots,a_k]_X \ a_1\langle\ldots\langle a_k$	After thread *X* performs actions *a*$_1$,.., *a*$_k$ in sequence, the action predicates corresponding to the actions are ordered in the same sequence.
AR1 $\mathrm{Receive}(X, p(t_1))[match\ p(t_1) as\ p(t_2)]_X$ $\mathrm{Receive}(X, p(t_2))$	It is used to model a method for obtaining information about structure of terms as they are received.
HASH1 $Compute(X, HASH_K\{t\}) \supset Has(X, k) \wedge Has(X, t)$	The principal possesses both *t* and *k* if it has the hashed term *t* using the *k* key.
HASH3 $\mathrm{Receive}(X, HASH_K\{t\}) \supset \exists Z.Computes(Z, HASH_K\{t\}) \wedge Send(Z, HASH_K\{t\})$	If a hashed term is received by *X*, then another principal *Z* must have been hashed and sent before.

HASH4 $Has(X, HASH_K\{t\}) \supset Compute(X, HASH_K\{t\}) \vee (\exists Z, m.Computes(Z, HASH_K\{t\}) \wedge Send(Z, HASH_K\{t\}) \wedge Contains(m, HASH_K\{t\}))$	If the hashed term is possessed by *X*, then it either *X* hashes the term or another principal *Z* must have hashed the term and sent it in previously.
SEC $Honest(\bar{X}) \wedge PkDec(Y, ENC_k\{t\}, \bar{k}) \supset \bar{Y} = \bar{X}$	In order to decrypt a term that has been encrypted with a public key, the principal needs to possess the corresponding private key.
ENC1 $Has(X, t) \wedge Has(X, k) \supset Computes(X, ENC_k\{t\}) \supset Has(X, ENC_k\{t\})$	If the term and key are known, then the principal *X* can be encrypted using *k* key and *t* term.
ENC3 $Computes(X, ENC_k\{t\}) \supset Has(X, ENC_k\{t\}) \supset Has(X, k) \wedge Has(X, t)$	If the principal possesses *t* and *k*, then it has been encrypted using the *t* term and *k* key.
VER $Honest(\bar{X}) Verify(Y, SIG_{\bar{k}}\{t\}, k) \wedge \bar{X} \neq \bar{Y} \supset \exists X.Send(X, m) Contains(m, SIG_{\bar{k}}\{t\})$	Signing a term cannot be denied by the principal (signature un-forgeability).
PENC $PKDec(X, ENC_k\{t\}, \bar{k}) \supset \exists Z.Send(Z, m) \wedge Contains(m, ENC_k\{t\})$	If a term can be decrypted by *X* using its private key, then another principal *Z* must have encrypted the term and sent it previously.

A matching conversation is formulated from the authentication property (ϕ_{PKBU}). The following part is the proof of the security guarantee for the nodes' role in PKBU protocol. The explanation of this formal proof is as follows:

1. The *CN* knows that every action is in proper order i.e., AA1, AR1, and AA4 are lined up to show that a sequence of actions has been performed by the *CN*.
2. Once the m_{BU} is received and verified by the *CN*, some entity ($\hat{MN}$) will have the $SIG_{\bar{K}_{MN-CN}}\{t\}$ and $ENC_k\{t\}$. They sent out the m_{BU} as shown in line (2) in Table 5.2.
3. The private key is needed by the *CN* for decrypting a term encrypted with the related public key, as explained in line (3) in Table 5.2.
4. If the term and the key are known, then the principal $\hat{MN}$ is able to encrypt it using the *k* key and *t* term, as shown in line (4) in Table 5.2.
5. The principal possesses the *t* and *k* if the term t and the key *k* are used by the $\hat{MN}$ for encryption, as shown in line (5) in Table 5.2.
6. If the private key is being used successfully by the $\hat{CN}$ for decryption, then another principal ($\hat{MN}$) must have encrypted and sent the term previously, as described in line (6) in Table 5.2.
7. The *CN* has the verified signed term *t* by using public key *k* if the *MN* is honest, as described in line (7) and (8) in Table 5.2.
8. The correctness of the send and receive message between the *MN* and *CN* in a PKBU protocol is shown in line (9) and (10) in Table 5.2.
9. If the *CN* receives a value that is hashed, then another entity $\hat{HA}$ must have $HASH_{K_{HA-CN}}(CN, m_{\mathrm{InitCR}})$ it and sent it out, as explained in line (11) in Table 5.2.

10. The *HA* possesses both the term *t* and the key *k* if it has successfully hashed a term using *t* and *k*, as shown in line (12) in Table 5.2.
11. If the *CN* has a term that is hashed, then either another principal has hashed the term and sent it out previously, or the *CN* itself has hashed the term, as indicated in line (13) in Table 5.2.
12. The correctness of the send and receive messages between the *HA* and the *CN* in the PKBU protocol is shown in line (14) and line (15), respectively, in Table 5.2.
13. All the actions in line (16) matched as indicated in line (1), (9), (10), (14) and (15) in Table 5.2. Therefore, the *CN* can now deduce that the session authentication in the security property is guaranteed by the PKBU protocol.

Table 5.2: Proof of PKBU in PCL language

Line	**PKBU written in PCL language**	
(1)	**AA1,** **AR1,AA4**	$[PKBU : CorrespondingNode]_{CN}$ $\text{Receive}\,(CN, \{\hat{HA}, \hat{CN}, m_{\text{InitCR}}, HASH_{K_{CN-HA}}(CN, m_{\text{InitCR}}))$ $< Send\,(CN, \hat{MN}, \hat{CN}, am_{\text{InitCR}})$ $< \text{Receive}\,(CN, \{\hat{MN}, \hat{CN}, SIG_{\bar{K}_{MN-CN}}\{\|ENC_k\{HoA \| COA\}\|\}, m_{\text{BU}}\})$
(2)	**AR1,HASH3**	$\theta_{PKBU\,2}[\text{Receive } \hat{MN}, \hat{CN}, m; \text{ match } m/\text{CN}, m_{\text{BU}}, ver, dec; \text{match}$ $ver/Verify\,(\hat{CN}, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})\text{ match } dec/PKDec$ $(CN, ENC_k\{t\}, \bar{k})]_{CN}\ \text{Receive}(CN, \{\hat{MN}, \hat{CN}, m_{\text{BU}}, Verify$ $(\hat{CN}, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})\}) \supset \exists Z.Send\,(CN, m) \wedge Contains$ $(m, SIG_{\bar{k}}\{t\}) < \text{Receive}(CN, \{\hat{MN}, \hat{CN}, CN, m_{\text{BU}}, Verify\,(\hat{CN}$ $, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})\}) \wedge \text{Receive}(CN, \{\hat{MN}, \hat{CN}, m_{\text{BU}},$ $PKDec(CN, ENC_k\{t\}, \bar{k})\}) \supset \exists X.Send\,(X, m) \wedge$ $Contains(m, ENC_k\{t\})$
(3)	**SEC**	$Honest\,(MN) \wedge PkDec(CN, ENC_k\{t\}, k)$

(4)	**ENC1**	$Has(MN,t) \wedge Has(MN,k) \supset Computes(MN, ENC_k\{t\}) \supset Has(MN, ENC_k\{t\})$
(5)	**ENC3**	$Computes(MN, ENC_K\{t\}) \supset Has(MN, ENC_K\{t\}) \supset Has(MN,k) \wedge Has(MN,t)$
(6)	**PENC**	$PKDec(CN, ENC_K\{t\}, \bar{k}) \supset \exists MN .Send(MN .m) \wedge Contatins(m, ENC_K\{t\})$
(7)	**VER**	$Honest(MN) \wedge Verify(CN, SIG_{\bar{K}}\{t\}, k) \wedge CN \neq MN \supset \exists MN .Send(MN, m) \wedge Contains(m, SIG_{\bar{K}}\{t\})$
(8)	Γ_{PKBU1}	$Has(Z, \bar{K}_{MN-CN}) \equiv Computes(Z, Verify(CN, SIG_{\bar{K}_{MN-CN}}\{t\}))$ $Has(X, k) \equiv Computes(X, PKDec(CN, ENC_k\{t\}, \bar{k}))$
(9)	**(2)-(8)**	$\theta_{PKBU2}[\text{Receive } \hat{MN}, \hat{CN}, m;\ \text{match } m/\text{CN}, m_{\text{BU}}, ver, dec;\ \text{match } ver/Verify(\hat{CN}, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})\ \text{match } dec/PKDec(CN, ENC_k\{t\}, \bar{k})]_{CN}\ Has(Z, \bar{K}_{MN-CN}) \equiv Has(Z, Verify(\hat{CN}, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})) \supset Computes(Z, Verify(\hat{CN}, SIG_{\bar{K}_{MN-CN}}\{t\}, K_{MN-CN})) \supset Has(Z, \bar{K}_{MN-CN}) \wedge Has(X, k) \equiv Has(X, PKDec(CN, ENC_k\{t\}, \bar{k})) \supset Computes(X, PKDec(CN, ENC_k\{t\}, \bar{k})) \supset Has(X, k)$
(10)	**AA4,**	$Honest(\hat{CN}) \wedge Honest(\hat{MN}) \supset Send(MN, \hat{MN}, \hat{CN}, m_{BU}) < \text{Re}ceive(CN, \hat{MN}, \hat{CN}, m_{BU})$
(11)	**HASH3,** **ARP**	$\theta_{PKBU2}[\text{Receive } \hat{HA}, \hat{CN}, m;\ \text{match } m_{\text{InitCR}}/HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})]_{CN}\ \text{Re}ceive(CN, \{\hat{HA}, \hat{CN}, m_{\text{InitCR}}, HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})\}) \supset \exists Z .(Computes(Z, HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})) \wedge Send(Z, \{CN, m_{\text{InitCR}}, HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})\})) < \text{Re}ceive(CN, \{\hat{HA}, \hat{CN}, CN, m_{\text{InitCR}}, HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})\}))$
(12)	**HASH1**	$Computes(Z, HASH_{K_{HA-CN}}(y, m_{\text{InitCR}})) \supset Has(Z, K_{HA-CN}) \wedge Has(Z, \{y, m_{\text{InitCR}}\})$
(13)	**HASH4**	$Has(CN, HASH_{K_{HA-CN}}\{t\}) \supset Computes(CN, HASH_{K_{HA-CN}}\{t\}) \vee \exists HA, m .Computes(HA, HASH_{K_{HA-CN}}\{t\}) \wedge Send(HA, m) \wedge Contains(m, HASH_{K_{HA-CN}}\{t\})$

(14)	**(13),**	$\theta_{PKBU\,2}$[Receive $\hat{HA}, \hat{CN}, m$; match $m_{\text{InitCR}}/HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})]_{CN} Honest(\hat{CN}) \wedge Honest(\hat{HA}) \wedge Has(Z, K_{HA-CN}) \supset Z = \hat{CN} \vee Z = \hat{HA}$
(15)	**(11), (14)**	$\theta_{PKBU\,2}$[Receive $\hat{HA}, \hat{CN}, m$; match $m_{\text{InitCR}}/HASH_{K_{HA-CN}}(CN, m_{\text{InitCR}})]_{CN} Honest(\hat{CN}) \wedge Honest(\hat{HA}) \supset Send(HA, \hat{HA}, \hat{CN}, m_{\text{InitCR}}) < \text{Receive}(CN, \hat{HA}, \hat{CN}, m_{\text{InitCR}})$
(16)	**(1),(6),(9),** **(10),(14)**	$\theta_{PKBU\,2} \wedge \theta_{PKBU\,3}\, Honest(\hat{MN}) \wedge Honest(\hat{CN}) \wedge Honest(\hat{HA}) \supset Send(HA, \hat{HA}, \hat{CN}, m_{\text{InitCR}}) < \text{Receive}(CN, \hat{HA}, \hat{CN}, m_{\text{InitCR}}) < Send(MN, \hat{HA}, \hat{CN}, m_{BU}) < \text{Receive}(CN, \hat{MN}, \hat{CN}, m_{BU}) <$

The formal verification of the PKBU protocol has further confirmed our intuitive analysis that our protocol has satisfied the security requirements, and defended against known attacks. In the next section, we will focus on the simulation modeling of the PKBU protocol.

5.3 Simulation Modelling

The network simulator used to program the proposed solution is the INETMANET-2.0 framework in OMNeT++ (Varga, 2010). OMNeT++ has powerful graphical tools that can display the simulation statistics and it is event-driven. In order to denote the network design, it uses hierarchically-linked domains. The advantage of this simulator is that it is possible to run different configuration on the layout and connection without having to reprogram it every time. Only appropriate variables and certain codes need to be modified to perform the necessary tasks and test. The OMNeT ++ network simulator was developed to give a generic component architecture. This allows the designers to define the network elements, protocols, or the wireless channels as model components or “modules”, easily.

C++ is the programming language used in OMNeT++. In the OMNeT++ simulation, the components required are: the NED file which contains the network definitions where all the nodes and connections between them are defined; the .CC files and .msg class where .cc files act as the main body of the simulation code. It defines the node's behaviour and how it handles and forwards messages; and the .msg component defines the message packet sent between the nodes and is automatically generated by the simulation.

5.3.1 Private Key-based Binding Update Protocol Description and Design

Figure 5.1 describes the PKBU network topology that has been created. It shows the “nodes” being connected together via “links”. These nodes can be blocks, entities, modules, etc, while the links can either be the channels, connections, etc. To ease the implementation of PKBU, a pair of .ned files is created. One of these .ned files is for each simple module while another .ned file is for the network that contains these simple modules. In OMNeT ++, once the model has been created, the parameters must be defined thus, some functions that are needed by the model can be executed. They are configured via an INI file and the owner of these parameters might be the model, or its modules or sub-modules. They can be accessed in their normal mode as well as the source mode. We implement our specific message for a model evaluation using the message files. They contain specific codes of various defined parameters, as well as the set values from other C++ programs.

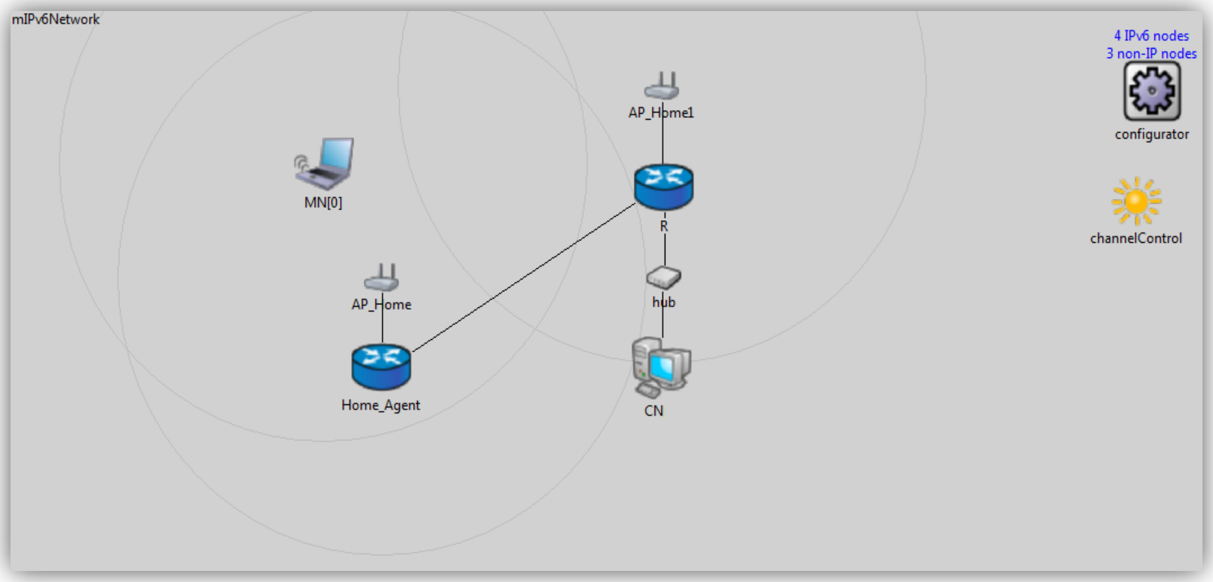


Figure 5.1: Topology of PKBU Network

- **Modules in PKBU**

Figure 5.1, shows a network file, called MIPv6Network_H.ned which contains the following entities:

- A single stationary *CN* connected to the Internet through the router *R*.
- The home link contains a single *MN*.
- The home link is represented by an *HA*.
- There are two access points: *AP_Home* and *AP_Home1*.

The location of the nodes in the network is shown in Figure 5.1. The following modules are applied to the entities in the PKBU protocol:

- **CorrespondentNode6:** It is an IPv6 host that comes with MIPv6 support. It has a Binding Cache that is updated with every received *BU*. The *CN* that uses this CorrespondentNode6 module is shown in Figure 5.2.

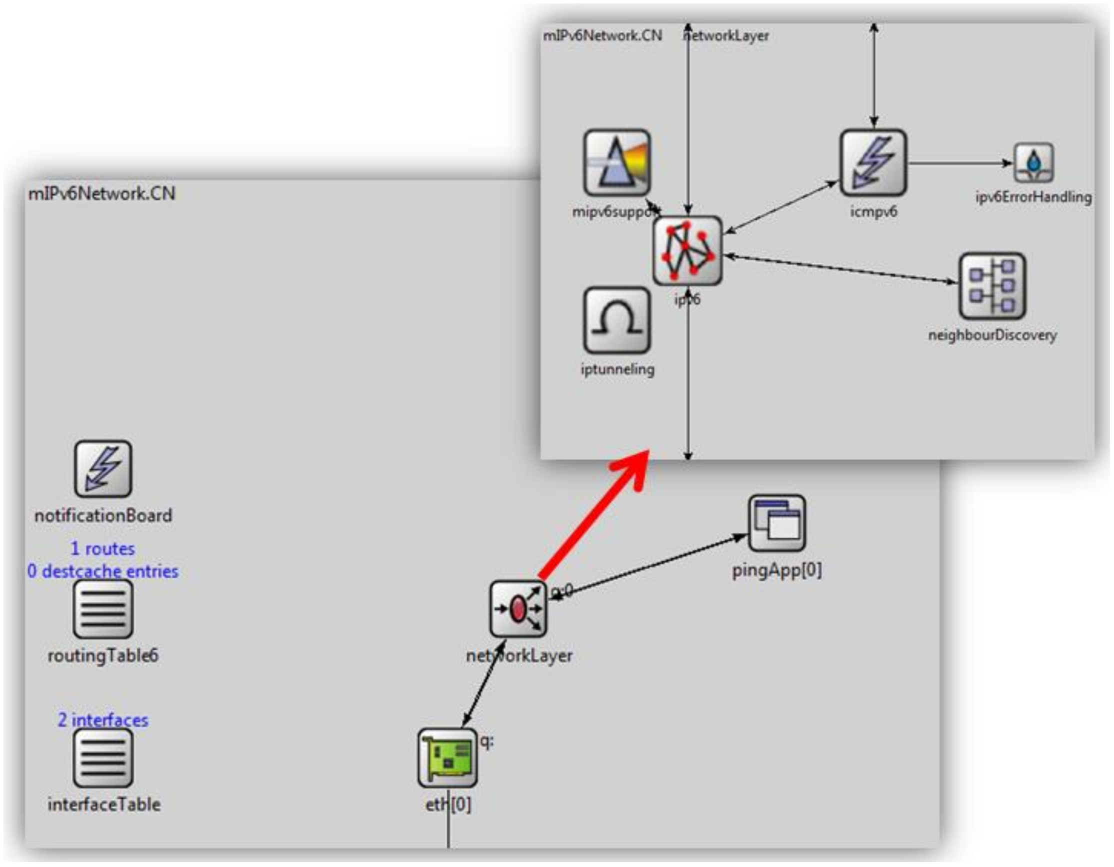


Figure 5.2: The *CN* Hierarchy Model

- **WirelessHost6:** An entity that has one wireless card in infrastructure mode, and supports handover as well as the MIPv6 protocol. Figure 5.3 shows the *MN* which uses the WirelessHost6 module.

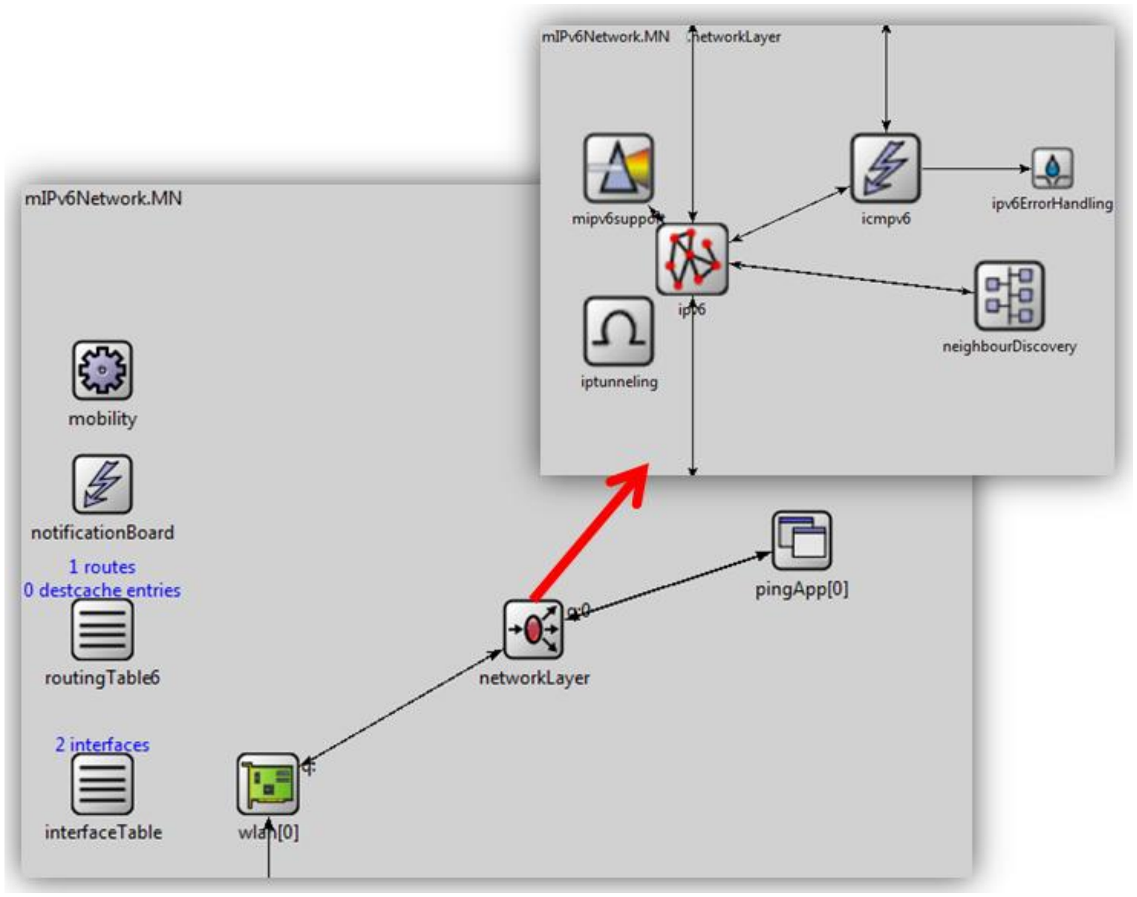


Figure 5.3: The *MN* Hierarchy Model

- **HomeAgent6:** An operating *HA* that supports MIPv6 and is configured from an IPv6 router. The usage of the HomeAgent6 on a *HA* is shown in Figure 5.4.

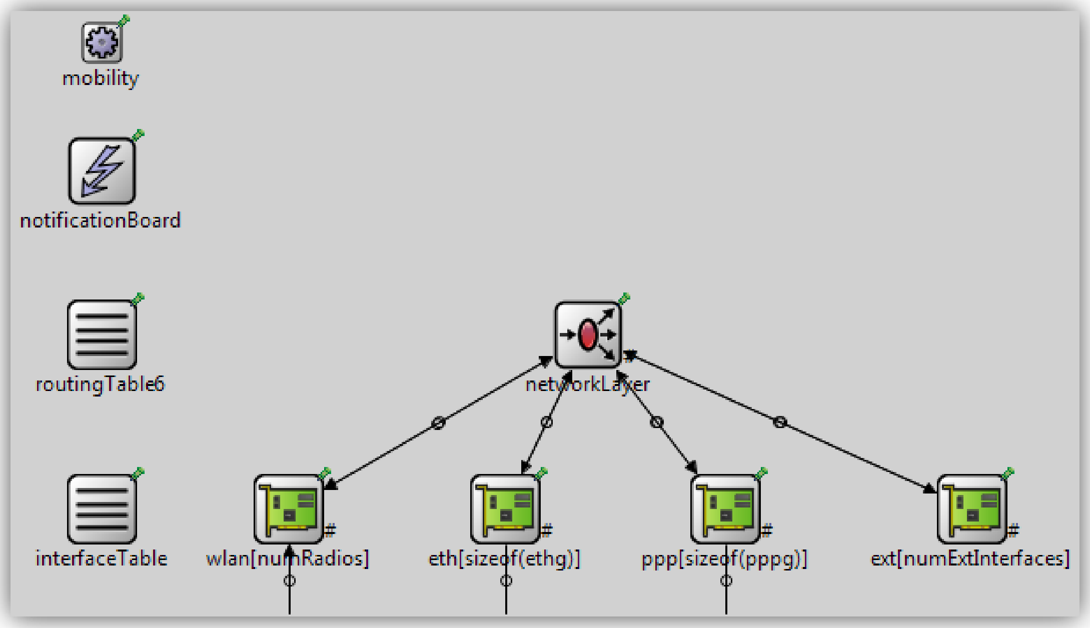


Figure 5.4: The Home_Agent Hierarchy Model

- **AccessPoint**: A standard access point that supports multiple wireless radios and Ethernet ports. Figure 5.5 illustrates the *AP_Home* which uses the AccessPoint module.

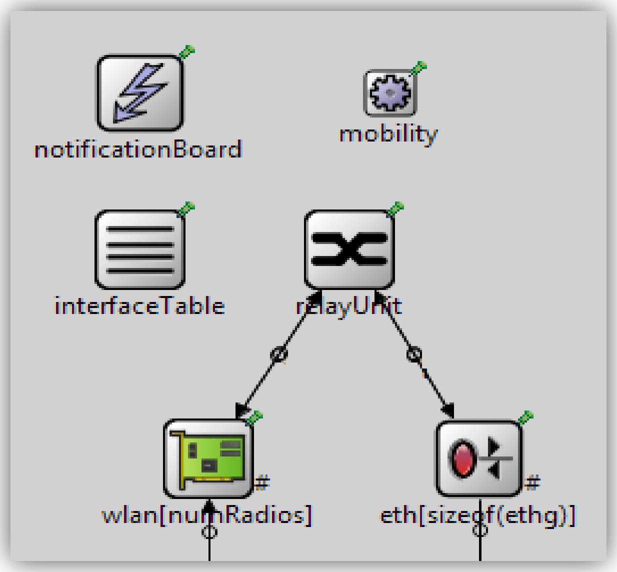


Figure 5.5: The AP_Home Hierarchy Model

- **EtherHub:** Messages are broadcasted to every other port once it arrives at one port. Similar data rates on all connecting incoming and outgoing rates are required in order for the model to function correctly.

- **ChannelControl:** If a network model contains mobile or wireless nodes, then it will have exactly one ChannelControl module. This module is kept informed of the movements and location of the nodes, and it will determine which of the nodes are within interference distance or which is within the communication. This type of information will be then used for transmission by the nodes' radio interface.

- **Modules connection**

Communication between the modules is done by message passing. The module also handles timers or timeouts by sending itself messages (self-messages). These messages could either come from a class cMessage, or from a class that is cMessage-based. They are delivered to the module's handleMessage method where we will redefine and add codes into. Everything that we want the module to do is contained in the handleMessage() method. The messages are defined for interpretation when there are messages travelling between different entities, for example, *HA*-*MN*, *HA*-*CN*, or *MN*-CN. The MgmtAP and MgmtSTA classes are used to manage the frames of the *HA*, the *MN* and the *CN*.

5.3.2 Running Simulation

This section discusses the PKBU steps after running the simulation.

- First, in order for the *MN* to receive the *AP*'s broadcast beacons, it has to be within the range of the *AP*. Before the *MN* and the *AP* start exchanging messages, the authentication and association messages have to be exchanged between the nodes.
  - This is done with the *MN* sending a message containing the hashed *HoA* and the public key of the *MN* to the *AP*. The message also contains a request for the *CN*'s public key.
- In order for this message (in the form of an IPv6 packet) to be sent through the airframe, it has to be encapsulated.
  - This means that certain fields have to be set before sending it as an "*AirFrame*" packet. The process of encapsulation has been written while the

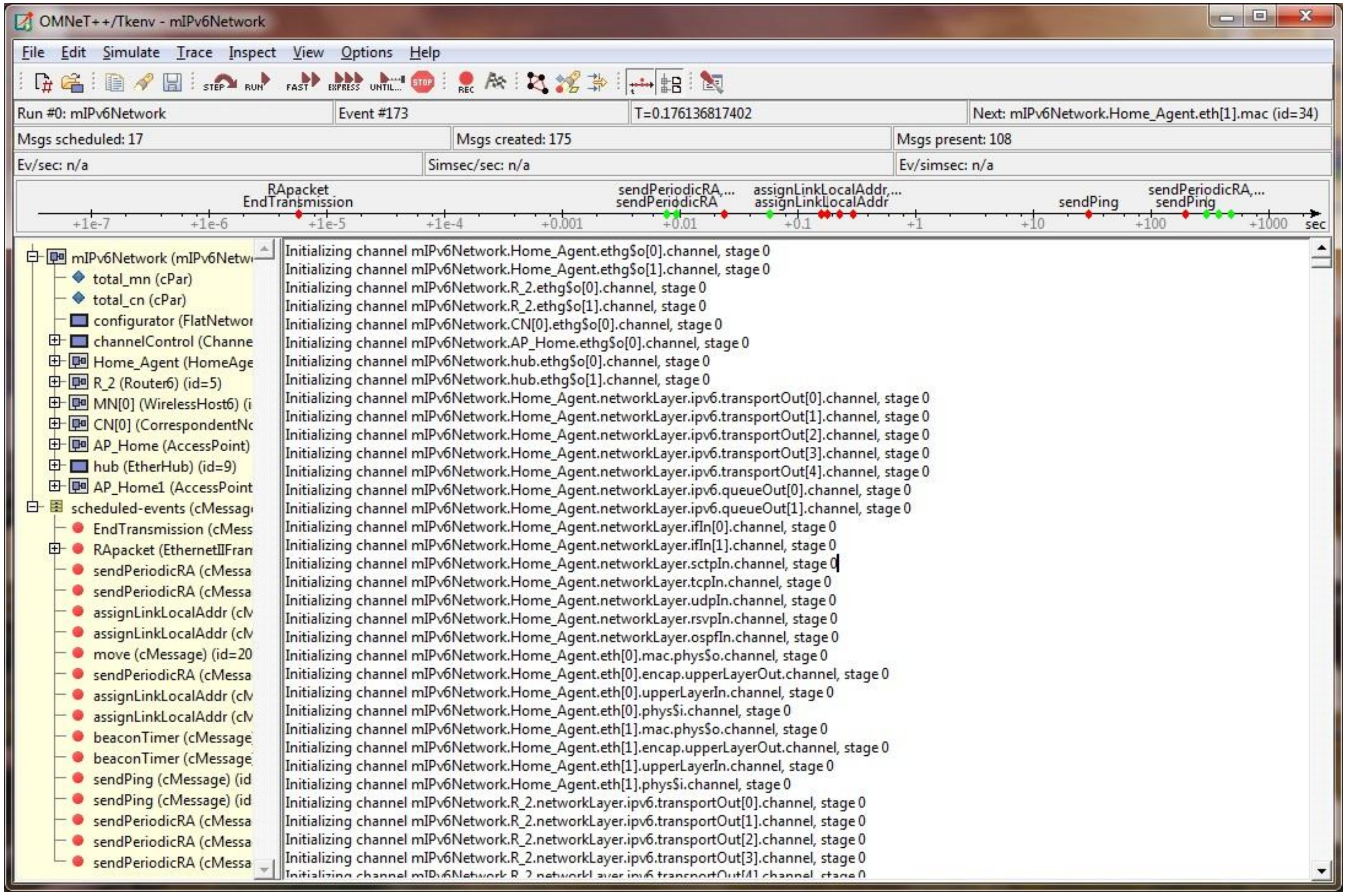
    AP’s IPv6 address has been set to the “*ReceiverAddress*” variable in its function.

- The packet is ready to be sent to the *“MAC”* layer, which is the next layer.
    - First, all “*initialise ()”* functions will be called once the simulation begins.
    - Then the “*ReceiveChangeNotification*” function is called whenever the host changes its location. The timer that was defined earlier during the initialising process is set to a pre-configured value. The value is then sent to the “*HandleTimer ()*” function. During the simulation, this function will receive different types of messages, and will react accordingly to those messages.
- The *AP* will send an acknowledgement back to the *MN* when it receives the *MN*’s message.
- The *AP* will direct the *MN*’s message to the *HA*, which will, in turn, send it to the *CN*’s router.
- Once the *CN* receives the message, the *“HandleDataFrame ()”* function will be executed.
    - This function will then de-encapsulate the message and extract the information contained in it. The extracted message is processed and the appropriate information is sent back. The following step is to send that processed message with the relevant information back to the *MN*.
- The *CN*, in turn, will receive the message from the *MN*. The *MN*’s *HoA* and *CoA* which were encrypted by the *CN*’s public key and signed by *MN*’s private key is stored in this message.
    - Before sending the message via the airframe form of IPv6 packet, it has to be encapsulated. The *“HandleDateFrame ()”* is called again when the *CN*

receives the message. The function will de-encapsulate the message and extract the information in it. The *CN* then checks the validity of the message signature and the correctness of the *HoA*.

- The *CN* will authenticate the *MN*'s signature using the *MN*'s public key.
- The *CN* will decrypt the message which contains the *MN*'s *CoA* and *HoA*.
- The hash value of the *HoA* is calculated and compared with the hash value of the first message.
    - The message is rejected if a negative result is received by either the *MN* or the *CN*. However, if the result is positive, then the *CN* is assured of the correctness of the *MN*'s *HoA* and *CoA*. The *MN* assures the *CN* by using this method to validate the ownership of the *MN*'s *HoA*, and the reachability of the *CoA*. This completes the last step.

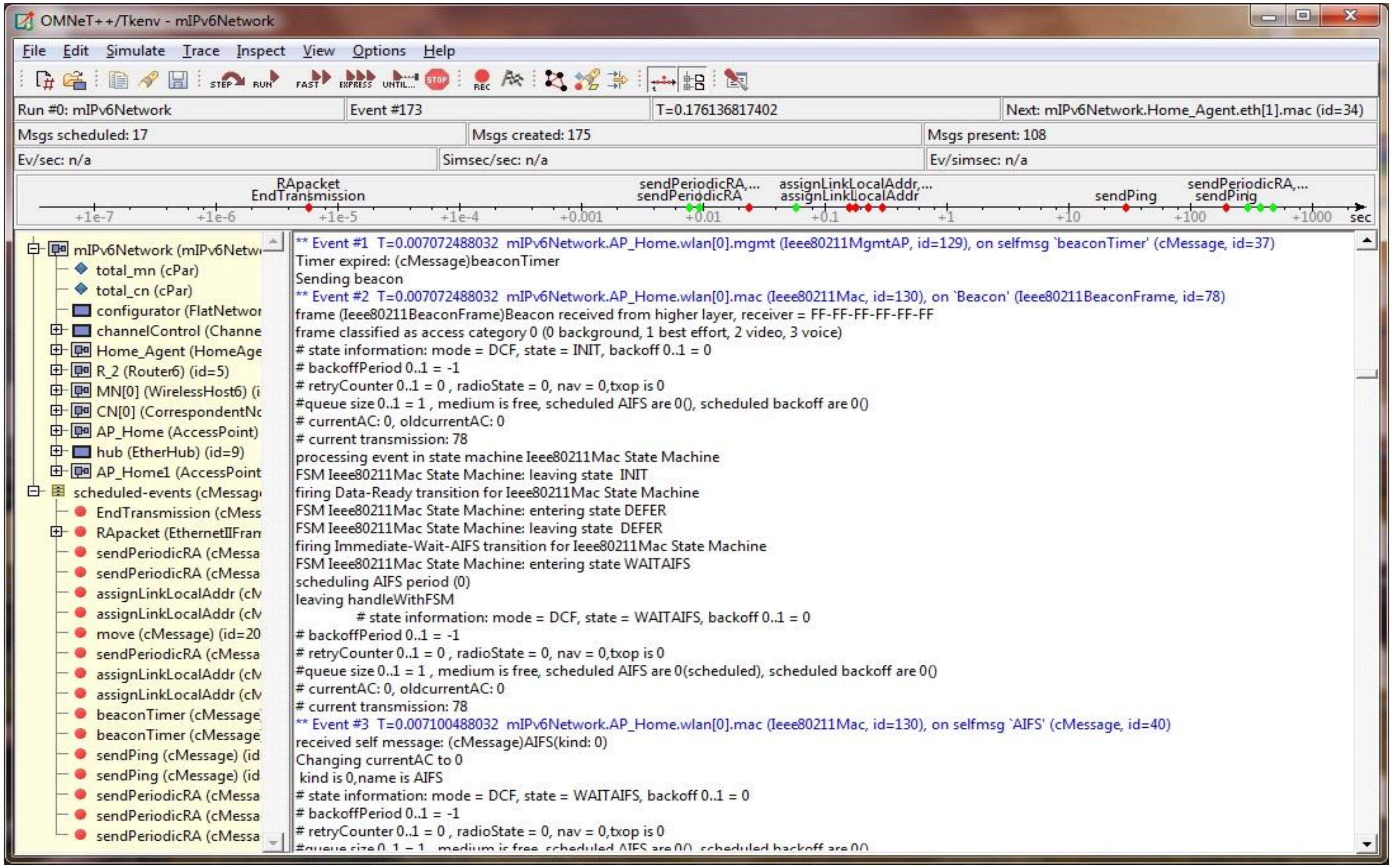
Figure 5.6 shows the initialising channel for PKBU after running the project. Figure 5.7 and Figure 5.8 show some of the events of the nodes.



```
Run #0: mIPv6Network          Event #173          T=0.176136817402          Next: mIPv6Network.Home_Agent.eth[1].mac (id=34)
Msgs scheduled: 17            Msgs created: 175            Msgs present: 108
Ev/sec: n/a                   Simsec/sec: n/a              Ev/simsec: n/a
```

```
Initializing channel mIPv6Network.Home_Agent.ethg$o[0].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.ethg$o[1].channel, stage 0
Initializing channel mIPv6Network.R_2.ethg$o[0].channel, stage 0
Initializing channel mIPv6Network.R_2.ethg$o[1].channel, stage 0
Initializing channel mIPv6Network.CN[0].ethg$o[0].channel, stage 0
Initializing channel mIPv6Network.AP_Home.ethg$o[0].channel, stage 0
Initializing channel mIPv6Network.hub.ethg$o[0].channel, stage 0
Initializing channel mIPv6Network.hub.ethg$o[1].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.transportOut[0].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.transportOut[1].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.transportOut[2].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.transportOut[3].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.transportOut[4].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.queueOut[0].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ipv6.queueOut[1].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ifIn[0].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ifIn[1].channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.sctpIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.tcpIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.udpIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.rsvpIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.networkLayer.ospfIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[0].mac.phys$o.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[0].encap.upperLayerOut.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[0].upperLayerIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[0].phys$i.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[1].mac.phys$o.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[1].encap.upperLayerOut.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[1].upperLayerIn.channel, stage 0
Initializing channel mIPv6Network.Home_Agent.eth[1].phys$i.channel, stage 0
Initializing channel mIPv6Network.R_2.networkLayer.ipv6.transportOut[0].channel, stage 0
Initializing channel mIPv6Network.R_2.networkLayer.ipv6.transportOut[1].channel, stage 0
Initializing channel mIPv6Network.R_2.networkLayer.ipv6.transportOut[2].channel, stage 0
Initializing channel mIPv6Network.R_2.networkLayer.ipv6.transportOut[3].channel, stage 0
Initializing channel mIPv6Network.R_2.networkLayer.ipv6.transportOut[4].channel, stage 0
```

Figure 5.6: Channel Initialising in PKBU


OMNeT++/Tkenv - mIPv6Network
File Edit Simulate Trace Inspect View Options Help
STEP RUN FAST EXPRESS UNTIL... STOP REC
RApacket
EndTransmission
sendPeriodicRA,...
sendPeriodicRA
assignLinkLocalAddr,...
assignLinkLocalAddr
sendPing
sendPeriodicRA,...
sendPing
+1e-7 +1e-6 +1e-5 +1e-4 +0.001 +0.01 +0.1 +1 +10 +100 +1000 sec

Run #0: mIPv6Network | Event #173 | T=0.176136817402 | Next: mIPv6Network.Home_Agent.eth[1].mac (id=34)

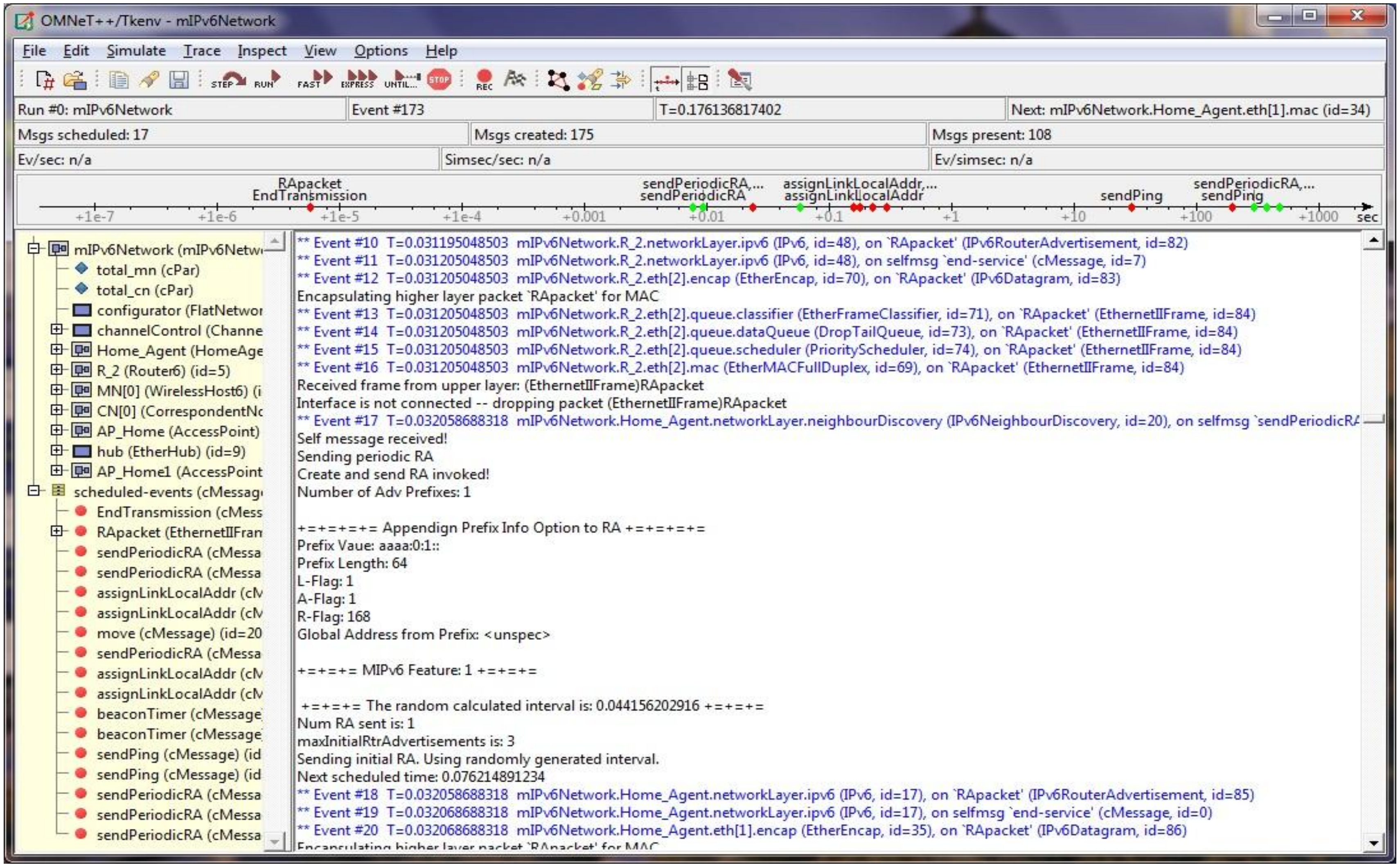
Msgs scheduled: 17 | Msgs created: 175 | Msgs present: 108

Ev/sec: n/a | Simsec/sec: n/a | Ev/simsec: n/a

```
mIPv6Network (mIPv6Netw
  total_mn (cPar)
  total_cn (cPar)
  configurator (FlatNetwor
  channelControl (Channe
  Home_Agent (HomeAge
  R_2 (Router6) (id=5)
  MN[0] (WirelessHost6) (i
  CN[0] (CorrespondentNc
  AP_Home (AccessPoint)
  hub (EtherHub) (id=9)
  AP_Home1 (AccessPoint
scheduled-events (cMessag
  EndTransmission (cMess
  RApacket (EthernetIIFran
  sendPeriodicRA (cMessa
  sendPeriodicRA (cMessa
  assignLinkLocalAddr (cM
  assignLinkLocalAddr (cM
  move (cMessage) (id=20
  sendPeriodicRA (cMessa
  assignLinkLocalAddr (cM
  assignLinkLocalAddr (cM
  beaconTimer (cMessage
  beaconTimer (cMessage
  sendPing (cMessage) (id
  sendPing (cMessage) (id
  sendPeriodicRA (cMessa
  sendPeriodicRA (cMessa
  sendPeriodicRA (cMessa
```

```
** Event #1  T=0.007072488032  mIPv6Network.AP_Home.wlan[0].mgmt (Ieee80211MgmtAP, id=129), on selfmsg `beaconTimer' (cMessage, id=37)
Timer expired: (cMessage)beaconTimer
Sending beacon
** Event #2  T=0.007072488032  mIPv6Network.AP_Home.wlan[0].mac (Ieee80211Mac, id=130), on `Beacon' (Ieee80211BeaconFrame, id=78)
frame (Ieee80211BeaconFrame)Beacon received from higher layer, receiver = FF-FF-FF-FF-FF-FF
frame classified as access category 0 (0 background, 1 best effort, 2 video, 3 voice)
# state information: mode = DCF, state = INIT, backoff 0..1 = 0
# backoffPeriod 0..1 = -1
# retryCounter 0..1 = 0 , radioState = 0, nav = 0,txop is 0
#queue size 0..1 = 1 , medium is free, scheduled AIFS are 0(), scheduled backoff are 0()
# currentAC: 0, oldcurrentAC: 0
# current transmission: 78
processing event in state machine Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: leaving state  INIT
firing Data-Ready transition for Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: entering state DEFER
FSM Ieee80211Mac State Machine: leaving state  DEFER
firing Immediate-Wait-AIFS transition for Ieee80211Mac State Machine
FSM Ieee80211Mac State Machine: entering state WAITAIFS
scheduling AIFS period (0)
leaving handleWithFSM
            # state information: mode = DCF, state = WAITAIFS, backoff 0..1 = 0
# backoffPeriod 0..1 = -1
# retryCounter 0..1 = 0 , radioState = 0, nav = 0,txop is 0
#queue size 0..1 = 1 , medium is free, scheduled AIFS are 0(scheduled), scheduled backoff are 0()
# currentAC: 0, oldcurrentAC: 0
# current transmission: 78
** Event #3  T=0.007100488032  mIPv6Network.AP_Home.wlan[0].mac (Ieee80211Mac, id=130), on selfmsg `AIFS' (cMessage, id=40)
received self message: (cMessage)AIFS(kind: 0)
Changing currentAC to 0
 kind is 0,name is AIFS
# state information: mode = DCF, state = WAITAIFS, backoff 0..1 = 0
# backoffPeriod 0..1 = -1
# retryCounter 0..1 = 0 , radioState = 0, nav = 0,txop is 0
```

Figure 5.7: Start PKBU Events


OMNeT++/Tkenv - mIPv6Network
File Edit Simulate Trace Inspect View Options Help
STEP RUN FAST EXPRESS UNTIL... STOP REC
RApacket
EndTransmission
sendPeriodicRA,...
sendPeriodicRA
assignLinkLocalAddr,...
assignLinkLocalAddr
sendPing
sendPeriodicRA,...
sendPing
+1e-7 +1e-6 +1e-5 +1e-4 +0.001 +0.01 +0.1 +1 +10 +100 +1000 sec

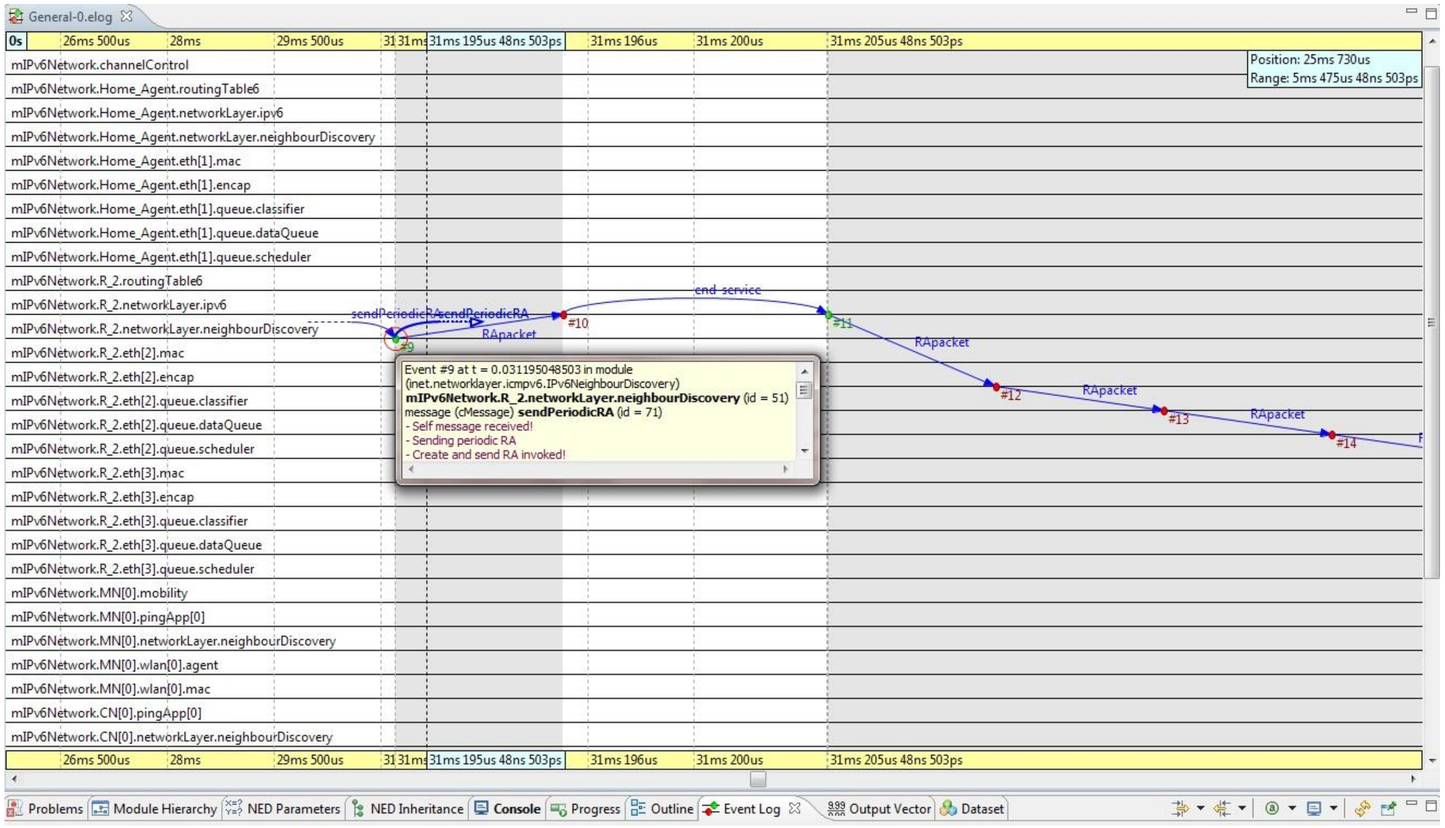
```
Run #0: mIPv6Network          Event #173          T=0.176136817402          Next: mIPv6Network.Home_Agent.eth[1].mac (id=34)
Msgs scheduled: 17            Msgs created: 175                            Msgs present: 108
Ev/sec: n/a                   Simsec/sec: n/a                              Ev/simsec: n/a
```

```
mIPv6Network (mIPv6Netw
   total_mn (cPar)
   total_cn (cPar)
   configurator (FlatNetwor
   channelControl (Channe
   Home_Agent (HomeAge
   R_2 (Router6) (id=5)
   MN[0] (WirelessHost6) (i
   CN[0] (CorrespondentNc
   AP_Home (AccessPoint)
   hub (EtherHub) (id=9)
   AP_Home1 (AccessPoint
scheduled-events (cMessag
   EndTransmission (cMess
   RApacket (EthernetIIFran
   sendPeriodicRA (cMessa
   sendPeriodicRA (cMessa
   assignLinkLocalAddr (cM
   assignLinkLocalAddr (cM
   move (cMessage) (id=20
   sendPeriodicRA (cMessa
   assignLinkLocalAddr (cM
   assignLinkLocalAddr (cM
   beaconTimer (cMessage
   beaconTimer (cMessage
   sendPing (cMessage) (id
   sendPing (cMessage) (id
   sendPeriodicRA (cMessa
   sendPeriodicRA (cMessa
   sendPeriodicRA (cMessa
```

```
** Event #10  T=0.031195048503  mIPv6Network.R_2.networkLayer.ipv6 (IPv6, id=48), on `RApacket' (IPv6RouterAdvertisement, id=82)
** Event #11  T=0.031205048503  mIPv6Network.R_2.networkLayer.ipv6 (IPv6, id=48), on selfmsg `end-service' (cMessage, id=7)
** Event #12  T=0.031205048503  mIPv6Network.R_2.eth[2].encap (EtherEncap, id=70), on `RApacket' (IPv6Datagram, id=83)
Encapsulating higher layer packet `RApacket' for MAC
** Event #13  T=0.031205048503  mIPv6Network.R_2.eth[2].queue.classifier (EtherFrameClassifier, id=71), on `RApacket' (EthernetIIFrame, id=84)
** Event #14  T=0.031205048503  mIPv6Network.R_2.eth[2].queue.dataQueue (DropTailQueue, id=73), on `RApacket' (EthernetIIFrame, id=84)
** Event #15  T=0.031205048503  mIPv6Network.R_2.eth[2].queue.scheduler (PriorityScheduler, id=74), on `RApacket' (EthernetIIFrame, id=84)
** Event #16  T=0.031205048503  mIPv6Network.R_2.eth[2].mac (EtherMACFullDuplex, id=69), on `RApacket' (EthernetIIFrame, id=84)
Received frame from upper layer: (EthernetIIFrame)RApacket
Interface is not connected -- dropping packet (EthernetIIFrame)RApacket
** Event #17  T=0.032058688318  mIPv6Network.Home_Agent.networkLayer.neighbourDiscovery (IPv6NeighbourDiscovery, id=20), on selfmsg `sendPeriodicRA
Self message received!
Sending periodic RA
Create and send RA invoked!
Number of Adv Prefixes: 1

+=+=+=+= Appendign Prefix Info Option to RA +=+=+=+=
Prefix Vaue: aaaa:0:1::
Prefix Length: 64
L-Flag: 1
A-Flag: 1
R-Flag: 168
Global Address from Prefix: <unspec>

+=+=+= MIPv6 Feature: 1 +=+=+=

 +=+=+= The random calculated interval is: 0.044156202916 +=+=+=
Num RA sent is: 1
maxInitialRtrAdvertisements is: 3
Sending initial RA. Using randomly generated interval.
Next scheduled time: 0.076214891234
** Event #18  T=0.032058688318  mIPv6Network.Home_Agent.networkLayer.ipv6 (IPv6, id=17), on `RApacket' (IPv6RouterAdvertisement, id=85)
** Event #19  T=0.032068688318  mIPv6Network.Home_Agent.networkLayer.ipv6 (IPv6, id=17), on selfmsg `end-service' (cMessage, id=0)
** Event #20  T=0.032068688318  mIPv6Network.Home_Agent.eth[1].encap (EtherEncap, id=35), on `RApacket' (IPv6Datagram, id=86)
```

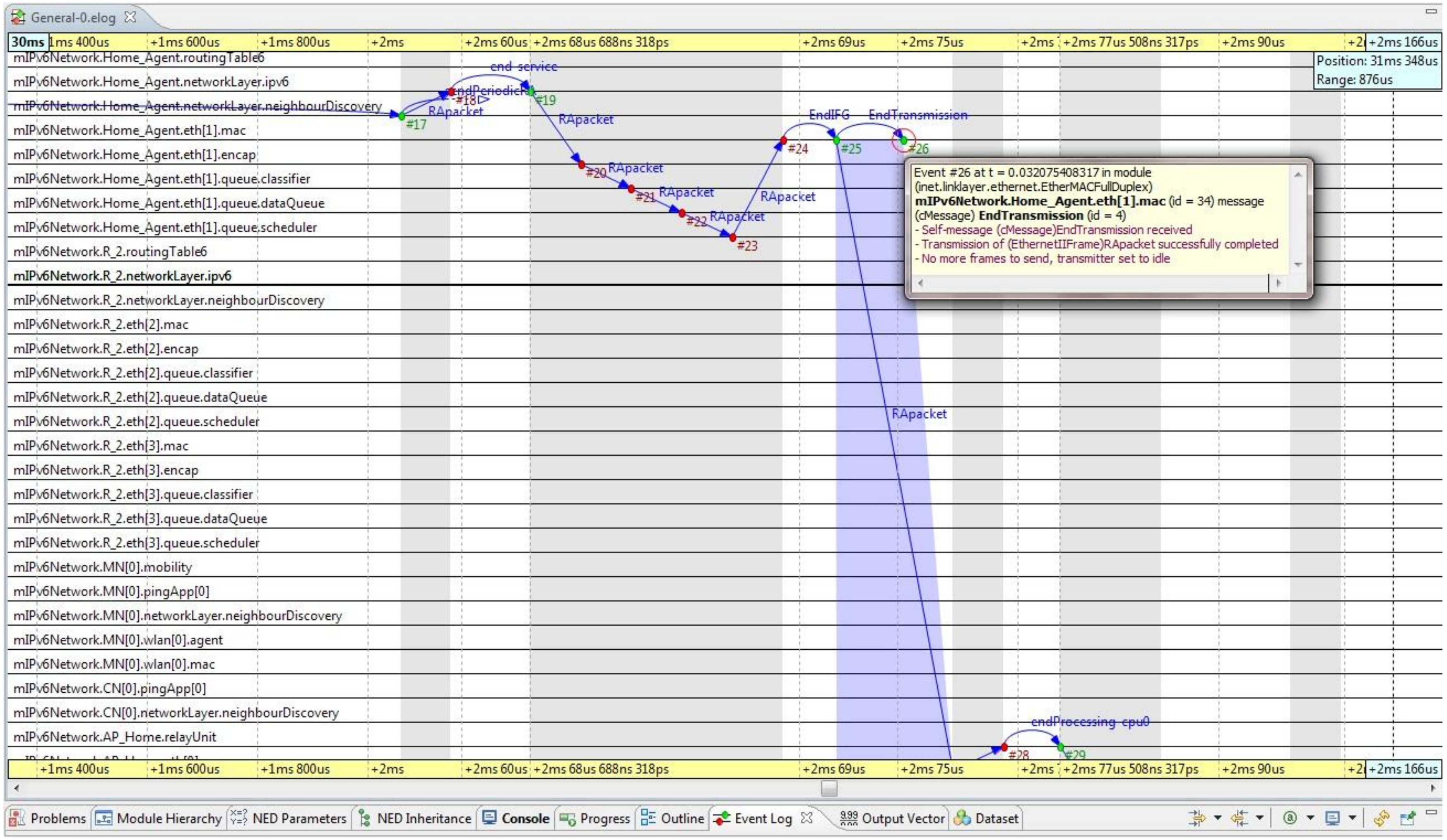
Figure 5.8: Message Exchanged between the Nodes

Figure 5.9 and Figure 5.10 are the sequence charts for PKBU. By clicking on each event, the corresponding log lines in the event can be viewed.


General-0.elog
0s
26ms 500us
28ms
29ms 500us
31ms 195us 48ns 503ps
31ms 196us
31ms 200us
31ms 205us 48ns 503ps
Position: 25ms 730us
Range: 5ms 475us 48ns 503ps
mIPv6Network.channelControl
mIPv6Network.Home_Agent.routingTable6
mIPv6Network.Home_Agent.networkLayer.ipv6
mIPv6Network.Home_Agent.networkLayer.neighbourDiscovery
mIPv6Network.Home_Agent.eth[1].mac
mIPv6Network.Home_Agent.eth[1].encap
mIPv6Network.Home_Agent.eth[1].queue.classifier
mIPv6Network.Home_Agent.eth[1].queue.dataQueue
mIPv6Network.Home_Agent.eth[1].queue.scheduler
mIPv6Network.R_2.routingTable6
mIPv6Network.R_2.networkLayer.ipv6
mIPv6Network.R_2.networkLayer.neighbourDiscovery
mIPv6Network.R_2.eth[2].mac
mIPv6Network.R_2.eth[2].encap
mIPv6Network.R_2.eth[2].queue.classifier
mIPv6Network.R_2.eth[2].queue.dataQueue
mIPv6Network.R_2.eth[2].queue.scheduler
mIPv6Network.R_2.eth[3].mac
mIPv6Network.R_2.eth[3].encap
mIPv6Network.R_2.eth[3].queue.classifier
mIPv6Network.R_2.eth[3].queue.dataQueue
mIPv6Network.R_2.eth[3].queue.scheduler
mIPv6Network.MN[0].mobility
mIPv6Network.MN[0].pingApp[0]
mIPv6Network.MN[0].networkLayer.neighbourDiscovery
mIPv6Network.MN[0].wlan[0].agent
mIPv6Network.MN[0].wlan[0].mac
mIPv6Network.CN[0].pingApp[0]
mIPv6Network.CN[0].networkLayer.neighbourDiscovery
sendPeriodicRA
sendPeriodicRA
RApacket
end service
#9
#10
#11
RApacket
#12
RApacket
#13
RApacket
#14
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Progress
Outline
Event Log
Output Vector
Dataset

```
Event #9 at t = 0.031195048503 in module
(inet.networklayer.icmpv6.IPv6NeighbourDiscovery)
mIPv6Network.R_2.networkLayer.neighbourDiscovery (id = 51)
message (cMessage) sendPeriodicRA (id = 71)
- Self message received!
- Sending periodic RA
- Create and send RA invoked!
```

Figure 5.9: Elog File for Event 9 to 14


General-0.elog
30ms
+1ms 400us
+1ms 600us
+1ms 800us
+2ms
+2ms 60us
+2ms 68us 688ns 318ps
+2ms 69us
+2ms 75us
+2ms 77us 508ns 317ps
+2ms 90us
+2ms 166us
Position: 31ms 348us
Range: 876us
mIPv6Network.Home_Agent.routingTable6
mIPv6Network.Home_Agent.networkLayer.ipv6
mIPv6Network.Home_Agent.networkLayer.neighbourDiscovery
mIPv6Network.Home_Agent.eth[1].mac
mIPv6Network.Home_Agent.eth[1].encap
mIPv6Network.Home_Agent.eth[1].queue.classifier
mIPv6Network.Home_Agent.eth[1].queue.dataQueue
mIPv6Network.Home_Agent.eth[1].queue.scheduler
mIPv6Network.R_2.routingTable6
mIPv6Network.R_2.networkLayer.ipv6
mIPv6Network.R_2.networkLayer.neighbourDiscovery
mIPv6Network.R_2.eth[2].mac
mIPv6Network.R_2.eth[2].encap
mIPv6Network.R_2.eth[2].queue.classifier
mIPv6Network.R_2.eth[2].queue.dataQueue
mIPv6Network.R_2.eth[2].queue.scheduler
mIPv6Network.R_2.eth[3].mac
mIPv6Network.R_2.eth[3].encap
mIPv6Network.R_2.eth[3].queue.classifier
mIPv6Network.R_2.eth[3].queue.dataQueue
mIPv6Network.R_2.eth[3].queue.scheduler
mIPv6Network.MN[0].mobility
mIPv6Network.MN[0].pingApp[0]
mIPv6Network.MN[0].networkLayer.neighbourDiscovery
mIPv6Network.MN[0].wlan[0].agent
mIPv6Network.MN[0].wlan[0].mac
mIPv6Network.CN[0].pingApp[0]
mIPv6Network.CN[0].networkLayer.neighbourDiscovery
mIPv6Network.AP_Home.relayUnit
end-service
sendPeriodicRA
RApacket
#17
#18
#19
#20
#21
#22
#23
#24
#25
#26
#28
#29
EndIFG
EndTransmission
endProcessing-cpu0
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Progress
Outline
Event Log
Output Vector
Dataset

Event #26 at t = 0.032075408317 in module (inet.linklayer.ethernet.EtherMACFullDuplex) **mIPv6Network.Home_Agent.eth[1].mac** (id = 34) message (cMessage) **EndTransmission** (id = 4)
- Self-message (cMessage)EndTransmission received
- Transmission of (EthernetIIFrame)RApacket successfully completed
- No more frames to send, transmitter set to idle

Figure 5.10 : Elog File for Event 17 to 29

The following scenarios describe the possible main attacks and their simulation:

1. The DoS attacks can be shown in the following scenario. The *CN* corresponds with the *MN1* and vice versa. For example, if a data stream is being transmitted between the two nodes, the attacker node (*AN1*), as shown in Figure 5.11, will send a *BU* message containing the *HoA* of the *MN1*as its *HoA*, and the *CoA* of *MN2* is sent to the *CN*. Once the *CN* receives the *BU* message, it updates its binding cache with an entry of the *MN1*. Also, this entry will be updated to the *CoA* with the *MN2* address.

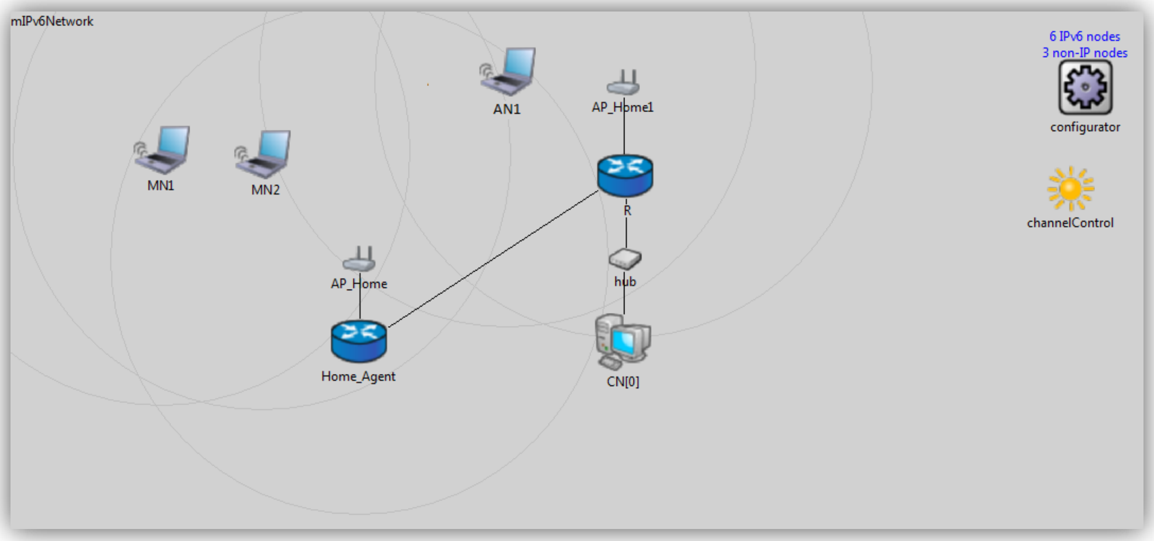


Figure 5.11: Attacks Simulation for PKBU Protocol in OMNeT ++

2. Once the attacks begin, the packets that are meant for the *MN1* from the *CN* will go to *MN2*, instead. This happens due to the illegitimate binding cache entry that was created when the *AN1* sent the *BU* message to the *CN*. The packet that was supposed to be delivered to the *MN1* has been blocked.

3. Any unnecessary data traffic associated with the attacker must now be handled by the *CN*. It is used as a packet reflector by the *AN1* or the attack perpetrator.

4. In another method of attack, the *BU* message is sent to the *CN* by the *AN* and the *AN* sets its IP address as the CoA of the *MN1*.

5. Another method of attack is session hijacking, that can be explained in the following scenario. Due to the illegitimate Binding Cash (*BC*) entry that was created when the *AN* sent the *BU*, any packet exchanged between the *MN* and the *CN* will actually happen between the *CN* and the *AN*. Thus, the *MN1* has been denied any packet rightfully meant for it.

6. The *AN* can also act as the Man-In-The-Middle (MITM). It can set the IP address as the *CoA* of the *MN1* in the *BC* entry of the *CN* by sending a *BU* message to the *CN*. This allows any message meant for the *MN* to be captured by the *AN*.

7. As for the MITM, the *AN* can eavesdrop on the packets and send the packets to the *MN*. It can also edit these packets and send those incorrect packets to the *MN*.

8. Based on the first scenario, the *MN2* is being attacked by unwanted and unsolicited streams of data traffic. It can also be described as the *CN* being attacked by data that is streamed to "nowhere". In the second scenario, the *CN* creates a binding with the *MN'sCoA* but sends subsequent data traffic to the *AN* believing that it is the new *CoA* of the *MN*. However, the *MN* will not receive the intended data. In this type of attack, the data packets will go to the *AN* instead of being exchanged between the *MN* and the *CN*.

9. In all scenarios mentioned in steps 1 to 8, there are four legitimate nodes (*HA*, *MN1*, *MN2*, *CN*) while there is one illegitimate node (*AN1*). This shows that the whole mobile network service can be put at risk by a single comprehensive attack.

10. We have implemented these attacks as well as presented the mitigation plan where the legitimate entities involved can work together to counter the attacks. The

mitigation plan must first detect false binding attacks, MITM attacks, and DoS attacks, reliably. In this context, the usefulness of PKBU is evident because it has rendered the *BU* messages secure and valid before allowing the *MN* to send them to the *CN*.

11. We propose the creation of a safe *HoA* as well as appropriate digital signatures, encryption algorithms, and hash functions within the proposed PKBU protocol.

- **Detection and Mitigation Simulation**

The following steps explain the simulated detection and mitigation plan:

1. The *HoA* of *MN1* can be reclaimed if the *AN* sends a fake *BU* to the *CN*. It can do this by sending a different *CoA* in the *BU* message. The *CN* will proceed to compare the hash value of the *HoA* in the *BU* message with the original hash value of the *HoA*. If the comparison result is negative, that means the *MN* does not own the new *HoA*.

2. The point of detection could be the *CN* itself when it is updating the *BC* after the *BU* is received. In PKBU, the *BU* packet is de-encapsulated after the *CN* receives it. It can proceed with ensuring that the signature used to sign the *BU* message is valid. Based on the PKBU framework, (as in the Chapter 4), the *MN* signs the *BU* message. The Elliptic Curve Digital Signature Algorithm (ECDSA) is used to digitally sign the *BU*. If the verification of the signature is correct, the *CN* can be sure that the packet was sent by the *MN*. The digital signature (r, s) of the *BU* message can be verified using the following steps:

   1. Acquire a valid copy of the sender public key *(E, P, n, Q)*.
   2. Verify r and s are integers in the interval [1, n-1].
   3. Compute *w = s-1 mod n and h(m).*
   4. Compute *u1= h(m).w mod n* and *u2 = r.w mod n*.

5. Compute $u1P+u2Q = (x_0 , y_0)$ and $v = x_0 \bmod n$.

6. The signature is accepted only if $v=r$.

In their research, A. Khaled and M. AL-Kayali found that the Elliptic Curve Discrete Logarithm Problem (ECDLP) algorithm provides strong security by merely using smaller keys. As the key is small, the memory required to store it would also be small. This keeps the size of the data transferred between the nodes to the minimum, thus, leading to shorter transmission time. The applications in the *MNs* require strong security features which are only achievable by using long keys. By using ECC, the *MN* can maintain their cost while still be able to provide strong security by using smaller size keys.

The processing time is greatly reduced in ECC, especially if the binary field $GF(2^k)$ (Appendix A) is applied. The *MN* can manage the elliptic curve domain parameters and the private key as well as generate the ECDSA signatures. In addition, the system is only limited to 162-bit size curves.

3. The *CN* can function as the detection point itself. Once the verification of the signature is done after it has de-encapsulated the packet, it will begin to decrypt the message. The *MN* will use the *CN*'s public key to encrypt the message before sending the packet to the *CN*. Based on the false binding update and the MITM attack scenarios (as in Section 4.6) the *AN* can eavesdrop on the *BU* packet. The whole message has to be signed using the private key belonging to *MN1*, which the *AN* does not have. If the *AN* decides to use parameters other than the original *MN* parameters as a signature, then the *CN* will not able to positively verify the signature since the *CN* uses the *MN's* parameter as the basis for verification.

4. Once the *CN* has finished the decryption process, the *MN's HoA* and *CoA* are retrieved.

5. In the PKBU framework (as in Chapter 4), the *CN* will receive packets from the *MN1* via the *HA*. The hash value of *MN1's HoA* is contained in the packet. It is used by the *CN* to verify the *HoA* ownership. The *CN* will call a function to calculate the *HoA* hash value and then compare it with the *HoA* hash value it receives from the *HA*. If the function returns a TRUE value, then it is verified that the sender of the packet is actually *MN1*. The *CN* will proceed to update the *BC* entry in regard to the *MN*, otherwise the *CN* will reject the update. The verified packet from the *MN* is now accepted by the *CN* and the traffic between the *CN* and the *MN* can now begin with the *MN1*'s new *CoA*.

6. This is a classic example of defeating the DoS attacks, collaboratively. At the end of the simulation, it is evident that the attack was launched by a malicious entity and the legitimate nodes had worked together to detect the DoS, MITM, and session hijacking attacks. It is evident that attack detection can be done in a collaborative manner by using the PKBU protocol.

5.4 Summary

This chapter discusses the evaluation of the PKBU protocol. The protocol was first formally evaluated with respect to the security requirements, and its effectiveness in countering known security attacks, as described in Chapter 2. The evaluation results confirmed that the PKBU protocol satisfies the security requirements, and it is effective in defending *BUs* against the known attacks. This chapter also presents the PKBU protocol reachability design, ownership, and protection of *BU* message against the known attacks. Initially, the PKBU establishes connections between nodes based on Infrastructure-less algorithms, following which it verifies the *MN*'*s* reachability at the new IP address in a foreign network. The PKBU then determines whether the *MN* owns the *HoA* and the *CoA*. A simulation of PKBU model was constructed using OMNeT ++. Model validation was carried out using an OMNeT ++ debugger theoretical calculation.

The features of the proposed PKBU protocol can be summarized, as follows:

- It can protect the *BU* message against MITM attacks, session hijacking attacks, and DoS attacks, and make the connection between the *MN* and the *CN* secure.
- It can check the address ownership of the *MN*.
- It can check the reachability of the *CoA*.
- It can complement the security services to enhance data integrity, data confidentiality, and node authentication.

The following chapter concludes this thesis with recommendations for further work.

Chapter 6 : CONCLUSIONS AND FUTURE WORK

6.1 Introduction

This chapter presents the conclusion of the research. It summarises the objectives set, the achievements and contribution of the research. It also highlights related area for future research.

The findings from earlier researches have revealed the vulnerability of various mobile Internet protocols to security threats, and the need for authenticating the *HoA* and the *CoA*. In general, the results indicate that the location information in IPv6 must be authenticated. The *MNs* should authenticate themselves every time they move to foreign networks. In the absence of a proper authentication process, the packet flow from one node to another might be intercepted by a malicious node that could redirect the flow to its location or to another fake IP address. As a consequence, the service meant for the a legitimate receiver will be denied. These security issues occur because existing protocols do not have effective authentication methods to verify that the user is a valid user, or do not conceal the *HoA* and the *CoA* location data. These shortcomings will, inevitably, lead to malicious attacks. Hence, the objectives of this study include: proposing methods to authenticate the *HoA* and the *CoA*, to protect the *BU* message against MITM attacks, session hijacking attacks, and DoS attacks; and ensure a secure connection between the *MN* and the *CN*.

To achieve the objectives we proposed the PKBU protocol. An overview of the PKBU as well as the detailed design the PKBU were presented. This was followed by a detailed description of the PKBU framework, and an informal analysis of the protocol. The informal analysis was based on the requirements, and attacks such as MITM attacks, session hijacking attacks, and DoS attacks. The results show that using this protocol, the connection between the *CNs* and the *MNs* is secure, and the *BU* is protected against the MITM attacks, session hijacking attacks, and DoS attacks. This has been achieved through various methods such as: using asymmetric algorithm instead of symmetric algorithm; using the Infrastructure-less algorithms instead of the Infrastructure-based algorithms; and using suitable methods to check the IP address ownership, and reachability. We then discussed the outcomes of the security requirements analysis, and the evaluation of the protocol. The protocol was formally analysed and then verified using the Protocol Composition Logic (PCL) method. We simulated attack scenarios and compared existing protocols with the proposed PKBU protocol.

6.2 Achievement of the Objectives

We have achieved all the research objectives stated in section 1.5, as follows:

1. A method to assert address ownership:

   We proposed a method to assert the address ownership of the *MN* by creating 128-bit MIPv6 addresses based on the *MN*'s private key and computing one-way hash function. This provides a way of authenticating the *MN*s authority. It also allows the *CN* to validate the *MN* and ensure that it is not a malicious node by checking the address ownership. Our proposed method to assert the address ownership of the *MN* relies on the assistance of the *MN* home link. It requires the home link to enable the *CN* to verify the authenticity of the *MN's HoA* ownership. This is done by using a 128-bit IPv6 address, which consists of a 64-bit subnet prefix and a 64-bit identifier. The strong cryptographic binding between the *MN* and the *CN* provided by this method is implemented in the interface identifier. Our implementation also requires that the *MN* should have its own private key and public key. In this way, PKI is not needed to prove the binding ownership of the *MN*’s *HoA*. The *MN* retrieves its private and public key pair, and the user's ID is used to obtain the node's private and public key pair. In the proposed method, the *HoA* of the *MN* is verified using the secure interface ID in order to check the ownership of the *MN*’s IP address. It is based on the *MN's* private key and a valid subnet prefix.

2. A method to verify the reachability of the *MN*:

   In our method, the *MN’s* public key and the hash value of the *MN's HoA* is sent from the *MN* to the *CN* through the *HA*. The *MN* will directly send another message to the *CN*. In this message, the *MN* encrypts the *MN’s CoA* and the *HoA* using the *CN’s* public key. The encrypted message will be signed with the *MN’s* private key. When the *CN* receives these messages from the *MN*, it uses the *MN’s* public key to verify the signature, then it decrypts the message and obtain the *MN’s CoA* and *HoA*. Then, the *CN* calculates the hash value of the *HoA* and compares it with the hash value from the message received from the *MN* via the *HA*. If one of the signature verification, decryption or comparison of hash value receives a negative result, the message will be rejected. If the results of the checking operations and the validation process are positive, the *CN* is assured that the *MN's HoA* and the *CoA* are correct and that the *MN* can be reached at this the *CoA*.

3. A method that is protecting the *BU* messages against MITM, session hijacking, and DoS attacks, and to ensure a secure connection between the *MN* and the *CN*.

We proposed the PKBU protocol. PKBU was evaluated using different attack scenarios, and show the results that the protocol can be used to protect the *BU* message against security threats such as MITM attacks, DoS attacks, and session hijacking attacks.

6.3 Contributions

The main outcome of this research is the development of the PKBU protocol. The protocol incorporates the features and functionalities to meet all the research objectives as discussed above. Thus, the contributions from the research can be summarised as follows:

1. The PKBU protocol:
    - device's Interface ID and private key is a new decentralised authentication solution. It ensures a more secure communication for the delivery of the *HoA* and the *CoA*. There is no vulnerable point for an attack to take place, thus, making the authentication infrastructure secure;
    - provides reasonable assurance that the user's address actually belongs to a valid user, and has not been spoofed by an attacker;
    - has a authentication process to prove that the location of the communicating device is at the correct IP, hence, preventing any address spoofing.
2. Ensuring secure communication for the *BU* message, especially, between the *MNs* and the *CNs*.
3. A solid *MN* authentication method is provided in the security protocols. This can be expanded to include user verification to prevent device and identity theft.

4. Protecting the *BU* message against MITM attacks, session hijacking attacks, and DoS attacks by using various methods and strategies, as follows:
    - Using an Infrastructure-less algorithm instead of an Infrastructure-based algorithm, and using asymmetric cryptography instead of symmetric cryptography.
    - Development of a method to check the ownership of the *MN's HoA* because the *MN* must ensure the *CN* that:
        a. the *MN* owns the *HoA*
        b. the actual *BU* request belongs to the *MN*;
    - Development of a method to verify that the *MN* is reachable at the claimed *CoAs*.
5. In the informal and formal analyses, the PKBU protocol has successfully proven to provide protection for the *MN* against false binding update. It does not require high processing power, and it has a relatively low latency since it requires fewer messages, lower number of hash functions, and reduced Round-Trip Time.
6. The results of the evaluation of the performance of the PKBU protocol using INETMANET-2.0 framework in OMNeT ++ simulation, clearly show the PKBU protocol is protecting the *BU* against common attacks. In addition, the protocol has also been informally evaluated against standard security requirements, and against common attacks. Formal methods have also been used to verify the PKBU security features.

6.4 Future Work

In view of the rapid increase in the use of mobile devices for communication, there are many potential areas of mobile Internet communication security for future research. We would like to suggest the following area to be researched as a follow-up to the work reported in this thesis.

The main goal of PKBU is to protect the binding message update between the *MN* and the *CN* in MIPv6. Thus, improving the security of the connection between the *MN* and the *HA* or between *HA* and the *CN* should be further explored. Also the idea behind the PKBU can be used to make the secure connection and protect binding update message against attacks in Proxy Mobile IPv6.

In this research, we did not consider the handover latency, and the packet loss during a handover. The MIPv6 supports a handover that changes its point of attachment to the network when a *MN* moves to a new IP subnet. The basic handover procedure for the mobile IP consists of two components - L2 handover, and L3 handover. The term L2 handover denotes its support for roaming at the link layer, while the L3 handover occurs at the network layer. Usually, the L3 handover is independent of the L2 handover, although it must precede the L3 handover. The long handover latency in the MIPv6 degrades the perceived quality of service (QoS), especially, the real-time services. The handover latency can be eliminated and eventually, the packet loss can be overcome by using a buffering mechanism. This is a promising area for further research in the MIPv6 protocol.

References

Al-Kayali, A. K. M. (2004). Elliptic curve cryptography and smart cards. *SANS Institute, 17*.

Anari, Z., & Mehdizadeh, A. (2008). *Security enhancement of route optimization in Mobile IPv6 Networks.* Serdang, Selangor; Universiti Putra Malaysia.

Anoop, M. (2007). *Elliptic curve cryptography*. *An implementation guide*, online Implementation Tutorial, Tata Elxsi, India, Retrieved from http://www.tataelxsi.com/whitepapers/ECC_Tut_v1_0.pdf.

Arkko, J., Calhoun, P., Guttman, E., Nelson, D., & Wolff, B. (2000). AAA solutions. *The Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-ietf-aaa-solutions-01.

Arkko, J., Devarapalli, V., & Dupont, F. (2004). RFC 3776: Using IPsec to protect mobile IPv6 signaling between mobile nodes and home agents. *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc3776.txt.

Arkko, J., Haddad, W., & Vogt, C. (2007). RFC 4866: Enhanced route optimization for mobile IPv6. *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc4866.txt.

Arkko, J., Nikander, P., & Montenegro, G. (2002). Selection of MIPv6 security level using a hashed address. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-arkko-mipv6-select-hash-00.

Arkko, J., Vogt, C., & Henderson, T. (2011). RFC 5206: Host mobility with the host identity protocol. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-ietf-hip-rfc5206-bis-03

Aura T. *Mobile IPv6 Security*. Security Protocols. Paper presented at the 10th International Workshop, Cambridge, UK, April 17-19, 2002. Revised Papers: Springer; 2004. pp. 215-234.

Aura, T. (2005). *Cryptographically generated addresses (CGA)*. Paper presented at the 6th Information Security Conference (ISC'03), vol 2851, pp. 29-43.

Aura, T., & Arkko, J. (2002). MIPv6 BU attacks and defenses. *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/draft-aura-mipv6-bu-attacks-01.

Aura, T., Roe, M., & Arkko, J. (2002). *Security of Internet location management.* Paper presented at the 18th Annual Computer Security Applications Conference, Washington, DC, USA.

Belding, R., Elizabeth, M., Sun, Y., & Perkins, C. E. (2001). Global connectivity for IPv4 mobile ad hoc networks. *Internet Engineering Task Force (IETF)*, Retrieved from draft-royer-manet-globalv4-00.txt.

Bidan, C., & de Sousa Júnior, R. T. (2013). Trust-based security for the OLSR routing protocol. Computer Communications, 36,Issues 10–11, June 2013, pp 1159-1171.

Boneh, D., & Franklin, M. (2003). Identity-based encryption from the Weil pairing. *Siam Journal on Computing, 32*(3), pp. 586-615.

Borralleras, C., Lucas, S., Navarro-Marset, R., Carbonell, E., & Rubio, A. (2009). Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. *Springer Berlin / Heidelberg,* Vol 5663/2009, pp. 294-305.

Bos, J., Özen, O., & Hubaux, J.-P. (2009). *Analysis and optimization of cryptographically generated addresses*. Paper presented at the 12th International Conference on Information Security, pp.17-32.

Boyko, V., MacKenzie, P., & Patel, S. (2000). *Provably secure password-authenticated key exchange using Diffie-Hellman.* Paper presented at the 19th International Conference on Theory and Application of Cryptographic Techniques, Berlin, Heidelberg, pp. 156-171.

Cao, Z., Deng, H., Ma, Y., & Hu, P. (2007). *Integrating identity based cryptography with cryptographically generated addresses in mobile IPv6*. Paper presented at the *ICCSA'07* International Conference on Computational Science and Its Applications*, 2,* pp. 514-525.

Conta, A., & Deering, S. (1998). RFC 2473: Generic packet tunneling in IPv6 specification. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc2473.txt.

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc3280.txt.

Cremers, C. (2008). *On the protocol composition logic PCL.* Paper presented at the ACM Symposium on Information, Computer and Communications Security, New York, NY, USA, pp. 66-76.

Datta, A., Derek, A., Mitchell, J. C., & Pavlovic, D. (2003a). *A derivation system for security protocols and its logical formalization.* Paper presented at the 16th IEEE Computer Security Foundations Workshop, Pacific Grove, CA, USA, pp. 109 -125.

Datta, A., Derek, A., Mitchell, J. C., & Pavlovic, D. (2003b). *Secure protocol composition.* Paper presented at the ACM Workshop on Formal Methods in Security Engineering, New York, NY, USA, pp. 11–23.

Datta, A., Derek, A., Mitchell, J. C., & Pavlovic, D. (2004). *Abstraction and refinement in protocol derivation.* Paper presented at the 17th IEEE Computer Security Foundations Workshop, Pacific Grove, California, pp. 30–45.

Datta, A., Derek, A., Mitchell, J. C., & Pavlovic, D. (2005). A derivation system and compositional logic for security protocols. *Journal of Computer Security,* 13(3), pp. 423-482.

Datta, A., Derek, A., Mitchell, J. C., & Roy, A. (2007). Protocol composition logic (PCL). *Electronic Notes in Theoretical Computer Science*, 172 , pp. 311-358.

Datta, A., Derek, A., Mitchell, J. C., & Warinschi, B. (2006). *Computationally sound compositional logic for key exchange protocols.* Paper presented at the 19th IEEE Computer Security Foundations Workshop Venice, pp. 321-334.

De Dormale, G. M., Bulens, P., & Quisquater, J.-J. (2004). *An improved Montgomery modular inversion targeted for efficient implementation on FPGA.* Paper presented at the IEEE International Conference on Field-Programmable Technology, Brisbane, Australia , pp. 441 - 444.

Debaert, C., & Gilbert, H. (2002). *The RIPEMD L and RIPEMD R improved variants of MD4 are not collision free*. Paper presented at the Fast Software Encryption, pp. 52-65.

Deering, S., & Hinden, R. (2006). RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc2460.txt.

Deng, R. H., Zhou, J., & Bao, F. (2002). *Defending against redirect attacks in mobile IP*. Paper presented at the 9th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 59 - 67.

Diffie, W., & Hellman, M. (1976). New directions in cryptography, *IEEE Transactions on Information Theory, 22*(6), pp. 644-654.

Doraswamy, N., & Harkins, D. (2003). *IPSec: the new security standard for the Internet, intranets, and virtual private networks*, NJ, United States: Prentice Hall.

Dupont, F., & Combes, J. (2006). RFC 4651: Care-of Address Test for MIPv6 using a State Cookie. *Network Working Group*, Retrieved from http://www.ietf.org/rfc/rfc4651.txt.

Durgin, N., Mitchell, J., & Pavlovic, D. (2001). *A compositional logic for protocol correctness*. Paper presented at the 14th IEEE Computer Security Foundations Workshop, Cape Breton, Novia Scotia, Canada, pp. 241—255.

Ehmke, M., Forsgren, H., Grahn, K., Karlsson, J., Karvi, T., & Pulkkis, G. (2009). Securing Control Signaling in Mobile IPv6 with Identity-Based Encryption. *Issues in Informing Science and Information Technology*, 6, pp. 645-667.

Elshakankiry, O. (2010). *Securing home and correspondent registrations in mobile IPv6 networks.* Manchester, England, University of Manchester.

Eronen, P., & Arkko, J. (2006). *Authentication components: Engineering experiences and guidelines.* Paper presented at the 12th International Conference on Security, LNCS 3957, pp. 68-77.

Feng, B., Deng, R., Qiu, Y., & Zhou, J. (2005). Certificate-Based binding update protocol (CBU). *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/id/draft-qiu-mip6-certificated-binding-update-03.txt.

Georgiades, A. (2011). *A security protocol for authentication of binding updates in Mobile IPv6.* London: Middlesex University.

Georgiades, A., Luo, Y., Lasebae, A., & Comley, R. (2006). *Dual identity return routability for the security of mobile Ipv6 binding updates within the distributed authentication protocol.* Paper presented at the 6th WSEAS International Conference on Applied Informatics and Communications, Elounda, Agios Nikolas, Crete Island, Greece, pp. 406-411.

Goldwasser, S., Micali, S., & Rivest, R. L. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing,* 17(2), pp. 281-308.

Gunderson, S. H. (2008). Global IPv6 statistics-measuring the current state of IPv6 for ordinary users. *Google White Paper,*1-20, Retrieved from http://meetings.ripe.net/ripe-57/presentations/Colitti-Global_IPv6_statistics_-_Measuring_the_current_state_of_IPv6_for_ordinary_users_.7gzD.pdf

Haddad, W., Dupont, F., Madour, L., & Arkko, J. (2005). Applying Cryptographically Generated Addresses to Optimize MIPv6 (CGA-OMIPv6). *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-haddad-mip6-cga-omipv6-03.

Haddad, W., & Krishnan, S. (2004). Optimizing Mobile IPv6 (OMIPv6). *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-haddad-mipv6-omipv6-01.

Hankerson, D. R., Vanstone, S. A., & Menezes, A. J. (2004). *Guide to elliptic curve cryptography*, New York: Springer-Verlag, 19, pp. 173-193.

Harel, D., Kozen, D., & Tiuryn, J. (2001). Dynamic logic, ACM SIGACT News, 32, pp. 9-17.

Harkins, D., & Carrel, D. (1998). RFC 2409: The Internet Key Exchange (IKE). *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/rfc2409.

Hasan, S. S., & Hassan, R. (2013). Enhancement of Return Routability Mechanism for Optimized-NEMO Using Correspondent Firewall. *ETRI Journal,35*(1).

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, *12*(10), pp. 576-580.

Huang, J. (2007). *FPGA implementations of elliptic curve cryptography and Tate pairing over binary field.* University of North Texas.

Jesper, J. (2006). The Most Misunderstood Windows Security Setting of All Time. *TechNet Magazine,* Retrieved from http://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx.

Johnson, D., Perkins, C., & Arkko, J. (1996). RFC 2002: IP Mobility Support. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc2002.txt.

Johnson, D., Perkins, C., & Arkko, J. (2004). RFC 3775: Mobility support in IPv6. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc3775.txt.

Kavitha, D., & Murthy, K. S. (2010, March). Security Analysis of Binding Update Protocols in Route Optimization of MIPv6. *International Conference on in Recent Trends in Information*, Telecommunication and Computing (ITC), 2010 IEEE, pp. 44-49.

Kavitha, D., Murthy, K. E., Sathyanarayana, B., & Reddy, V. R. (2010). An Efficient Hierarchical Certificate Based Binding Update Protocol for Route Optimization in Mobile IPv6. Global Journal of Computer Science and Technology, 10(2).

Kempf, J., Gentry, C., & Silverberg, A. (2002). Securing IPv6 neighbor discovery using address based keys (ABKs). *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-kempf-secure-nd-01.

Kent, S. (2005a). RFC 4302: IP authentication header. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/rfc4302.

Kent, S. (2005b). RFC 4303: IP Encapsulating Security Payload (ESP). *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc4303.txt.

Kohl, J., & Neuman, C. (1993). RFC 1510: The Kerberos network authentication service (v5), *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc1510.txt.

Koo, J.-D., & Lee, D.-C. (2007). Extended Ticket-Based Binding Update (ETBU) Protocol for Mobile IPv6 (MIPv6) Networks. *IEICE Transactions on Communications*, *90*(4), pp. 777-787.

Koo, J., Koo, J., & Lee, D. (2006). *A new authentication scheme of binding update protocol on handover in mobile IPv6 networks.* Paper presented at the International Conference on Emerging Directions in Embedded and Ubiquitous Computing, Seoul, Korea, pp. 972-978 .

Krawczyk, H., Canetti, R., & Bellare, M. (1997). RFC 2104: HMAC: Keyed-hashing for message authentication. *Internet Engineering Task Force (IETF)*, Retrieved from http://www.rfc-editor.org/rfc/rfc2104.txt.

Kroeselberg, D., Patil, B., Tschofenig, H., & Korhonen, J. (2012). RFC 6618: Mobile IPv6 Security Framework Using Transport Layer Security for Communication between the Mobile Node and Home Agent. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/rfc6618.

Kumar, S. S. (Ed.) (2006). ruhr-universität bochum, Universitätsbibliothek.

Le, F., & Faccin, S. M. (2001). Dynamic Diffie Hellman based Key Distribution for Mobile IPv6. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/draft-le-mobileip-dh-00.

Lee, C. C., Li, C. T., & Hsu, C. W. (2013). A three-party password-based authenticated key exchange protocol with user anonymity using extended chaotic maps. *Nonlinear Dynamics*, 1-8.

Lowe, G. (1997). *Casper: A compiler for the analysis of security protocols. Paper presented at the 10th Computer Security Foundations Workshop*, Rockport, MA, pp. 18 - 30.

Mankin, A., Bradner, S., & Schiller, J. (2003). A Framework for Purpose Built Keys (PBK). *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/draft-bradner-pbk-frame-06.

Mankin, A., Patil, B., Harkins, D., Nordmark, E., Nikander, P., Roberts, P., et al. (2001). Threat models introduced by Mobile IPv6 and requirements for security in mobile IPv6. *Internet Engineering Task Force (IETF),* http://tools.ietf.org/html/draft-ietf-mobileip-mipv6-scrty-reqts-02.

Menezes, A. J. (1993). *Elliptic curve public key cryptosystems,* Berlin: Springer.

Metz, C. (1999). AAA protocols: authentication, authorization, and accounting for the Internet. *IEEE Internet Computing,* 3(6), pp.75-79.

Miller, V. (1986). *Use of elliptic curves in cryptography.* Lecture notes in computer sciences; 218 on Advances in cryptology CRYPTO 85, New York: Springer-Verlag, pp. 417-426 .

Modares, H. (2009). *A scalar multiplication in elliptic curve cryptography with binary polynomial operations in Galois Field.* Kuala Lumpur-Malaysia: University of Malaya.

Modares, H., Salem, Y., Salleh, R., & Shahgoli, M. T. (2010). A Bit-Serial Multiplier Architecture for Finite Fields Over Galois Fields. *Journal of Computer Science, 6*(11), pp. 1237-1246.

Modares, H., Moravejosharieh, A., Lloret, J., & Salleh, R. (2013). A Survey of Secure Protocols in Mobile IPv6. Journal of Network and Computer Applications, http://dx.doi.org/10.1016/j.jnca.2013.07.013.

Moon, J. S., Lee, S. H., Lee, I. Y., & Byeon, S. G. (2010). *Authentication Protocol Using Authorization Ticket in Mobile Network Service Environment.* Paper presented at the 3rd International Conference on Human-Centric Computing (HumanCom) Cebu, Philippines pp. 1-6.

Nandi, M. (2005). Designs of efficient secure large hash values, Cryptology ePrint report, Retrieved from http://eprint.iacr.org/2004/296.

Needham, R. M., & Schroeder, M. D. (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), pp. 993-999.

Neuman, C., Yu, T., Hartman, S., & Raeburn, K. (2005). RFC 4120: The Kerberos network authentication service (V5). *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc4120.txt.

Nikander, P., Arkko, J., Aura, T., & Montenegro, G. (2003). *Mobile IP version 6 (MIPv6) route optimization security design.* Paper presented at the 58th IEEE Vehicular Technology Conference Fall 2003, Orlando, 2004 - 2008.

Nikander, P., Arkko, J., Aura, T., Montenegro, G., & Nordmark, E. (2005). RFC 4225: Mobile IP version 6 route optimization security design background. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc4225.txt.

O'shea, G., & Roe, M. (2001). Child-proof authentication for MIPv6 (CAM). *ACM SIGCOMM Computer Communication Review,* 31(2), 4-8.

Perkins, C. (1997). Design Principles and practices, Boston, MA: Addison-Wesley.

Perkins, C. (2002). RFC 3344: IP mobility support for IPv4. *Internet Engineering Task Force (IETF)*, Retrieved from http://www.ietf.org/rfc/rfc3344.txt.

Perkins, C. (2006). RFC 4449: Securing Mobile IPv6 route optimization using a static shared key. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/rfc4449.

Perkins, C. (2010). RFC 5944: IP Mobility Support for IPv4. *Internet Engineering Task Force (IETF)*, http://tools.ietf.org/html/rfc5944.

Perkins, C., Johnson, D., & Arkko, J. (2011). RFC 6275: Mobility Support in IPv6. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/rfc6275.

Perlman, R. (1999). An overview of PKI trust models., *IEEE Network, 13*(6), pp. 38-43.

Rathi, S., & Thanuskodi, K. (2009). A Secure and Fault tolerant framework for Mobile IPv6 based networks. *International Journal of Computer Science and Information Security, IJCSIS, 5*, pp. 46-55.

Ren, K., Lou, W., Zeng, K., Bao, F., Zhou, J., & Deng, R. H. (2006). Routing optimization security in mobile IPv6. *Computer Networks,* 50(13), pp. 2401-2419.

Riedel, I. (2003). Security in ad-hoc networks: Protocols and elliptic curve cryptography on an embedded platform. Bochum, Germany: Ruhr-Universitaet.

Rigney, C., Willens, S., Livingston, R., Merit, A., & Simpson, W. (2000). RFC 2865: Remote authentication dial in user service (RADIUS). *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/rfc2865.

Rivest, R. (1992). RFC 1321: The MD5 Message-Digest Algorithm. *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc1321.txt.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21*(2), pp. 120-126.

Robert, S. (2003). Introduction à Mobile IP. *Institute for Information and Communication Technologies (IICT).*

Roe, M., Aura, T., O’Shea, G., & Arkko, J. (2002). Authentication of Mobile IPv6 Binding Updates and Acknowledgments. *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/draft-roe-mobileip-updateauth-02.

Rossi, A., Pierre, S., & Krishnan, S. (2012). Secure Route Optimization for MIPv6 Using Enhanced CGA and DNSSEC, *Mobile Computing and Networking Laboratory (LARIM)* , Ecole Polytechnique de Montreal.

Roy, A., Datta, A., Derek, A., Mitchell, J. C., & Seifert, J. P. (2006). *Secrecy analysis in protocol composition logic.* Paper presented at the 11th Asian computing science conference on Advances in computer science: secure software and related issues, INRIA Institut Natl de Recherche en Info et en Automatique, Keio University Keio University, Japan, pp. 197-213.

Schneier, B. (1996). *Applied cryptography: protocols, algorithms, and source code in C,* New York, NY: John Wiley & Sons, Inc.

Seo, K., & Kent, S. (2005). RFC 4301: Security architecture for the internet protocol. *Internet Engineering Task Force (IETF),* Retrieved from http://tools.ietf.org/html/rfc4301.

Shoup, V. (2000). *Using hash functions as a hedge against chosen ciphertext attack.* Paper presented at the 19th International Conference on Theory and Application of Cryptographic Techniques, Bruges, Belgium,1807, pp. 275-288.

Soliman, H. (2004). Securing Mobile IPv6 Signaling. In G. L. GmbH (Ed.), *Mobile IPv6: Mobility in a Wireless Internet* (pp. 149): Boston, MA: Addison-Wesley.

Stoneburner, G. (2001). *Underlying Technical Models for Information Technology Security*: Gaithersburg, MD: U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology.

Sudanthi, S. (2003). Mobile IPv6. *Information Security Reading Room,* 1, pp. 1-14.

Suresh, V. B., Antonioli, D., & Burleson, W. P. (2013, June). On-chip lightweight implementation of reduced NIST randomness test suite. *IEEE International Symposium on InHardware-Oriented Security and Trust (HOST),* 2013 (pp. 93-98).

Tanenbaum, A. S., & Van Steen, M. (2003). Distributed systems principles and paradigms. *Network, 3*, p. 26.

Tung, B. (2005). The Moron's Guide to Kerberos. *That's Version 1.2.2 of the Guide*, p. 11.

Tuomas, A. (2005). RFC 3972: Cryptographieally generated addresses (CGA). *Internet Engineering Task Force (IETF),* Retrieved from http://www.ietf.org/rfc/rfc3972.txt.

Varga, A. (2010). OMNeT++ *Modeling and tools for network simulation,* New York: Springer, pp.35-59.

Vogt, C. (2005). *Credit-Based Authorization for Concurrent IP-Address Tests.* in Proceedings of the IST Mobile and Wireless Communications Summit.

Vogt, C., Bless, R., Doll, M., & Kuefner, T. (2005). *Early binding updates for mobile IPv6.* Paper presented at the Wireless Communications and Networking Conference, IEEE, pp. 1440 - 1445.

Wang, X., Yin, Y., & Yu, H. (2005). *Finding collisions in the full SHA-1.* Paper presented at the 25th Annual International Cryptology Conference, Barbara, California, USA, pp. 17-36.

Werapun, W., & Unakul, A. (2004). *Secure Mobile IPv6 Binding Updates with Identiy-based Signature.* Paper presented at the International Conference on Electronics Packaging.

Xiao, H. (2003). *Trust management for mobile IPv6 binding update.* In Proceedings of Security and Management, ACM, pp. 469-474.

Yan G., Wen D., Olariu S., and Weigle M. C (2012), Security challenges in vehicular cloud computing. *IEEE Transactions on Intelligent Transportation Systems*, 14(1), pp.284-294.

Yeh, L. Y., Yang, C. C., Chang, J. G., & Tsai, Y. L. (2013). A secure and efficient batch binding update scheme for route optimization of nested NEtwork MObility (NEMO) in VANETs. Journal of Network and Computer Applications,36(1), 284-292.

Ying, Q., & Feng, B. (2010). *Authenticated binding update in Mobile IPv6 networks.* Paper presented at the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, pp. 307 - 311.

Yoon, H.-S., Kim, R.-H., Hong, S.-B., & Youm, H.-Y. (2006). *PAK-based Binding Update Method for Mobile IPv6 route optimization.* Paper presented at the International Conference on Hybrid Information Technology, 2006. ICHIT'06., Cheju Island, pp. 617 - 623.

Yu, T., & Astrand, L. (2012). RFC 6649: Deprecate DES, RC4-HMAC-EXP, and Other Weak Cryptographic Algorithms in Kerberos. *Internet Engineering Task Force (IETF)*, Retrieved from http://tools.ietf.org/html/rfc6649.

Zuleger, O. (2005). Mobile internet protocol v6. Retrieved from http://www.hznet.de/ipv6/mipv6-intro.pdf

APPENDIX A

Cryptography encrypts and decrypts data with mathematics, and it facilitates sending or storing sensitive information over insecure networks. Thus, cryptanalysis is the science of breaching secure communication.

Cryptography is meant to facilitate message exchanges between two people, so they are not understood by others (Wang, Yin, & Yu, 2005). In other words, people A and B should be able to communicate securely using encryption over an insecure channel, and any eavesdroppers should not be able to read or alter the clear text (Figure A.1).

In such a situation, the person sending, receiving or controlling data is called an entity. If the entity sends legitimate data, it is called a sender. An entity that receives data is called a receiver, who may in fact be an entity trying to destroy data security services meant for senders and receivers. The bad guy disguised as a sender or a receiver can be called an adversary, attacker, enemy, eavesdropper, opponent, or intruder (Jesper, 2006).

Cryptographic strength is measured by observing how much time and resources are used to recover plain text. Then, when encrypting the plaintext, a cryptographic algorithm associates itself with a private key to resolve the ciphertext which uses unique values each time. Encrypted data security is relative cryptographic algorithm strength as well as key confidentiality (Schneier, 1996).

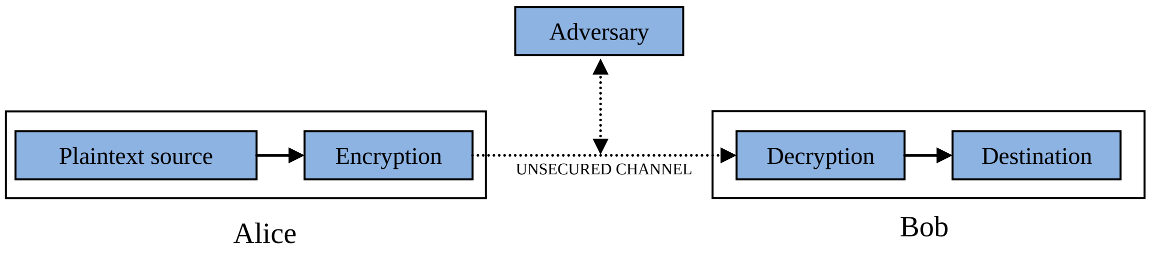


Figure A.1: Two-part communication

- **Symmetric**

Symmetric-key is a type of cryptography in which two parties wishing to communicate use a secret shared key valid for both encryption and decryption. Here, the parties need to trust each other and not reveal the secret-key to anybody. Symmetric-key is advantageous especially when large amounts of data get encrypted, but can be problematic when key control is for numerous users (Riedel 2003; Stoneburner, 2001). Figure A.2 illustrates a symmetric-key example (Al-Kayali, 2004; Hankerson, Vanstone, & Menezes, 2004).

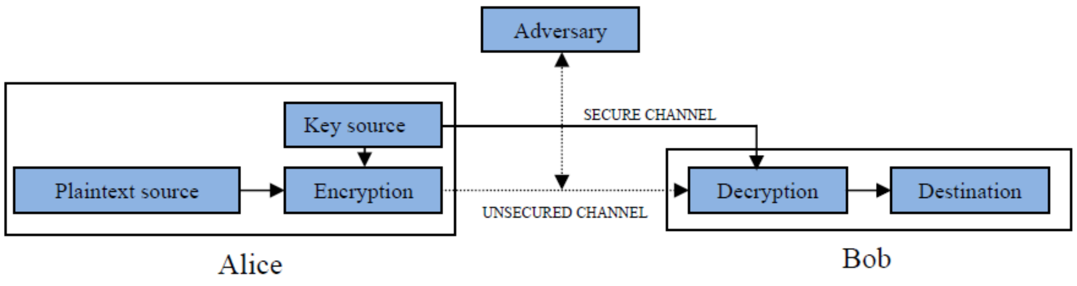


Figure A.2: Symmetric-Key

- **Asymmetric**

Public key cryptography (PKC) is different than a public key, and was put forth in 1976 by Diffie and Hellman. These keys do not encrypt or decrypt messages, but help come up with a single shared secret key.

PKC involves two types of keys, namely, public and private keys. For example, user *A* wishes to send a message to *B*. *A* uses *B'*s public key to encrypt a message, while *A*'s private key is used to sign the message. The receiver *B* decrypts the message with his private key, then uses *A*'s public key to authenticate the signature. Encryption key size is set according to the preferred level of security, because size determines how challenging it will be to recover the encrypted data computationally without a secret key. An example of a public key example is given in Figure A.3, where the key pair *(e,d)* is chosen by *B* with *'e'* as the public key sent by *B* to *A* over any channel and the private key *'d'* is kept. *A* encrypts the message to send *B* using *B'*s public-key, then *B* decrypts the ciphertext *'c'* using *'d'* (Modares, 2009).

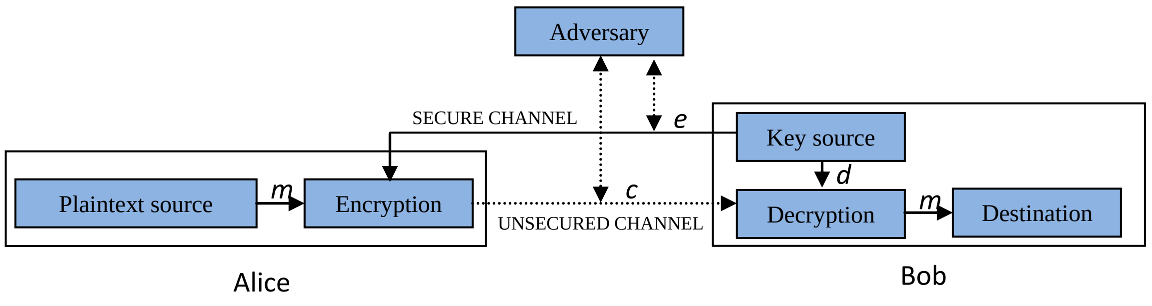


Figure A.3: Public Key

- **Elliptic Curve Cryptography**

Neal Koblitz and Victor Miller first proposed Elliptic Curve Cryptography (ECC) in 1985. ECC systems offer public keys, as the same as provided by RSA systems. Security depends on the Elliptic Curve Discrete Logarithm Problem (ECDLP) which can be solved by algorithms with fully exponential time. However, the integer factorization problem would be solved by sub-exponential-time algorithms (Hankerson et al., 2004). This way, ECC can provide comparable security. Different public key cryptography methods provide this nowadays, but with smaller key sizes and memory requirements (Table A.1). For instance, a 1024-bit RSA key normally offers the same level of security as a 160-bit elliptic curve key. Smaller key sizes have various advantages like storage, speed and efficient power and bandwidth usage, since storing keys require less space and faster arithmetic processes. In cases when public-key cryptography is used in mobile devices or RFID, for instance, the advantages are essential, which is why ECC is the cryptographic system of choice for this thesis.

Table A.1: NIST Recommended Key Size

Symmetric-key	ECC	RSA	Comment
64 bit	128 bit	700 bit	Short period security
80 bit	160 bit	1024 bit	Medium period Security
128 bit	256 bit	2048	Long period Security

Basically, to achieve backward compatibility and security as in smaller, low-end controlled devices, existing protocols may use ECC based algorithms.

Elliptic curves can offer a group structure of points on the curves, determined over a finite field for implementing cryptographic schemes. The curve elements, or points, act as group identity parts. To perform group operation, some finite field based arithmetic operations can be done, as presented later (Kumar, 2006).

Calculating points in the prime field elliptic curve can be done in different ways, a direct method being (Anoop, 2007):

$$y^2=x^3+ax+b \text{ where } 4a^3+27b^2\neq 0$$

If the *‘a’* and *‘b’* values are changed, various elliptic curves result. Since a fundamental operation of ECC is multiplying points, to calculate the public key in elliptic curve cryptography, the private key is multiplied by the curve’s generator point *‘G’*, where the private key is a random number in the interval [1, n-1], and *‘n’* is the curve’s order (Anoop, 2007).

Depending on how complex and difficult the Elliptic Curve Discrete Logarithm Problem is, ECC is accordingly secure. If *‘P’* and *‘Q’* are points on the curve, then $kP=Q$; where a scalar *‘k’* is multiplied by a point *‘P’* to get another point, *‘Q’* on the curve (Modares, 2009).

- **Diffie-Hellman Key Exchange**

Symmetric-key cryptosystems are faster than public-key, and are normally used for encryption and decryption. They require the entities' mutual agreement (i.e. Diffie-Hellman key exchange) on a secret key prior to beginning cryptography. The Elliptic Curve Diffie-Hellman (ECDH) key exchange method is illustrated as follows (Huang, 2007).

Figure A.4 shows Alice and Bob and two parties who are communicating and agree on an elliptic curve *'E'* over finite field Fq as well as on a point $P\epsilon E(F_q)$. Alice chooses a secret integer 'k', computes $A = g^k \ mod \ p$ and sends it to Bob. Bob does the same, chooses a secret integer$l$, computes $B = g^l \ mod \ p$, and sends it to Alice. She computes $K_{ab} = B^k \ mod \ p$. On his end, Bob computes $K_{ab} = A^l \ mod \ p$. Finally, the result is the same for both parts $(K_{ab}=K_{ba})$ (Modares, 2009).

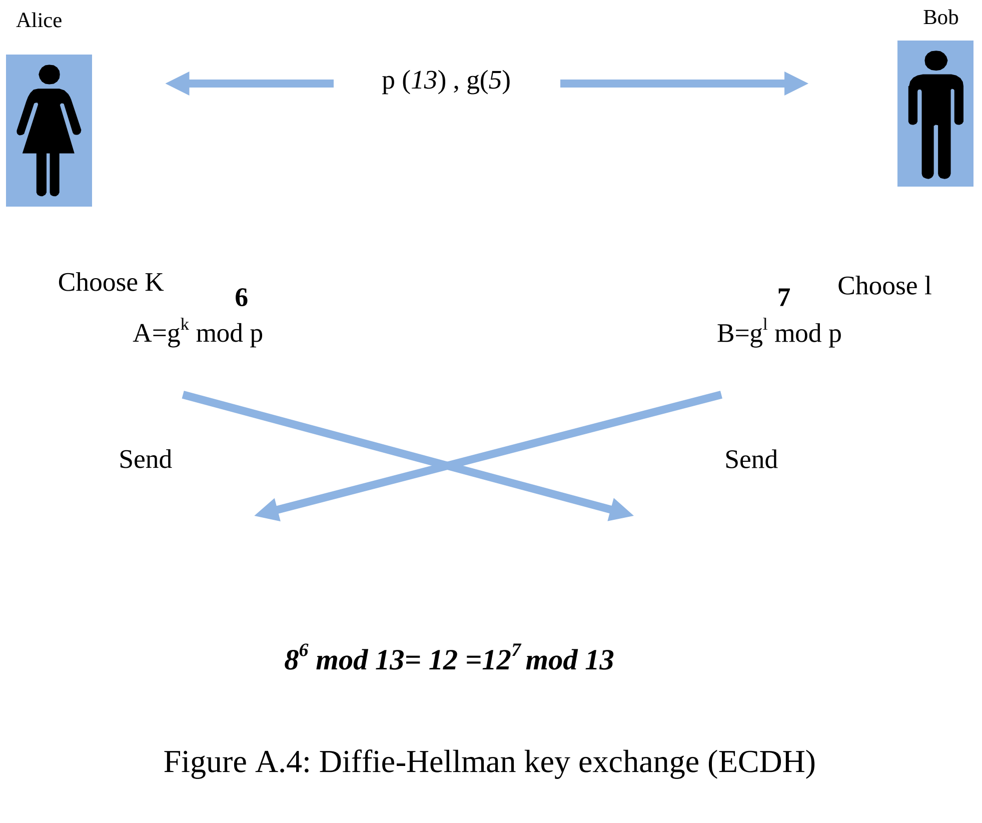


Figure A.4: Diffie-Hellman key exchange (ECDH)

- **Digital Signatures**

Digital Signature Algorithms (DSAs) help authenticate electronic messages or applications. Signatures have individual properties that only one single individual with a private key can come up with. However, anybody getting the message can validate a signature. For instance, if a sender, *A*, sends a message to a receiver, *B*, the message should be signed by *A* with a private key for message authentication, and *A*'s public key can verify the signature. As such, if *B* knows *A*'s public key, *B* can verify the signature (Huang, 2007).

A variation of DSA is Elliptic Curve Digital Signature Algorithm (ECDSA) which works on elliptic curves. In DSA, if user *A* sends *B* a signed message, both *A* and *B* must agree on EC domain parameters. A possesses a private key *'dA'* and a public key $QA = dA * G$, where *'dA'* is a random number less than *'n'* (*'n'* is the curve order) and *'G'* is the generator point (Huang, 2007).

- **ECDSA Key Generation**

ECDSA first performs key generation as:

1. An elliptic curve *'E'* defined over $GF(2^m)$ is chosen. The number of points on 'E' should be divisible by a large prime number *'n'*.

2. A point $P = (x,y) \in GF(2^m)$ of order 'n' is selected.

3. A random integer '*d*' in interval [1, n-1] is chosen, and acts as the private key.

4. $Q=dP$ is calculated.

5. The public key is *(E, P, n, Q)*.

- **ECDSA signature Generation**

The sender signs a message, m, as:

1. A random integer number k in the range [1, n-1] is calculated.

2. Compute $(x_1, y_1) = kP = k(x,y)$, and set $r = x_1$ mod n. If *'r'* equals to zero, then go back to step 1. This means the private key is r=0, and 'd' is not part of the signing equation $[s = k^{-1}(h(m) + dr)modn]$.

3. Calculate k-1mod n.

4. Then computes $S = k^{-1}(h(m) + dr)modn$, where $h(m)$ is the hash value obtained from a suitable hash algorithm (i.e. Secure Hash Algorithm, SHA-1).

The protocol remains secure so long as the private key stays secret. If the key gets lost or altered, it could be denied. Stamps or perhaps some central tracking could help prevent this (Modares et al., 2010).

- **Hash Function**

Hash is a function that can be easily calculated from the set of all finite binary strings, $\{0, 1\}^*$ to a fixed length binary string set, $\{0, 1\}^n$. Hash functions are well known in digital signature systems (Debaert & Gilbert, 2002; Goldwasser, Micali, & Rivest, 1988), public key encryption schemes (Shoup, 2000), message authentication codes (Krawczyk, Canetti, & Bellare, 1997), etc. Collision or image resistant hash functions are used to create such systems. Naturally, a collision resistant hash function is a hash function, $H(.)$, in which two different inputs (collision pair) are difficult to find $(X \neq Y)$, so $H(X) = H(Y)$. If an image resistant hash function receives a random image, it inverts it so it is not easy to find (Nandi, 2005).

So far, 14 hash algorithm kinds are in use (Wang et al., 2005). It is argued that LM cryptographic functions (LanManager), MD5, SHA-0/SHA-1, and SHA-2. SHA-0 and SHA-1 are very similar and often taken as a single algorithm.

- **Finite Fields**

All elliptic curve operations mentioned earlier are based on real numbers. However, operations over the real numbers are inaccurate and slow, whereas cryptographic operations need to be accurate and fast. Therefore, the curve cryptography can be defined over finite fields to operate EC efficiently and accurately. A finite field is a set of a finite number of elements. Figure A.5 describes elliptic curve operation hierarchy. It consists of cryptographic protocols, Elliptic Curve point operations and basic Galois Field operations.

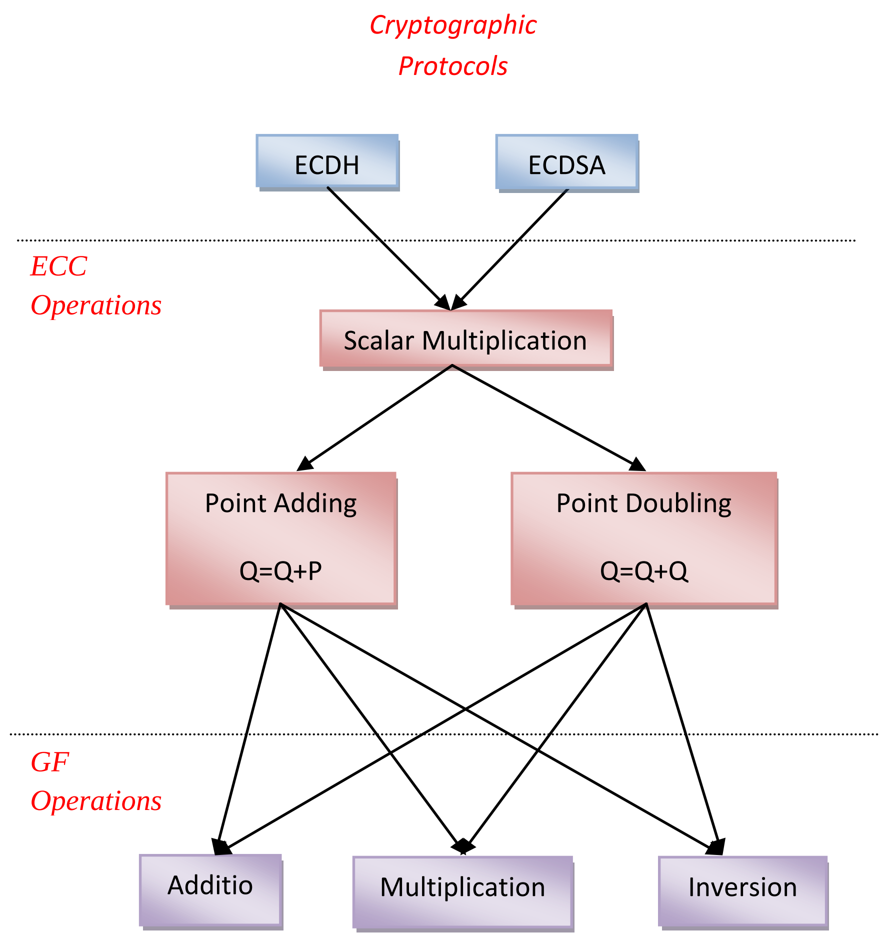


Figure A.5: Hierarchy of Elliptic Curve Operations

ECC hierarchy consists of three levels. The highest level is cryptography protocols. ECDH and ECDSA are two categories of cryptography protocol that are used to provide services like Public-key, signature and key agreement. Elliptic Curve point operation level consists of Scalar Multiplication, which utilizes point adding and point doubling. In the last level, the operations are GF addition, GF multiplication, GF squaring and GF inversion.

The rest of this part explains EC operations on finite fields. The operations are defined on affine coordinate system. Affine coordinate system is a normal coordinate system that represents each point by the vector (x, y).

- **EC on Prime field F_p**

The equation to retrieve all the possible points on the curve over the prime field is $y^2 mod\, p = x^3 + ax^2 + b\, mod\, p$, where $4a^3 + 27b^2\ mod\ p \neq 0$. The elements of the finite field are integers between 0 and $p - 1$ thus all the operations such as addition, subtraction, multiplication, division include integers between 0 and $p - 1$. The prime number *'p'* is the finitely large number of points on the EC in order to make the cryptosystem secure (Tata, 2007).

- **Point Addition**

Let *'J'* and *'K'* be the two points on the curve where $J = (x_J, y_J)$ and $K = (x_K, y_K)$ and $L=K+J$ where $L=(x_L,y_L)$ and *'s'* is the incline of the line through *'J'* and *'K'* then:

$S = (y_J - y_K)/(x_J - x_K)\ mod\ p,$

$x_L = s^2 - x_J - x_K\ mod\ p$

$y_L = -y_J + s\ (x_J - x_L)\ mod\ p$

So if $K=-J$ for example $K = (x_J, -y_J\ mod\ p)$ then $J+K=O$, where *'O'* is the point at infinity. If $K=J$, then $J+K=2J$ needs to use point doubling equation too $J + K = K + J$.

- **Point Subtraction**

If *'J'* and *'K'* are two points on the curve where $J = (x_J, y_J)$ and $K = (x_K, y_K)$ Then:

$J - K = J + (-K)$ *where* $-K = (x_k, -y_k\ mod\ p)$

In certain implementation of point multiplication such as NAF, point subtraction is used (Riedel, 2003; Stoneburner, 2001b).

- **Point Doubling**

Assume $J=(x_J, y_J)$ and $y_J \neq 0$ so the calculation of $L= 2J$ where $L = (x_L, y_L)$ is:

$s = (3x_J^2 + a) / (2y_J) mod\ p$ (*'S'* is the tangent at point *'J'* and *'a'* is one of the parameters selected with the EC)

$x_L = s^2 – 2x_J\ mod\ p$

$y_L = -y_J + s(x_J – x_L)\ mod\ p$

And *'$\mathcal{O}$'* is the point at infinity if $y_J=0$ then $2J= \mathcal{O}$.

- **Elliptic Curve (EC) on Binary field F_{2^m}**

The equation to retrieve all the possible points on the curve over the binary field is F_{2^m} is $y^2 + xy = x^3 + ax^2 + b$**,** where $b \neq 0$. The elements of the finite field are integers of length at most *'m'* bits, which they can be considered as a binary polynomial of degree $m – 1$. On the other hand, in binary polynomial the coefficients can only be 0 or 1. Thus, all the operations such as addition, subtraction, multiplication and division include polynomials of degree $m – 1$ or lesser. The m is the finitely large number of points on the EC to make the cryptosystem secure (Tata, 2007).

- **Point Addition**

'J' and *'K'* are two points on the curve which $J = (x_J, y_J)$ and $K = (x_K, y_K)$ and $L=K+J$ where $L= (x_L,y_L)$ and is the incline of the line through *'J'* and *'K'* then:

$s = (y_J + y_K)/(x_J + x_K)$

$x_L = s^2 + s + x_J + x_K + a$

$y_L = s (x_J + x_L) + x_L + y_J$

So if $K=-J$ for example $K = (x_J, x_J+y_J)$ then $J+K= \mathcal{O}$, where *'$\mathcal{O}$'* is the point at infinity. And if $K=J$ then $J+K=2J$. It needs to use point doubling equation also $J + K = K + J$.

- **Point Subtraction**

If *'J'* and *'K'* are two points on the curve where $J = (x_J, y_J)$ and $K = (x_K, y_K)$ Then:

$J - K = J + (-K)$ *where* $-K = (x_k, x_k+y_k)$

In certain implementation of point multiplication such as NAF, point subtraction is used (Riedel, 2003; Stoneburner, 2001b).

- **Point Doubling**

Assume $J=(x_J, y_J)$ and $y_J \neq 0$ so calculating $L= 2J$ where $L = (x_L, y_L)$ is:

$s = x_J + y_J/ x_J$, (*'S'* is the tangent at point *'J'*)

$x_L = s^2 + s + a$ (*‘a’* is one of the parameters selected with the EC)

$y_L = x_J^{\,2} + (s + 1)*x_L$

And *‘$\mathcal{O}$’* is the point at infinity if $xj=0$ then $2J= \mathcal{O}$.

Elliptic Curve Domain parameters

- **Domain parameters for EC over field Fp**

Elliptic curve over F_p has a list of domain parameters which includes ‘*p’, ‘a’,’ b’,’ G’, ‘n’* and ‘*h’*parameters.

‘a’ and *‘b’*: define the curve $y^2 \bmod p= x^3 + ax + b \bmod p$.

‘p’: prime number defined for finite field F_p

‘G’: generator point (X_G,Y_G) on the EC that is selected for cryptography operations.

‘n’: The Elliptic curve order.

‘h’: if $\#E(Fp)$ is the number of points on an elliptic curve, then *‘h’* is co-factor where $h=\#E(Fp)/n$.

- **Domain parameters for EC binary fields**

Elliptic curve over F_{2^m} has a list of domain parameters which include *‘m’, f(x), ‘a’, ‘b’, ‘G’, ‘n’* and *‘h’* parameters.

‘m’: an integer to finite field $F_2{}^m$.

F(x): the irreducible polynomial of degree m that is used for elliptic curve operations.

'a' and *'b'*: define the curves $y^2 + xy = x^3 + ax^2 + b$.

'G': the generator point (x_G, y_G) on the EC that is selected for cryptography operations.

'n': the order of the elliptic curve.

'h': if $\#E(F_{2^m})$ is the number of points on an elliptic curve then *'h'* is cofactor where $h=\#E(F_{2^m})/n$.

• Field Arithmetic

Modular arithmetic and polynomial arithmetic are two different types applied in ECC operations depending on the chosen field. In Modular arithmetic, over a number *'p'* arithmetic covers the number in the interval [0 and $p - 1$]. Modular Arithmetic contains Addition, Subtraction, Multiplication, Division, Multiplicative Inverse and Finding *x mod y* operations. Polynomial Arithmetic contains Addition, Subtraction, Multiplication, Division, Multiplicative Inverse and Irreducible Polynomial (Tata, 2007). Polynomial arithmetic plays an important role in a number of areas of engineering and software verification. In particular, polynomial limitation solving has a long and successful history in the developing tools to provide termination of programs (Borralleras, Lucas, Navarro-Marset, Carbonell, & Rubio, 2009). This research is based on polynomial arithmetic. Therefore, this part gives an overview of polynomial arithmetic.

EC over field F_{2^m} includes arithmetic of integer with length m bits. The binary string can be declared as polynomial:

Binary String: $(a_{m-1} \ldots a_1\ a_0)$

Polynomial: $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_2x^2 + a_1x + a_0$ where $a_i = 0$.

For example: $x^3 + x^2 + 1$ is polynomial for a four-bit number 1101_2.

- **Addition**

If $A = x^3 + x^2 + 1$ and $B = x^2 + x$ are two polynomial then $A+B$ called polynomial addition that returns $x^3 + 2x^2 + x + 1$ after taking *mod 2* over coefficients $A + B = x^3 + x + 1$.

On binary representation, polynomial addition can be achieved by simple XOR of two numbers. For example, over $\mathrm{GF}(2^4)$ there are 16 elements where $\mathrm{f(x)} = \mathrm{x}^4 + \mathrm{x} + 1$ as follows:

0 (0000)	1 (0001)	x(0010)
$x + 1$(0011)	x^2(0100)	$x^2 + 1$(0101)
$x^2 + x$(0110)	$x^2 + x + 1$(0111)	x^3(0100)
$x^3 + 1$ (1001)	$x^3 + x$ (1010)	$x^3 + x + 1$ (1011)
$x^3 + x^2$(1100)	$x^3 + x^2 + 1$(1101)	$x^3 + x^2 + x$ (1110)
$x^3 + x^2 + x + 1$ (1111)		

So if: $\left.\begin{array}{l} A = 1101_2 \\ \\ B = 0110_2 \end{array}\right\} A+B=A \text{ XOR } B \Rightarrow A+B=1011_2$

- **Subtraction**

If $A = x^3 + x^2 + 1$ and $B = x^2 + x$ are two polynomials, then A-B is called polynomial subtraction that returns $x^3 - x + 1$ after taking *mod 2* over coefficients $A - B = x^3 + x + 1$.

On binary representation, polynomial addition can be achieved by a simple XOR of two numbers same as the Addition operation in F_{2^m}:

$$\left.\begin{array}{l} A = 1101_2 \\ \\ B = 0110_2 \end{array}\right\} A\text{-}B = A \text{ XOR } B \Rightarrow A+B = 1011_2$$

- **Multiplication**

If $A = x^3 + x^2 + 1$ and $B = x^2 + x$ are two polynomials, then A*B is called polynomial multiplication that returns $x^5+x^3 + x^2 + x$, $m=4$. The result should be reduced to a degree less than 4 by irreducible polynomial $x^4 + x + 1$.

$x^5 + x^3 + x^2 + x \ (mod\ f(x)) = (x^4 + x + 1)x + x^5 + x^3 + x^2 + x$

$= 2x^5 + x^3 + 2x^2 + 2x = x3$ (after reducing the coefficient on *mod 2*)

$A = 1101_2$

$B = 0110_2$

$A * B = 1000_2$

- **Division**

*a * b⁻¹*

$a * b^{-1}$ *(mod f(x))* has the same result of *a/b(mod f(x)).* So in order to find *a/b(mod f(x)),a* * b^{-1} *(mod f(x))* can be used. Instead of this, b^{-1} is the multiplicative inverse of *'b'*.

- **Multiplicative Inversion**

There are 16 powers for g where the element $g = (0010)$ is a generator:

g^0 = (0001)	g^1= (0010)	g^2 = (0100)	g^3 = (1000)
g^4 = (0011)	g^5 = (0110)	g^6 = (1100)	g^7 = (1011)
g^8 = (0101)	g^9 = (1010)	g^{10} = (0111)	g^{11} = (1110)
g^{12} = (1111)	g^{13} = (1101)	g^{14}= (1001)	g^{15} = (0001).

The multiplicative identity is $g^0 = (0001)$ and the multiplicative inverse for $g^9 = (1010)$is:

g^9= (1010) is g^{-9}mod 15 = g^6 mod 15

To assure that $g^6$is the multiplicative inverse of g^9 their multiplication result should be 1.

Proofing that$g^6 \times g^9 \equiv 1 \; mod \; f(x)$

$g^6 \cdot g^9 =$ (1010) · (1100)

$(x^3 + \mathrm{x}) \cdot (\mathrm{x}^3 + \mathrm{x}^2)$ mod f(x)

$(x^6 + x^5 + x^4 + x^3)$ mod f(x)=1 (which is the multiplicative identity)

This means $g^{-9} \equiv g^6$mod f(x)

APPENDIX B

• Analysis of Security Services and Architectures for Binding Updates

In order to minimise and address the security risks and attacks discussed above, the following security services should be provided (Modares, 2009; Riedel, 2003; Stoneburner, 2001a) :

• Authentication Mechanisms and Architecture in Mobile IP

The aim of encryption is to keep data private in case it gets intercepted. To authenticate and verify system users, various techniques exist to ensure that the data received is intact and real.

Security protocols normally use some methods of authentication, and followed by a protected data exchange. Sometimes, the states are closely integrated like in TLS (Transport Layer Security), or at other times, the states can be separate and belong to different endpoints, like in EAP (Extensible Authentication Protocol) and Kerberos (Eronen & Arkko, 2006).

Public Key Infrastructure (PKI) can be used in decentralized authentication, and does not need an on-line authentication server. Current PKIs require a centralized server because they do not do well with authorization. Authentication and authorization data are often integrated into an authentication, authorization and accounting (AAA) server (Arkko, Calhoun, Guttman, Nelson, & Wolff, 2000; Modares et al., 2010).

- **Key Exchange Protocols**
- **Kerberos**

Kerberos is a network authentication protocol (Kohl & Neuman, 1993; TUNG, 2005) that uses secret-key cryptography in order to provide strong authentication for client/server applications. It is based on the key distribution model developed by Needham and Schroeder (Needham & Schroeder, 1978) who wanted to eliminate the need to present private or secret information (a password) by exposing information. Kerberos fundamental architecture is the KDC (Key Distribution Centre) which is responsible for storing the authentication data and uses it to securely authenticate users and services.

Secure authentication means:

• Does not occur in plaintext;

• Does not rely on authentication by the host operating system;

• Does not trust IP addresses; and

• Does not require network host physical security.

The KDC is basically a third-party which carries out authentication services, with many, vital functions, where each KDC stores a database of users, servers, and secret keys. Kerberos clients are normal network applications, adapted to authentication.

Total security is crucial since KDCs contain secret keys for every network user and server. For instance, if an attacker impersonates an authenticated system user, he would gain administrative access to the KDC with all its Kerberos resources.

To encrypt Kerberos, Data Encryption Standard (DES) algorithm is used, and adding encryption or check sum algorithms are available. Kerberos also supports the Cyclic Redundancy Check (CRC-32), Message-Digest Algorithm (MD4, MD5), and the DES algorithms for checksums.

- **AAA - Authentication, Authorization and Accounting**

On the Internet, a client belonging to one administrative domain (home domain) usually has to utilize resources from a different administrative domain (foreign domain). In dealing with a client, a foreign domain agent (attendant) probably needs the client's certification for authentication prior to granting resource access (Arkko et al., 2000). Certification should be understood by the foreign domain; however, credentials are mostly allocated and understood only by the home domain, to help create secure channels to the MN.

The AAA protocols (Metz, 1999) were first created for dial-up users with secure access to an ISP. AAA defines a framework for coordinating individual security and network elements like user authentication, authorization to access services, and service accounting for billing over many network technologies and platforms. The AAA protocol offers distributed services by communicating with an AAA client. Before yielding network access, the AAA server uses its user database for authenticating and validating end-users who must provide unique identification like a password, a cryptographic key or biometric data to be compared against the information in the database. Thus, when data match, the user is granted network access, if it does not, access is denied.

After authorization, the server defines the services a user may access, including providing an IP address, and allowing access to, or supporting certain applications. Accounting supervises network use and gathers data regarding resource usage to help with billing and auditing.