SECURING HOME AND CORRESPONDENT REGISTRATIONS IN MOBILE IPv6 NETWORKS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2010

By
Osama Elshakankiry
School of Computer Science

Contents

Word count: 44,241

List of Tables

List of Figures

Abstract

The Mobile IPv6 (MIPv6) protocol enables mobile nodes (MNs) to remain connected to other correspondent nodes (CNs) while roaming the IPv6 Internet. Home and correspondent registrations are essential parts of the MIPv6 protocol, whereby MNs register their care-of addresses (CoAs) with their home agents (HAs) and with their CNs, respectively. Security provision for home and correspondent registrations is a fundamental part of the MIPv6 protocol and has been an open research issue since the early stages of the protocol.

This thesis examines state-of-the-art protocols for securing home and correspondent registrations in MIPv6 networks. The strengths and weaknesses of these protocols are discussed. The investigation of these protocols leads to the proposal of an enhanced home registration protocol and a family of correspondent registration protocols. The Enhanced Home Registration (EHR) protocol extends the basic home registration protocol defined in MIPv6 to support the location authentication of MNs to their HAs. The EHR is based on novel ideas of segmenting the IPv6 address space, using a symmetric CGA-based technique for generating CoAs, and applying concurrent CoAs reachability tests. As a result, EHR is able to reduce the likelihood of a malicious MN being successful in luring an HA to flood a third party with useless packets using MIPv6. In addition, EHR enables HAs to help in correspondent registrations by confirming MNs' CoAs to CNs. Simulation studies of EHR have shown that it only introduces a marginal increase in the registration delay, but a significant increase in the signalling overhead as a cost of supporting the location authentication of MNs.

The thesis also proposes a family of correspondent registration protocols. These protocols rely on the assistance of home networks to confirm the MNs' ownership of the claimed HoAs and CoAs. The protocols consist of three phases: a creation phase, an update phase and a deletion phase. Informal and formal protocol analyses have confirmed the protocols' correctness and satisfaction of the required security properties. The protocols have been simulated extensively and the results show that they produce lower registration delay and a reduction in the signalling overhead during update and deletion phases. This is at the cost of a varying increase, depending on the protocol variant, in the registration delay and signalling overhead during the creation phase.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the "Copyright") and s/he has given The University of Manchester the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

ii. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the John Rylands University Library of Manchester. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

iii. The ownership of any patents, designs, trade marks and any and all other intellectual property rights except for the Copyright (the "Intellectual Property Rights") and any reproductions of copyright works, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and exploitation of this thesis, the Copyright and any Intellectual Property Rights and/or Reproductions described in it may take place is available from the Head of School of Computer Science (or the Vice-President).

Acknowledgements

I would like to express my deep gratitude to my first supervisor Dr. Andy Carpenter for his support, encouragement, and guidance during this research. I would also like to express my special thanks to my second supervisor Dr. Ning Zhang for her numerous helpful discussions, invaluable suggestions, and comments on my work.

My deepest gratitude and appreciation to my wife and children for their help and support in difficult times. Your love, understanding and patience sustained me through to the end of my Ph.D.

I cannot thank my parents enough for their love and sacrifice throughout my whole life. Over the years they have continued to support and encourage me. I would like to thank them for everything.

Last but not least, I would like to thank my beloved country "Egypt" for its sponsorship to do this research and giving me this opportunity.

Dedication

I dedicate this thesis to my beloved mum and late dad who passed away on Saturday, 7^{th} August 2010.

Definitions

Glossary of Terms and Acronyms

The following gives the alphabetical list of terms and acronyms used in the thesis.

BA – Binding Acknowledgement; a mobility message sent by a home agent and a correspondent node to a mobile node to acknowledge the binding of a care-of address.

BACoT – Binding Acknowledgement with Care-of Token; a mobility message sent by a home agent to a mobile node to acknowledge the binding of a care-of address and deliver a fresh care-of token.

BHR – Basic Home Registration; a home registration process that allows a mobile node to (de)register a care-of address with a home agent.

BU – Binding Update; a mobility message sent by a mobile node that contains the mobile node's home address and current care-of address.

BUCoT – Binding Update with Care-of Token; a binding update message sent by a mobile node to a home agent, which contains a care-of token.

CA – Certification Authority; a trusted entity that issues and revokes public-key certificates.

CGA – Cryptographically Generated Address; an IPv6 address in which the interface identifier part is generated using a cryptographic one-way hash function that takes the address owner's public key and some auxiliary parameters as its input.

CN – Correspondent Node; a peer node either mobile or stationary that communicates with a mobile node.

CoA – Care-of Address; a mobile node's transient address that gives the mobile node's location while away from home link.

CoT – Care-of Test; a mobility message sent by a correspondent node to a mobile node as a response to a Care-of Test Init message.

CoTI – Care-of Test Init; a mobility message sent by a mobile node to initiate the return routability procedure and request a care-of keygen token from a correspondent node.

Correspondent Registration – A process that is initialized by a mobile node to notify a correspondent node about the mobile node's current location while away from home link.

DAD – Duplicate Address Detection; a test for checking whether an address is already in use.

EHR – Enhanced Home Registration; an enhanced home registration protocol that supports the location authentication of mobile nodes to their home agents, see Chapter 4.

HA – Home Agent; a router located on a mobile node's home link with which the mobile node has registered one of its current care-of addresses.

HoA – Home Address; a mobile node's permanent address that identifies the mobile node in its home link.

Home Registration (HR) – A process that is initialized by a mobile node to notify a home agent about the mobile node's current location while away from home link.

HoT – Home Test; a mobility message sent by a correspondent node to a mobile node as a response to a Home Test Init message.

HoTI – Home Test Init; a mobility message sent by a mobile node to initiate the return routability procedure and request a home keygen token from a correspondent node.

INF-based protocol – A correspondent registration protocol for use when no prior relationship exists between mobile nodes' home links and correspondent nodes.

IPSec – Internet Protocol Security; a set of protocols that provides security to IP and upper-layer protocols.

IPSec AH – IPSec Authentication Header; a member of the IPSec protocol suite, which is used to protect the authenticity and integrity of an IP packet.

IPSec ESP – IPSec Encapsulating Security Payload; a member of the IPSec protocol suite, which is used to protect the confidentiality, authenticity, and integrity of an IP packet.

LT – Binding Lifetime; a lifetime for the binding of home and care-of addresses of a mobile node.

MIPv6 – Mobile IPv6; an IP-layer mobility protocol that enables mobile nodes to remain connected to other correspondent nodes while roaming the IPv6 Internet.

MN – Mobile Node; a mobile device that has a home link and a permanent home address on that link. It can change links and maintain reachability using its home address.

PK-based protocol – A correspondent registration protocol for use when a mobile node's home link has a certified public/private key pair.

PKI – Public-Key Infrastructure; an infrastructure including all entities involved in managing public-key certificates.

Public-key certificate – A digital document, issued and digitally signed by a CA, that binds the name of a public-key owner to his public key.

RO – Route Optimization; a mode of communication that allows packets to be routed directly between a mobile node and a correspondent node.

RR – Return Routability; a procedure that is currently used to protect correspondent registrations in the Mobile IPv6 protocol.

SA – Security Association; a bundle of algorithms and keys that is used to specify parameters controlling security operations in IPSec.

SK-based protocol – A correspondent registration protocol for use when a secret key is shared between a mobile node's home link and a correspondent node.

Notation

The following notation has been used throughout the thesis.

K_I	Entity I's secret key (a secret random value generated and only known by I).
K_{I-J}	A secret key shared between entities I and J.
(PK_I, SK_I)	Entity I's public/private RSA key pair.
K_{BM}	A binding management secret key.
K_{BC}	A binding confirmation secret key.
$ENC_K(x)$	A ciphertext of a data item x encrypted with key K.
$DEC_K(x)$	The inverse function of encryption function $ENC_K(x)$, i.e. it denotes decryption of ciphertext x using key K.
$SIG_I(x)$	Entity I's signature on a data item x.
$H(x)$	A collision-resistant one-way hash function, such as SHA1.
$MAC_K(x)$	A keyed hash function using key K, such as HMAC_SHA1.
LT_{BReq}	A binding lifetime requested by a mobile node.
LT_{BRem}	A remaining lifetime for the binding of home address and care-of address at the mobile node and the home agent.
N_I	A fresh random nonce generated by an entity I.
$CGA\ Parameters$	Auxiliary parameters used in generating a CGA-based address; they include a 128-bit random number (called a modifier), an eight-bit number (called a collision count), and a 64-bit subnet prefix of the CGA-based address.
$x \mid\mid y$	Concatenation of data items x and y.

Chapter 1

Introduction

The rapid growth in the number of wireless Internet devices combined with a desire of the users of these devices to remain connected to the Internet lead to the development of IP-based mobility protocols. These protocols allow mobile Internet hosts (called mobile nodes (MNs)) to remain connected to other hosts (called correspondent nodes (CNs)) while roaming the Internet. However, as will be shown, these protocols also add inherent security risks to those already in the Internet today. Most of these risks are introduced from redirecting traffic intended for an MN from one address to another. The signalling for such redirection requires protection. If not protected, it can have detrimental effects on the entire Internet [1]. The need to reduce vulnerabilities can be seen by examining existing attacks that have been launched on Internet devices.

1.1 Network Security Threats and Attacks

A security threat can be informally defined as an entity or program with the capability to cause harm or distort normal security operations by exploiting vulnerabilities in a system [2, 3]. In these papers, five types of threats to computer networks are identified:

- **Errors and omissions:** comprising of network errors, system faults and omissions. They cause harms to the system, or create vulnerabilities in the system and can be caused by all types of users (novice and expert users).
- **Deliberate software threats:** comprising of network worms, viruses, Trojans and denial of service.

- **Insider threats:** comprising of disgruntled employees and threats from legitimate users of the systems.
- **Cyber-threats:** comprising of terrorism, political warfare, organised crimes, phishing, and pharming.
- **Natural disaster:** comprising of fire, flooding, earthquakes, and hurricane.

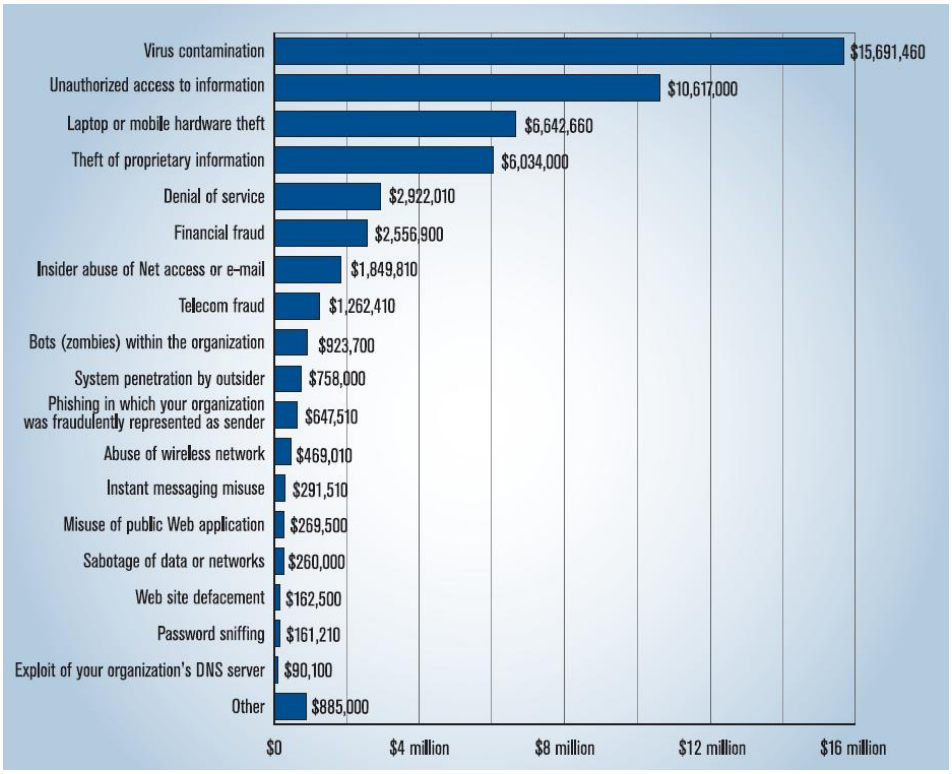

Figure 1.1: Dollar amount losses due to security threats [4]

Figure 1.1 shows the losses in US dollars caused by different types of threats obtained from CSI/FBI crime survey [4]. As shown, viruses topped the list; however, other new types of threats, such as Cyber-threats, are beginning to emerge. These types of threats accounted for a significant amount of dollar losses compared to previous years.

1.2 IP-based Mobility Protocols

IP-based mobility protocols aim to achieve two goals: first, allow MNs to be reachable as long as they are connected somewhere to the Internet, and second,

maintain ongoing connections between MNs and their CNs. These goals are achieved by assigning an MN two IP addresses: a permanent home address (HoA) that identifies the MN in its home link and a transient care-of address (CoA) that gives the MN's location while away from its home link. The HoA remains the same during movements and the CoA changes every time the MN changes its IP connectivity. When the MN moves to a foreign link and configures a new CoA, it initiates a home registration process to register the CoA with a router in its home link (called a home agent (HA)). This registration allows the HA to create a binding for the MN between its HoA and its CoA. Using this binding, the HA intercepts all packets on the home link destined for the MN's HoA and forwards them through a tunnel to the MN's registered CoA, i.e. to the MN's current location [5, 6, 7, 8]. Thus, via the HA's binding for the MN, the MN is reachable at any location on the Internet using its CoA, and its ongoing connections are maintained using the HoA.

IP-based mobility protocols also define a correspondent registration process that allows an MN to register its CoA with a CN. The correspondent registration improves routing (called route optimization (RO)) by enabling packets to be sent directly between the MN and the CN. The correspondent registration is voluntary in the sense that either the MN or the CN can refuse to do it, and they can continue communicating via the MN's home link [5].

In the home/correspondent registration, the MN sends the HA/CN a mobility message called a 'Binding Update (BU)' that contains the MN's HoA and current CoA. The HA/CN first stores this binding in its binding cache, which indicates packets destined to that HoA should be forwarded/sent to the bound CoA. The HA/CN then replies to the MN by returning a 'Binding Acknowledgement (BA)' mobility message that acknowledges the binding of the current CoA.

The BUs sent from MNs during home and correspondent registrations, i.e. the location management feature of IP-based mobility protocols, are obviously very sensitive; they modify routing to enable mobility in the network. By spoofing BUs, an attacker can divert traffic to itself or to another node and prevent the original MN from receiving the traffic destined to it. This opens up possibilities for many attacks, discussed later in Section 2.2.1, such as denial of service, man-in-the-middle, connection hijacking, and impersonation attacks [5, 9]. Therefore,

the biggest security vulnerability of IP-based mobility protocols is the authentication and authorization of BUs sent from MNs. Consequently, a security provision for home and correspondent registrations is a fundamental part of IP-based mobility protocols. It is believed that the deployment of an IP-based mobility protocol without securing home and correspondent registrations could result in a breakdown of the entire Internet [1].

The current specifications of IPv6-based mobility protocols use IPSec Encapsulating Security Payload (ESP) [10] to protect home registrations and a Return Routability (RR) procedure to protect correspondent registrations [5, 11]. In addition, sequence numbers are used within BUs and BAs to protect against replay attacks. IPSec and sequence numbers can protect home registrations against outsider attacks, i.e. an attacker cannot send a spoofed or a replayed BU message instead of the MN. They can also partially protect against insider attacks, i.e. the MN cannot send a BU on behalf of another MN that is using the same HA. However, they fail to prevent the MN from lying about its current location, i.e. the MN might provide a false CoA to the HA and thus redirect data to this address.

The RR procedure does not provide sufficient protection for correspondent registrations. Furthermore, it is inefficient as it requires the exchange of six messages and takes about 1.5 round-trip times between the MN and the CN to complete one correspondent registration. The procedure checks the MN's reachability at both the HoA and the CoA. In the procedure, the CN generates and sends two secret keys as plaintext to the MN's home and care-of addresses. The MN and the CN use the two keys to establish a session key that is used to protect the subsequent correspondent registration. However, an attacker that can capture both the keys can generate the session key and successfully impersonate the MN. As a result, the procedure only reduces the number of attackers from any node in the Internet to those that are on the route between the CN and the MN's home link. In addition, the procedure increases signalling overhead and registration delay as it requires six messages and about 1.5 round-trip times per correspondent registration. Other protocols, discussed later in Chapter 3, were proposed to overcome the RR's limitations and to protect correspondent registration; however, they also have their limitations in terms of needing a security infrastructure, increasing registration delay, increasing signalling overhead, and/or increasing load at MN side.

The limitations of the solutions currently used to protect home and correspondent registrations have motivated the research described here to investigate ways of improving security provisions for home and correspondent registrations in IPv6-based mobility protocols.

1.3 Research Hypotheses

Existing work on securing home registrations fails to protect against legitimate MNs behaving maliciously. A malicious MN could pretend to own a third party's address and lure an HA to flood that victim with useless packets. Similarly, existing solutions for securing correspondent registrations do not provide sufficient protection and/or cause high registration delay and signalling overhead. As mentioned above, securing home and correspondent registrations are a fundamental part of the deployment of IP-based mobility protocols without the breakdown of the entire Internet.

The first hypothesis of this research is that it is possible to improve home registrations to support location authentication of MNs to their HAs. This could be achieved by cryptographically configuring an MN's CoA based on a secret key shared between the MN and an HA, and by applying a concurrent CoA reachability test. By doing so, the HA could verify authenticity of the CoA and verify the MN's reachability at the CoA with a marginal increase in the registration delay. In addition, this could enable HAs to participate in correspondent registrations by confirming MNs' CoAs to CNs.

The second hypothesis is that an efficient and secure correspondent registration could be achieved by involving HAs in the registration (to confirm MNs' HoAs and CoAs) and by designing the registration to occur in three phases: a creation phase, an update phase and a deletion phase. The authentication of an MN and a home link to a CN, and the establishment of registration session keys could be done when they do not impact registration delay. They could take place in the creation phase where the MN and the CN communicate through the MN's home link. In addition, by using these session keys in the update and the deletion phases, the security strength of the correspondent registration could be increased and the registration delay and signalling overhead could be reduced.

The third hypothesis is that all stationary hosts, such as popular web servers, could be protected against different attacks caused as a side effect of the location

management feature of IP-based mobility protocols by finding a way to differentiate between redirectable and non-redirectable IP addresses. This could prevent attackers from falsely claiming that stationary hosts' addresses are their HoAs or CoAs. Consequently, it could significantly reduce possible attacks that could be launched due to IP-based mobility protocols.

1.4 Research Aim and Objectives

The aim of this research is to improve the security as well as the efficiency of home and correspondent registrations in IP-based mobility protocols. To restrict the scope of the discussion, this thesis is limited to an improvement in the Mobile IPv6 (MIPv6) mobility protocol. The reasons for this are threefold. Firstly, to support the rapid growth of the Internet, it is expected that IPv6/MIPv6 will replace the current IPv4/MIPv4 in the next few years. Secondly, MIPv6 is the base for all IPv6-based mobility protocols; other IPv6-based mobility protocols such as Fast Handover Mobile IPv6 [6] and Hierarchical Mobile IPv6 [8] are extensions of MIPv6 that improve handover latency. Thirdly, simulation packages such as NS-2 and OPNET [12, 13] implement MIPv6, and thus could be used to simulate and evaluate existing solutions as well as the proposed solutions.

In order to achieve this aim, the objectives of this thesis are as follows.

1. To thoroughly understand the security and performance needs of home and correspondent registrations;

2. To identify weaknesses in the existing solutions for securing home and correspondent registrations;

3. To overcome the weaknesses and to advance the state of the art by designing an enhanced home registration protocol that supports location authentication of MNs to their HAs, i.e. allows HAs to verify MNs' ownership of claimed CoAs;

4. To overcome the weaknesses and to advance the state of the art by designing a family of correspondent registration security protocols that could be used in different scenarios depending on the relationship between MNs' home links and CNs;

5. To perform security analysis and performance evaluation of the designed protocols. This will show improvements offered and identify any negative impacts of the designed protocols.

1.5 Research Method

It was possible to identify the following key tasks that need to be completed as part of the research project.

1.5.1 Literature Review

The first task was to study the general area of interest. In the context of this project, a starting point of addressing the issues related to IP-based mobility was given. From this, MIPv6 protocol was selected as an IPv6-based mobility protocol that will be used in the next few years to support mobility in the Internet. The next point was to identify security risks that could be launched by applying the MIPv6 protocol. The security and performance requirements of home and correspondent registration processes used in the MIPv6 protocol were then identified. Related existing work was identified and reviewed against the requirements. Gaps where further work was necessary were identified. Design hypotheses based on state-of-the-art work were identified, and theoretical work on designing protocols based on the hypotheses could start. From this, the area of securing home and correspondent registrations was identified as the field of research. At this point, the literature review was not finished; reviewing relevant literature carried on throughout the project. As new work was published, this too was reviewed and where necessary taken into account.

1.5.2 Theoretical Work

Upon the initial intensive literature review, solutions for the issue of providing security to home and correspondent registrations in MIPv6 protocol were identified. The ideas were repeatedly refined by taking input from existing work. Considerations were given to minimising any additional overheads incurred by the proposed solutions. At the conclusion of this stage, two novel protocols were proposed. The first protocol was the Enhanced Home Registration (EHR) protocol and the second was a family of correspondent registration security protocols.

1.5.3 Security Analysis

On completion of the design stage, the next stage was to verify the security of the proposed protocols. The security analysis strategy included three methods: informal analysis, formal analysis and comparison with related work. The informal analysis was used to analyse the proposed protocols against the security requirements and risks identified in the literature review stage. The formal analysis was used to provide a rigid and thorough means of testing the correctness of the proposed. The formal analysis was carried out using both the framework of protocol composition logic [14, 15] and the Casper/FDR2 model checker [16, 17]. Finally, a comparison of the proposed protocols and their most relevant work was done to demonstrate the merits of the security (and performance) they provide over the related work.

1.5.4 Simulation Modelling

After the security analysis stage, the next stage was to evaluate the performance of the proposed protocols. Evaluation was carried out using the OPNET^{TM} Modeler version 14.5 simulation package [13]. The first step was to design and construct the simulation models. Once the simulation models were constructed, they were validated. Validation was performed by driving a mathematical model of a simplified simulation model and comparing the simulated results with the results obtained from the mathematical model. In addition, the OPNET debugger was used to prove that the proposed protocols operate correctly by outputting different processes carried out by all involved entities, and by outputting relevant packet information, i.e. source address, destination address, packet contents, and packet size. Once the validation was complete, the simulation models could be run with confidence that they were accurate.

1.5.5 Evaluation

When simulation was completed, the last stage was to process the results and produce graphs. These graphs were used to compare the performance of the proposed protocols with that of the most related existing work. Conclusions were then drawn from these comparisons.

1.6 Achievements and Novel Contributions

The research work presented in this thesis has led to the following achievements and novel contributions:

1. The design of a novel symmetrical CGA-based technique to cryptographically generate and verify an IPv6 address. It makes use of a secret key shared between the participants, i.e. address owner and address verifier, in the address generation and verification processes. As a result, it reduces the computational and communication costs of the participants compared to the traditional CGA-based technique, which requires digital signature generation and verification processes. This in turn resulted in the suitability for use by MNs to configure new CoAs and by HAs to verify the authenticity of those CoAs. It aims to reduce the likelihood of a malicious MN being successful in stealing a third party's address by forcing the MN to attempt about (2^{61}) tries to be able to produce the same address.

2. The design of a novel concurrent CoA reachability test that allows an HA to register and use an MN's new CoA while concurrently verifying the MN's reachability at that CoA. This test has no effect on the registration delay as it runs parallel to data transfer to and from the new CoA. It also allows the HA to make use of a secret key, called the 'node key', to produce tokens sent to the MNs. This enables the HA to verify that the token returned by the MN is indeed its own without forcing the HA to remember a list of all tokens it has handed out. In addition, the test allows the HA to limit the number of BUs that can be received from unreachable CoAs, which prevents malicious MNs from bypassing the test by continuously sending BUs without involving the test. Even though a concurrent CoA reachability test is not entirely new in the MIPv6 context, this is the first concurrent CoA reachability test between HAs and MNs that incorporates the node key concept, is able to prevent malicious MNs from bypassing the test, and has no effect on the registration delay.

3. The proposal of a novel method of segmenting IPv6 address space to differentiate between redirectable and non-redirectable IPv6 addresses. The method uses two bits of the IPv6 64-bits interface identifier field to distinguish between redirected-from addresses (i.e. MNs' HoAs), redirected-to

addresses (i.e. MNs' CoAs), and non-redirectable addresses (i.e. stationary hosts' addresses). This method needs to be deployed on a global scale on the IPv6 Internet to be able to protect stationary hosts, as well as other MNs located at their home links, against attacks that could be launched via abuse of current and future mobility protocols.

4. The design of a novel home registration protocol called the Enhanced Home Registration (EHR) protocol. It incorporates the novel symmetrical CGA-based technique, the novel concurrent CoA reachability test, and the novel segmenting IPv6 address space method mentioned above to allow HAs to verify MNs' ownership of claimed CoAs without incurring a negative impact on registration delay. The protocol introduces a marginal increase in the registration delay, but a significant increase in the signalling overhead as a cost of supporting location authentication of MNs.

5. The design of a family of novel correspondent registration security protocols that rely on the assistance of an MN's home link to enable a CN to securely authenticate an MN's ownership of a HoA and a CoA. The protocols can be used in different scenarios depending on the relationship between MNs' home links and CNs:

   – SK-based protocol for use when a secret key is shared.

   – PK-based protocol for use when the home link has a certified public/private key pair.

   – INF-based protocol for use when no prior relationship exists.

   The protocols are designed in three phases: a creation phase, an update phase and a deletion phase, where most of the authentication and keys establishment take place in the creation phase (while the MN and the CN communicate through the MN's home link). By carrying out authentication and keys establishment in the creation phase, and by using these keys in the update and deletion phases, security strength is increased and registration delay is reduced. In addition, the protocols offload computational and communication to the home link from the MN, thus reducing computing and memory requirements as well as power consumption for the MN.

6. Security and performance evaluation of the proposed protocols. Specifically, (a) informal analysis of the protocols against the security requirements and well-known attacks; (b) formal verifications of the security properties using the framework of protocol composition logic and the Casper/FDR2 model checker; and (c) evaluation of the performance of the protocols using OPNET simulation package, which has been compared to that of related work.

Parts of the research work presented in this thesis have been published in the following conference proceedings.

1. Osama Elshakankiry, Andy Carpenter and Ning Zhang. A New Secure Binding Management Protocol for Mobile IPv6 Networks, In *Proceeding of the 4th International Symposium on Information Assurance and Security (IAS08)*, IEEE CS Press, Naples, Italy, September 2008, pp. 281-286.

2. Osama Elshakankiry, Andy Carpenter and Ning Zhang. A Novel Scheme for Supporting Location Authentication of Mobile Nodes, In *Proceeding of the Second International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec2010)*, LNICST, Catania, Italy, May 2010, pp. 91-102.

1.7 Thesis Structure

The thesis is structured as follows. Chapter 2 describes the current specification of the Mobile IPv6 protocol; it defines the basic home registration and the return routability procedure, which are currently used to protect home and correspondent registration processes, respectively. It also presents the security risks and attacks and outlines the security services that define the security requirements for securing home and correspondent registration processes. Chapter 3 gives a survey of state-of-the-art protocols relevant to the protection of the correspondent registration process. The strengths and weaknesses of these protocols are then identified.

Chapters 4 to 6 present our novel research work, contributions, and results. Chapter 4 presents the design, implementation and performance evaluation of our EHR protocol. This chapter includes the efficiency analysis and the performance comparison of the proposed EHR protocol with the basic home registration protocol. Chapter 5 presents the design of a family of novel correspondent registration

security protocols. Chapter 6 presents the security analysis and performance evaluation of the protocols proposed in Chapter 5. The security analysis is conducted both informally against requirements and attacks, and formally using both protocol composition logic and Casper/FDR2 model checker. Again the performance of the protocols are measured and compared with the most related existing protocols. A brief description of the methods chosen for formal verification of the protocols' security properties is also provided. Finally, Chapter 7 concludes the thesis and suggests directions for future work. A brief introduction to the basic cryptographic techniques that are used in the design of home and correspondent registration protocols is given in Appendix A. Full details of simulation model validation and formal protocol verification are also presented in the appendices. A brief introduction to the OPNET Modeler simulation package and the CryptoSys Cryptography Toolkit that are chosen for performance evaluation is also given.

Chapter 2

The Mobile Internet Protocol version 6 (Mobile IPv6)

This chapter gives an insight into the current specification of the Mobile IPv6 protocol. In detail, Section 2.1 presents an overview of the protocol, its main components, and the two possible modes of communications allowed between an MN and a CN. Section 2.2 details security risks and attacks that could be launched via abuse of the location management feature of the protocol. It also presents the security services and performance requirements needed to overcome these attacks. Section 2.3 gives an overview of the Internet Protocol Security (IPSec), which is currently used to protect control traffic between MNs and HAs. Section 2.4 is devoted to the description of the basic home registration process included in the protocol, which enables an MN (while away from home link) to register its current location with an HA. It also analyses the security services provided by the protocol to the process against the requirements specified in Section 2.2. Section 2.5 presents a detailed description of the return routability (RR) procedure included in the protocol to protect correspondent registrations. It also identifies the security and performance limitations of the procedure against the requirements specified in Section 2.2. Finally, Section 2.6 summarises the chapter.

2.1 Basic Operations

Mobile IPv6 (MIPv6) is an IP-layer mobility protocol for the IPv6 Internet [5]. It is designed to allow a mobile node to always be reachable with one address,

i.e. a home address, regardless of the mobile node's actual location. It also makes it possible for the mobile node to maintain existing connections with other correspondent nodes across different locations.

The MIPv6 protocol introduces three network entities: a mobile node (MN), a home agent (HA) and a correspondent node (CN). The MN is a mobile device that has a home link and a permanent home address (HoA) on that link. The MN also acquires one or more transient care-of addresses (CoAs) when it roams to some foreign link. The HA is a router located on an MN's home link with which the MN has registered one of its current CoAs. The HA maintains registrations of the MNs that are away from home and their current CoAs. In addition, when the MN is away from home, the HA intercepts packets on the home link destined for the MN's HoA and forwards them to the MN's registered CoA. The CN is a peer node that communicates with an MN. The CN may be either mobile or stationary, and it does not have to be aware of the MIPv6 protocol.

In order for an MN to receive data packets destined for its HoA while away from home, it must register one of its current CoAs with an HA on its home link. When the MN moves to some foreign link, it automatically detects its movement using the Router Discovery protocol [18]. In addition, the MN automatically obtains a new CoA corresponding to the subnet prefix of the foreign link using either stateless or stateful address auto-configuration [19, 20]. The MN may have multiple CoAs at the same time with different subnet prefixes (e.g. in the case of overlapping wireless networks). The MN registers one of its current CoAs with the HA by running a home registration process. The MN initiates the home registration by sending a Binding Update (BU) message to the HA, which contains the MN's HoA and CoA. The HA stores this binding, i.e. association between the HoA and the CoA, in its Binding Cache (which will be explained later in greater detail). Using this binding, the HA intercepts and forwards packets destined for the HoA to the bond CoA. The HA concludes the home registration by returning a Binding Acknowledgement (BA) message to the MN to acknowledge the binding of the CoA. The MN's CoA registered at the HA is called the MN's primary CoA.

There are different ways for an MN to communicate with a CN. While the MN is at home, all traffic between the MN and the CN is routed to and from the HoA using the normal IPv6 routing mechanisms [21], i.e. no special procedure is required. When the MN is away from its home link, two modes of communication are possible: bidirectional tunnelling and route optimization. In the bidirectional

tunnelling mode, all traffic is routed indirectly via the MN's home link. In detail, this means packets from the CN are routed to the MN's home link where the HA intercepts and forwards them through a tunnel to the MN's CoA. Packets to the CN are tunnelled from the MN to the HA ('reverse tunnelled') and then routed normally from the home link to the CN. Tunnels between MNs and HAs are normally performed using IPv6 encapsulation [22]. However, when these tunnels need to be secured, they are replaced by IPSec tunnels. The bidirectional tunnelling mode is shown in Figure 2.1.

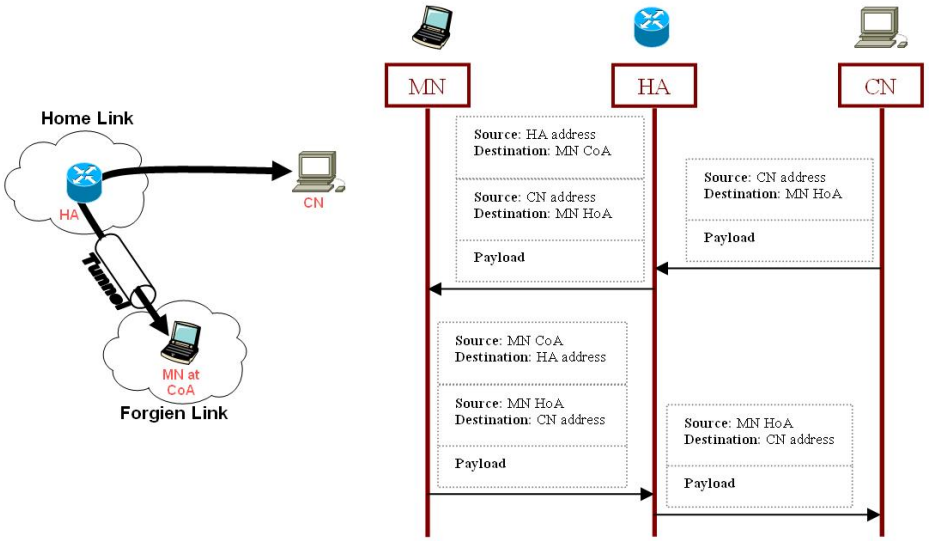

Figure 2.1: Bidirectional tunnelling communication

The route optimization (RO) mode allows packets to be routed directly between the MN and the CN, as shown in Figure 2.2. This mode requires the CN to be aware of the MIPv6 protocol and the MN to register its current location with the CN. The registration is performed by running a correspondent registration, in which the MN and the CN exchange BU and BA messages. As a result, the CN creates a Binding Cache entry and stores the binding between the MN's HoA and CoA. Subsequently, the CN sends all packets destined for the MN to its CoA instead of its HoA. The RO mode allows the presumably shorter path between the MN and the CN to be used and minimizes traffic levels at the HA as well as at the home link [23].

In order for the optimized route to be used, two new types of routing headers are included in the MIPv6 protocol: the 'Home Address Destination Option'

header and the 'Type-2' header. When routing a packet directly to the CN, the MN uses its CoA as the source address and includes its HoA in a 'Home Address Destination Option' header. The header indicates that although the source address is the CoA, the packet is actually from the node whose address is the HoA. After receiving the packet, layer 3 at the CN replaces the source address with the HoA before passing the packet to the upper layer protocol. Similarly, when routing a packet directly to the MN, the CN sets the destination address to the MN's CoA and includes the MN's HoA into the 'Type-2' header. The header indicates that although the packet is destined for the CoA, it is really intended for the HoA. When the MN receives the packet, layer 3 at the MN replaces the destination address with the HoA before passing the packet to the upper layer protocol. Thus, through the use of the two new headers, the MIPv6 operation and the use of the CoA are transparent to the upper layer protocol, as far as the upper layer protocol is concerned, and the HoA is used for all communications.

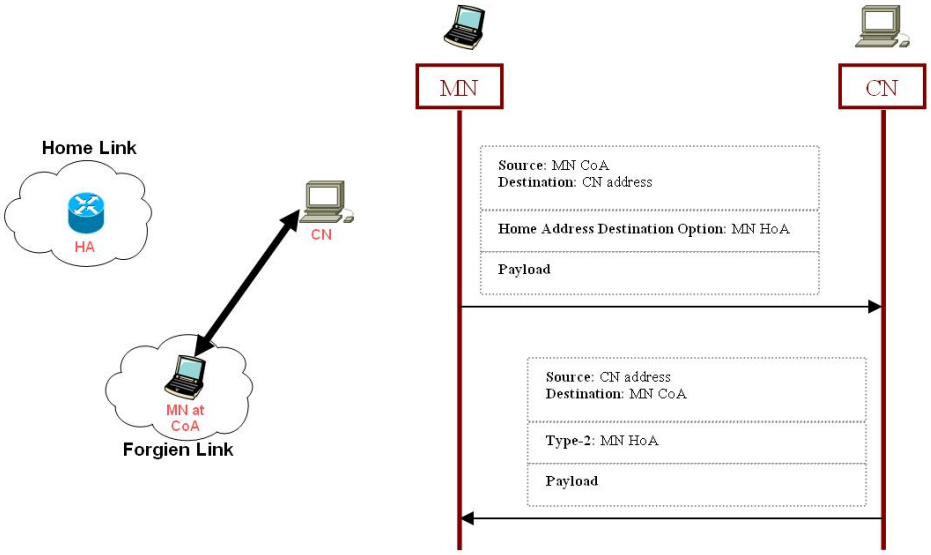

Figure 2.2: Route optimization communication

The MIPv6 protocol also defines a new mobility header called the 'Alternate Care-of Address'. This header may be used to hold the MN's CoA in BU messages sent by the MN. Normally, a BU message specifies the CoA in the source address field of the IPv6 header. However, the MN may specify a different CoA by including it in the 'Alternate Care-of Address' header. When the 'Alternate Care-of Address' header is included in the BU message, the recipient must use it

as the CoA for the binding rather than using the source address of the message as the CoA. This header is provided for use when the MN wishes to register a CoA that cannot be used as a topologically correct source address, when the MN wishes to deregister its CoA with a CN while it is away from home, or when the security mechanism used does not protect the IPv6 header.

The MN maintains a list called 'Binding Update List'. This list specifies all of the bindings that the MN has or is trying to establish with CNs. The binding with the MN's HA is stored in the list as well. The MN uses this list to determine whether packets for a specific CN should be sent directly to the CN or tunnelled via the HA. The list is also used to determine to whom the MN should send BU messages when roaming to another link. On the other hand, both the HA and the CN maintain a 'Binding Cache' of the accepted bindings received from other MNs. The Binding Cache at the CN allows it to send packets directly to the MN's CoA. The HA uses its Binding Cache to determine which MNs it is serving as a home agent.

2.2 Security and Efficiency of Binding Updates

The location management feature of the MIPv6 protocol makes it possible for an attacker to send a spoofed BU message to an HA and/or to a CN. By spoofing BUs, the attacker can launch different types of attacks including session hijacking, man-in the-middle (MITM), and denial-of-service (DoS) [9]. This section examines the security and efficiency of BUs used in MIPv6 protocol. Section 2.2.1 first presents security risks and attacks imposed by the introduction of the binding feature of the MIPv6 protocol. Section 2.2.2 then describes security services that should be provided to prevent such attacks, and the performance requirements that should be considered while providing these security services.

2.2.1 Security Risks and Attacks

Most of the attacks introduced by the binding feature of the MIPv6 protocol cause inappropriate bindings that misinform other entities (HAs and CNs) about the location of an MN, making them redirect traffic intended for the MN to a wrong destination. The attacks can be summarized as follows [5, 9]:

- **Malicious MNs flooding attacks.** A legitimate MN sends a spoofed BU message to an HA and/or a CN. In the spoofed message, the MN sets the home address to its own HoA and the care-of address to a victim node's address, falsely claiming that it has moved to the victim's location. As a result, the HA/CN would redirect all subsequent packets, which should otherwise be delivered to the MN itself, to the victim's address, flooding the victim with excessive unwanted data. This is shown in Figure 2.3.

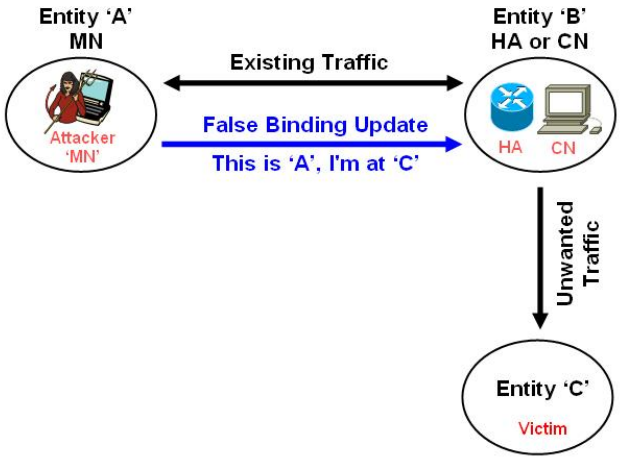

Figure 2.3: A malicious MN flooding attack

- **False Binding Update attacks.** An attacker impersonates another (mobile) node and sends a spoofed BU message. As a result, the attacker would redirect traffic destined for the original (mobile) node to itself (or to a third node). This is shown in Figure 2.4 and can be summarized as follows:

  - **Session hijacking attacks.** The attacker sends a spoofed BU, in which the home address is set to the (home) address of a victim (mobile) node, and the care-of address is set to the attacker's address. In this way, the attacker can hijack existing connections and launch DoS attacks against the victim (mobile) node as packets destined for the (mobile) node will be directed elsewhere. This is shown in Figure 2.4(a).

  - **Denial-of-Service (DoS) attacks.** The attacker sends a spoofed BU, in which the home address is set to the (home) address of a victim (mobile) node, and the care-of address is set to the address of a victim third node.

In this way, the attacker can launch DoS attacks against both of the victim (mobile) node and the victim third node. This is shown in Figure 2.4(b).

– **Man-In-The-Middle (MITM) attacks.** The attacker sends spoofed BUs to two CNs in order to set itself as a MITM between them. This is shown in Figure 2.4(c).

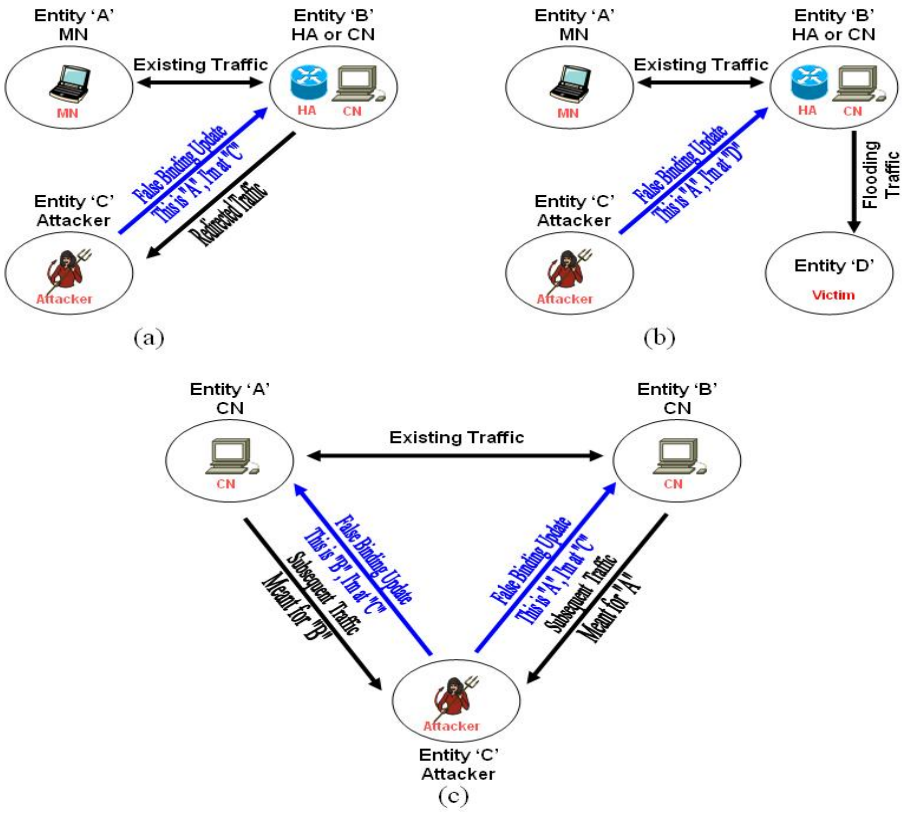

Figure 2.4: False Binding Update attacks

- **Return-to-home spoofing attacks.** An attacker claims to be an MN that is currently away from its home link. The attacker starts a communication with a CN and sends a spoofed BU in which the home address is set to the (home) address of a victim (mobile) node, and the care-of address is set to the attacker's address. The attacker next starts to download a heavy stream of data, such as video streaming, from the CN. The attacker then sends a spoofed BU to remove its Binding Cache entry at the CN, claiming that it has returned to the home link. As a result, the CN sends subsequent traffic to the victim's address, flooding the victim with excessive unwanted data.

- **Replay attacks.** An attacker replays a BU that an MN had sent earlier to an HA or to a CN. Consequently, the HA (or the CN) redirects subsequent traffic to the MN's old location. This causes DoS attacks to both of the MN and the node (if any) that is currently located at that location.

- **Resource exhaustion DoS attacks.** A security protocol that could be used to protect BUs may increase the participants' vulnerability to resource exhaustion DoS attacks. An attacker may exploit the protocol features to exhaust memory and/or computing resources of a CN. The attacker can flood the CN with BUs that cause it to perform computationally expensive cryptographic operations or to create a lot of states in its memory.

It is expected that the correspondent functionality of MIPv6 will be deployed in most IPv6 nodes. In addition, the addresses of MNs are indistinguishable from those of stationary ones. Thus, the above attacks are applicable to the whole Internet. The attacker can be anywhere on the Internet and all Internet nodes are potential targets. To summarise, both BUs exchanged between an MN and its HA and between an MN and its CN in MIPv6 protocol need protection. Otherwise, the MN and the CN are vulnerable to DoS, MITM, impersonation, and hijacking attacks. Third parties are also vulnerable to DoS attacks.

2.2.2 Security Services and Performance Requirements

In order to minimise and address the security risks and attacks discussed above, the following security services should be provided [5, 9]:

- **Authentication**

The authentication service provides assurance that a BU message has originated from the entity that owns the claimed home and care-of addresses. Authenticating a home address ensures that the entity is authorized to create a binding for that home address. In addition, authenticating a care-of address ensures that the entity is indeed located at that care-of address.

- **Integrity**

The integrity service ensures that the received BU message contains the same binding data as sent.

- **Freshness**

The freshness service prevents attackers from replaying a previously authenticated BU message.

- **Resource exhaustion DoS resistance**

The resource exhaustion DoS resistance protects CNs against memory and CPU exhaustion DoS attacks. A CN should set a limit on the amount of resources used for managing locations of MNs. The CN also should not retain any state about individual MNs until it receives an authentic BU from that MN, i.e. the honesty of the MN is proved. Furthermore, the CN should use computationally inexpensive cryptographic operations to authenticate the MN, or should delay expensive operations (if any) until the MN provides some assurance of its honesty.

In addition, the following performance requirements should also be considered while providing the above security services [24]:

- **Cost-effective operation at MN**

The MN's capabilities should be considered through minimising the operational load at the MN. The operational load includes:

  - Computational cost: the number of computationally expensive cryptographic operations performed by the MN.
  - Communication cost: the number and length of messages sent/received by the MN.

- **Support delay-sensitive application**

The delay introduced as the result of securing BUs should be considered. A longer delay may significantly impact delay-sensitive applications.

2.3 Internet Protocol Security (IPSec)

Internet Protocol Security (IPSec) is a standard framework for securing IP communications. IPSec is a set of protocols that provides security to IP and upper-layer protocols for secured exchange of IP packets [25, 26]. It provides two types of security algorithms: symmetric ciphers for encryption, and keyed one-way hash

functions for authentication and integrity. It can be used to provide data confidentiality, data integrity, data authenticity, and optional anti-replay protection.

IPSec supports two modes of operation: transport mode and tunnel mode. The transport mode protects the payload and upper-layer headers of each IP packet. In this mode, an IPSec header is inserted between the IP header and the upper-layer protocol header. On the other hand, the tunnel mode protects the entire IP packet. In this mode, the entire IP packet to be protected is encapsulated in a new IP packet and an IPSec header is inserted between the outer and inner IP headers.

IPSec has two basic protocols to provide security: IPSec Authentication Header (IPSec AH) [27] and IPSec Encapsulating Security Payload (IPSec ESP) [10]. The IPSec AH is used to protect the authenticity and integrity of an IP packet with a keyed cryptographic hash value. The proof of authenticity and integrity is based on the possession of a secret key that is used to calculate a MAC value for the IP packet. Figure 2.5 depicts the IPSec AH header format. The 'Next Header' field identifies the type of transport protocol used in the upper layer. The 'Payload Length' field specifies the length of AH header. The 'Reserved' field is reserved for future use and must be set to zero. The 'Security Parameter Index (SPI)' field, in combination with the destination IP address and security protocol (IPSec AH), uniquely identifies the Security Association (SA) for this datagram. The 'Sequence Number' field contains a monotonically increasing counter value (sequence number). Finally, the 'Authentication Data' field is a variable-length field that contains the Integrity Check Value (ICV) for the attached packet (including the AH header itself).

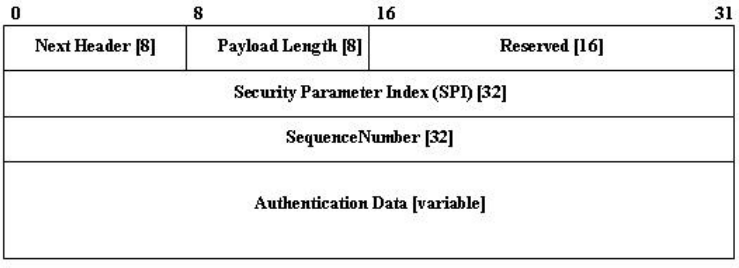

Figure 2.5: IPSec AH header format

Figure 2.6 depicts the coverage of the authentication protection for IPSec AH in transport mode and in tunnel mode. The ICV is computed first at the

transmitter by the use of a keyed one-way hash function that is also known to the receiver. Then, the ICV is recomputed at the receiver and compared to match the received value for authentication integrity. The ICV computation excludes values of the IP header that are subject to change during transmission such as time-to-live, flags, and header checksum.

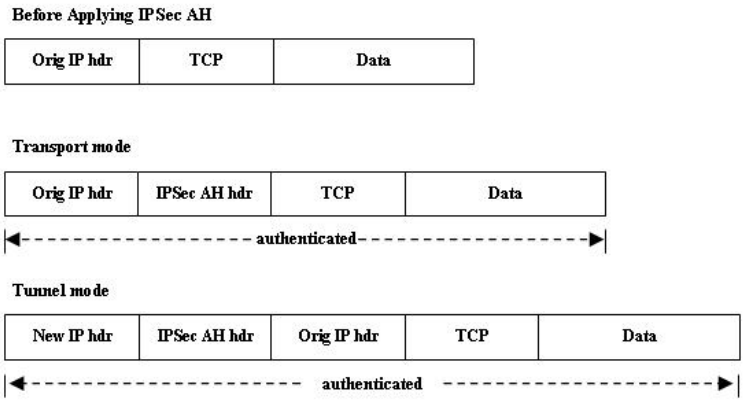

Figure 2.6: IPSec AH authentication protection

The IPSec ESP can provide confidentiality, integrity, authenticity, and optional anti-replay services to the IP packets it is protecting. Figure 2.7 depicts the IPSec ESP header and trailer format. The IPSec ESP header consists of 'Security Parameter Index (SPI)' and 'Sequence Number' fields that are exactly the same as in IPSec AH. The IPSec ESP trailer consists of three fields. The 'Padding' field is used to adjust the size of the plaintext (consisting of the Payload Data, Pad Length, and Next Header, as well as the Padding) to the size required by the encryption algorithm used. The 'Pad Length' field contains the number of pad bytes inserted by the encryption algorithm. The 'Next Header' field is the same as in IPSec AH. The figure also shows the 'Authentication Data' field, which contains the ICV computed over the packet and the ESP header and trailer (not including the authentication data itself). This field is optional and is included only if the authentication service is required.

Figure 2.8 depicts the protection coverage of authentication and encryption for IPSec ESP in transport mode and tunnel mode. The ESP header is inserted in exactly the same way as the AH header. The ESP trailer is inserted after the payload data and before the ESP authentication data. The ICV computation steps are the same as in IPSec AH.

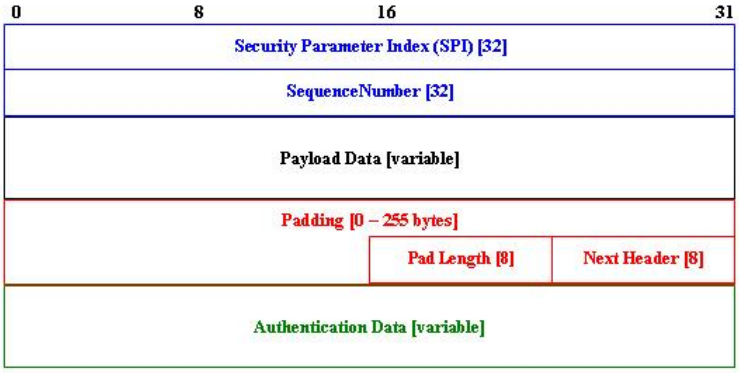

Figure 2.7: IPSec ESP header, trailer, and authentication data format

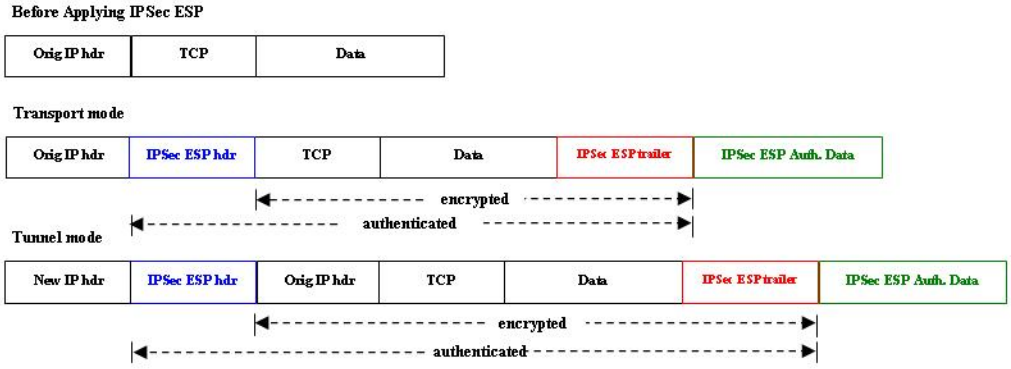

Figure 2.8: IPSec ESP authentication and encryption protection

IPSec uses Security Associations (SAs) to specify parameters controlling security operations. The SAs are represented through data records stored in the communicating entities that contain parameters such as encryption and authentication algorithms, secret keys, IPSec mode (transport or tunnel), IPSec protocol (AH or ESP), and so forth. SAs can be set up manually or by a security association and key-management protocol that performs an entity authentication. SAs are unidirectional, so two SAs have to be set up for secured bi-directional communication. IPSec can provide anti-replay protection only if dynamic keying is used. The Internet Key Exchange (IKE) is the most widely adopted protocol for SA negotiation and keying material provisioning [28, 29]. It uses either pre-shared secrets or public keys for authenticating IPSec, negotiating security services and generating shared keys.

2.4 Standard Home Registration

Home registration is a process that is initialized by an MN to inform an HA about the MN's current location. It is performed through the use of BU and BA mobility messages. When the MN roams away from the home link, it initiates the process to request the HA to serve as the home agent for its HoA by registering its current CoA with the HA. The process also allows the MN to update the HA with the MN's new CoA after roaming to a new foreign link, to extend the lifetime of a registration that is about to expire, or to delete a registration after returning to the home link.

Security provision for home registrations is a fundamental part of the MIPv6 protocol to avoid the various attacks stated in Section 2.2.1. The MN uses services of the HA and they belong to the same administrative domain. Thus, it is assumed that the MN and the HA know each other in advance and can use a pre-shared secret (or other authentication infrastructure such as certificates) to establish a bidirectional IPSec Security Association (SA), which can be then used in protecting home registrations. As a result, the base specification of the MIPv6 protocol [5] uses IPSec Encapsulating Security Payload (ESP) [10] and sequence numbers to protect control traffic between MNs and HAs [11]. The control traffic includes BU and BA mobility messages, and is carried by the Mobility Header protocol in IPv6 [21]. A home registration process is shown in Figure 2.9 and can be summarised as follows [5].

An MN initiates a home registration by sending a BU mobility message to an HA. The BU message contains the MN's HoA, the MN's current CoA, a sequence number, and a binding lifetime request. The MN must include its current CoA in the new 'Alternate Care-of Address' mobility header even if the CoA appears as the source address of the BU message. That is because IPSec ESP in transport mode does not protect the IPv6 header. To protect against replay attacks, the MN sets the sequence number to a value that is greater than the value sent in the previous BU to this HA (if any). In addition, if the purpose of the BU is to delete the MN's binding entry at the HA, the MN will set the CoA equal to its HoA and the binding lifetime request to 'zero'. Finally, if the MN does not receive a valid matching BA message within a retransmission interval of one second, the MN will resend the BU message to the HA. The retransmission interval is doubled upon each retransmission, until either a matched response is received or the retransmission interval reaches a maximum retransmission interval

of 32 seconds. At this point, if there is only one HA in the home link, the MN will continue to periodically retransmit the BU message at this slower rate until acknowledged. Otherwise, the MN will reinitiate the home registration by instead trying a different HA.

When the HA receives the BU message, it authorizes the sender of the message, and verifies the authenticity, integrity, and freshness of the message. The HA first authorizes the sender of the BU to ensure that it is actually authorized to create a binding for the claimed HoA. This authorization is meant to prevent an MN from using its SA to send a BU on behalf of another MN that is using the same HA and is provided by using the claimed HoA in identifying the IPSec SA that must be used. The HA next uses the identified IPSec SA to verify the authenticity and integrity of the BU message. The HA then uses the sequence number enclosed in the BU to check the freshness of the message, i.e. the HA confirms that the given sequence number is greater than the sequence number carried in the last valid BU from this HoA (if any).

If all verifications are positive, i.e. the received BU message is valid, the HA will decide whether it needs to perform a Duplicate Address Detection (DAD) test [20] for the MN's HoA. The HA runs the DAD test for a given HoA only if it does not have an existing Binding Cache entry for that HoA. The DAD test is done to ensure that no other node on the home link is using the MN's HoA when the BU arrives.

If the DAD test succeeds (or if there is no need to perform the DAD test), the HA will create, update, or delete a Binding Cache entry for the received HoA. If the BU deletes an existing binding entry, the HA will delete any existing entry in its Binding Cache for this HoA and will stop intercepting packets on the home link that are addressed to the HoA. In addition, the HA will return an accepted BA message to the MN acknowledging the deletion of the binding. If the BU creates a new binding entry or updates an existing one, the HA will store the CoA and sequence number carried in the BU message and will grant a binding lifetime in its Binding Cache entry for the given HoA. The granted binding lifetime is set by the HA depending on the binding lifetime requested in the BU and on the remaining valid lifetime for the subnet prefix of the given HoA. The HA determines the remaining valid lifetime for this prefix based on its own Prefix List entry [18]. The HA then sends an accepted BA message to the MN acknowledging the binding of the CoA. The BA message contains the

MN's HoA, the granted binding lifetime, a sequence number that is equal to the sequence number enclosed in the BU, and optionally, a binding refresh advice. The binding refresh advice, which must be shorter than the granted binding lifetime, may be included in the BA message suggesting the MN refresh its home registration at this shorter interval.

On the other hand, if any of the above verifications is negative or if the DAD test fails, the HA will reject the binding and will reply with a BA message in which the status field is set to a value indicating the rejection of the binding and the cause of the rejection.

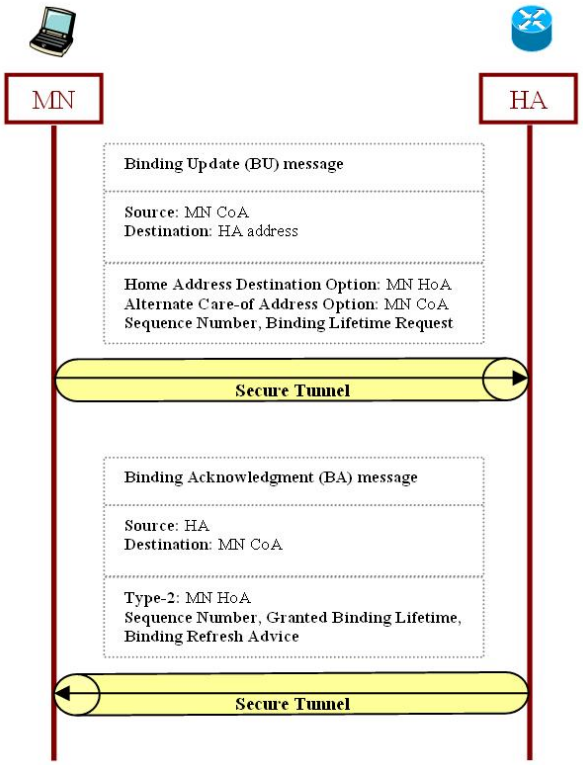

Figure 2.9: Home registration

Upon receiving the BA message from the HA, the MN identifies the IPSec SA that must be used. The MN next verifies the authenticity and integrity of the BA message. The MN then verifies that the sequence number enclosed in the BA matches the sequence number sent by the MN to this HA as maintained in the corresponding Binding Update List entry. If any of the above verifications is negative, the MN will discard the BA message without any further action. Otherwise, the MN will examine the status field of the message. If the status

field indicates that the BU was accepted, then the MN will stop retransmitting the BU and will update the corresponding entry in its Binding Update List to indicate that the BU has been acknowledged. In addition, if the given granted binding lifetime is less than the requested binding lifetime sent in the BU being acknowledged, the MN will subtract the difference between these two lifetime values from the remaining binding lifetime. On the other hand, if the status field indicates that the BU was rejected, then the MN will take steps to fix the error and retransmit the BU with a new sequence number.

The use of the IPSec ESP protocol and sequence numbers can partially protect home registrations against the attacks mentioned in Section 2.2.1. Specifically, it can prevent attackers from sending spoofed or replayed BU messages. In addition, it can prevent a legitimate MN from sending a BU on behalf of another MN that is using the same HA. However, as it cannot authenticate the claimed CoA, it cannot prevent a legitimate MN from lying about its current location by providing a false CoA and thus causing the HA to redirect traffic to this address. This weakness was previously ignored as the MIPv6 protocol assumed that an MN could only register one CoA, and if the MN cheated the HA with a fake CoA then the MN would lose the communication with the HA, thus losing its mobility. However, recent research [30, 31] has suggested that MNs could be multi-homed. In such a scheme, a multi-homed MN can (1) have multiple HoAs connected to different home links, and (2) bind a HoA to multiple CoAs. As a result, the MN may cheat one or more of its HAs with victim addresses while maintaining mobility through other HAs. If this cheating is successful, it is possible for the cheated HAs to flood the victims located at the fake CoAs with unwanted packets. Therefore, the home registration must enable HAs to verify the authenticity of the claimed CoAs to prevent these attacks.

2.5 Standard Correspondent Registration

Correspondent registration is a process that is initialized by an MN to notify a CN about the MN's current location while away from home link. It is performed through the use of a return routability (RR) procedure and a registration, an exchange of a BU message and an optional BA message. When the MN receives a tunnelled packet from its HA, it infers that the CN that sent the original packet is unaware of the MN's current location. Therefore, the MN may initiate

the process to inform the CN of its location. The process also allows the MN to update and/or delete binding information in the CN.

The purpose of the RR procedure is to assure the CN that the MN is able to receive messages sent to the claimed HoA as well as the claimed CoA. It is also used to establish a session key between the MN and the CN, which is used to protect the subsequent registration. The RR procedure consists of two tests: a home address test and a care-of address test. The MN can initiate the two tests simultaneously by sending a Home Test Init (HoTI) message and a Care-of Test Init (CoTI) message to the CN. The CoTI message is sent directly to the CN, while the HoTI message is 'reversed tunnelled' from the MN to the HA, which then forwards it to the CN. The CN responds to the HoTI message by sending a Home Test (HoT) message containing a secret home keygen token. The HoT message is addressed to the MN's HoA and is forwarded by the HA to the MN's current CoA. In addition, the CN responds to the CoTI message by sending a Care-of Test (CoT) message containing a secret care-of keygen token. The CoT message is routed directly to the MN's CoA. If the MN really owns both addresses, it will receive the two tokens from which it can compute a session key. The session key is used by the MN and the CN to protect subsequent BU and BA messages. The correspondent registration for updating a CN with an MN's location while away from the home link is shown in Figure 2.10 and can be summarised as follows.

• **Steps 1 and 2:**
After the MN has completed a home registration, it can initiate a correspondent registration with a specific CN. The MN first creates a Binding Update List entry for the CN and sets it in a 'Route_Pending' state. The MN then sends a HoTI message to the CN (via the home link) requesting a home keygen token, where HoTI = {Src=HoA, Des=CN, $Cookie_1$}. At the same time, the MN also sends a CoTI message directly to the CN requesting a care-of keygen token, where CoTI = {Src=CoA, Des=CN, $Cookie_2$}. The cookies, $Cookie_1$ and $Cookie_2$, are 64-bit random numbers generated by the MN. The MN stores the cookies at the list entry and compares them later with the responses from that CN to verify that responses match with requests.

• **Steps 3 and 4:**
Upon receiving a HoTI message, the CN generates a home keygen token ($Token_1$) and sends a HoT message to the MN (via the home link). Specifically, the CN

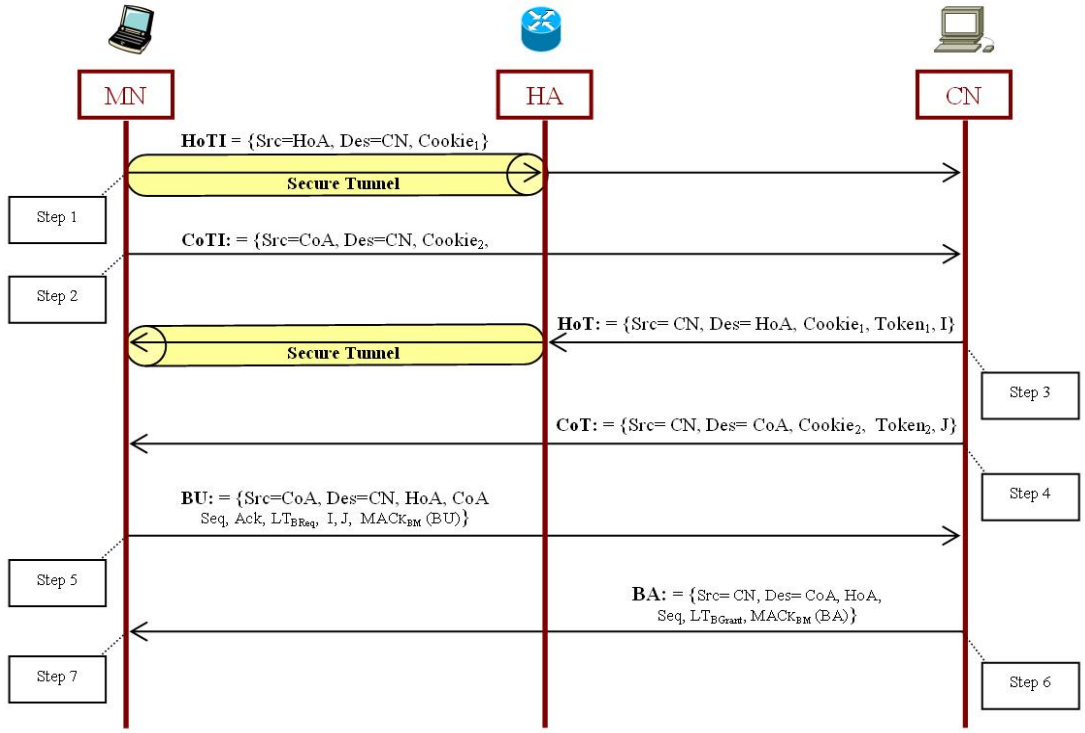

Figure 2.10: Standard correspondent registration - including the RR procedure

first selects a home nonce N_I that is identified by a home nonce index I. The CN next generates the home keygen token, i.e. $Token_1$ = First (64, HMAC_SHA1 (K_{CN}, (HoA || N_I || 0))), where || denotes string concatenation; K_{CN} is a secret random value generated and only known by the CN (CN's key); '0' is a single zero octet that is used to distinguish home keygen and care-of keygen tokens from each other; HMAC_SHA1 ( ) denotes a keyed hashing MAC scheme using hash function SHA1; and First (n, M) denotes the first n bits of message M. The CN then sends out the HoT message to the MN's HoA, i.e. HoT = {Src=CN, Des=HoA, $Cookie_1$, $Token_1$, I}. The HA intercepts the HoT message and forwards it to the MN's registered CoA via a secure tunnel.

Similarly, upon receiving a CoTI message, the CN first generates a care-of keygen token ($Token_2$), i.e. $Token_2$ = First (64, HMAC_SHA1 (K_{CN}, (CoA || N_J || 1))), where N_J is a care-of nonce that is selected by the CN and identified by a care-of nonce index J. The home and care-of nonces, i.e. N_I and N_J, may be the same. The CN then sends out a CoT message directly to the MN, i.e. CoT = {Src=CN, Des=CoA, $Cookie_2$, $Token_2$, J}. The home and care-of nonce

indices, i.e. I and J, will be returned later by the MN to remind the CN of which nonce value is used in generating $Token_1$ and $Token_2$.

The K_{CN} is a 160-bit random number generated by the CN. It is called the 'node key' and is used to produce the keygen tokens sent to the MNs. The K_{CN} is not shared with any other entity, i.e. it is known only to the CN. Generating tokens using K_{CN} allows the CN to verify that the keygen tokens used by the MN in authorizing a BU are indeed its own without forcing the CN to remember a list of all tokens it has handed out. The K_{CN} can either be a fixed value or regularly updated. An update of K_{CN} can be done at the same time as an update of a nonce, e.g. N_I, so that the index I identifies both the nonce and the key. A CN can generate a fresh K_{CN} at any time; this avoids the need for secure persistent key storage for K_{CN}.

- **Step 5:**

Upon receiving a HoT or a CoT message, the MN uses the CN's address, i.e. the source address of the message, as an index to search its Binding Update List. If a list entry is found with a 'Route_Pending' status, the MN will verify that the cookie enclosed in the message matches the cookie sent by the MN to this CN as maintained in the list entry. If no entry is found (or if the matched entry is not in a 'Route_Pending' status), the MN will discard the received message without any further action. Otherwise, the MN will record the given keygen token and nonce index in the list entry and wait for the second message. When the MN receives both the HoT and CoT messages, it hashes the two tokens together to form a 160-bit binding management key K_{BM}, i.e. K_{BM} = SHA1 ($Token_1$ || $Token_2$). K_{BM} is the shared secret key established between the MN and the CN via the RR procedure. The MN then sends a BU message to the CN, i.e. BU = {Src=CoA, Des=CN, HoA, CoA, Seq, Ack, LT_{BReq}, I, J, $MAC_{K_{BM}}$(BU)}, where Seq is a sequence number that is greater than the sequence number sent in the previous BU to this CN (if any); Ack is an acknowledge bit that may be set by the MN to request a BA message returned by the CN; LT_{BReq} is a binding lifetime request; I and J are the home and care-of nonce indices received previously from the CN; and $MAC_{K_{BM}}$(BU) is a keyed hash value, i.e. $MAC_{K_{BM}}$(BU) = First (96, HMAC_SHA1 (K_{BM}, (CoA || CN || BU))), where BU indicates the binding update message itself. In addition, the MN will change the status of the list entry to indicate the completion of the RR procedure. Specifically, if the MN sets the Ack bit, then it will change the status to a 'Binding_Pending' indicating that

it is waiting for an acknowledgement; otherwise, it will change the status to a 'Binding_Complete' indicating that the correspondent registration is complete.

- **Step 6:**

When the CN receives a BU message, it first searches its Binding Cache for an entry indexed by the HoA. If no entry exists, the CN will accept any sequence number value enclosed in the received BU message. Otherwise, the CN verifies that the given sequence number is greater than the sequence number received in the last valid BU from this HoA. If the verification fails, then the CN will send back a rejected BA message that contains the last accepted sequence number. If the verification succeeds, the CN will use the home and care-of nonce indices enclosed in the BU message (i.e. I and J) to regenerate the two tokens (i.e. $Token_1$ and $Token_2$). The CN then hashes the two tokens together to form K_{BM} and verifies the validity of the received BU message, as shown in Figure 2.11. If the verification succeeds, the CN will update the entry with the CoA and sequence number carried in the BU message. In addition, the CN will grant a binding lifetime in the entry based on the binding lifetime requested by the MN in the BU message. Finally, the CN checks the Ack bit given in the BU message; if it is set (i.e. the MN requests an acknowledgement), the CN will send back an accepted BA message, i.e. BA = {Src= CN, Des= CoA, HoA, Seq, LT_{BGrant}, $MAC_{K_{BM}}$(BA)}, where Seq is a sequence number that is equal to the sequence number given in the BU being acknowledged; LT_{BGrant} is the granted binding lifetime; and $MAC_{K_{BM}}$(BA) = First (96, HMAC_SHA1 (K_{BM}, (CN || CoA || BA)), where BA indicates the binding acknowledgement message itself.

The CN may decide to refuse the BU for many reasons, such as when it does not have sufficient resources, when one or both of the given nonce indices are not recognizable, or when the validity verification fails. If this is the case, the CN will return a rejected BA message in which the status field is set to a value indicating the cause of the rejection.

- **Step 7:**

Upon receiving a BA message, the MN uses the CN's address, i.e. the source address of the message, as an index to search its Binding Update List. If a list entry is found, the MN will verify that the sequence number enclosed in the message matches the last sequence number sent by the MN to this CN as maintained in the list entry. If the verification succeeds, the MN will use K_{BM}

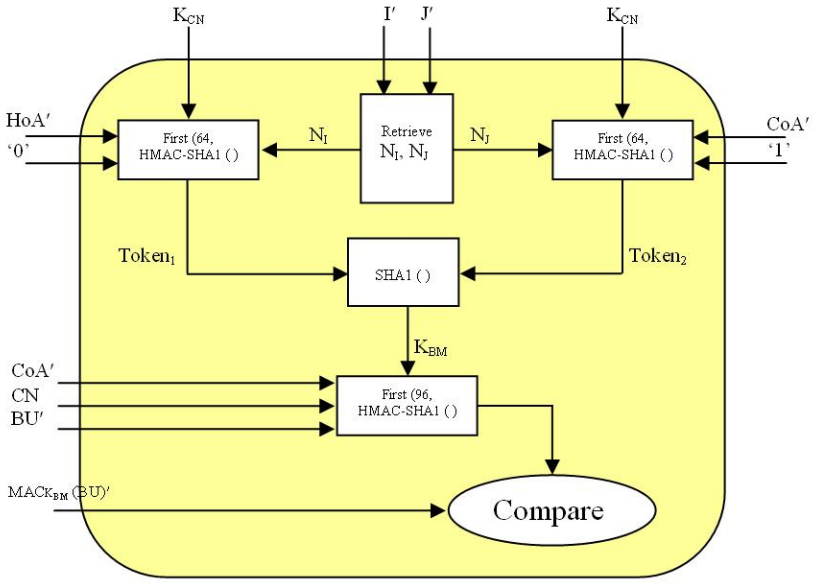

Figure 2.11: Verification of a BU in standard correspondent registration

to verify the validity of the received BA message as shown in Figure 2.12. If any of the above verifications is negative or if no list entry exists for that CN, the MN will discard the received message without any further action. Otherwise, the MN will examine the status field of the message. If the status field indicates that the BU was accepted, the MN will update the status of the list entry to 'Binding_Complete', indicating that the BU has been acknowledged. In addition, the MN adjusts the remaining binding lifetime depending on the given granted binding lifetime. On the other hand, if the status field indicates that the BU was rejected, the MN will examine the cause of the rejection and take steps to fix the error and retransmit the BU with a sequence number value greater than that used for the previous transmission of that BU. However, if the rejection indicates that the CN cannot provide route optimization, the MN will stop trying to initiate route optimization and will revert back to the use of bidirectional tunnelling.

The MN is responsible for retransmissions in the standard correspondent registrations [5]. If the MN sends a HoTI message or a CoTI message to the CN and does not receive a matching response within a retransmission interval, the MN will resend the corresponding message with a new cookie value. Similarly, if the MN sends a BU message in which the Ack bit is set and does not receive a matching BA message within a retransmission interval, the MN will resend the BU message with a new sequence number value greater than that used for the

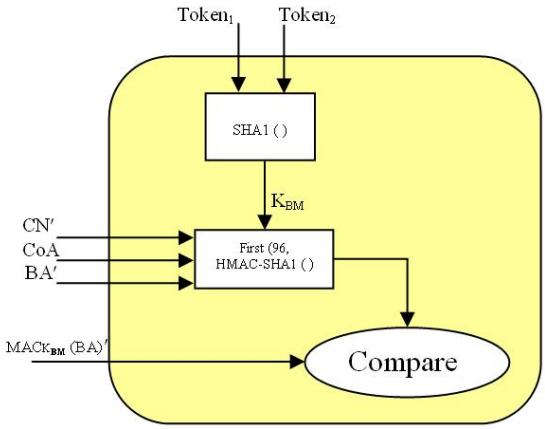

Figure 2.12: Verification of a BA in standard correspondent registration

previous transmission of this BU. In all cases, the initial retransmission interval is one second and is doubled upon each retransmission, until either a matched response is received or the retransmission interval reaches a maximum retransmission interval of 32 seconds. The MN may continue to send HoTI, CoTI, and BU messages at this slower rate indefinitely [5].

If the correspondent registration is meant to delete a previously established binding, then the CoTI and CoT messages are omitted from the RR procedure. The MN initiates the procedure by sending a HoTI message, and the CN responds by returning a HoT message that contains a secret home keygen token, i.e. $Token_1$. The session key is generated through hashing the token, i.e. K_{BM} = SHA1 ($Token_1$). Furthermore, in the BU message, the MN sets the CoA equal to its HoA and the binding lifetime request to 'zero'.

If the CN is also mobile, then all mobility messages (i.e. HoTI, CoTI, and BU messages) are sent to the CN's home address. In this case, if the mobile CN is currently located in its home link, it will receive the messages directly. Otherwise, the CN's home agent will intercept and forward the messages to the CN's current care-of address. In addition, while the mobile CN is away from its home link, it sends all of its responses (i.e. HoT, CoT, and BA messages) via its home agent (reverse tunnelling) as shown in Figure 2.13. Apart from these, the RR procedure for the stationary CN and the mobile CN cases are alike.

The RR procedure protects correspondent registrations from all attacks, except where on-path attackers are concerned [5]. It protects CNs against replayed

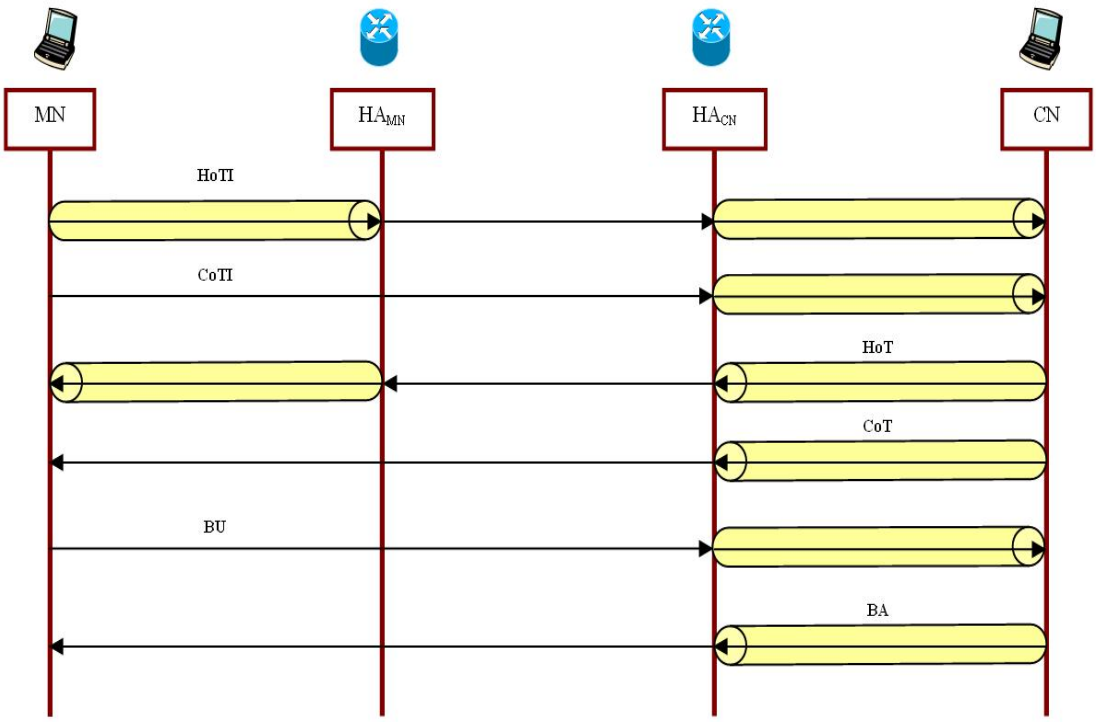

Figure 2.13: Standard correspondent registration - mobile to mobile

BUs through the use of the sequence number; an attacker cannot replay a previously authenticated BU message due to the sequence number included in the message. Also, the attacker cannot modify a BU message since the MAC verification would fail after such a modification. In addition, the RR procedure protects CNs against resource exhaustion DoS attacks. Specifically, the CN does not retain any state about individual MNs until it receives an authentic BU from that MN. Furthermore, it does not maintain the value for the K_{BM}, instead it forms the K_{BM} based on its secret key (K_{CN}), the given nonce indices, and the MN's addresses. Finally, as the RR procedure is dependent on the use of symmetric cryptography, it requires very little processing at the CN and at the MN.

The RR procedure increases the registration delay because the MN must wait for both address tests to conclude before it can register its CoA. As a result, a registration consumes about 1.5 round-trip times between an MN and its CN. The delay can be higher than 1.5 round-trip due to the home address test being relayed through the home link. In addition, the RR procedure requires the exchange of an extra four messages per registration, which increases signalling overhead.

Considering the on-path attackers, the RR procedure fails to provide sufficient protection for correspondent registrations. The RR procedure only limits the potential attackers from any node in the Internet to those that are on the route between the CN and the MN's home link. Attackers that can capture both the HoT and CoT messages can obtain the two tokens. Hence, they can hash the two tokens to form the session key and send a fake BU on behalf of the MN.

To limit the on-path attacks, the MIPv6 protocol requires the mobility messages tunnelled through the HA (i.e. the HoTI and HoT messages) during the RR procedure to be encrypted. This prevents attackers located on the MN's current foreign link from capturing the two tokens and launching attacks based on it. Furthermore, the MIPv6 protocol requires the MN to periodically (at most every seven minutes) rerun the RR procedure even with the absence of IP connectively change. This prevents attackers from launching attacks after moving away, i.e. attackers must be present on the path between the CN and the MN's home link to be able to launch the attack.

Based on the above discussion, we conclude that the RR procedure is inefficient as it fails to protect against on-path attacks; it requires the exchange of an extra four messages; it requires about 1.5 round-trip times between an MN and its CN; and it needs to be re-executed every few minutes. In other words, the RR procedure does not address the security problems efficiently and causes high registration delay as well as signalling overhead.

2.6 Chapter Summary

This chapter has presented an overview of the Mobile IPv6 protocol. Based upon the location management feature of the protocol, the security threats and attacks have been analysed, and the security services and performance requirements needed to address these attacks have been outlined. Finally, the chapter studied the standard home and correspondent registrations and highlighted their strengths and limitations in providing the required security services.

Before describing our proposed solutions for securing home and correspondent registrations in the MIPv6 protocol, the next chapter will investigate and discuss the existing correspondent registration security solutions.

Chapter 3

A Survey of Correspondent Registration Protocols

This chapter surveys existing protocols for protecting correspondent registrations in an attempt to identify their strengths and weaknesses with respect to the security risks and attacks mentioned in Section 2.2.1, as well as the security services and performance requirements mentioned in Section 2.2.2. The main goal of these correspondent registration protocols is to reduce registration delays and/or signalling overheads caused by the RR procedure (see Section 2.5) in a secure manner. Most of these protocols do not require frequent computation of fresh keys, thus reducing the number of redundant signalling messages generated by the RR procedure. They also have built-in security measures to offer additional security benefits over the RR procedure. The protocols can largely be divided into two classes. The first class is referred to as the 'infrastructure-less correspondent registration protocols'. The protocols in this class are suited to cases where participants do not have prior relationships among them. The second class of protocols relies on security infrastructure support and is, therefore, referred to as 'infrastructure-based correspondent registration protocols'.

Before surveying the existing correspondent registration protocols, we first present an overview of cryptographically generated IPv6 addresses.

3.1 Cryptographically Generated Addresses

The concept of cryptographically generated addresses (CGAs) was proposed to prevent stealing and spoofing of existing IPv6 addresses [32]. The basic idea is

to create a part of the IPv6 address from the address owner's public key, i.e. to bind the IPv6 address of an entity to the entity's public key [33, 34, 35].

A 128-bit IPv6 address consists of a 64-bit subnet prefix, which is used for routing, and a 64-bit interface identifier, which identifies a specific node in a network [36]. The interface identifier is normally derived from the node's link-layer address [36], but actually, almost any value is valid. Only two bits of the interface identifier have a special meaning, the 'U/L' bit (Universal/Local bit) and the 'I/G' bit (Individual/Group bit), which means that the remaining 62-bits can be generated by any method providing the resulting interface identifier is unique in the network.

A CGA-based address is an IPv6 address for which the interface identifier part is generated using a cryptographic one-way hash function that takes the address owner's public key and some auxiliary parameters as its input. The auxiliary parameters are a 128-bit random number, called a modifier, an eight-bit number, called a collision count, and a 64-bit subnet prefix of the CGA-based address [33]. The address owner can protect a message sent from the address by digitally signing it with the corresponding private key and sending the public key and the auxiliary parameters along with the signed message [33]. Upon receipt of the signed message, the recipient verifies the binding between the public key and the address by re-computing and comparing the hash value with the interface identifier part of the address. In addition, it authenticates the address by verifying the signature.

The early proposals of CGA suffered from "the small number of bits of the address to accommodate the result of the hash function", i.e. at most, 62 bits of the address can be used for the hash [32, 35]. If an attacker wished to impersonate a given CGA-based address, he/she would only need to attempt about 2^{61} (i.e. approximately $2.3 * 10^{18}$) tries to find an alternative public key that hashed to this address. This would leave open the possibility of brute-force attacks during the lifetime of the IPv6 protocol [37]. The shortage in the hash length problem is solved by Aura in [33] by using hash extensions.

The hash extension method [33] embeds a security parameter (Sec) into the CGA-based address. The Sec is a three-bit unsigned integer encoded in the three leftmost bits of the interface identifier that determines the strength of the address against brute-force attacks. This leaves just 59 bits for the actual hash output. The hash extension method applies a second hash function until the leftmost

16*Sec bits of the hash value are zero. This increases the cost of both address generation and brute-force attacks by the same factor while keeps the cost of address verification constant. In this way, the attacker would need to attempt about $2^{((59+16*Sec)-1)}$ tries to be able to bind its public key to a given CGA-based address.

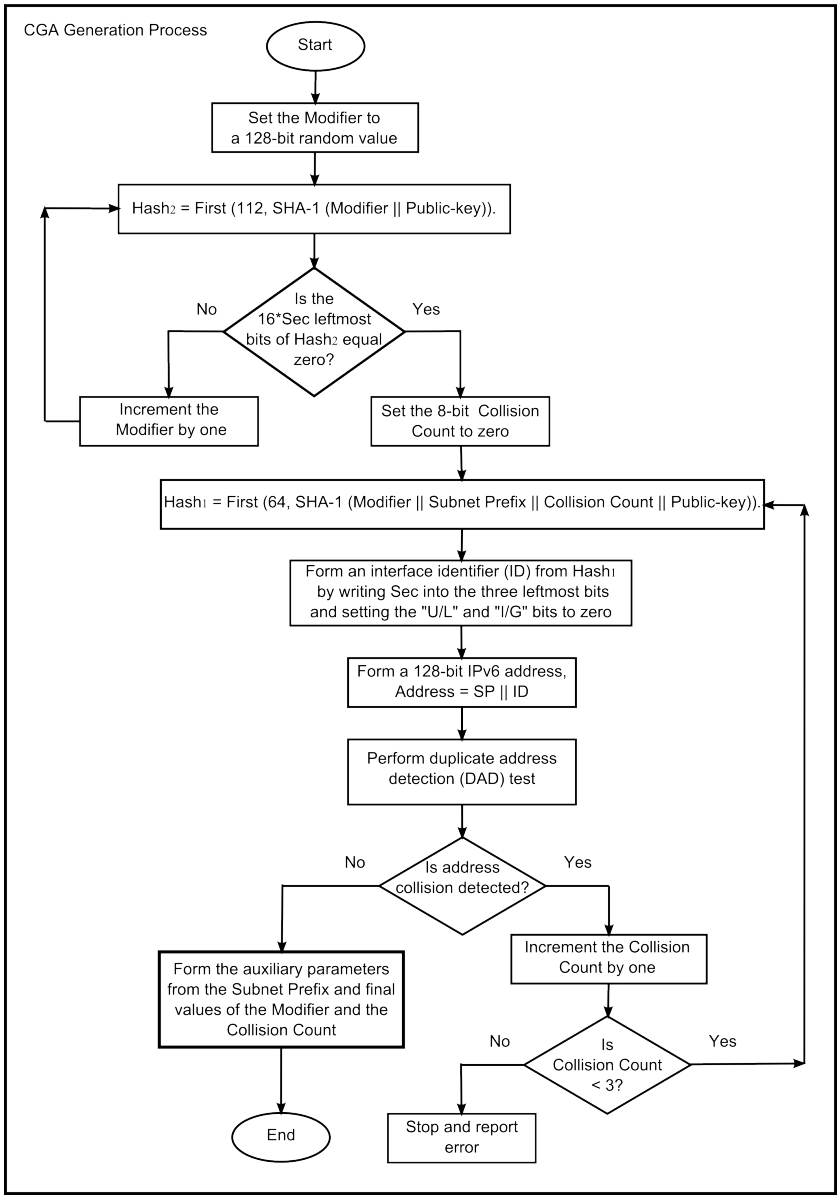

Figure 3.1: CGA-based address generation algorithm

The hash extension method theoretically solved the main weakness of the CGA-based method, especially when the address was generated offline by an

entity that had sufficient computing power. However, for Sec values greater than 0, the second hash function is nondeterministic, i.e. not guaranteed to terminate after a certain number of iterations, and when the Sec value equals 0, the cost of brute-force attacks is decreased to about 2^{58}.

A detailed process of generating and verifying a CGA-based address is given in [33] and can be illustrated in Figures 3.1 and 3.2.

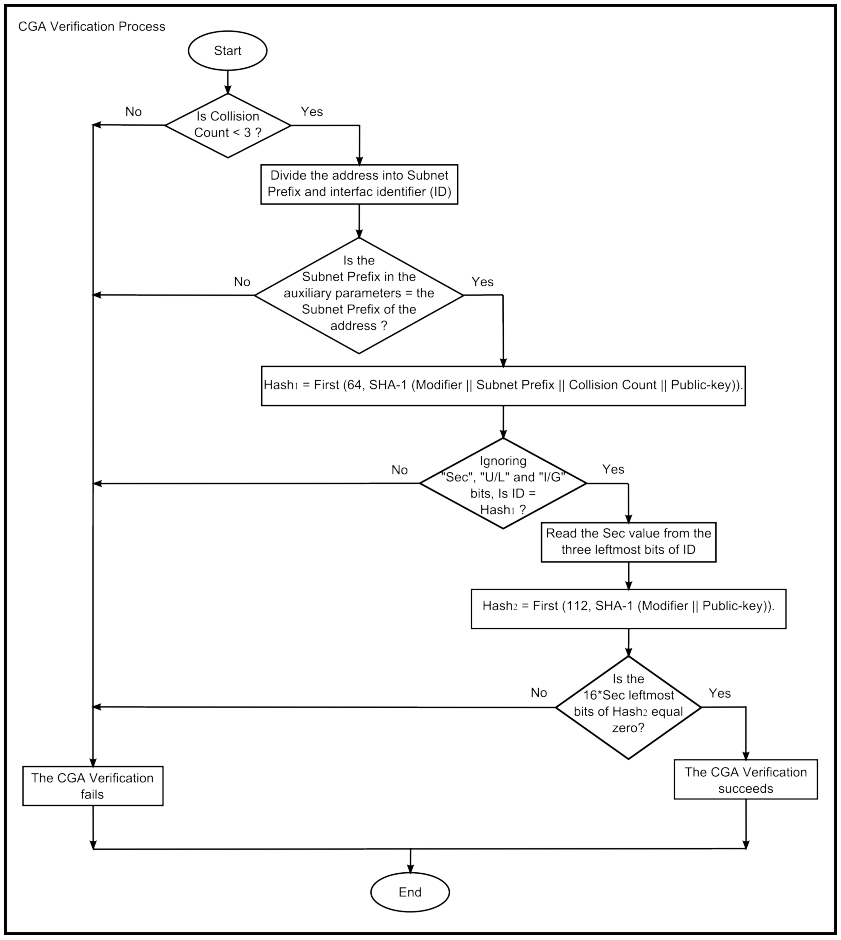

Figure 3.2: CGA-based address verification algorithm

The CGA-based technique provides public-key authentication for IPv6 addresses without using trusted third parties or PKI to convince others that the addresses are used by the owner of the public keys. As a result, the CGA-based technique suffers from the following limitations [38]. Firstly, it does not guarantee the owner's reachability at the address, i.e. an attacker can use its own public key to cryptographically generate a non-used address with a subnet prefix from a victim network. Secondly, although it can effectively stop attackers from

impersonating valid IPv6 addresses to launch attacks, it cannot thwart attacks on an entire network by redirecting data to a non-used address. Thirdly, it relies on public-key operations (in particular, digital signatures), which require heavy computations both to compute and verify the signature. Thus, it could expose the participants to denial-of-service attacks, especially when the participant is a mobile device with limited computational power or when the participant needs to verify digital signatures for a large number of peers at the same time. Finally, as a message needs to carry the address owner's public key and signature as well as auxiliary parameter values that are used to generate the address cryptographically, there is a certain amount of overhead incurred to the bandwidth consumption.

3.2 Infrastructure-less Correspondent Registration Protocols

The infrastructure-less correspondent registration protocols are proposed for use on a global basis between MNs and CNs belonging to different administrative domains. These protocols have a wide applicability and can be used generally on the Internet; they enable CNs to authenticate MNs without requiring any security infrastructure support.

3.2.1 Early Binding Update (EBU) Protocol

The Early Binding Update (EBU) protocol [39] is proposed to reduce the high registration delay caused by the RR procedure. It enhances the RR procedure by moving home address and care-of address tests to a handover phase where they do not impact the registration delay. The home address test occurs while the MN can still use its old CoA (i.e. performed before the handover), and the care-of address test runs in parallel with data transfer to and from the new CoA (i.e. done after the handover). It also utilises a Credit-Based Authorisation technique [39, 40] to limit the amount of data that the CN can send to the new CoA while concurrently running the care-of address test. This technique limits the data a CN can send to an MN's unconfirmed CoA by the data recently sent to or received from that MN. In this technique, the CN grants the MN credit for packets it sends to or receives from a confirmed CoA. When the MN sends

an early binding update message for registering a new CoA, subsequent packets consume the credit that the MN has collected up to then. The EBU protocol is shown in Figure 3.3 and is detailed in Appendix B.

Compared to the RR procedure, the EBU protocol reduces registration delay of correspondent registrations to about one one-way time between the MN and the CN. However, there is a cost for this reduction in delay. First, one or two additional messages are required per correspondent registration. Secondly, the MN may need to run the home address test periodically, which could increase the signalling overhead. Thirdly, complexity at the CN is increased for implementing the Credit-Based Authorisation technique. In addition to these costs, the EBU protocol suffers from the same on-path attacks applicable to the RR procedure.

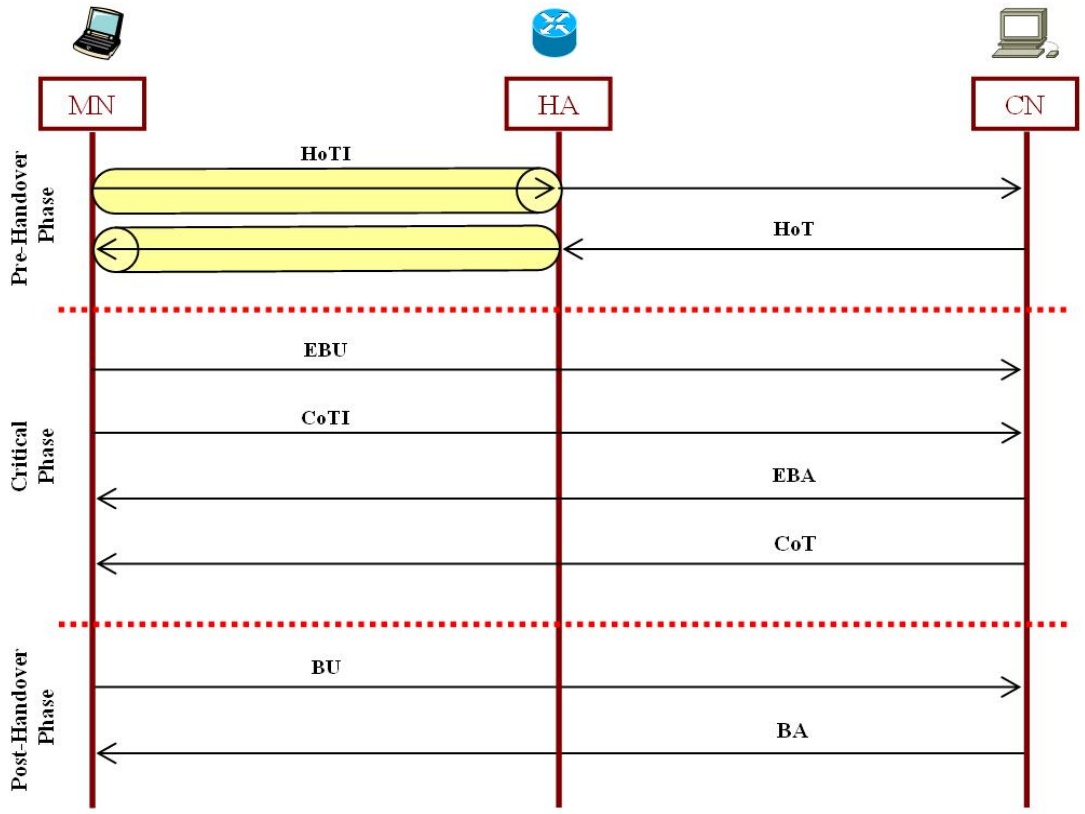

Figure 3.3: A correspondent registration protected by the EBU protocol

3.2.2 Purpose-Built Key (PBK) Protocol

The Purpose-Built Key (PBK) protocol [41] aims to authenticate the initiator of a network communication where the actual identity of the initiator is not important, but successive packets must come from the same initiator. Therefore, the protocol is suitable to use in correspondent registrations where a CN needs to be assured that a correspondent registration is actually coming from the same MN that initiates the communication. The PBK protocol is detailed in Appendix B.

The PBK protocol requires four messages during any correspondent registration, which reduces signalling overhead compared to the RR procedure. However, it cannot improve registration delay as it requires about 1.5 round-trip times between the MN and the CN. It also increases the CN's vulnerabilities to resource exhaustion DoS attacks; the CN is required to create a state and to perform two digital signature verifications during a protocol execution. The protocol is also vulnerable to return-to-home spoofing attacks as it cannot authenticate the claimed HoA. It is also vulnerable to MITM attacks affecting its initialisation. An attacker needs to be on the path during the initialisation phase to be able to intercept the hash of the public key, i.e. PBID, and send the hash of a different key. Finally, the protocol requires each of the MN and the CN to perform two public key operations per correspondent registration.

3.2.3 Child-proof Authentication for MIPv6 (CAM) Protocol

The Child-proof Authentication for MIPv6 (CAM) protocol [32] aims to authenticate correspondent registrations in Mobile IPv6 networks. In this protocol, an MN has a self-generated public/private key pair, and it configures its HoA as a CGA-based address (see Section 3.1).

The MN runs a correspondent registration by sending just one BU message to a CN, i.e. BU = {Src=CoA, Des=CN, HoA, PK_{MN}, Modifier, TS_{MN}, SIG_{MN}[H(CoA || CN || HoA || TS_{MN})]} where PK_{MN} is the MN's public key; Modifier is a random number indicating the value of the modifier used while configuring the HoA; TS_{MN} is MN's timestamp; H() denotes a hash function; and SIG_{MN}[] denotes MN's signature using the private key SK_{MN}.

When the CN receives the signed BU message, it first compares TS_{MN} against

its own clock to detect replayed BUs. The CN next verifies the binding between the PK_{MN} and the HoA by re-computing and comparing the hash value with the interface identifier part of the HoA. The CN then verifies the signature and the authenticity of the HoA by using the PK_{MN}. If any of the above verifications fails, the CN will reject the message. Otherwise, the CN will accept the message, create/update a Binding Cache entry, and store the binding between the MN's HoA and CoA.

An entire run of the CAM protocol requires one message and one one-way time between the MN and the CN, causing minimal signalling overhead as well as registration delay. However, the protocol is vulnerable to malicious MNs flooding attacks as it cannot authenticate the claimed CoA. In addition, as the protocol uses timestamps to detect replayed BUs, it requires the clocks of the MN and the CN to be synchronised. It also suffers from the same weaknesses of the CGA-based technique mentioned in Section 3.1.

3.2.4 Unauthenticated Diffie-Hellman-based Binding Update (UDHBU) Protocol

The Unauthenticated Diffie-Hellman-based Binding Update (UDHBU) protocol [42] makes use of the Diffie-Hellman key exchange protocol to derive a shared session key between an MN and a CN. The two nodes then use the shared secret in authenticating subsequent correspondent registrations. The UDHBU protocol is divided into two cases: an initial correspondent registration and a subsequent correspondent registration. It is shown in Figure 3.4 and is detailed in Appendix B.

The UDHBU protocol requires four messages during the initial correspondent registration and two messages during subsequent ones. It also reduces the registration delay of subsequent registrations to one one-way time between the MN and the CN. In addition, it protects CNs against resource exhaustion DoS attacks; the CN does not create any state for individual MNs until it receives an authentic BU from that MN in $Message_3$. Furthermore, the CN is required to perform just two exponential calculations during the initial correspondent registration; all other calculations performed by the CN are based on MAC operations, which are considered lightweight operations. Finally, it establishes a long-term shared key between the MN and the CN, which reduces the number of redundant signalling

messages caused by the RR procedure for frequently generating a fresh shared key.

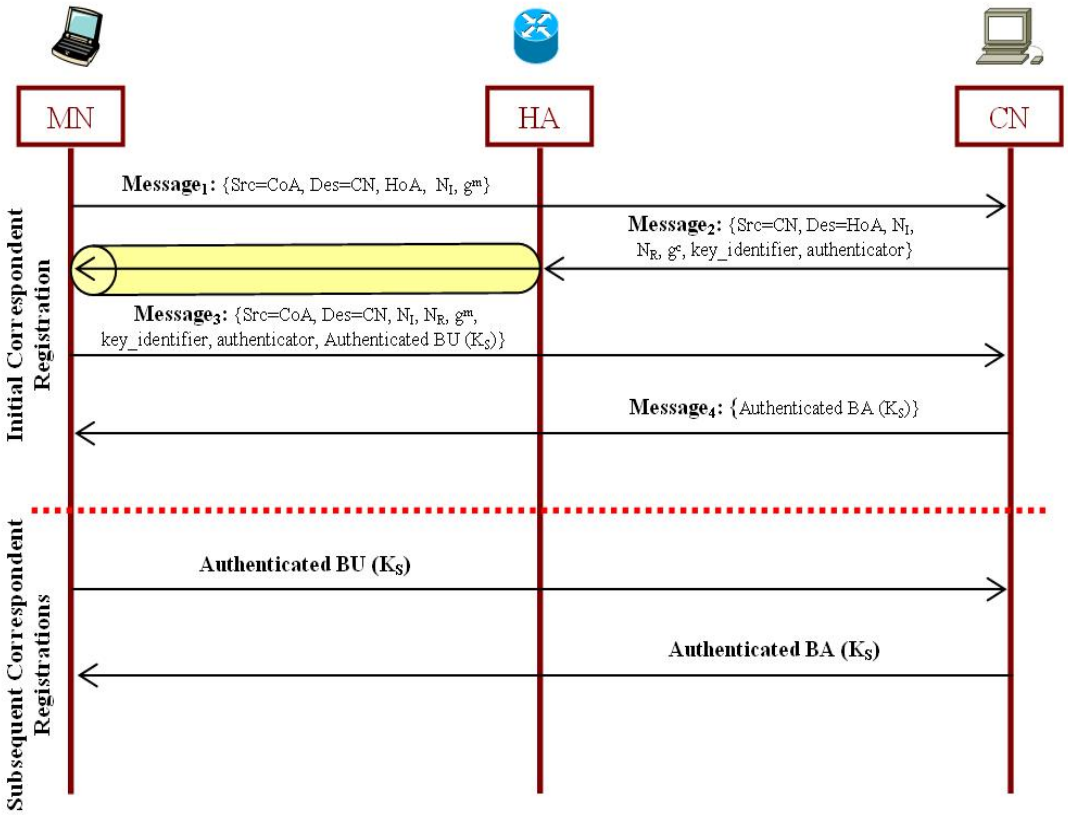

Figure 3.4: The UDHBU protocol

The UDHBU protocol is vulnerable to malicious MNs flooding attacks as it cannot authenticate the claimed CoA. In addition, it is vulnerable to MITM and false BU attacks as it involves an unauthenticated DH key exchange between the MN and the CN. An attacker on the path during the initial correspondent registration is able to intercept the DH public values and send its own DH public values. Furthermore, the attacker can initialise the protocol instead of the legitimate MN and gain the session key; subsequently it can use the key to send false BUs.

3.2.5 Optimizing Mobile IPv6 (OMIPv6) Protocol

The Optimizing Mobile IPv6 (OMIPv6) protocol [43] combines the use of the RR procedure and the Diffie-Hellman key exchange protocol to derive a shared session

key between an MN and a CN. It is similar to the UDHBU protocol mentioned in Section 3.2.4 and is also designed into an initial correspondent registration case and a subsequent correspondent registration case. However, in the initial correspondent registration, the MN and the CN first run the RR procedure to establish a shared secret. The two nodes then run the DH key exchange that is authenticated by the shared secret established from the RR procedure to derive a long-term shared session key. Apart from this, the OMIPv6 protocol is the same as the UDHBU protocol; subsequent correspondent registrations in the two protocols are identical.

The OMIPv6 protocol requires eight messages and about 2.5 round-trip times between the MN and the CN during the initial correspondent registration. However, it requires only two messages and about one one-way time between the MN and the CN during subsequent correspondent registrations. It also increases the protection, compared to the UDHBU protocol, provided to the CN against resource exhaustion DoS attacks by signing the DH messages with the shared secret established from the RR procedure. Apart from these, the OMIPv6 protocol has the same features, pros, and cons as the UDHBU protocol mentioned in Section 3.2.4.

3.2.6 Applying CGAs to Optimize Mobile IPv6 (CGA-OMIPv6) Protocol

The CGA-OMIPv6 protocol [44] combines the use of the RR procedure and the CGA-based technique. It applies the following set of optimizations to the Mobile IPv6 protocol: (1) an MN has a self-generated public/private key pair; (2) the MN configures its HoA based on the concept of CGAs mentioned in Section 3.1; (3) the MN provides a care-of address reachability proof by exchanging CoTI and CoT messages with a CN every time a new CoA is claimed; and (4) the MN provides initial HoA ownership and reachability proofs at initial correspondent registration with the CN. The MN provides the initial HoA ownership proof by signing the first BU message with its private key and by sending its public key along with the signed message. The initial HoA reachability proof is provided by exchanging HoTI and HoT messages with the CN. The CGA-OMIPv6 protocol is divided into two cases: an initial correspondent registration and a subsequent correspondent registration. The former enables the CN to authenticate the MN's

HoA and to verify the MN's reachability at the HoA as well as the CoA. It also enables the two nodes to securely exchange a secret session key. The latter is performed every time the MN wants to register a new CoA at the CN. It enables the CN to verify the MN's reachability at the new CoA. The CGA-OMIPv6 protocol is shown in Figure 3.5 and is detailed in Appendix B.

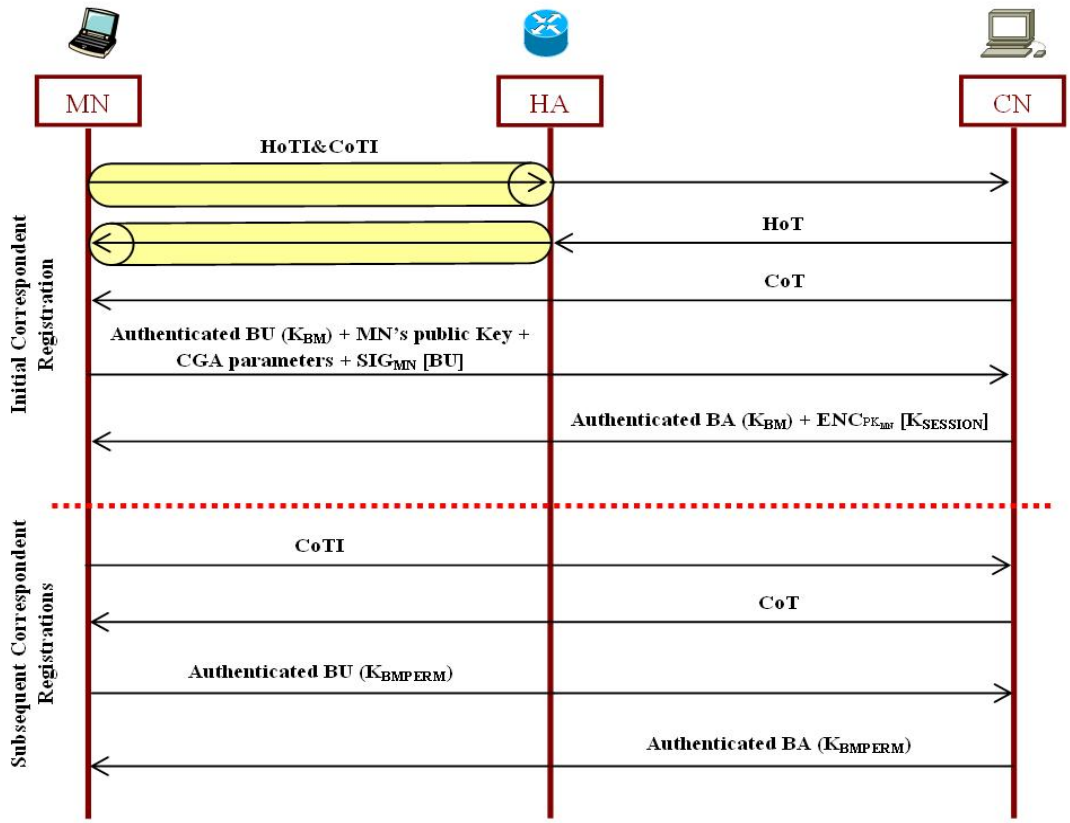

Figure 3.5: The CGA-OMIPv6 protocol

The CGA-OMIPv6 protocol requires five messages during the initial correspondent registration and four messages during subsequent ones. It also requires about 1.5 round-trip times between the MN and the CN during any correspondent registration. It employs a CGA-based HoA together with an initial HoA reachability test for authenticating the HoA, which partially protects against return-to-home spoofing attacks. It also employs a CoA reachability test whenever a new CoA is to be claimed, which partially protects third parties against DoS attacks. It also partially protects CNs against resource exhaustion DoS attacks; the CN uses K_{BM} established from the RR procedure to verify the authenticity and integrity of the BU message before running the CGA-based address verification

algorithm. In addition, it requires each of the MN and the CN to perform two public key operations during the initial correspondent registration.

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

The ERO-MIPv6 protocol [45] combines the use of the EBU and the CGA-OMIPv6 protocols (see Sections 3.2.1 and 3.2.6). It is shown in Figure 3.6 and is detailed in Appendix B.

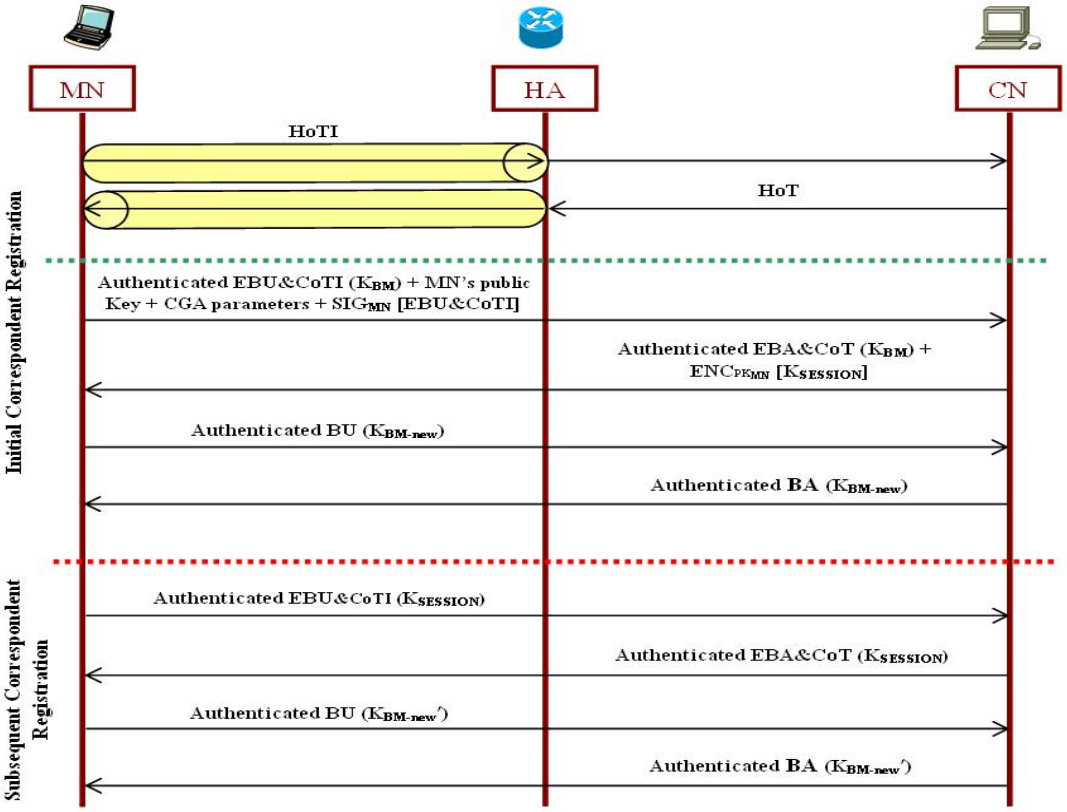

Figure 3.6: The ERO-MIPv6 protocol

The ERO-MIPv6 protocol requires six messages during the initial correspondent registration and four messages during subsequent ones. It optimises the registration delay to about one one-way time between the MN and the CN during any correspondent registrations. Apart from this optimisation, the ERO-MIPv6 protocol has the same pros and cons of the CGA-OMIPv6 protocol mentioned in

Section 3.2.6. In addition, it increases complexity for implementing the Credit-Based Authorisation technique at the CN, as in the EBU protocol.

3.3 Infrastructure-based Correspondent Registration Protocols

The infrastructure-based correspondent registration protocols rely on security infrastructure support for protecting correspondent registrations. These protocols can largely be divided into two subclasses. The first subclass is referred to as the secret-key based protocols, as they protect correspondent registrations based on a secret key shared between the participants. The second subclass relies on having a certified public/private key pair and is, therefore, referred to as the public-key based protocols. The infrastructure-based correspondent registration protocols assure the CN that a specific home address is used by the right MN, i.e. the shared secret key (or the public key of the public/private key pair) is related to the MN's HoA. In addition, they remove the need for a home address test leading to a reduction in registration delay and signalling overhead.

3.3.1 Secret-Key based Protocols

Secret-Key based protocols require the establishment of a shared secret between an MN and a CN. Therefore, the applicability of these protocols is limited to scenarios when the MN and the CN know each other in advance and can agree on a secret. A typical scenario where these protocols are applicable is within a corporation or among users who know each other.

3.3.1.1 Static Shared Key version 1 (SSKv1) Protocol

The Static Shared Key version 1 (SSKv1) protocol [46] is based on the pre-configuration of the data needed for creating the binding management key (K_{BM}). These data include a secret key (K_{CN}), a home nonce (N_I), and a care-of nonce (N_J). The SSKv1 protocol protects correspondent registrations in the same way as specified in the base specification of the MIPv6 protocol ([5], see also Section 2.5). However, there are two differences. First, K_{CN} is a shared secret known to both the CN and the MN. Second, N_I and N_J are shared secrets as well. Therefore, the SSKv1 protocol omits the RR procedure; the MN generates

the home and care-of keygen tokens instead of requesting them from the CN. As a result, an entire correspondent registration process only requires one BU message and, optionally, one BA message.

The SSKv1 protocol requires only two messages during any correspondent registration, which reduces signalling overhead compared to the RR procedure. It also reduces registration delay to about one one-way time between the MN and the CN. Furthermore, it increases the security strength by ensuring the home address ownership, as the shared secrets are related to the MN's HoA. Therefore, it protects against return-to-home spoofing attacks. However, the SSKv1 protocol is vulnerable to malicious MNs flooding attacks as it cannot authenticate the claimed CoA. It also requires the MN to store different secrets for each CN and the CN to store different secrets for each MN. In addition, it requires the CN to keep track of the most recent value of the sequence number for BU messages from each MN to protect against replay attacks. If the CN does not maintain the value of the sequence number from a particular MN, then the CN could be fooled into accepting a replayed BU message. Finally, it requires a manual reconfiguration for the shared key (K_{CN}) and the associated nonces (N_I, and N_J) at the MN and the CN whenever sequence numbers roll over, so that a new K_{BM} is computed and replay attacks are prevented.

3.3.1.2 Static Shared Key version 2 (SSKv2) Protocol

The Static Shared Key version 2 (SSKv2) protocol [47] enhances the SSKv1 protocol by applying a care-of address reachability test. It aims to verify MNs' reachability at claimed CoAs to partially protect third parties against malicious MNs flooding attacks. The SSKv2 protocol can be summarised as follows. Upon receiving a BU message from an MN, a CN rejects the message and returns a BA message to the claimed CoA containing a fresh token. When the MN receives the rejected BA message, it retransmits the BU message with the exact received token. When the CN receives the augmented BU message, it validates the received token; the CN re-computes the token and compares it to the received one. If the validation succeeds, the CN infers that the MN is reachable at the claimed CoA. Therefore, the CN accepts the binding and returns a normal BA message.

The SSKv2 protocol verifies the MNs' reachability at claimed CoAs, thus partially protecting third parties against malicious MNs flooding attacks. However,

it adds two additional messages and one additional round-trip time, i.e. it requires four messages and about 1.5 round-trip times between the MN and the CN during any correspondent registration. Apart from these, the SSKv2 protocol has the same features, pros, and cons as the SSKv1 protocol mentioned in the previous section.

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

The PAK-based BU protocol [48] requires an MN to share a password with a CN. It applies an optimised PAK scheme to enable the CN and the MN to authenticate each other, establish a one-time binding management key for protecting the subsequent correspondent registration, and verify the correctness of the home address as well as the location of the MN.

In the PAK-based BU protocol, an MN performs a correspondent registration by exchanging four messages with a CN. The first two messages are the optimised PAK scheme [48], in which the MN and the CN run a Diffie-Hellman-based password authenticated key exchange scheme [49]. The other two messages are the normal BU and BA, which are protected using the key established from the optimised PAK scheme.

The PAK-based BU protocol requires four messages and about 1.5 round-trip times between the MN and the CN during any correspondent registration. It also verifies the MN's reachability at the CoA, thus partially protecting third parties against malicious MNs flooding attacks. In addition, it increases the security strength by ensuring the home address ownership, as the password is related to the MN's HoA. Therefore, it protects against return-to-home spoofing attacks. However, it requires the MN to store a different password for each CN and the CN to store a different password for each MN. It also requires both the MN and the CN to perform two exponential calculations per correspondent registration.

3.3.1.4 Ticket-based Binding Update (TBU) Protocol

The TBU protocol [50] requires that the connection between an MN's home link and a CN is protected securely using IPSec ESP tunnel. It also requires that an HA's duty is extended to testify the legitimacy of an MN's HoA, facilitate mutual authentication between the MN and a CN, and establish a secret key and ticket between the two nodes. The TBU protocol is divided into two cases: an

initial correspondent registration and a subsequent correspondent registration. It is shown in Figure 3.7 and is detailed in Appendix B.

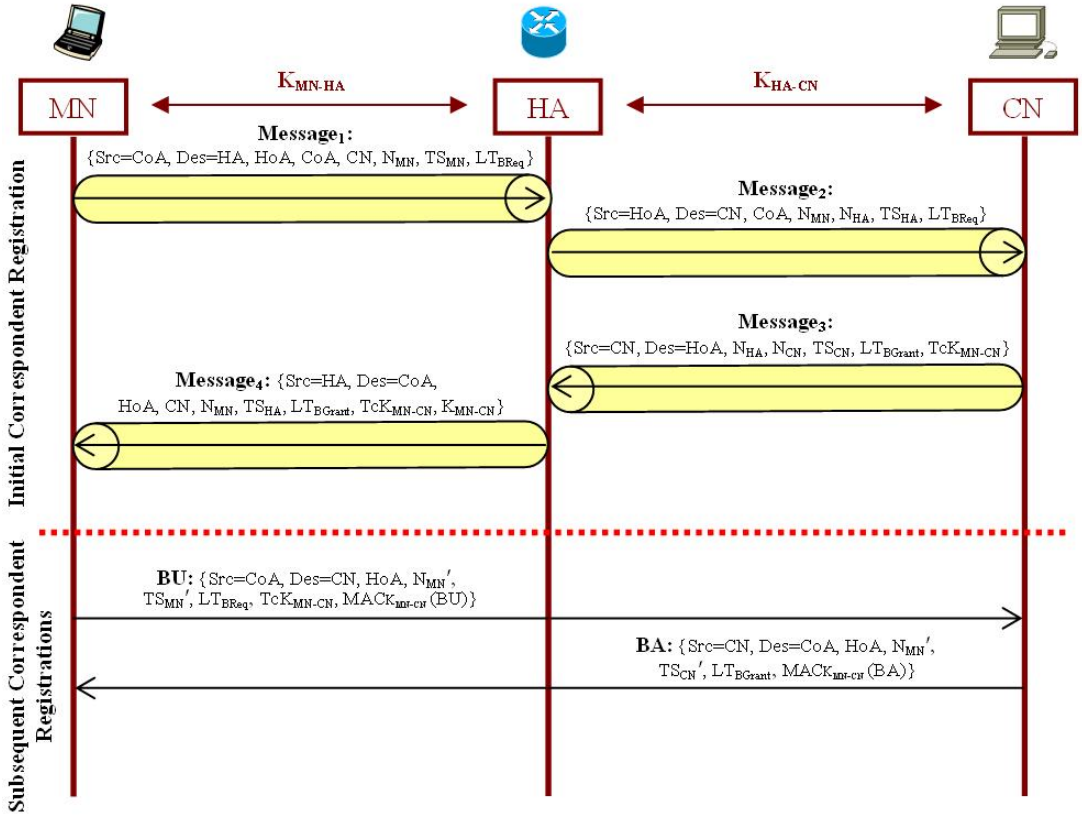

Figure 3.7: The TBU protocol

The TBU protocol requires only two messages and about one one-way time between the MN and the CN during subsequent correspondent registrations. This introduces low signalling overhead and registration delay. It also increases the security strength by ensuring the home address ownership, i.e. it protects against return-to-home spoofing attacks. In addition, it establishes a long-term secret key and ticket between the MN and the CN, thus reducing the number of redundant signalling messages caused by the RR procedure for frequently generating a fresh shared key.

The TBU protocol requires four messages and about one one-way time between the MN and the HA plus one one-way time between the HA and the CN during the initial correspondent registration. The TBU protocol also requires the clocks of the MN, the HA, and the CN to be synchronised as it uses timestamps to detect replayed messages. In addition, it increases the complexity at the HAs

for supporting correspondent registrations of MNs. Furthermore, it is vulnerable to malicious MNs flooding attacks as it cannot authenticate claimed CoAs.

3.3.2 Public-Key based Protocols

Public-key based protocols require that an MN (or an MN's home link) has a certified public/private key pair. They assume that the public key has been issued by a trusted CA and that a PKI is in place to provide the services of certificate application, issuance and revocation.

3.3.2.1 Certificate-based Binding Update (CBU) Protocol

The CBU protocol [51, 52] makes use of a digital signature scheme and a unilateral authenticated Diffie-Hellman key exchange protocol to derive a long-term shared secret between an MN and a CN. It requires that each home link has a certified public/private key pair (PK_H, SK_H) where the private key is kept by the HAs in the home link. It also requires that an HA's duty is extended to handle strong authentication on behalf of the MN during correspondent registrations. The HA testifies the legitimacy of the MN's HoA, facilitates authentication of the MN to the CN, and establishes a shared secret session key between the two nodes. The CBU protocol is shown in Figure 3.8 and is detailed in Appendix B.

The CBU protocol has the same pros as the TBU protocol mentioned in Section 3.3.1.4. In addition, it offloads public key operations and DH exponential calculations from MNs to their HAs. Furthermore, it partially protects HAs and CNs against resource exhaustion DoS attacks through the use of cookies during the key exchange, and through the prevention of the CN from retaining any state of individual MNs until it receives an authentic $EXCH_0$ message from that MN's home link.

The CBU protocol requires eight messages and about one round-trip time between the MN and the HA, plus two round-trip times between the HA and the CN, plus one one-way time between the MN and the CN during the initial correspondent registration. In addition, it has the same cons as the TBU protocol, but synchronised clocks are not required.

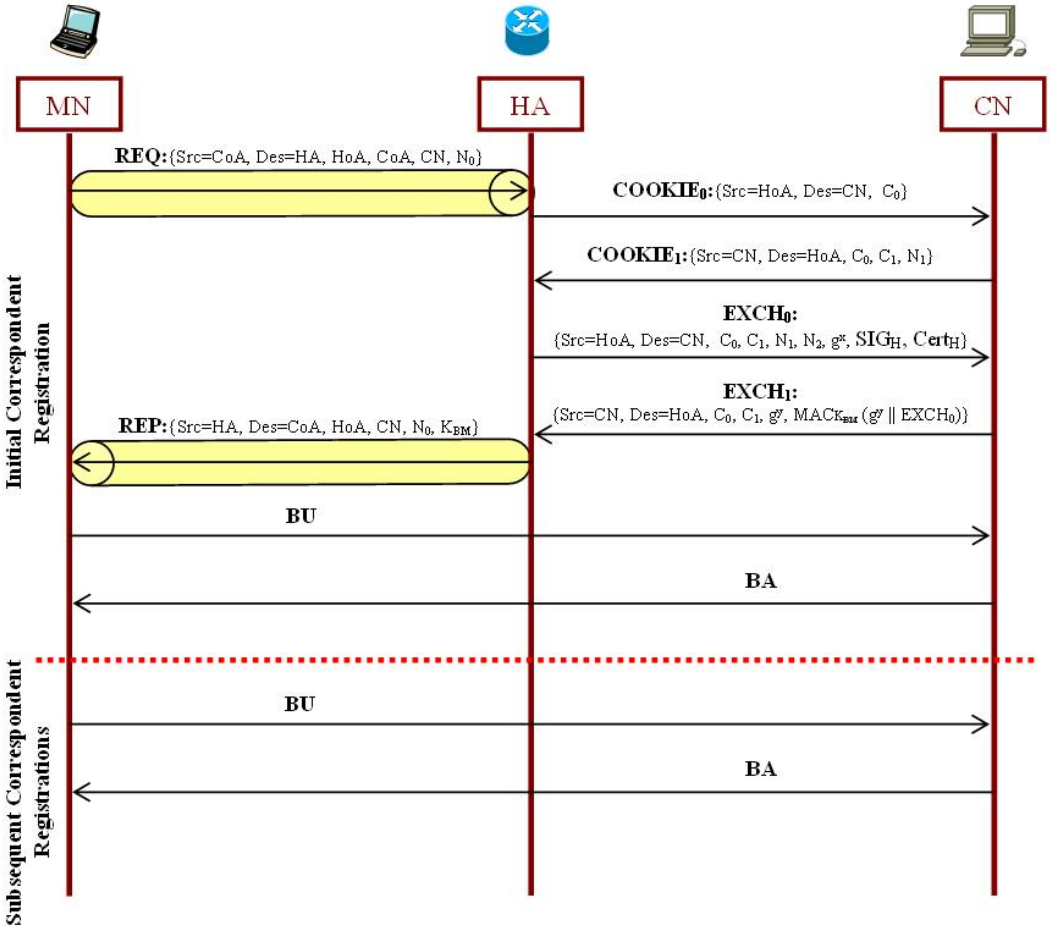

Figure 3.8: The CBU protocol

3.3.2.2 Hierarchical Certificate-based Binding Update (HCBU) Protocol

The HCBU protocol [23] enhances the CBU protocol by assuring an MN's ownership of a claimed CoA. It further extends an HA's duty to cover all the nodes currently under its home link, i.e. the HA's duty covers not only MNs belonging to its home link but also roaming MNs from other foreign links. In HCBU, when an MN roams to a foreign link and is configured a new CoA, it additionally obtains the foreign link's signature on the binding of (HoA, CoA). The signature convinces the MN's home link that the MN actually owns the claimed CoA. Subsequently, the home link proves the MN's ownership of both the HoA and CoA to the CN. The HCBU protocol is designed into an initial correspondent registration case and a subsequent correspondent registration case, where the former

case is designed into two steps: a pre-handover step and a post-handover step. The HCBU protocol is shown in Figure 3.9 and is detailed in Appendix B.

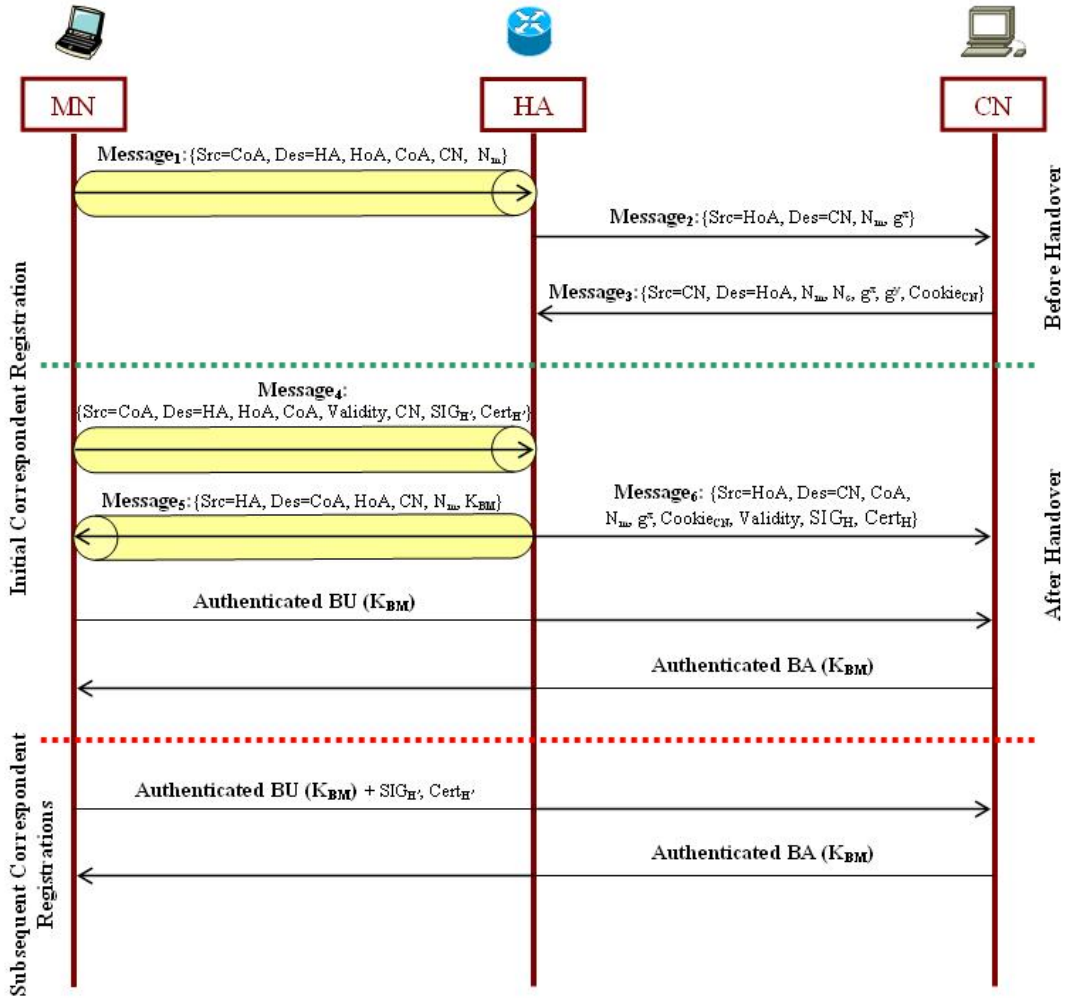

Figure 3.9: The HCBU protocol

The HCBU protocol has the same pros as the CBU protocol mentioned in the previous section. In addition, it authenticates the claimed CoA, thus achieving much stronger security strength by protecting third parties against malicious MNs flooding attacks. Furthermore, it reduces registration delay during the initial correspondent registration by taking advantage of the EBU protocol, i.e. the pre-handover step is performed before the handover. However, the MN may periodically need to repeat the pre-handover step (such as when handovers cannot be anticipated), which could increase the signalling overhead. The protocol also requires the use of trusted third parties (i.e. foreign links) to verify the CoAs

used by the MNs; it requires an infrastructure that supports this authentication service. Furthermore, it further increases the complexity at the HAs by extending their duty to additionally cover roaming MNs from other foreign links. Finally, as it requires foreign HAs to compute a signature for each roaming MN, this can significantly reduce throughput at foreign networks.

3.3.2.3 Extended Ticket-based Binding Update (ETBU) Protocol

The ETBU protocol [53] extends the TBU protocol (see Section 3.3.1.4) to handle with the case of two mobile nodes in simultaneous movement, i.e. when the CN is also mobile, and when the two nodes move simultaneously to different foreign links. It also makes use of a digital signature scheme and the CGA-based technique to provide mutual authentication between an MN and a CN. Specifically, it requires that (1) each home link has a certified public/private key pair where the private key is kept by HAs in the home link; (2) each mobile node has a self-generated public/private key pair; and (3) both the HoA and CoA of each mobile node are configured as CGA-based addresses with the same public key. In addition, it uses tickets issued by the MN and the CN instead of signatures to minimise the computing costs that are needed to verify the CGA-based addresses; a ticket is used instead of a new signature when the MN obtains a new CoA. Furthermore, it utilises the smooth handoff mechanism [54] to solve the simultaneous movement problem. That is, when an MN moves to a new foreign link, the home agent of the previous link intercepts and forwards the messages destined for the MN's previous CoA to the MN's new CoA.

The participants of the ETBU protocol are an MN, an MN's HA (HA_{MN}), a CN, and a CN's HA (HA_{CN}). Each HA executes the role of a verification server that checks on the validity of the messages forwarded from the other node, which offloads public-key operations from both the MN and the CN to its respective HA. The ETBU protocol is divided into two cases: an initial correspondent registration and a subsequent correspondent registration. It is shown in Figure 3.10 and is detailed in Appendix B.

The ETBU protocol does not require the MN to create a signature each time it obtains a new CoA. It also utilises the smooth handoff mechanism to handle the case of two mobile nodes in simultaneous movement. However, it requires six messages and about one one-way time between the MN and the HA_{MN}, plus one one-way time between the HA_{MN} and the HA_{CN}, plus one one-way time

between the HA_{CN} and the CN during the initial correspondent registration. In addition, it increases the HA_{CN}'s vulnerabilities to resource exhaustion DoS attacks; the HA_{CN} is required to perform digital signature verification during the initial correspondent registration. An attacker can start an excessive number of protocol runs by spoofing a large number of signed messages, which causes the HA_{CN} to verify the signatures before rejecting the messages. Apart from these, the ETBU protocol has all the pros and cons of the CBU protocol mentioned in Section 3.3.2.1.

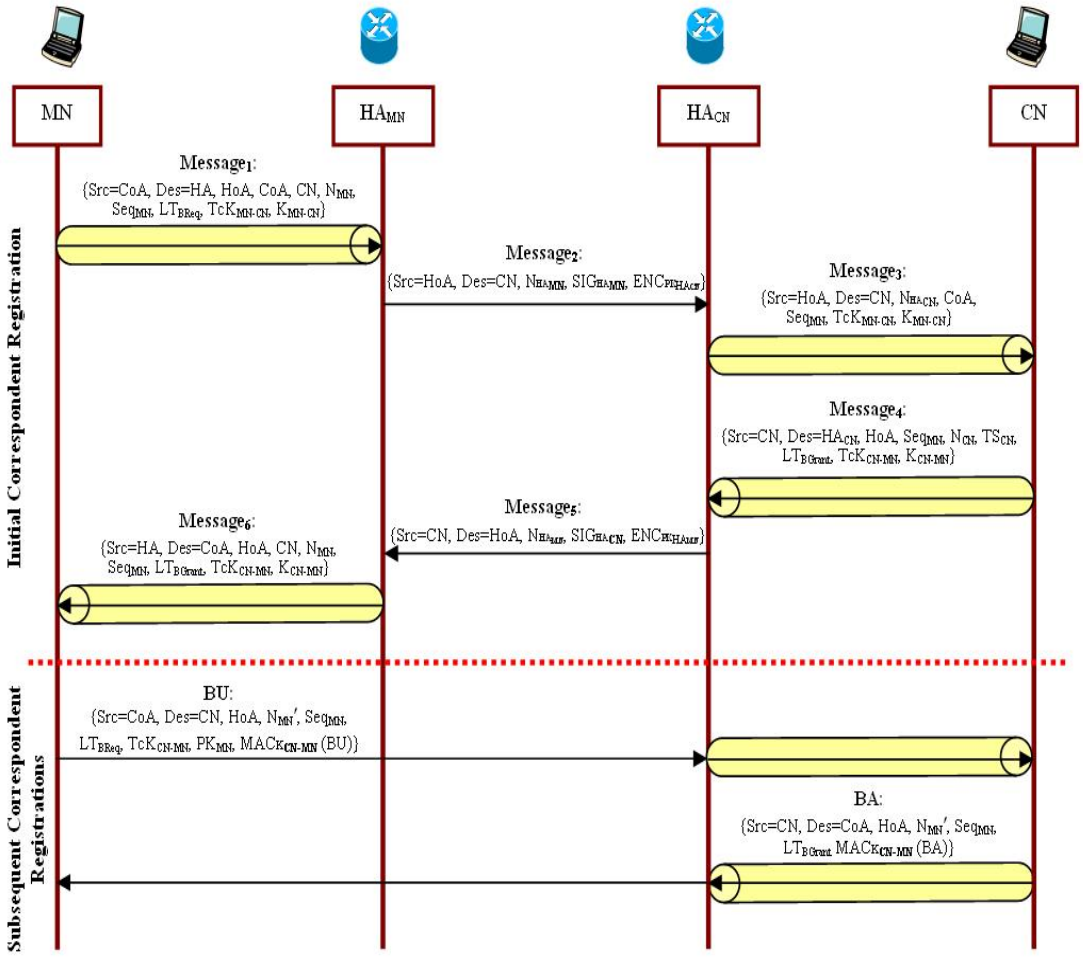

Figure 3.10: The ETBU protocol

3.4 Chapter Summary

This chapter has presented a survey of the existing protocols for protecting correspondent registrations in the MIPv6 protocol. The studied protocols include two classes: 'infrastructure-less correspondent registration protocols' and 'infrastructure-based correspondent registration protocols'. Furthermore, the latter class includes secret-key based and public-key based protocols. Tables 3.1, 3.2, 3.3, and 3.4 summarise how well each of the studied protocols meets the security services and the performance requirements set out in Section 2.2.2. Based on this, we conclude that:

1. It is preferable to design a correspondent registration protocol into cases to reduce iterating courses of the entire protocol; to avoid unnecessarily repeating some sections of the protocol. That is, designing the protocol into an initial correspondent registration case, a subsequent correspondent registration case, and a correspondent registration deletion case could elevate efficiency by reducing iterating registration courses.

2. A correspondent registration protocol could be designed in a way that the registration procedure is completed with the help of HAs to reduce the computing costs of the MNs.

3. None of the protocols, except the HCBU, are capable of guaranteeing the authenticity of the claimed care-of address, thus they cannot protect third parties against DoS attacks and malicious MNs flooding attacks. In addition, the HCBU protocol requires an infrastructure, i.e. the use of trusted third parties (foreign links), to support this authentication service. Furthermore, the HCBU protocol requires both a foreign access router and a CN to perform computationally expensive signature generation and verification operations each time an MN moves to a different foreign link. This can significantly reduce throughput at both the foreign network and the CN.

The next chapter presents a novel protocol for securing home registrations in the MIPv6 protocol; it presents the design and evaluation of the Enhanced Home Registration (EHR) protocol.

	RR	EBU	PBK	CAM	UDHBU	OMIPv6	CGA-OMIPv6	ERO-MIPv6	INF-based	SSKv1	SSKv2	PAK-based	TBU	SK-based	CBU	HCBU	ETBU	PK-based
Protocol needs security infrastructure support	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Protocol authenticates HoAs	△	△	x	△△	△	△	△△△	△△△	△△△	✓	✓	✓	✓	✓	✓	✓	✓	✓
Protocol authenticates CoAs	△	△	△	x	x	x	△	△	x	x	△	△	x	✓	x	✓	△△	✓
Protocol ensures integrity of BUs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Protocol ensures freshness of BUs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Protocol protects CNs against resource exhaustion DoS attacks	✓	✓	x	x	✓	✓	△	△	△	✓	✓	✓	✓	✓	△	△	x	△
Protocol protects against on-path attacks	x	x	x	✓	x	▽	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

△: Satisfies the security services by using an address reachability test.
△△: Satisfies the security services by using a CGA-based address.
△△△: Satisfies the security services by using a CGA-based address and an address reachability test.
▽: Satisfies the security services by running the RR procedure first.

Table 3.1: Security requirements vs. state of the art

	RR	EBU	PBK	CAM	UDHBU	OMIPv6	CGA-OMIPv6	ERO-MIPv6	INF-based	SSKv1	SSKv2	PAK-based	TBU	SK-based	CBU	HCBU	ETBU	PK-based
Number of messages in the first registration	6	8	4	1	4	8	5	6	9	2	4	4	4	8	8	8	6	8
Number of messages in subsequent registrations	6	8	4	1	2	2	4	4	4	2	4	4	2	4	2	2	2	4
Number of messages in deregistrations	4	4	4	1	2	2	2	2	2	2	2	4	2	2	2	2	2	2
Registration delay in the first registration	1.5⋆	0.5	1.5	0.5	1.5	2.5⋆	1.5⋆	0.5	0.5[a] + 0.5[b] + 1⋆	0.5	1.5	1.5	0.5[a] + 0.5[b]	0.5[a] + + 0.5[b] + 1	1[a] + 2[b] + 0.5	1[a] + 0.5	0.5[a] + 0.5[b] + 0.5[c]	0.5[a] + + 0.5[b] + 1
Registration delay in subsequent registrations	1.5⋆	0.5	1.5	0.5	0.5	0.5	1.5	0.5	0.5	0.5	1.5	1.5	0.5	0.5	0.5	0.5	0.5	0.5
Deregistration delay	1.5⋆	0.5	1.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.5	0.5	0.5	0.5	0.5	0.5	0.5

(De)registration delay is the number of round-trip times between the MN and the CN.
⋆: One round-trip is done through the MN's home network.
a: round-trip between the MN and its HA.
b: round-trip between the MN's HA and the CN.
c: round-trip between the CN and its HA.

Table 3.2: Performance requirements vs. state of the art (a)

		RR	EBU	PBK	CAM	UDHBU	OMIPv6	CGA-OMIPv6	ERO-MIPv6	INF-based	SSKv1	SSKv2	PAK-based	TBU	SK-based	CBU	HCBU	ETBU	PK-based
Operational load at MN in the first registration	Public key	0	0	2	1	0	0	2	2	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	2	2	0	0	0	0	0	2	0	0	0	0	0	0
	HMAC	4	8	0	0	2	6	4	8	2	6	9	4	0	3	2	2	0	2
	Secret key	0	0	0	0	0	0	0	0	2	0	0	0	2	3	2	3	3	2
Operational load at MN in subsequent registrations	Public key	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
	HMAC	4	8	0	0	2	2	4	6	4	5	8	4	2	4	2	2	2	4
	Secret key	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
Operational load at MN in deregistration	Public key	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
	HMAC	4	4	0	0	2	2	2	2	4	4	4	4	2	4	2	2	2	4
	Secret key	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Operational load at CN in the first registration	Public key	0	0	2	1	0	0	2	2	2	0	0	0	0	0	1	1	0	3
	Exponential	0	0	0	0	2	2	0	0	0	0	0	2	0	0	2	2	0	0
	HMAC	7	11	0	1	4	11	9	11	9	5	10	4	1	6	4	5	0	6
	Secret key	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0	1	0
Operational load at CN in subsequent registrations	Public key	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
	HMAC	7	11	0	1	2	2	5	7	8	4	9	4	1	8	2	2	6	8
	Secret key	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0
Operational load at CN in deregistration	Public key	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
	HMAC	5	5	0	1	2	2	2	2	3	3	3	4	1	3	2	2	6	3
	Secret key	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0
Operational load at an entity is the number of public-key, exponential, keyed hash function, and secret-key operations performed by the entity during a protocol execution																			

Table 3.3: Performance requirements vs. state of the art (b)

		RR	EBU	PBK	CAM	UDHBU	OMIP$_{v6}$	CGA-OMIP$_{v6}$	ERO-MIP$_{v6}$	INF-based	SSK$_{v1}$	SSK$_{v2}$	PAK-based	TBU	SK-based	CBU	HCBU	ETBU	PK-based
Operational load at HA in the first registration	Public key	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	2	4	2
	Exponential	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0
	HMAC	0	0	0	0	0	0	0	0	2	0	0	0	1	5	3	1	0	2
	Secret key	0	0	0	0	0	0	0	0	2	0	0	0	4	2	2	3	2	2
Operational load at HA in subsequent registrations	Public key	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Exponential	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	HMAC	0	0	0	0	0	0	0	0	3	0	0	0	0	3	0	0	0	3
	Secret key	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
Operational load at HA$_{CN}$ in the first registration	Public key	0	-	-	-	-	-	-	-	0	-	-	-	-	0	-	-	-	0
	Exponential	0	-	-	-	-	-	-	-	0	-	-	-	-	0	-	-	-	0
	HMAC	0	-	-	-	-	-	-	-	0	-	-	-	-	0	-	-	-	0
	Secret key	0	-	-	-	-	-	-	-	0	-	-	-	-	0	-	-	-	0

Operational load at an entity is the number of public-key, exponential, keyed hash function, and secret-key operations performed by the entity during a protocol execution

Table 3.4: Performance requirements vs. state of the art (c)

Chapter 4

The Enhanced Home Registration (EHR) Protocol

Section 2.4 investigated the basic home registration process included in the MIPv6 protocol to enable an MN to register its current CoA with an HA. The investigation showed that the HA could not authenticate the given CoA. Therefore, the MN could lie about its current location and lure the HA to redirect traffic to a third party causing a DoS attack against that third party.

This chapter presents an enhanced home registration process to support location authentication of MNs to their respective HAs. This is called the Enhanced Home Registration (EHR) protocol. The EHR protocol allows an HA to verify that a claimed CoA is indeed an MN's real location. As a result, the EHR protocol can reduce the likelihood of a malicious MN being successful in luring an HA to flood a third party with useless traffic using the MIPv6 protocol.

This chapter is organized as follows. Section 4.1 gives an overview of the EHR protocol and details the design of its three building blocks: the symmetric CGA-based technique, the concurrent CoA reachability test, and the segmenting IPv6 address space method. Section 4.2 presents a detailed description of the EHR protocol and its informal security analysis in comparison to the basic home registration protocol. Section 4.3 presents the simulation and evaluation of the EHR protocol compared to the basic home registration protocol. It first discusses the design and construction of the simulation model. It next validates the simulation model. It then presents the simulation results, discusses the implications of the results and draws conclusions. Finally, Section 4.4 summarises the chapter.

4.1 The EHR Protocol Overview

The EHR protocol extends the basic home registration protocol defined in the MIPv6 base document [5] by making use of a combination of three ideas. Firstly, it uses a novel lightweight version of the traditional CGA-based technique to cryptographically generate and verify MNs' CoAs. This is called the symmetric CGA-based technique. This technique makes use of a secret key shared between an MN and its HA in the CoA generation and verification processes. It is used to reduce the likelihood of a malicious MN being successful in stealing other nodes' addresses. Secondly, the EHR protocol applies a novel concurrent CoA reachability test to verify an MN's reachability at a claimed CoA. The test enables an HA to register and use an MN's new CoA while concurrently verifying the MN's reachability at that CoA. Thirdly, the EHR protocol optionally uses a novel method called the segmenting IPv6 address space method to differentiate between hosts based on their IPv6 address. The method divides the IPv6 address space into three parts: home addresses, care-of addresses, and stationary addresses. Therefore, it reduces the number of targets that are vulnerable to DoS attacks launched by malicious MNs.

4.1.1 The Symmetric CGA-based Technique

The symmetric CGA-based technique is a novel lightweight CGA-based technique that is used for the cryptographic generation and verification of an IPv6 address. It requires a secret key to be shared between the participants, i.e. address owner and address verifier, and is thus suitable for use between MNs and their respective HAs.

If an MN wants to cryptographically generate a CoA that could be verified by a number of HAs, the key used in the address generation and verification processes must be shared with all of the HAs. To achieve this, when the MN is in an initial state (not registered with any HA), it generates a random number that represents a new key. The MN uses this key to cryptographically generate subsequent CoAs when roaming away from home and securely sends this key to each HA when first registering with that HA. The HA stores this key in its Binding Cache entry for the MN's HoA and uses it to verify the MN's claimed CoA as well as subsequent CoAs. Using this method, the MN can register any of its CoAs with all of its home links. In addition, the original pre-shared secret

key with each of the home links (if any) is protected from brute force attacks, as it is not used in the address generation or verification processes.

The details of the generation and verification processes of the symmetric CGA-based technique are largely similar to those described by Aura [33] and mentioned in Section 3.1. When generating a CGA-based CoA, an MN uses two input values: (1) a 64-bit subnet prefix, and (2) a secret key shared between the MN and its HA. The generation process is depicted in Figure 4.1 and detailed as follows:

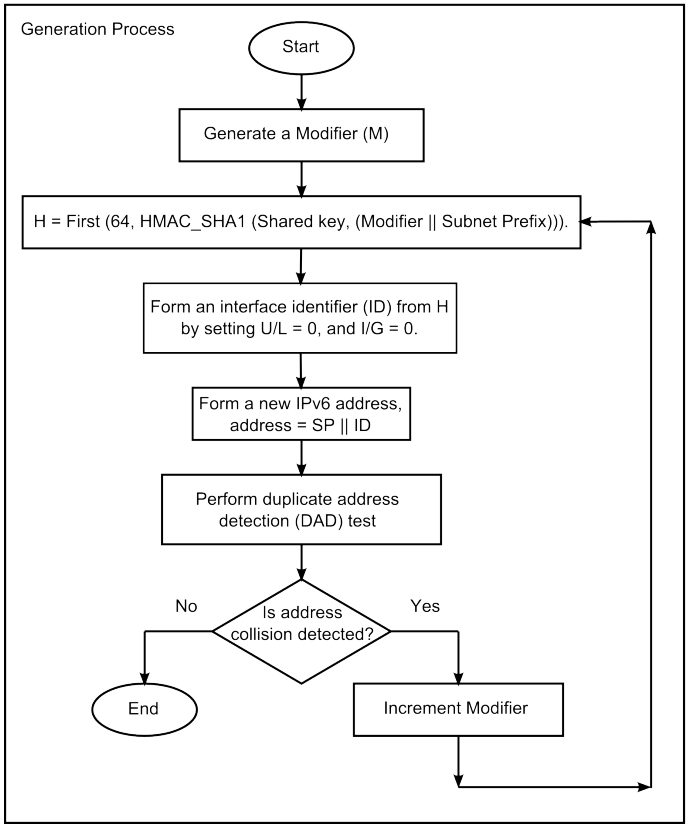

Figure 4.1: Symmetric CGA-based address generation algorithm

Symmetric CGA-based Generation Algorithm:

1. Generate a random 128-bit number, called a modifier. This modifier is used to further randomise the addresses generated from the same subnet prefix and shared key.

2. Concatenate from left the modifier and the subnet prefix. Execute the HMAC_SHA1 function on the concatenation using the secret key shared

with the HA and get the leftmost 64 bits of the output. The result is H, i.e. H = First (64, HMAC_SHA1 (shared key, (modifier || subnet prefix))).

3. Form an interface identifier from H by setting U/L and I/G bits to zero and zero, respectively.

4. Concatenate the 64-bit subnet prefix and the 64-bit interface identifier to form a 128-bit IPv6 address with the subnet prefix to the left and the interface identifier to the right.

5. Perform a Duplicate Address Detection (DAD) test. If an address collision is detected, increment the modifier by one and go back to step 2.

The outputs of the address generation algorithm are (1) a new CGA-based IPv6 address (i.e. a CoA), and (2) a 128-bit number representing the final value of the modifier. The modifier is carried in BU messages that are sent by MNs to their HAs in order to convey the modifier value. In addition, if a BU message is to create a new binding, the secret key is also encrypted and attached to the BU message; the MN encrypts the secret key using the IPSec secret key shared by the MN and the HA.

When an HA receives a BU message from an MN, it verifies the claimed CGA-based CoA. The verification process requires the inputs of an IPv6 address, a 128-bit modifier, and a shared secret key. The process is depicted in Figure 4.2 and detailed as follows:

Symmetric CGA-based Verification Algorithm:

1. Divide the IPv6 address into 64-bit subnet prefix and 64-bit interface identifier.

2. Concatenate from left the modifier and the subnet prefix. Execute the HMAC_SHA1 function on the concatenation using the secret key shared with the MN and get the leftmost 64 bits of the output. The result is H, i.e. H = First (64, HMAC_SHA1 (shared key, (modifier || subnet prefix))).

3. Compare the calculated hash value (H) obtained from step 2 with the 64-bit interface identifier obtained from step 1; differences in the U/L and I/G bits are ignored. If the calculated hash value differs, the verification fails. Otherwise, the verification succeeds.

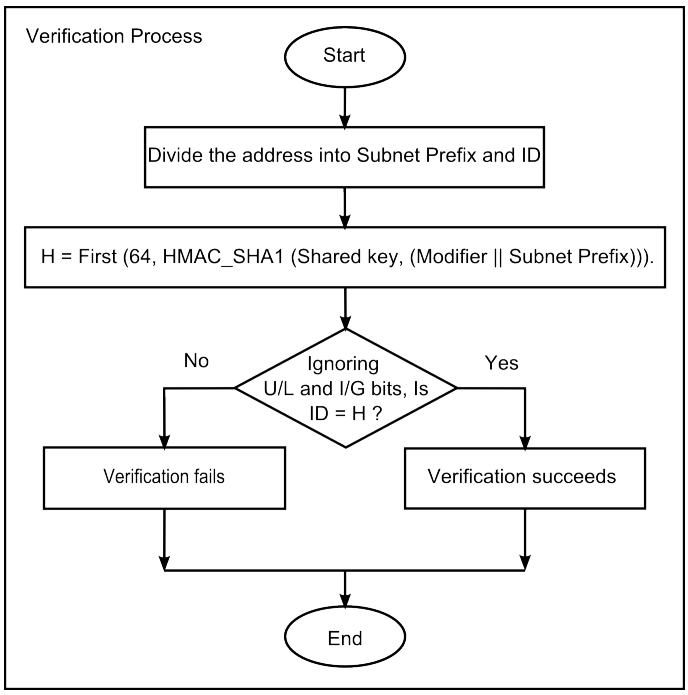

Figure 4.2: Symmetric CGA-based address verification algorithm

If the verification fails, the HA will reject the received BU message and reply with a BA message indicating that the binding is rejected due to a failure in CoA verification. However, if the verification succeeds, the HA will gain confidence that the CGA-based CoA was generated by the MN within that specific foreign link and that it either belongs to the MN itself or it is a non-used address. This is because with the symmetric CGA-based technique in place, a malicious MN will need to attempt about (2^{61}) tries to be able to produce a CoA that matches a third-party's IPv6 address.

Compared to the traditional CGA-based technique, the symmetric CGA-based technique has the following advantages. Firstly, the MN (i.e. address owner) and the HA (i.e. address verifier) are not required to generate and verify a digital signature, respectively, in order to verify the authenticity of the CoA (i.e. CGA-based address). Therefore, it reduces the computational overhead imposed on the MN and the HA, which also protects the MN and the HA against DoS attacks (see Section 3.1). Secondly, it removes the need to include the MN's public key and signature in BU messages sent to the HA to be able to verify the CoA, and this could reduce signalling overhead. On the other hand, the symmetric CGA-based technique cannot overcome the following limitations that exist in

the traditional CGA-based technique. Firstly, it does not guarantee the MN's reachability at the CoA, i.e. a malicious MN can use the shared secret key to cryptographically generate a non-used address with a subnet prefix from a victim network. Secondly, it cannot thwart attacks on an entire network by redirecting data to a non-used address. Furthermore, it suffers from the same hash-length shortage problem that exists in the early CGA-based techniques, i.e. at most, 62 bits of the address can be used for the hash. Basically, 62 bits are not sufficient to provide strong security and real protection against brute force attacks [55, 56, 57]. If a malicious MN wishes to steal a third party's IPv6 address, the MN will need to attempt about (2^{61}) tries to find a modifier that, when used with the subnet prefix and the shared secret key, produces the same address.

One possible way to increase the hash length is to use the hash extension method proposed by Aura in [33] (see Section 3.1). However, this method also increases the cost of address generation, thus increasing the latency in configuring a new CoA. Moreover, one of the hash functions included in this method is nondeterministic, i.e. not guaranteed to terminate after a certain number of iterations, thus the hash extension method is not suitable for use in the context of generating CoAs. A second possibility is to make use of the idea of lossless compression proposed by Hwang in [58]. A lossless compression algorithm allows the exact original data to be reconstructed from the compressed data; it guarantees that the original and the decompressed data are identical. When an MN generates a CGA-based CoA, it calculates a longer hash value and then compresses it to 62 bits. Specifically, in step 2 of the symmetric CGA-based generation algorithm, the MN gets the leftmost X bits of the keyed hash, i.e. H = First (X, HMAC_SHA1 (shared key, (modifier || subnet prefix))) where $64 \leq X \leq 160$. The MN then compresses H to 62 bits using a lossless compression algorithm. In the same way, when an HA verifies a CGA-based CoA, it decompresses the 62 bits from the interface identifier part of the CoA to obtain the original X bits again. The HA then compares these X bits with a freshly calculated hash value. In this way, the MN will need to attempt about (2^{X-1}) tries to be successful in stealing a third party's IPv6 address. An initial experiment that attempted to use the second technique has not resulted in any compression occurring. The reason for this is currently unknown.

4.1.2 The Concurrent CoA Reachability Test

The idea of generating CoAs cryptographically is complemented by the idea of concurrent CoAs reachability tests to validate the MNs' reachability at claimed CoAs. A concurrent CoA reachability test allows an HA to register and use an MN's new CoA while concurrently verifying an MN's reachability at that CoA. The reachability test uses two additional messages: a Binding Acknowledgement with Care-of Token (BACoT) message and a Binding Update with Care-of Token (BUCoT) message.

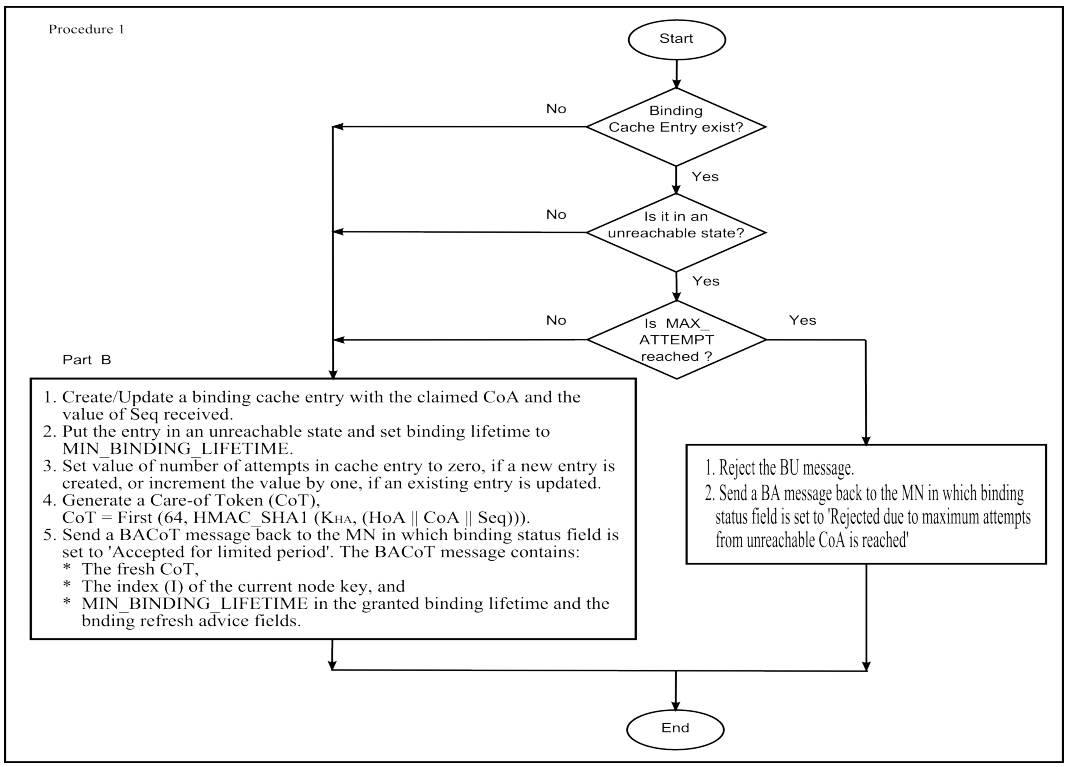

Figure 4.3: Procedure 1 - executed by an HA upon receipt of a valid BU message

The reachability test is initiated as soon as an HA receives a valid BU message from an MN. The HA replies by sending a BACoT message to the MN. The BACoT message acknowledges the binding of the new CoA and delivers a fresh care-of token to the MN. The MN uses the received token to show its presence at the new CoA, i.e. the MN sends a BUCoT message containing the received token to the HA. When the test concludes, the HA sends a BA message to the MN acknowledging the receipt of the token; hence, the successful completion of the reachability test.

A care-of token is a 64-bit number that is produced using the idea of a 'node key' [5]. The node key is only known to an HA, and it allows the HA to verify that a token enclosed in a BUCoT message is indeed its own. The HA generates a fresh node key at regular intervals and identifies it by an index. The HA produces a fresh care-of token based on its active node key as well as values of the MN's HoA, the MN's claimed CoA, and the sequence number received in a valid BU message. The HA may use the same node key with all of the MNs it is in communication with to avoid the need to store a token per MN.

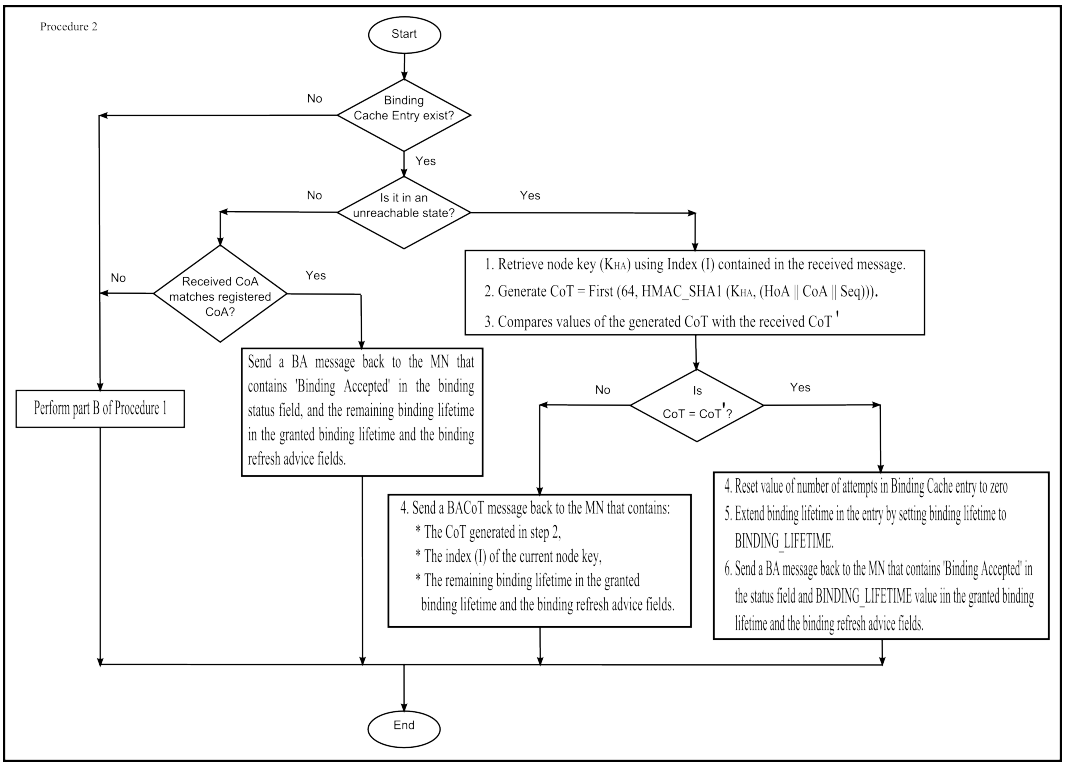

Figure 4.4: Procedure 2 - executed by an HA upon receipt of a valid BUCoT message

The reachability test limits the number of valid BU messages that can be received from an unreachable CoA as well as the amount of traffic that can be sent to an unreachable CoA. While running the test, i.e. after sending a BACoT message and before receiving a BUCoT message, an HA is not yet able to validate an MN's reachability at a claimed CoA. Therefore, the test allows the HA to limit the number of BUs that can be received from that unreachable CoA to a MAX_ATTEMPT value, which prevents the MN from bypassing the test by

continuously sending BUs without involving the test. In addition, the test allows the HA to set the granted binding lifetime between the MN's HoA and the unreachable CoA to a MIN_BINDING_LIFETIME value, which limits the amount of data sent to the unreachable CoA.

The operational procedure of the concurrent CoA reachability test is summarised as follows:

1. When an HA receives a valid BU message from an MN, the HA performs Procedure 1, which is also shown in Figure 4.3:

   Procedure 1:

   1.1 The HA checks whether it has a Binding Cache entry for the HoA enclosed in the message. If the HA does not have one (i.e. the MN wants to create a new binding), then go to step 1.4.

   1.2 The HA checks the state of the entry. If the state is reachable (i.e. the previous run of the reachability test concluded successfully), then go to step 1.4.

   1.3 The HA checks number of attempts in the entry. If MAX_ATTEMPT is reached, then the HA rejects the message and replies to the MN with a BA message in which the binding status field is set to 'Rejected due to maximum attempts from unreachable CoA has been reached'. Otherwise, the HA continues to step 1.4.

   Part B:

   1.4 The HA creates/updates a Binding Cache entry with the claimed CoA and the value of sequence number (Seq) received. The HA puts the entry in an unreachable state and sets the granted binding lifetime in the entry to a MIN_BINDING_LIFETIME value. In addition, the HA either sets the value of the number of attempts in the entry to zero (if a new entry is created) or increments it by one (if an existing entry is updated).

   1.5 The HA then uses its active node key (K_{HA}) as well as the values of the HoA, CoA, and Seq enclosed in the BU message to generate a fresh

care-of token CoT, i.e. CoT = First (64, HMAC_SHA1 (K_{HA}, (HoA || CoA || Seq))).

1.6 Finally, the HA sends a BACoT message to the MN in which the binding status field is set to 'Accepted for limited period'. The BACoT message contains the fresh CoT and the index (I) of the current node key. The granted binding lifetime and the binding refresh advice fields, in this BACoT message, are set to a MIN_BINDING_LIFETIME value, which indicates that the MN must send a BUCoT message as soon as possible.

2. Upon receipt of a valid BACoT message from an HA, the MN sends a BUCoT message, which contains the CoT and I, back to the HA requesting a longer lifetime by the binding lifetime request field.

3. Upon receipt of a valid BUCoT message from an MN, the HA performs Procedure 2, which is also shown in Figure 4.4:

Procedure 2:

2.1 The HA checks whether it has a Binding Cache entry for the HoA enclosed in the message. If the HA does not have one, then the HA considers the received message a BU message, not a BUCoT message, and responds by performing part B of Procedure 1.

2.2 The HA checks the state of the Binding Cache entry. If the state is reachable, then the HA compares the values of the CoA enclosed in the message and the CoA registered at the HA for that MN. If they are matched, which means the MN is resending the BUCoT message because it did not receive a BA message, the HA replies to the MN by sending a BA message that contains a 'Binding accepted' value in the binding status field and the remaining binding lifetime value in the granted binding lifetime field. Otherwise, if the two CoAs are not matched, the HA again considers the received message a BU message and responds by performing part B of Procedure 1. On the other hand, if the state of the entry is unreachable, the HA continues to step 2.3.

2.3 The HA uses the index (I) enclosed in the received message to retrieve its node key (K_{HA}). The HA next generates a CoT using the retrieved

node key, as well as the value of the HoA enclosed in the received message and the values of the CoA and Seq retrieved from the Binding Cache entry for that HoA, i.e. CoT = First (64, HMAC_SHA1 (K_{HA}, (HoA || CoA || Seq))).

2.4 The HA then verifies that the generated CoT matches the received CoT (**Verification-1**). If the verification succeeds, the HA will (1) reset the value of the number of attempts in the cache entry to zero; (2) extend the binding lifetime in the cache entry by setting the granted binding lifetime to a BINDING_LIFETIME value; and (3) send a BA message that contains a 'Binding accepted' value in the binding status field and a BINDING_LIFETIME value in the granted binding lifetime field. Otherwise, if the verification fails, the HA will send a BACoT message containing the generated CoT and the index (I) of the current node key to the MN. The granted binding lifetime and the binding refresh advice fields are set to the remaining binding lifetime in this BACoT message.

4. Upon receipt of a valid BA message with a 'Binding accepted' status from an HA, the MN accepts the message and the current run of the test ends.

4.1.3 The Segmenting IPv6 Address Space Method

As just discussed, by generating CoAs cryptographically and by testing MNs' reachability at claimed CoAs, HAs gain confidence that CoAs claimed by MNs match with the MNs' real locations. However, there is still a chance that a malicious MN could falsely claim a third party's address as its CoA. To do this, the third party's address must have a long lifetime, and the malicious MN must be located on the path between an HA and the third party. In this case, the malicious MN could attempt about (2^{61}) tries to cryptographically generate a CoA that matches the third party's IPv6 address. Furthermore, as the malicious MN is on the path between the HA and the third party, it would be able to intercept the BACoT message and send the BUCoT message. Therefore, the ideas mentioned in Sections 4.1.1 and 4.1.2 are complemented with the idea of segmenting IPv6 address space into three parts (home addresses, care-of addresses, and stationary addresses) to reduce the number of targets that are vulnerable to DoS attacks launched by malicious MNs.

The segmenting IPv6 address space method distinguishes the addresses used by MNs from those used by stationary nodes (SNs). Furthermore, it distinguishes the addresses used by MNs as HoAs from those used by MNs as CoAs. This can be done by dividing the IPv6 addresses into two groups: those that identify SNs (group 1) and those that identify MNs (group 2). Furthermore, group 2 can be further divided into those that identify MNs located at their home links (group 2.1) and those that identify MNs located at foreign links (group 2.2). In this way, the IPv6 addresses that are vulnerable to flooding attacks launched by malicious MNs are scoped to group 2.2 addresses. In other words, this method (i.e. segmenting IPv6 address space) on its own can protect the IPv6 addresses in group 1 and group 2.1 against malicious MNs flooding attacks.

The segmenting IPv6 address space method uses two bits of the IPv6 64-bits interface identifier field to distinguish between SNs' addresses and MNs' addresses, and between MNs' HoAs and MNs' CoAs. As shown in Table 4.1, the first bit, i.e. the Mobile/Stationary (M/S) bit, is used to indicate whether an address is for a mobile or a stationary node, and the second bit, i.e. the Home/Care-of (H/C) bit, is used to indicate whether the address is for a mobile node at its home link or at a foreign link. The M/S and H/C bits are part of the addresses; hence if a malicious MN changes them that will change the address and the address owner will not be flooded by any directed packets. Consequently, the proposed method prevents malicious MNs from impersonating either stationary nodes or other MNs located at their home links. In addition, the method does not impact the availability of the IPv6 address space. It divides the address space into two groups, but the availability of the address space remains the same.

M/S	H/C	
0	X	Stationary nodes (stationary IPv6 addresses)
1	0	Mobile nodes at foreign links (CoAs)
1	1	Mobile nodes at home links (HoAs)
X means either 0 or 1.		

Table 4.1: M/S and H/C bits

The use of the proposed method must be deployed on a global scale on the IPv6 Internet. It requires changing the way every IPv6 node in the world chooses an IPv6 address, which seems unrealistic. However, the author can argue this requirement as follows. Firstly, "IPv6 is still in its infancy in terms of general

worldwide deployment" [59, 60] which makes it possible to be changed to support the proposed method. Specifically, 5.5% of networks on the Internet could handle IPv6 traffic by early 2010. Additionally, only 1.45% of the top 1000 websites had an IPv6 website in January 2010. Secondly, the current IPv6 lacks any way to know about a node from its address making it necessary to find a way to differentiate between nodes, especially with the rapid growth of the number of mobile devices connected to the Internet. Thirdly, the author believed that the benefits of the proposed method far outweigh the costs, as it will not only be used to support location authentication of mobile nodes but will also be used in other applications to differentiate between redirectable and non-redirectable IPv6 addresses, such as in MIPv6 route optimization, in protecting against future address stealing [61], and in future protocols that allow redirecting of IP packets from one IPv6 address to another one.

In summary, in the context of supporting location authentication of MNs to HAs, the segmenting IPv6 address space method could protect nodes that use stationary IPv6 addresses as well as MNs' HoAs from being attacked as the result of using MIPv6 protocol. This is because, with this method in place, it is not possible for an MN to falsely claim that a SN's address or another MN's HoA is its CoA.

4.2 The EHR Protocol Description

The EHR protocol is based on three ideas that (1) cryptographically generate MNs' CoAs based on a shared secret key; (2) verify MNs' reachability at claimed CoAs; and (3) differentiate between different address types. The EHR protocol adds the three ideas mentioned above to the basic home registration protocol (see Section 2.4) to help HAs authenticate MNs' CoAs. The whole picture of the EHR protocol is illustrated in Figures 4.5 and 4.6 and is detailed as follows.

1. When the MN roams into a foreign link, it uses a secret key along with the foreign link's subnet prefix to configure a new CoA using the symmetric CGA-based generation algorithm. The MN also sets the M/S and the H/C bits in the new CoA to one and zero, respectively. The MN then sends a BU message to notify the HA about the new CoA. The BU message contains a 128-bit number representing the value of the modifier generated by the MN while running the symmetric CGA-based generation algorithm. In addition,

if the current run of the EHR protocol is to create a new binding at the HA, the secret key is encrypted and carried in the BU message. If the MN does not receive a matching response within a retransmission interval of one second, the MN will resend the BU message to the HA. The MN doubles the retransmission interval upon each retransmission in the same way, as specified in the base specification of the MIPv6 protocol ([5], see also Section 2.4).

2. Upon receipt of a BU message, the HA verifies the authenticity, integrity, and freshness of the message, using an IPSec SA and a sequence number, as in the basic home registration protocol. If any of these verifications fails, the HA will discard the received message without any further action. Otherwise, the HA will check that the values of the M/S and H/C bits in the claimed CoA are 'one' and 'zero', respectively. If positive, the HA will execute the symmetric CGA-based verification algorithm described in Section 4.1.1 (see Figure 4.2). If the outcome of the algorithm is positive, the HA will perform Procedure 1 described in Section 4.1.2 (see Figure 4.3), which results in either (1) rejecting the received BU message and sending a BA message to the MN with the binding status field set to 'Rejected due to maximum attempts from unreachable CoA has been reached', or (2) accepting the received BU message and sending a BACoT message to the MN with the binding status field set to 'Accepted for limited period'. Otherwise, if the values of the M/S and H/C bits are not 'one' and 'zero', respectively, and/or the address verification fails, the HA will reject the received BU message and reply with a BA message in which the binding status field is set to 'Rejected due to failure in CoA verification'.

3. Upon receipt of a BACoT message, the MN verifies the authenticity, integrity, and freshness of the message as in the basic home registration protocol. If positive, the MN will send a BUCoT message, which contains the CoT and I, back to the HA requesting a longer binding lifetime. Otherwise, if any of these verifications is negative, the MN will discard the received message without any further action. If the MN does not receive a matching response within a retransmission interval of one second, the MN will resend the BUCoT to the HA.

4. Upon receipt of a BUCoT message, the HA verifies the authenticity, integrity, and freshness of the message as in the basic home registration protocol. If any of these verifications is negative, the HA will discard the received message without any further action. Otherwise, the HA will perform Procedure 2 (described in Section 4.1.2, see Figure 4.4), which results in either (1) accepting the received BUCoT message and sending a BA message to the MN with the binding status field set to 'Binding Accepted', or (2) accepting the received BUCoT message and sending a BACoT message to the MN with the binding status field set to 'Accepted for limited period'.

5. Upon receipt of a BA message, the MN verifies the authenticity, integrity, and freshness of the message as in the basic home registration protocol. If negative, the MN will discard the received message without any further action. Otherwise, the MN will check the status of the message. If the status is 'Binding accepted', the MN will accept the message and the current run of the protocol will end. Otherwise, the MN will take steps to fix the error and retransmit the BU message, or will reinitiate the home registration by instead trying a different HA.

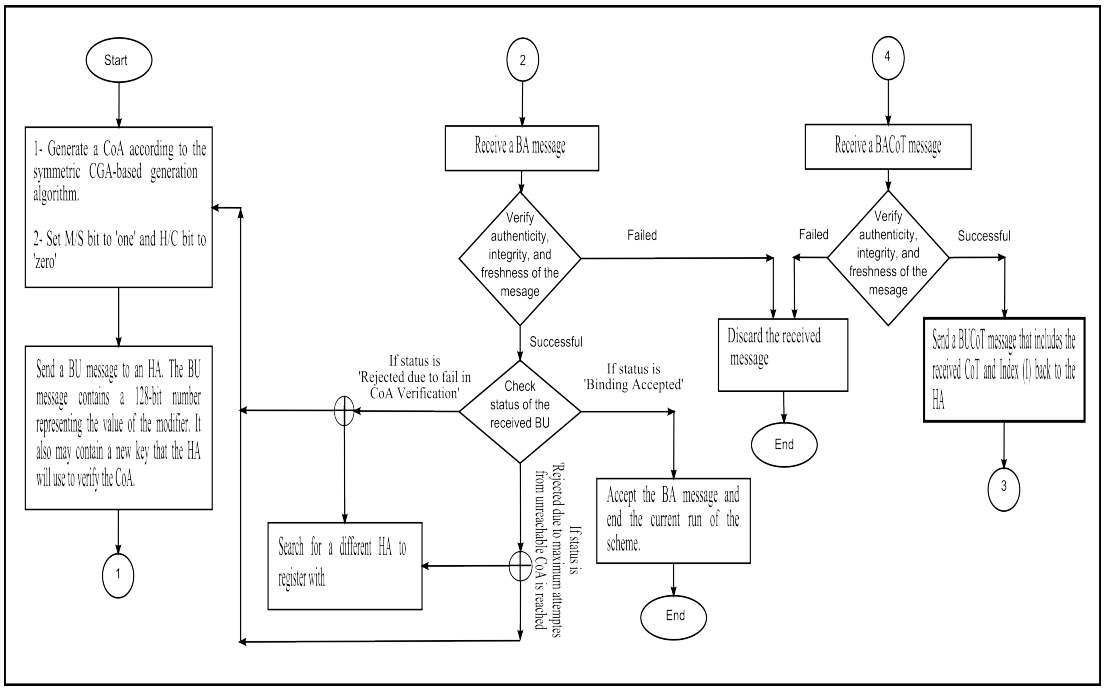

Figure 4.5: EHR protocol at mobile node side

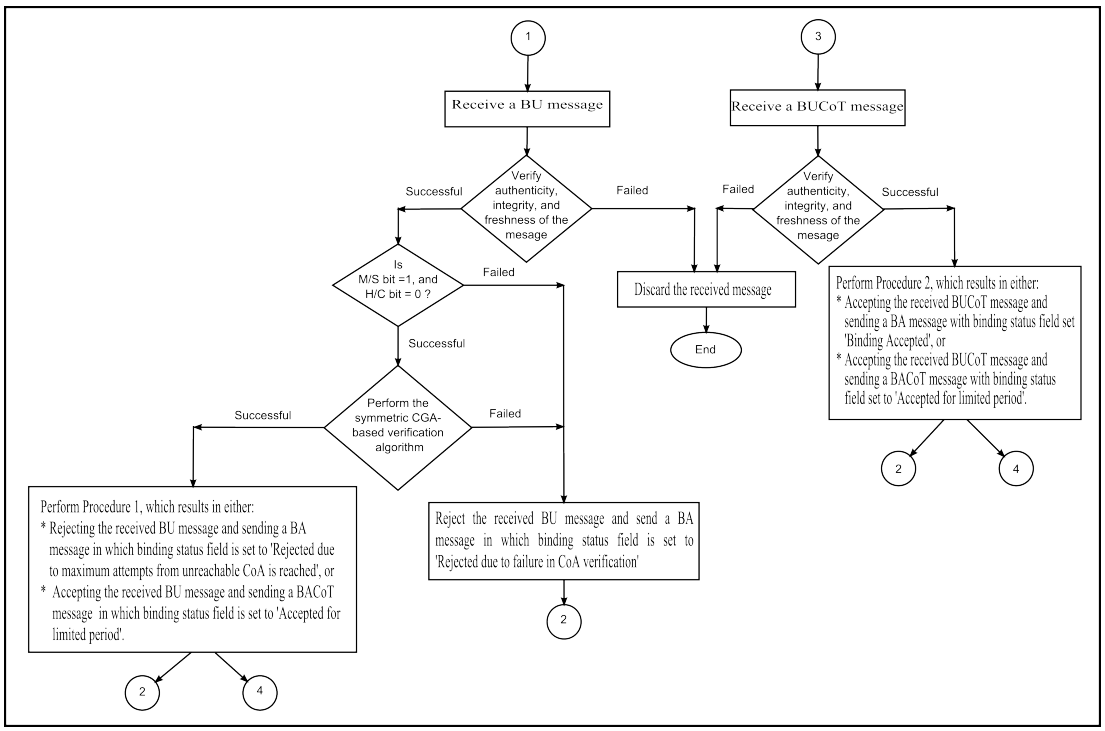

Figure 4.6: EHR protocol at home agent side

The EHR protocol is based on the BHR protocol; it also uses IPSec ESP and sequence numbers to protect home registrations. Therefore, the EHR protocol has the same security protection as the BHR protocol. Specifically, it can protect home registrations against outsider attacks; an attacker cannot send a spoofed or a replayed BU message instead of the MN. It also can prevent malicious MNs from falsely sending BU messages on behalf of other MNs.

Furthermore, the EHR protocol extends the BHR protocol to support the location authentication of MNs to their HAs. It adds the novel ideas of segmenting the IPv6 address space, using a symmetric CGA-based technique for generating CoAs, and applying concurrent CoAs reachability tests to the basic home registration protocol. As a result, the EHR protocol reduces the likelihood of a malicious MN being successful in luring an HA to flood a third party. Specifically, when the BHR protocol is in place, a malicious MN can be anywhere on the Internet and launch an attack against any node; all Internet nodes are potential targets. On the other hand, when the EHR protocol is in place, a malicious MN would need to (1) attempt about (2^{61}) tries to be able to produce a CoA that matches a third-party's IPv6 address; and (2) be on the path between the HA and the third

party. In addition, the malicious MN cannot target stationary nodes or other MNs at their home links.

4.3 Performance Evaluation

This section reports the performance evaluation of the EHR protocol by comparing it to the basic home registration (BHR) protocol. This is done by using the OPNET Modeler simulation package and the CryptoSys Cryptography Toolkit. A brief description of these is given in Appendix C. The performance is measured in terms of home registration delay (HR-Delay) measured in seconds and control signalling overhead measured in bits per second. The HR-Delay is defined as the total amount of time taken for the MN to receive an acknowledgement message (i.e. a BACoT in the EHR protocol or a BA in the BHR protocol) from the HA, once a BU message has been sent. The control signalling overhead is the total amount of Mobile IPv6 signalling traffic sent and received by the MN and the HA.

4.3.1 Simulation Modelling

In order to measure the performance of the EHR protocol, OPNET^{TM} Modeler version 14.5 has been used to simulate the performance of the protocol under varying network conditions. In particular, the performance is investigated when varying levels of background traffic on the network are applied and when various numbers of simultaneously roaming MNs are served by the same HA. The simulation results obtained are then compared to those when the BHR protocol is run.

The network model, depicted in Figure 4.7, is composed of three CNs that are connected via routers (R1 to R3) to the Internet. An HA and three access routers (AR1 to AR3) each one representing a different IPv6 subnet are also connected to the Internet. MNs, initially located at a home subnet, move to foreign subnets and return back to the home subnet along the same route. When the MNs are away from the home subnet, they register with the HA; i.e. the same HA serves all the MNs. The MNs are initially located at the same position; move from one subnet to another at the same speed; and wait the same interruption time at each subnet before moving to the next subnet. In this way, all the MNs will perform

handoff and consequently initiate the home (de)registration process at the same time.

The HA, the Rs, and the ARs are placed at a fixed distance of 1 km (kilometre) apart from the Internet. The value of 1 km has been chosen to make the validation of the simulation model straightforward, but without loss of generality. The HA and the ARs have been positioned in such a way that provide a continuous wireless coverage area for the MNs. Each MN is communicating with the three CNs at the same time and running three Internet applications, i.e. web browsing, email and file transfer. These applications are selected because they are likely to be the dominant applications in the Internet [62]. The MNs perform one hundred passes (movement between HA and AR3) with six handoffs in each pass (five registrations and one deregistration). The OPNET Modeler's documentation recommends thirty as an initial rule of thumb for the number of repetitions to use. However, one hundred passes have been chosen (i.e. five hundreds registrations) in the hope of averaging out any possible fluctuating factors.

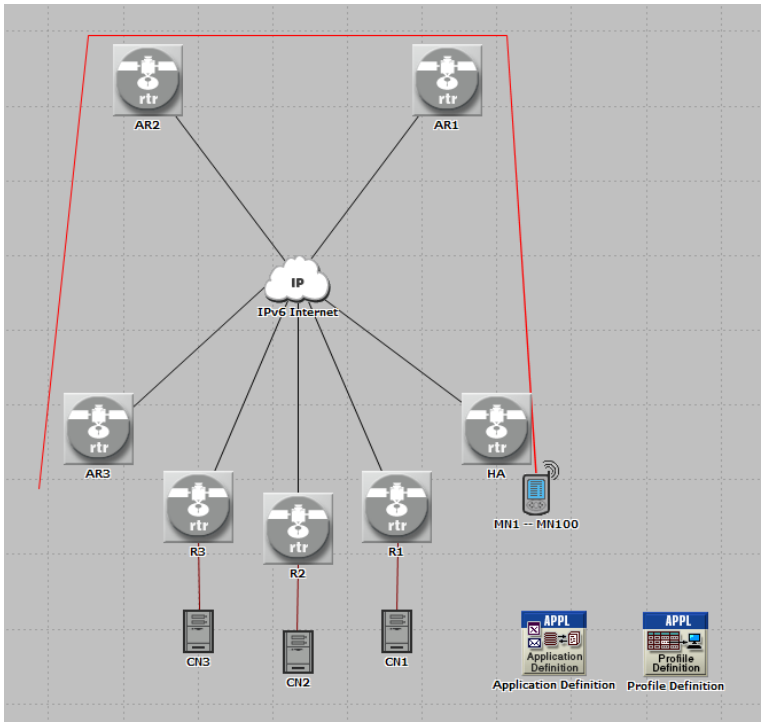

Figure 4.7: Simulation model

Three separate simulation studies are performed. The first one is created to investigate the impact of the EHR protocol on HR-Delay, i.e. whether the EHR

protocol has led to any performance degradation through increasing HR-Delay. In this study, ten background traffic scenarios ranging from 0% to 90% at 10% increments are applied to investigate how different volumes of traffic may affect the results of the HR-Delay. The second simulation study is to investigate the impact of number of MNs served by an HA on the performance of the EHR protocol. In this study, the number of simultaneously roaming MNs varied between one and one hundred as this is the default value for the maximum number of MNs that can be served by one HA (maximum Binding Cache size). The third simulation study is to investigate the control signalling overhead produced at the MN and the HA when both of EHR and BHR protocols are executed.

4.3.2 Simulation Model Validation

This section validates the simulation model so that any collected results could be considered reliable. For doing so, a validation process that consists of two phases is adopted. The first phase involves the use of the OPNET debugger to prove that the EHR protocol operates correctly. The OPNET debugger is applied to output the processes of generating and verifying both a CoA and a CoT. In addition, relevant packets' information (i.e. source address, destination address, value of modifier, value of CoT, and packet size) has been inspected during runtime. The second phase involves deriving a theoretical equation for the calculation of the HR-Delay under the condition that the underlying network is unloaded, i.e. the only traffic in the network is the home registration signalling messages. Next theoretical results are calculated. These are compared to the simulation results collected under the same conditions to validate the correctness of the simulation model.

4.3.2.1 The Validation Process: Phase One

The output from the OPNET debugger during simulation runtime is illustrated in Appendix C. Trace labels have been defined in the code to request output of specific information while monitoring simulation output. By observing the output, the messages exchanged were confirmed to be consistent with the expected mobility signalling exchanged between the MN and the HA.

4.3.2.2 The Validation Process: Phase Two

A simplified simulation model and a theoretical validation model are constructed to measure and calculate, respectively, the HR-Delay for the EHR protocol. The theoretically calculated results are compared to the simulation results collected under the same conditions, i.e. under the same assumptions and parameter value settings, to validate the correctness of the simulation model.

Theoretical Model

For the purpose of simulation and validation, the equation for calculating theoretical value of the HR-Delay is as follows:

$$\begin{aligned} HR-Delay &= \text{Delay for BU message} + \text{Delay for BACoT message} \\ &+ \text{Delay for HoA DAD test} \end{aligned} \quad (4.1)$$

The delays for BU and BACoT messages are functions of four delays, i.e. transmission delay, propagation delay, queuing delay, and processing delay. The BU and BACoT messages are exchanged between various nodes before they reach their final destination. Figures 4.8 and 4.9 illustrate the HR-Delay for each node involved in processing and/or forwarding of the BU and BACoT messages. These figures illustrate how the various delays that are experienced at each node are combined together to calculate the total HR-Delay for the validation model. The remainder of this section details how theoretical values for the five main delays are calculated.

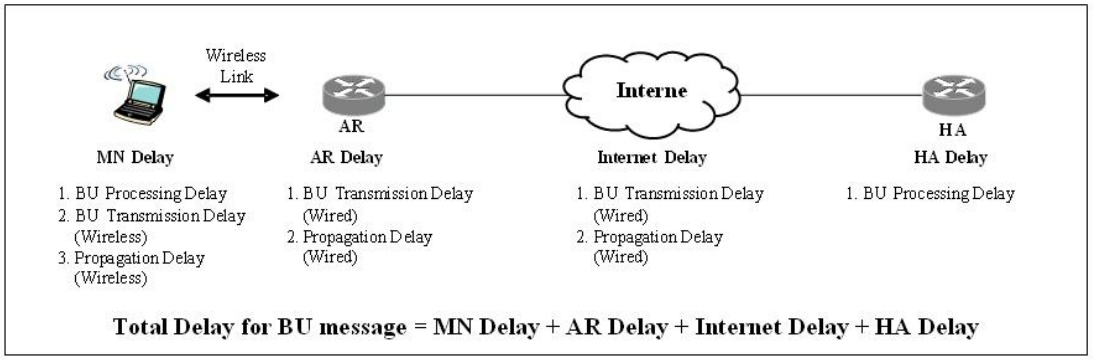

Figure 4.8: Theoretical delay for BU message

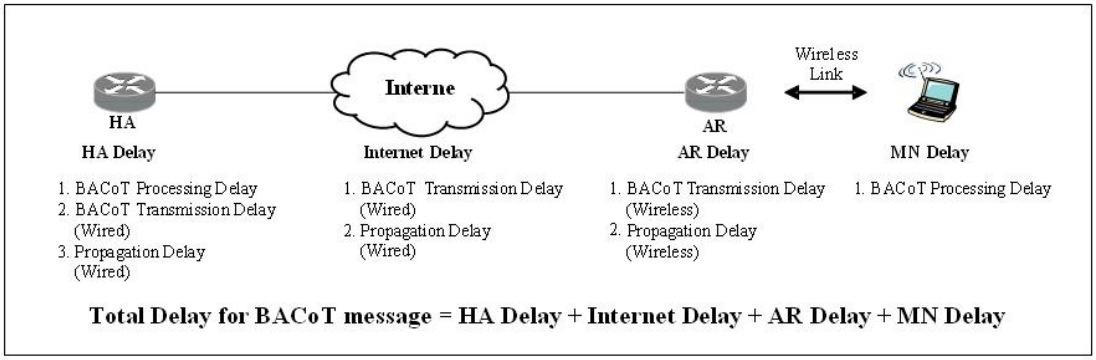

Figure 4.9: Theoretical delay for BACoT message

- **Transmission Delay**

  Transmission delay is the amount of time required to transmit all of the packet's bits onto the link. The equation for calculating transmission delay is as follows:

$$Transmission - Delay \quad = \quad \text{Packet Size / Bandwidth} \tag{4.2}$$

  Packet Size refers to the total amount of bits in a packet; whereas, Bandwidth specifies the data transmission rate of a link.

- **Propagation Delay**

  Propagation delay is the time taken by the packet's bits to propagate from one network node to another. The equation for calculating propagation delay is as follows:

$$Propagation - Delay \quad = \quad \text{Distance / Propagation Speed} \tag{4.3}$$

  Distance in metres refers to the distance between the two network nodes. Propagation Speed is usually equivalent to the Speed of Light which equals 300,000,000 metres per second in a vacuum (wired-link) and equals 299,702,547 metres per second in airspace (wireless-link).

- **Queuing Delay**

  Queuing delay consists of message arrival queuing delay and message transmission queuing delay. The former is the amount of time that an arrived message waits at the inbound network interface before being processed. The latter is the amount of time that a processed message spent waiting at the outbound network interface for transmission. The two delays depend entirely on the level of packet congestion at the interface. By assuming a 'zero' background

traffic, there are no extra packet queued at either the inbound interface or the outbound interface, hence queuing delays are considered as insignificant and negligible in this case. Therefore, the queuing delay is set to zero in the theoretical validation model.

- **Processing Delay**
  Processing delay refers to the amount of time taken to process an outgoing or incoming packet at the sender or receiver node respectively. The processing delay for the EHR protocol is based on a HMAC_SHA1 delay. The HMAC_SHA1 delay depends on processor speed whereby an increase in processor power leads to a decrease in the HMAC_SHA1 delay and vice versa. The HMAC_SHA1 function is used by the HA twice to verify MN's CoA and to generate a fresh CoT. The HMAC_SHA1 delay is measured on a DELL computer with the following configuration:

Operating System:	Window XP professional
Installed Memory:	2.00 GB
Processor Class:	Pentium D
Processor Speed:	3.00 GHz.

  1. The HMAC_SHA1 delay for generating ID of CoA = 5.011 microseconds
  2. The HMAC_SHA1 delay for generating CoT = 4.966 microseconds

- **HoA Duplicate Address Detection (DAD) Delay**
  HoA DAD delay is the amount of time taken by an HA, upon receipt of a BU message, to perform a DAD test on the MN's home link before sending back a BACoT message. The HA performs the DAD test to ensure that no other node on the home link is using the MN's HoA when the BU message arrives. The HA needs to perform a DAD test for an MN's HoA when it creates a new binding for that HoA. In other words, if the HA already has a Binding Cache entry for a HoA, it means that the HA has already performed the required DAD test before and there is no need to repeat it. The equation for calculating HoA DAD delay is as follows [63]:

$$\text{HoA DAD Delay} = T_{RT} + T_{WT} \tag{4.4}$$

  T_{RT} is the delay time in seconds that the HA must randomly wait after receiving a BU message before initializing the DAD test. Whereas, T_{WT} is the waiting time in seconds that the HA must wait after sending a neighbour solicitation message that includes MN's HoA, as the solicitation's target address,

without receiving a neighbour advertisement reply to indicate DAD test success. According to [20] the value of T_{RT} is between 0 and 1000 milliseconds and the value of T_{WT} is about 1000 milliseconds. Consequently, the minimum value of HoA DAD Delay is 1.0 seconds and the maximum value is 2.0 seconds. Therefore the theoretical value of the HoA DAD Delay is calculated as 1.5 seconds when the MN is roaming from the home link to a foreign link and is 0 seconds when the MN is either roaming from a foreign link to another foreign link or roaming back from a foreign link to the home link.

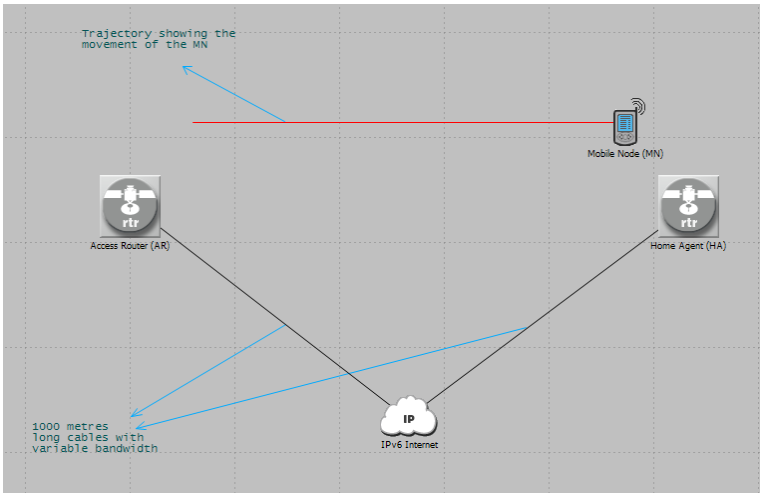

Figure 4.10: Simplified simulation model

Simplified Simulation Model

The simplified simulation model, depicted in Figure 4.10, is composed of a single MN that is located at a home link, an HA that represents the home link, and an access router (AR) that represents a foreign link. The model has the following assumptions. (1) The underlying network is unloaded, i.e. background traffic is set to 0%. (2) The HA and the AR are placed at a fixed distance (wired-distance) of 1000 metres apart from the Internet. (3) The distance between the MN and the AR (wireless-distance) is 300 metres. (4) The propagation speed in wired-links is set to 300,000,000 metres per second. (5) The propagation speed in wireless-links is set to 299,702,547 metres per second. (6) The IPv6 Internet cloud has a single router. (7) The amount of time taken by the HA to perform a DAD test on the MN's HoA is set to 1.5 seconds. (8) The secret key used in the CoA generation and verification processes is shared between the MN and the HA in advance. (9)

The wired-link bandwidth and the wireless-link bandwidth are variable, i.e. a different set of wired and wireless data transmission rates are applied.

Comparing Theoretical and Simulated Results

To validate the simulation model, Equation 4.1 is used to calculate a theoretical value for the HR-Delay at 0% background traffic load. This value is then compared to that obtained when the simulation model is run at 0% load. In order to further confirm the accuracy of the simulation model, it was validated at different wireless and wired data transmission rates. The validation results are shown in Figures 4.11 and 4.12. The results show a gap between the theoretical results and the simulation results. This gap is probably due to the processing delay that was used in the theoretical model. To validate this suspicion, the processing delay from Intel Pentium-133 was obtained and used to produce some adjusted theoretical results illustrated in Figures 4.13 and 4.14. The HMAC_SHA1 Delay was re-measured on a DELL computer with the following configuration:

Operating System:	Window 95
Installed Memory:	16 MB
Processor Class:	Pentium I
Processor Speed:	133 MHz.

1. The HMAC_SHA1 delay for generating ID of CoA = 155.47 microseconds

2. The HMAC_SHA1 delay for generating CoT = 155.42 microseconds

The adjusted theoretical results show that the gap between theoretical and simulation results has narrowed; hence, it confirmed that this insignificant gap is due to the value of the processing delay that was used in the theoretical model. As a result of the success of the adopted validation process, the simulation model has been successfully validated; hence any results collated by the simulation could be considered reliable.

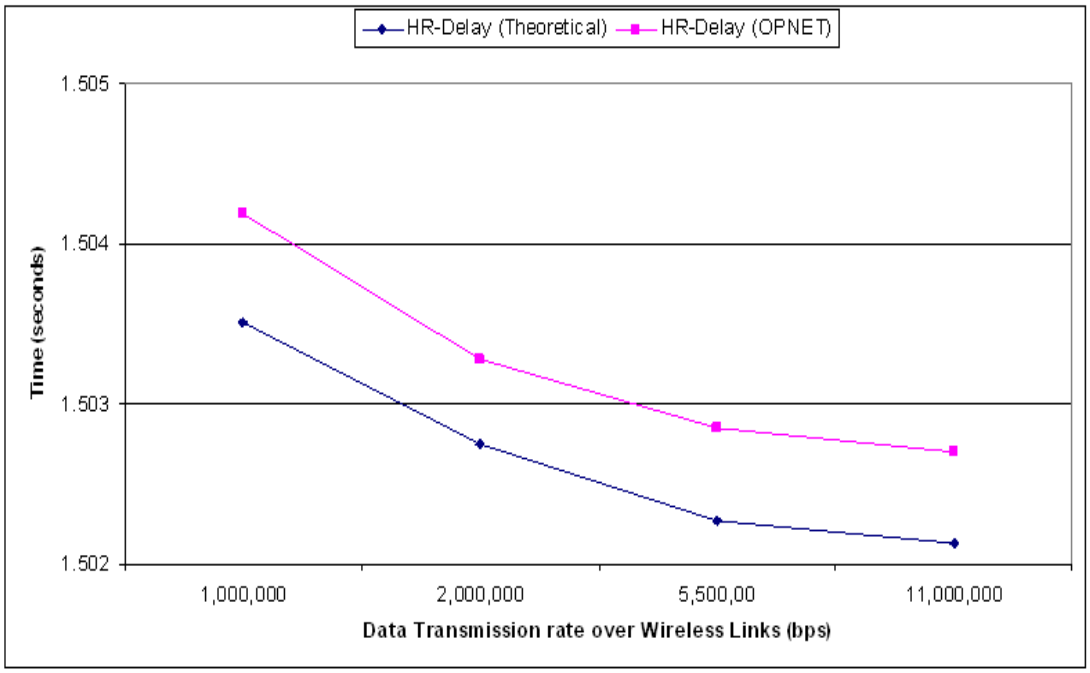

Figure 4.11: Theoretical and simulated results for HR-Delay at different wireless links' data transmission rates (wired data transmission rate is 1,544,000 bps)

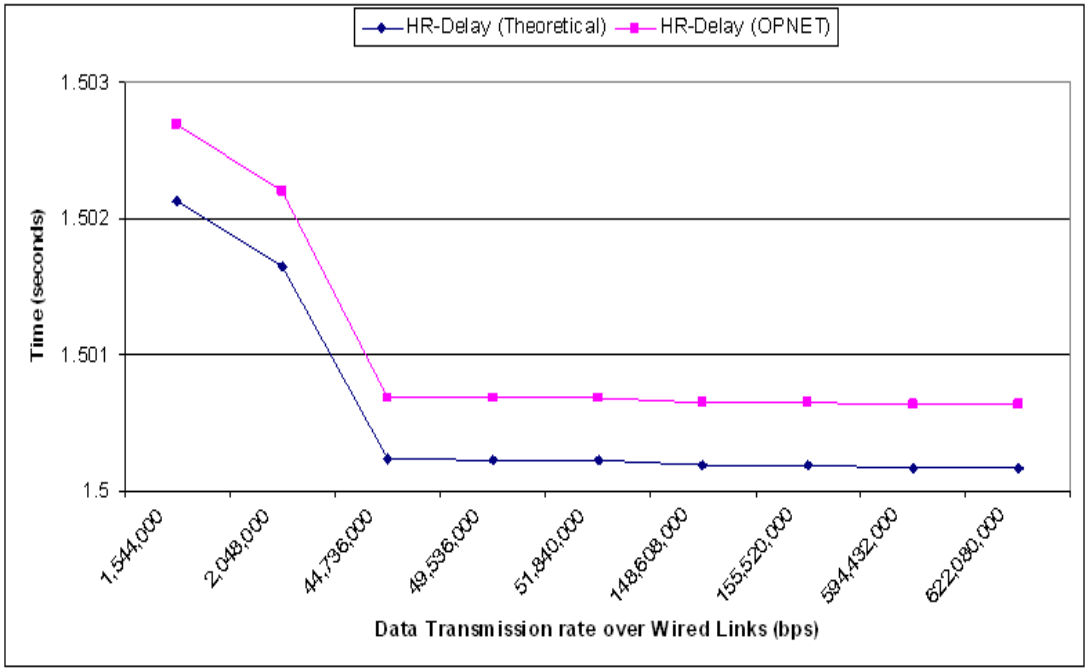

Figure 4.12: Theoretical and simulated results for HR-Delay at different wired links' data transmission rates (wireless data transmission rate is 11,000,000 bps)

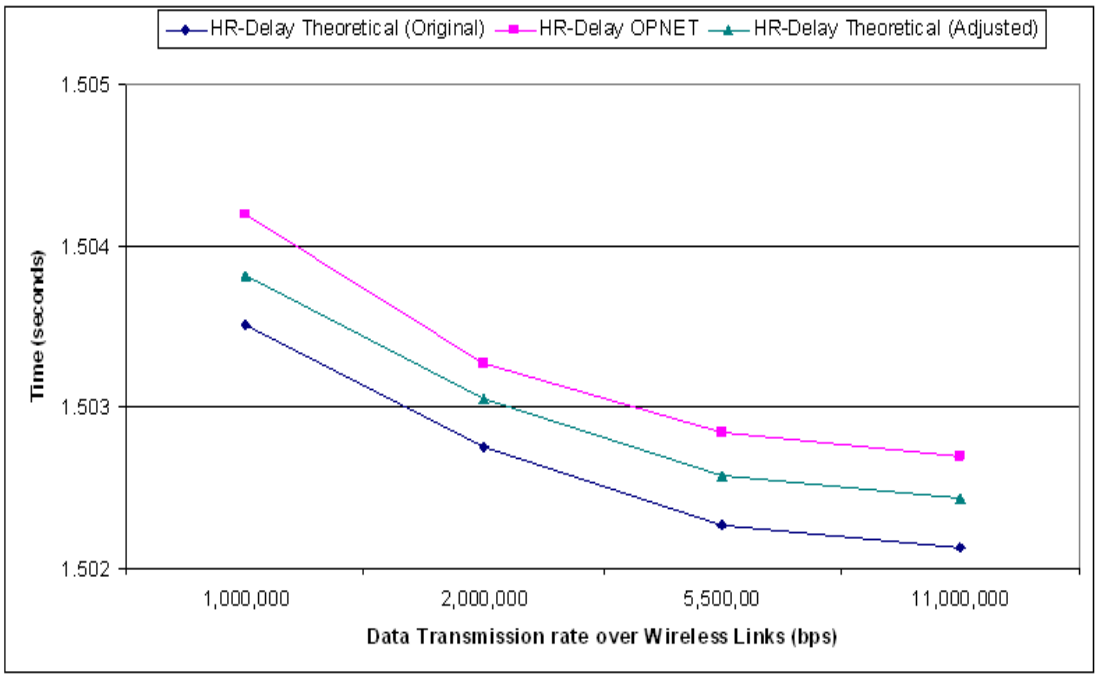

Figure 4.13: Adjusted validation results at different wireless data transmission rates (wired data transmission rate is 1,544,000 bps)

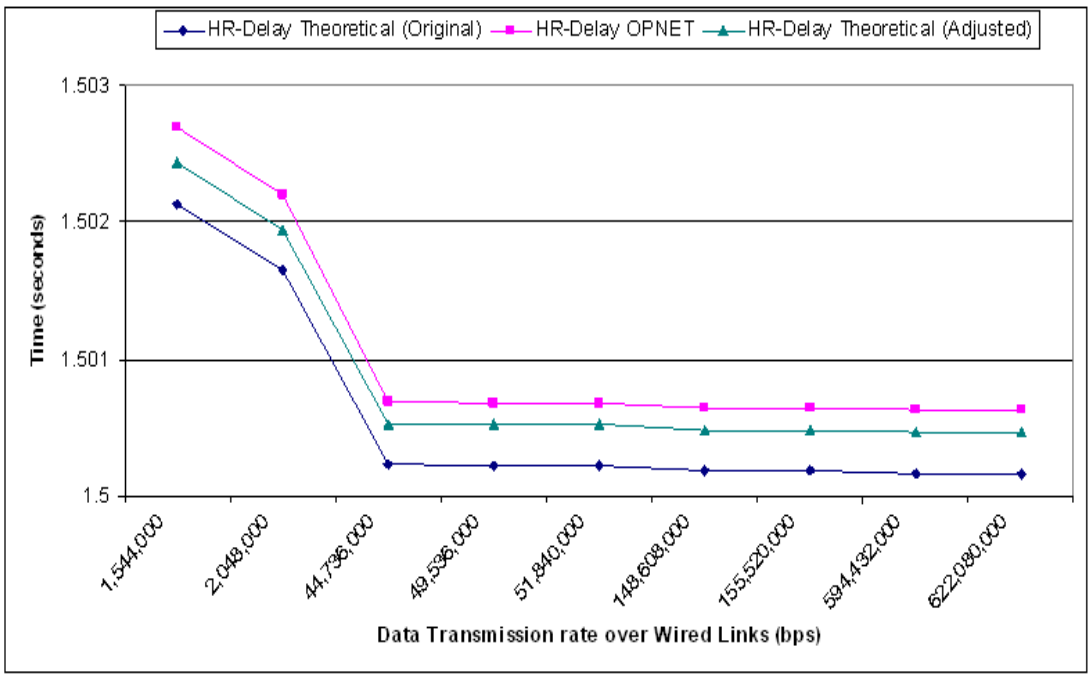

Figure 4.14: Adjusted validation results at different wired data transmission rates (wireless data transmission rate is 11,000,000 bps)

4.3.3 Simulation Results

This section presents and analyses simulation results obtained from the simulation study of both the HR-Delay and the control signalling overhead. It compares the results of both the EHR protocol and the BHR protocol.

4.3.3.1 Home Registration Delay

This section presents an analysis of the HR-Delay simulation results. A selection of the simulation results are shown in Figure 4.15 to Figure 4.25. In these results, the HoA DAD delay is set to zero even during the first registration of a CoA at an HA, i.e. when an MN roams from a home subnet to a foreign subnet. The reason for this is that the OPNET calculates this delay using a random number generator, which affects the accuracy of the collected results.

Figures 4.15, 4.16 and 4.17 show the HR-Delay during registration (handovers 1, 2, 3, 4, and 5) and deregistration (handover 6) processes at different network loads. The figures show that (1) in each of the two protocols, the HR-Delays at handovers number 1, 2, 3, 4, and 5 are nearly identical, i.e. during registration process; (2) the EHR protocol has a higher HR-Delay than the BHR protocol at handovers number 1, 2, 3, 4, and 5, i.e. during registration process; and (3) the HR-Delay is identical in the two protocols at handover number 6, i.e. during deregistration process. The reason for this is that in the deregistration process the two protocol operate the same way, i.e. there is no difference between them.

In order to further compare performance of the EHR protocol against performance of the BHR protocol, the average HR-Delays for registration and deregistration at different network loads are measured and illustrated in Figure 4.18 and Figure 4.19, respectively. As shown, the HR-Delays for registration and deregistration increase exponentially as the network load increases. This pattern is caused by the increase in the queuing delay experienced at each node. Figure 4.18 shows that, on average, the EHR protocol takes 3.76% longer than the BHR protocol to allow an MN to register a CoA. This is caused by (1) the additional two (computationally light) HMAC_SHA1 operations performed by the HA, and (2) the increase in the size of the BU and BACoT messages exchanged in the EHR protocol compared to the BU and BA messages exchanged in the BHR protocol. Figure 4.19 shows that, on average, the EHR protocol is identical to the BHR protocol in allowing the MN to deregister the CoA.

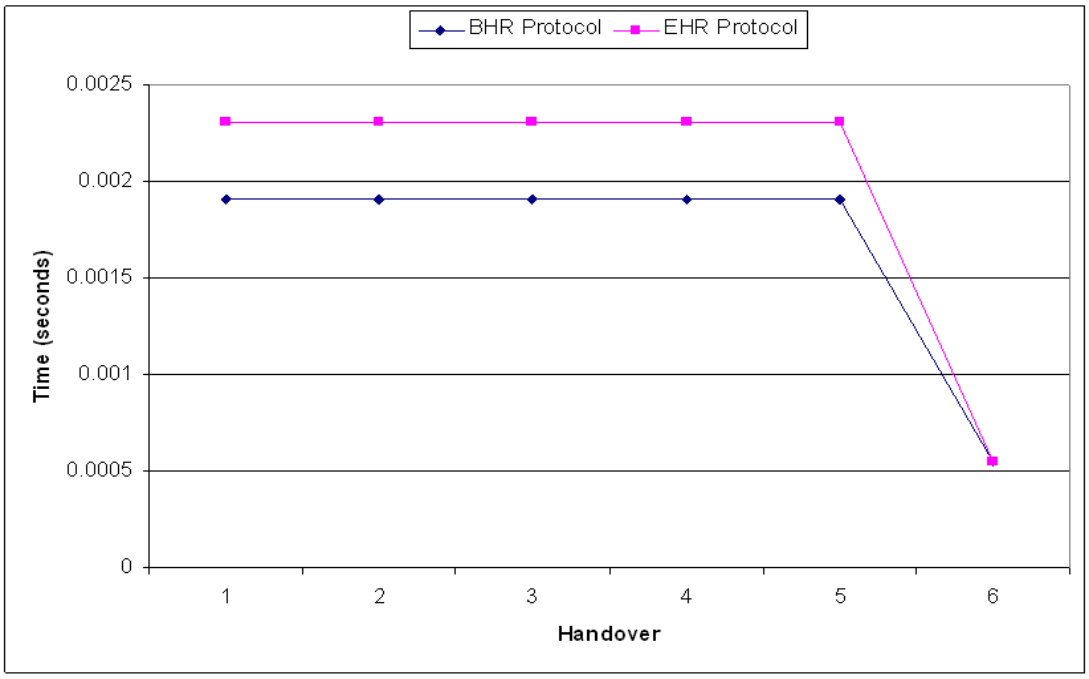

Figure 4.15: HR-Delay for BHR and EHR protocols vs. handover (one MN, three CNs, 0% load)

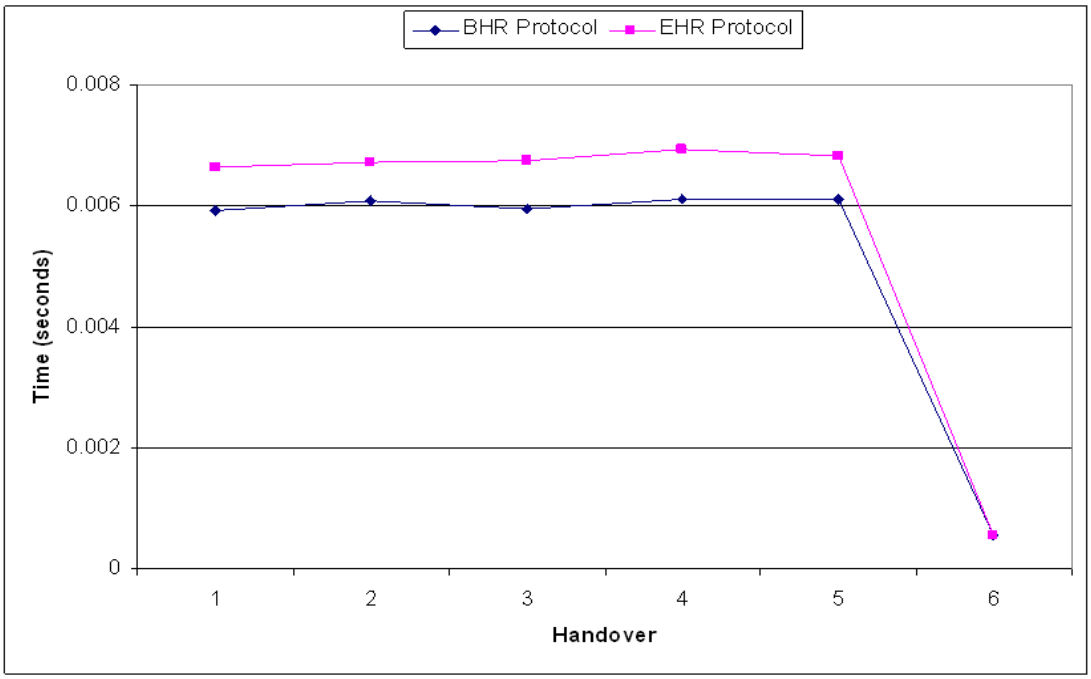

Figure 4.16: HR-Delay for BHR and EHR protocols vs. handover (one MN, three CNs, 30% load)

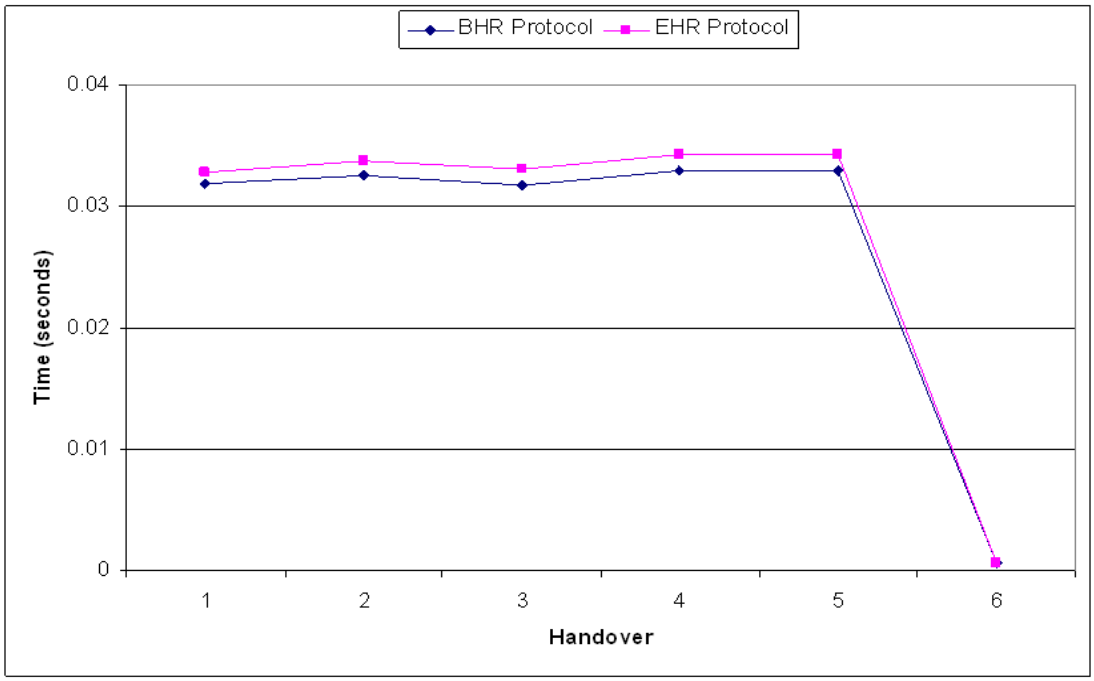

Figure 4.17: HR-Delay for BHR and EHR protocols vs. handover (one MN, three CNs, 80% load)

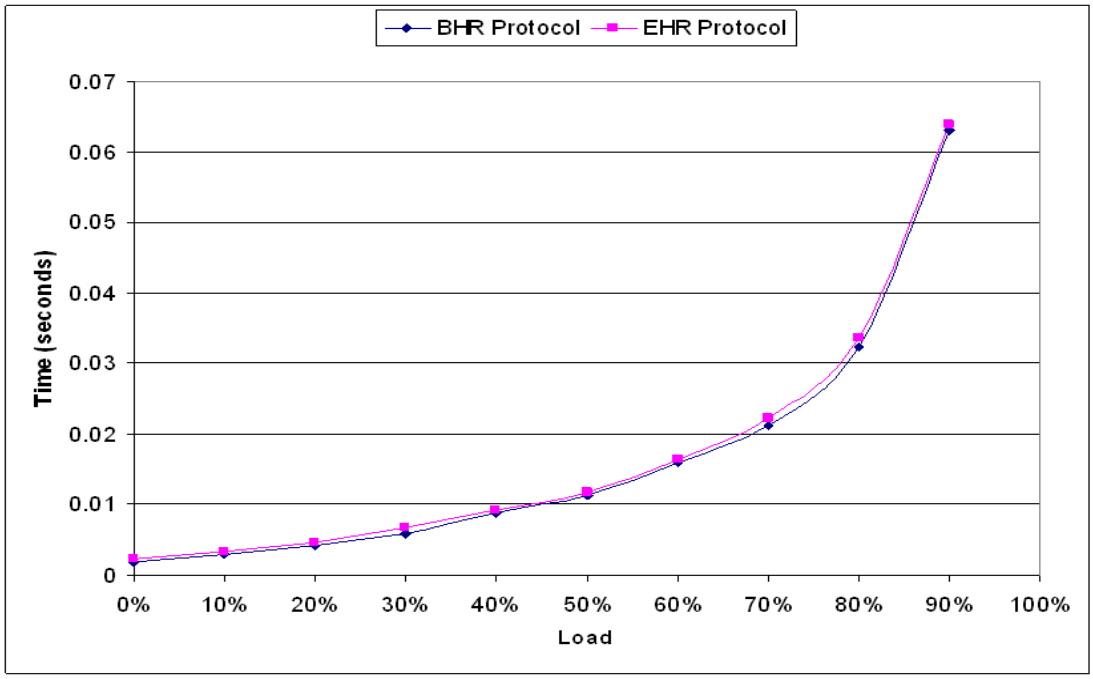

Figure 4.18: Average HR-Delay (registration) for BHR and EHR protocols vs. load (one MN, three CNs)

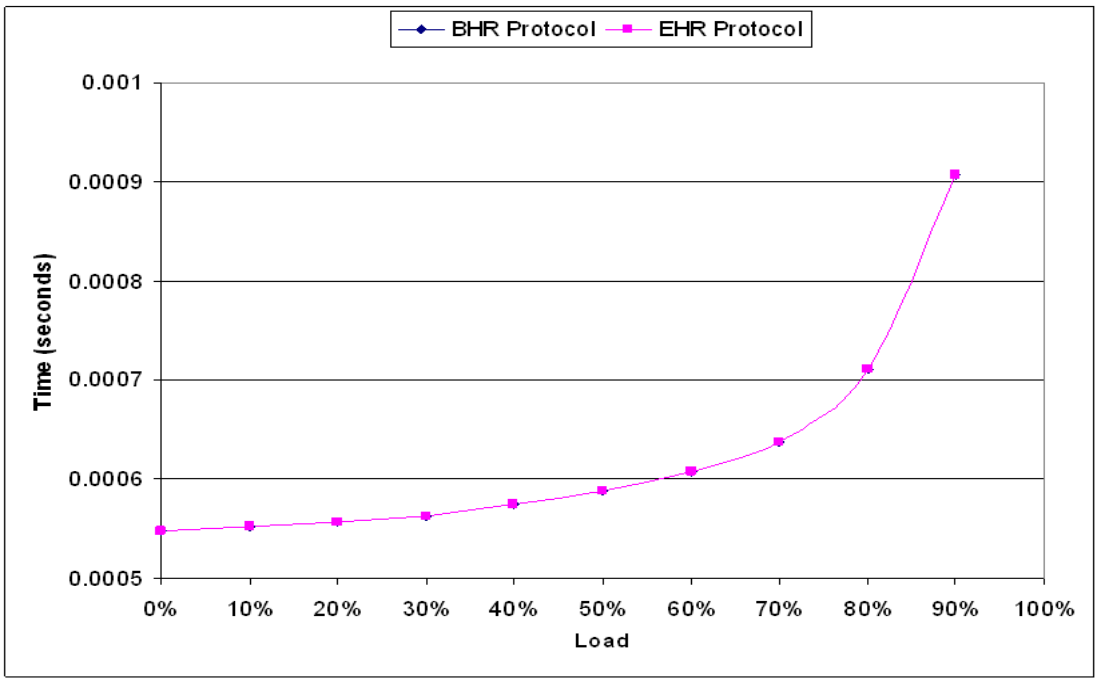

Figure 4.19: Average HR-Delay (deregistration) for BHR and EHR protocols vs. load (one MN, three CNs)

The effect of the number of simultaneously roaming MNs on the performance of the EHR protocol is depicted in Figures 4.20 to 4.23. In these figures, the mean result (Average), the minimum result (Min), the maximum result (Max), the mean result minus the standard deviation (Lower), and the mean result plus the standard deviation (Upper) are all presented. It can be seen from these figures that the HR-Delays for registration and deregistration increase linearly as the number of simultaneously roaming MNs increases. That is, the HR-Delay is proportional to the number of simultaneously roaming MNs. In addition, it can be seen that as the number of simultaneously roaming MNs increases, the deviation from the mean result increases. These are caused by the increased queuing time on the HA side, as a high number of simultaneously roaming MNs means a high number of BU messages to be processed at the HA.

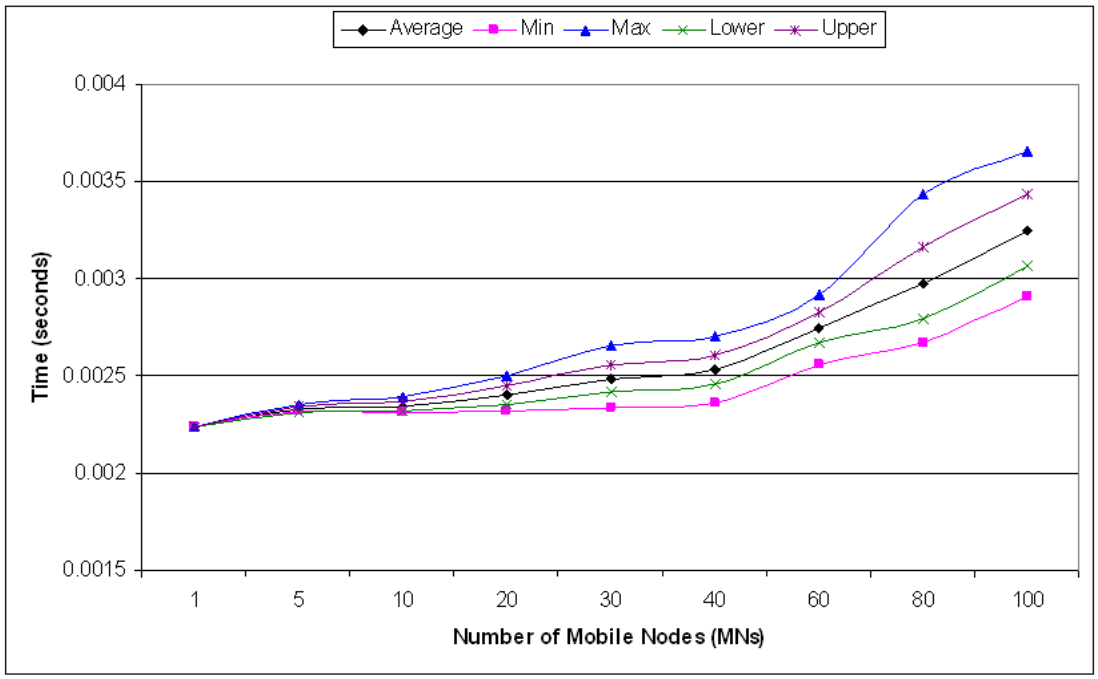

Figure 4.20: HR-Delay (registration) for EHR protocol vs. number of MNs (0% load)

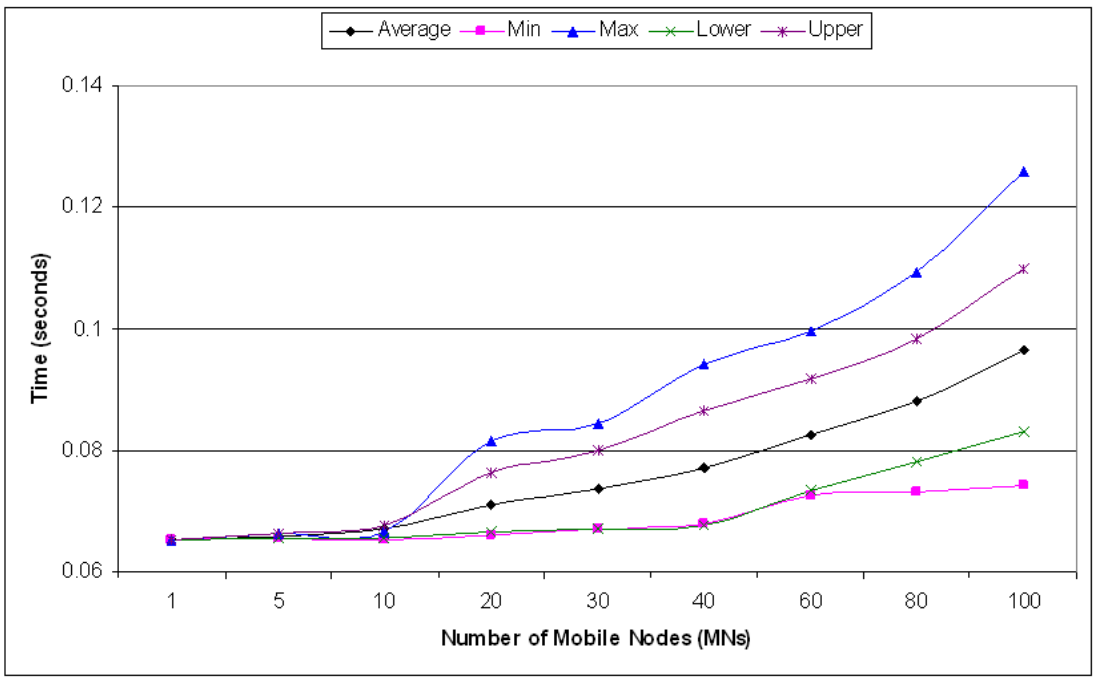

Figure 4.21: HR-Delay (registration) for EHR protocol vs. number of MNs (90% load)

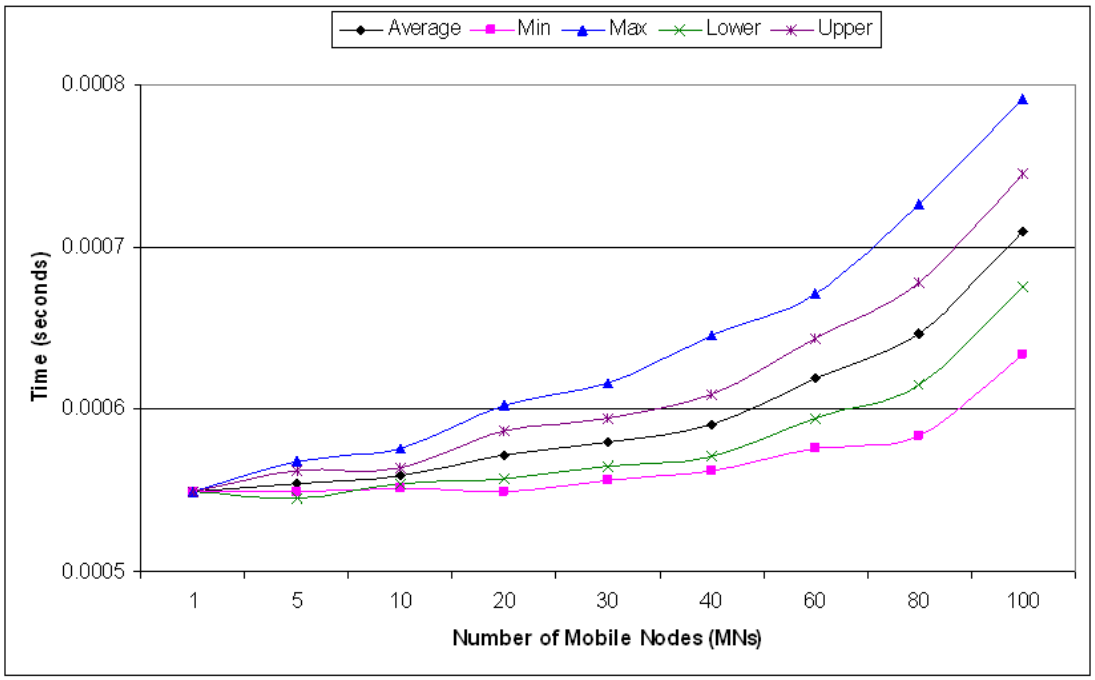

Figure 4.22: HR-Delay (deregistration) for EHR protocol vs. number of MNs (0% load)

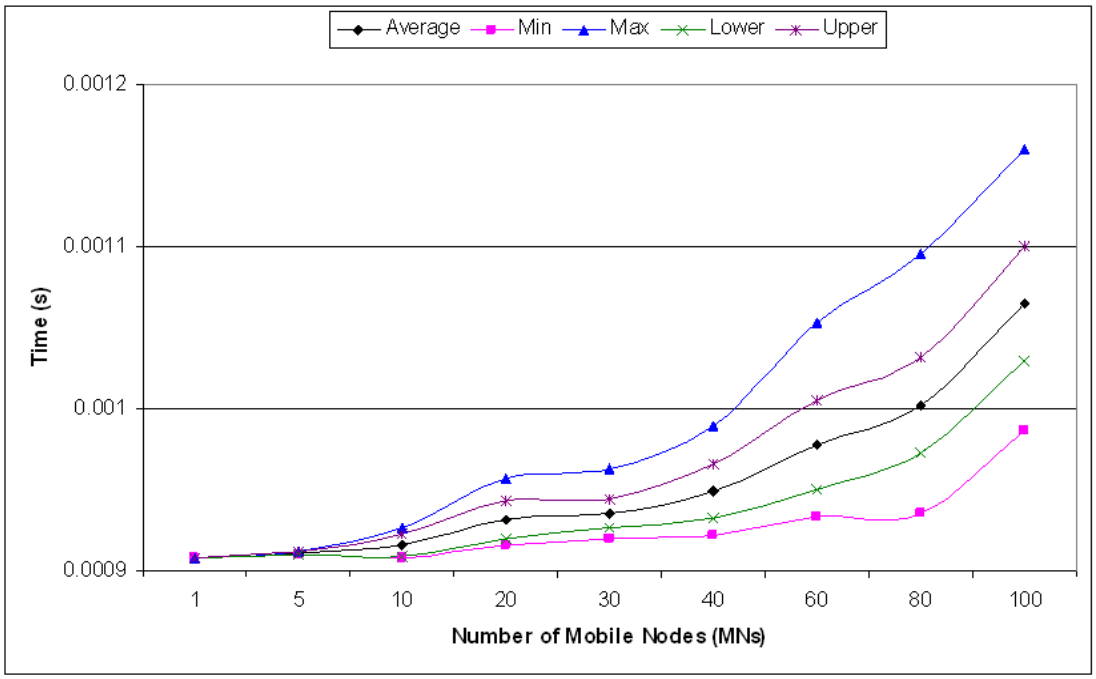

Figure 4.23: HR-Delay (deregistration) for EHR protocol vs. number of MNs (90% load)

Figure 4.24 and Figure 4.25 compare the performance of the BHR and EHR protocols as the number of simultaneously roaming MNs increases. As shown, the EHR protocol has higher rates of increase of HR-Delay than the BHR protocol during registration. This is because HAs in the EHR protocol are required to perform more operations during registration than in the BHR protocol. Consequently, the queuing time on the HA side in the EHR protocol increases faster than it does in the BHR protocol. Therefore, as the number of simultaneously roaming MNs increases, the HR-Delay in the BHR protocol increases less quickly than in the EHR protocol.

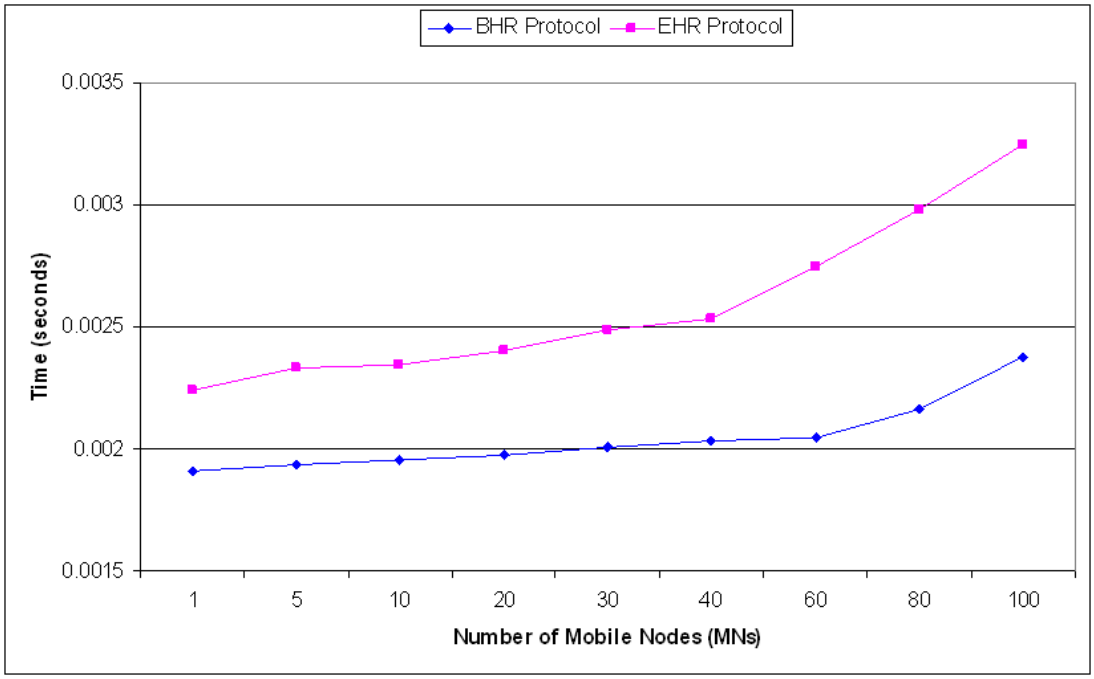

Figure 4.24: HR-Delay (registration) for BHR and EHR protocols vs. number of MNs (0% load)

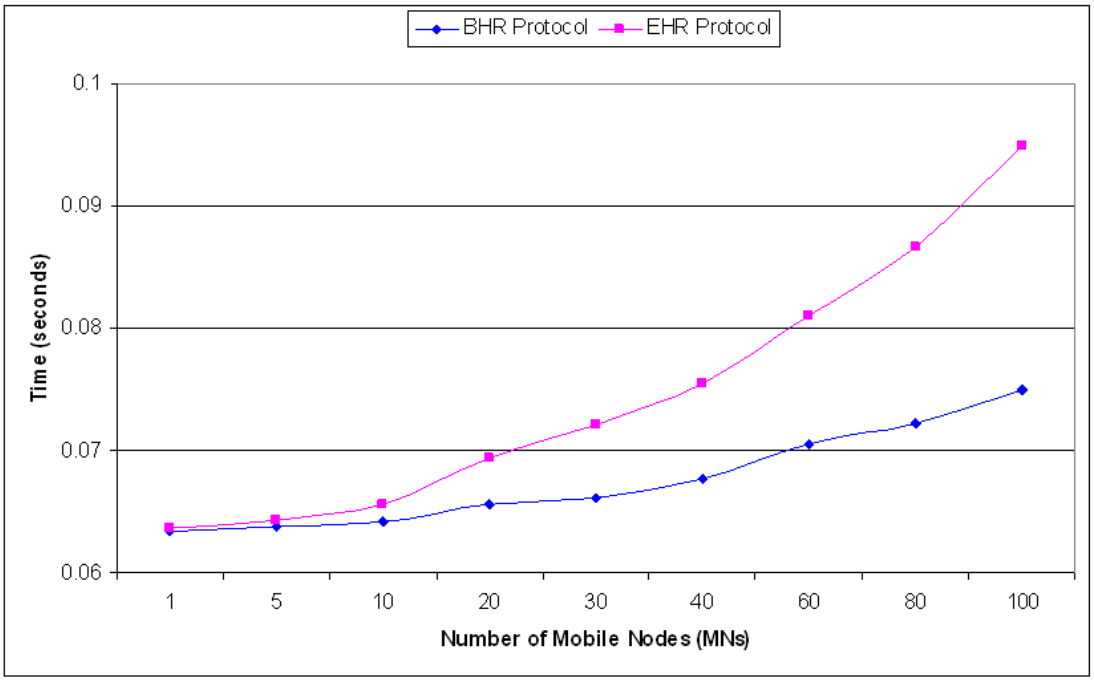

Figure 4.25: HR-Delay (registration) for BHR and EHR protocols vs. number of MNs (90% load)

4.3.3.2 Control Signalling Overhead

This section presents an analysis of control signalling overhead results for the BHR and EHR protocols. Figure 4.26 and Figure 4.27 show control signalling overheads at the MN side and HA side, respectively. Generally, the reason for any differences in control signalling overhead between BHR and EHR at an entity is due to the difference in the number and length of signalling messages received and/or sent at that entity. The following two observations can be made from these figures: (1) the amount of control signalling overhead at both the MN and HA are identical in the two protocols during deregistration processes; and (2) while registering a new CoA, control signalling overhead at both the MN and HA in EHR is about 126% higher than in BHR.

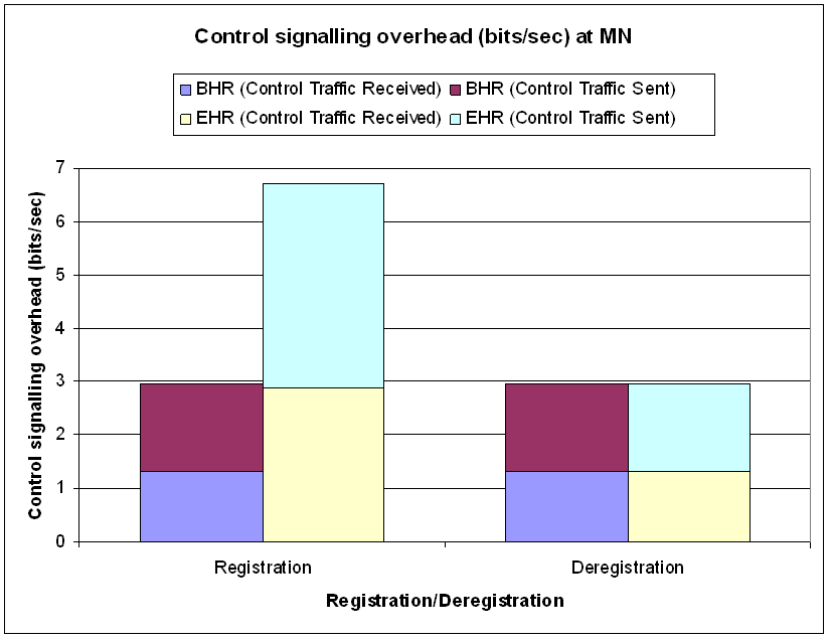

Figure 4.26: Control signalling overhead (bits/sec) for BHR and EHR protocols at MN

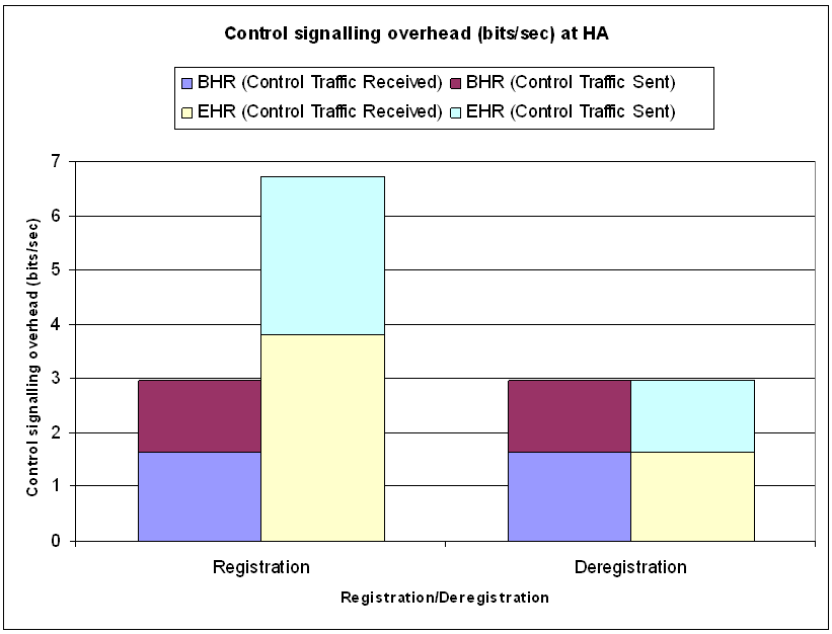

Figure 4.27: Control signalling overhead (bits/sec) for BHR and EHR protocols at HA

4.3.3.3 Discussions

From the simulation results presented, the following observations can be made:

- Firstly, the performance of the two protocols are identical during the deregistration process.
- Secondly, the performance of the EHR protocol and the BHR protocol in terms of HR-Delay is comparable.
- Thirdly, the impact of the increase in number of simultaneous roaming MNs that are served by the same HA on the performance of the EHR protocol is higher than in the BHR protocol.
- Fourthly, the EHR protocol doubles the control signalling overheads at both the MN and the HA as a price to pay for supporting the location authentication of MNs to their HAs.

Overall, the author concludes that if a comparison between both the EHR protocol and the BHR protocol is performed on the basis of efficiency as well as security, the EHR protocol renders itself superior.

4.4 Chapter Summary

This chapter has presented the design of a novel enhanced home registration (EHR) protocol that allows HAs to verify MNs' ownership of claimed CoAs. The EHR protocol makes use of a combination of three ideas. Firstly, it generates CoAs cryptographically using a symmetric CGA-based technique. Secondly, it applies a concurrent CoA reachability test to verify MNs' reachability at claimed CoAs. Thirdly, it uses a novel method to determine hosts' types based on their IPv6 addresses.

A simulation model of EHR has been constructed using the OPNET Modeler. The model has been validated using the OPNET Modeler debugger and using theoretical calculations. The analysis of simulation results has demonstrated that EHR only introduces a marginal increase in the registration delay, but a significant increase in the signalling overhead as a cost of supporting the location authentication of MNs.

In the next chapter, a family of correspondent registration protocols is presented. These protocols require the use of the EHR protocol between MNs and

their HAs in registering MNs' CoAs. They rely on the assistance of HAs to confirm the MNs' ownership of the claimed CoAs and HoAs to the CNs.

Chapter 5

A Family of Correspondent Registration Protocols

This chapter presents the design of a family of correspondent registration protocols. Section 5.1 describes the design objectives for the protocols. Section 5.2 outlines the design principles for the protocols and the common assumptions on which the design is based. Section 5.3 gives an overview of the protocols and their three phases: the creation phase, the update phase, and the deletion phase. Section 5.4 presents detailed descriptions of the protocols. Finally, Section 5.5 summarises the chapter.

5.1 Design Requirements

The security and performance requirements mentioned in Section 2.2.2 have been identified as the design objectives for the protocols. These requirements are:

- **(S1)** To provide authenticity of the claimed home address to assure the CN that the binding request has originated from the entity that owns the home address.
- **(S2)** To provide authenticity of the claimed care-of address to assure the CN that the entity that sent the binding request is actually located at the care-of address.
- **(S3)** To provide integrity of the binding request to detect the unauthorised modification of binding data.

- **(S4)** To provide freshness of the binding request to prevent replay attacks.
- **(S5)** To provide protection for all involved entities against resource exhaustion DoS attacks.
- **(P1)** To minimize delay introduced as a result of securing correspondent registrations.
- **(P2)** To minimize the number of expensive operations performed by the mobile node.
- **(P3)** To minimize the number and length of messages sent/received by the mobile node.

5.2 Design Preliminaries

Before presenting the design of the protocols, we first highlight the assumptions and principles behind their design.

5.2.1 Design Assumptions

The following assumptions have been used in the design of the protocols.

- **(A1)** The protocols are designed for both stationary and mobile CNs. When the CN is stationary, hereafter referred to as the stationary CN case, the protocol entities are an MN, an HA, and a CN. When the CN is also mobile, hereafter referred to as the mobile CN case, the protocol entities are an MN, an MN's HA (denoted as HA_{MN}), a CN, and a CN's HA (denoted as HA_{CN}).
- **(A2)** A mobile node and its home agent have a preconfigured bidirectional security association for encrypted and authenticated communication. They use IPSec ESP protocol to protect mobility-related messages exchanged between them. Therefore, (1) in the stationary CN case: the MN and HA share a secret key (K_{MN-HA}); and (2) in the mobile CN case: the MN and HA_{MN} share a secret key ($\mathrm{K}_{MN-HA_{MN}}$) and the CN and HA_{CN} share a secret key ($\mathrm{K}_{CN-HA_{CN}}$).
- **(A3)** A mobile node and its home agent agree on a validity period, where any mobility-related messages received by the home agent after this validity period

will be rejected. This validity period is to protect the home agent against replay attacks and should be kept reasonably short.

- **(A4)** A mobile node's home agent is a trusted entity. The mobile node and the correspondent node trust that the home agent does not misbehave on its own (both in the stationary and mobile CN cases).

- **(A5)** Cryptographic primitives are secure. That is, well-known primitives such as the AES secret-key algorithm [64], the RSA public-key algorithm [65], and the SHA-512 hash algorithm [66], are secure.

5.2.2 Design Principles

The following measures have been taken in the design of the correspondent registration protocols in order to satisfy the requirements mentioned in Section 5.1.

- **Measure 1:** The enhanced home registration (EHR) protocol, detailed in Chapter 4, is used between the MN and its HA. When the MN roams away from its home link, it uses the EHR protocol to register its new CoA with the HA. The EHR protocol enables the HA to securely authenticate the MN's ownership of the CoA. Therefore, it enables the HA to participate in correspondent registrations by confirming the MN's CoA to the CN.

- **Measure 2:** The idea of early binding updates ([39], see also Section 3.2.1) is used to minimise registration delay in the update phase. That is, when the CN receives an authentic binding update request from the MN, it registers the claimed CoA enclosed in the request and sends subsequent packets destined for the MN to that CoA while concurrently confirming the correctness of that CoA with the MN's home link. The CN limits the amount of data sent to the unconfirmed CoA through granting a short binding lifetime instead of using a Credit-Based Authorization technique ([39], see also Section 3.2.1). In this way, the complexity at the CN is not increased as it does not implement the technique.

- **Measure 3:** The remaining lifetime (LT_{BRem}) for the binding of HoA and CoA at the HA's Binding Cache entry for the MN, and at the MN's Binding Update List entry for the HA, is used as a timestamp to protect the HA against replay attacks. That is, as the HA is involved in the protocols, the HA is enabled to

verify the freshness of binding creation and confirmation requests received from the MN and the CN, respectively, through the use of LT_{BRem}. Specifically, the value of LT_{BRem} is enclosed in binding creation requests sent by the MN to the HA. It is also enclosed in binding confirmation requests sent by the CN to the HA. In the latter case, the MN encrypts the values of its HoA and CoA, the CN's address, and the LT_{BRem} using a secret key shared with the HA (Assumption **A2**). The MN then encloses the coded data to the binding update request sent to the CN, which forwards it to the HA in the binding confirmation request. In this way, the HA can detect replay attacks without the need of clock synchronization; the only requirement is having accurate clocks at the HA and the MN.

- **Measure 4:** The idea of increasing the maximum binding lifetime is used to reduce the number of redundant binding refreshes, thereby reducing signalling overheads. However, increasing the maximum binding lifetime could enable malicious MNs to flood foreign networks. For example, the MN could send a binding creation/update request to the CN requesting the maximum binding lifetime for its CoA. The MN would next start to download a heavy stream of data from the CN. The MN would then roam to a different network without informing the CN. As a result, the CN would continue to send subsequent traffic to the MN's old CoA, flooding the MN's old foreign network with excessive unwanted data. Therefore, it is recommended that the CN periodically run a care-of address reachability test, every few minutes, especially when the CN is sending heavy streams of data to the MN's CoA. That is, the CN generates and sends a fresh token to the MN's CoA for reachability reconfirmation. If the CN does not receive a response within a specific interval, it deactivates its cache entry for the MN and sends subsequent data to the MN's HoA.

5.3 Protocols Overview

The proposed correspondent registration protocols rely on the assistance of the MN's home link to enable the CN to securely authenticate the MN's ownership of the HoA and the CoA; the HA of the MN asserts to the CN that the MN is the legitimate owner of the HoA and it is indeed connected to the CoA. These protocols are in three categories depending on the relationship between MNs' home links and CNs: (1) the SK-based protocol for use when a secret key is

shared, (2) the PK-based protocol for use when the home link has a certified public/private key pair, and (3) the INF-based protocol for use when no prior relationship exists.

The three protocols are designed as shown in Figure 5.1. Each protocol is divided into three phases: the creation phase, the update phase, and the deletion phase. Each of the protocols has its own creation phase. However, they all have the same update and deletion phases. The creation phase is to allow the CN to securely identify the MN and to establish a session key between the two nodes. It also allows the MN to create a new binding at the CN. The update phase is to allow the MN to securely update the CN with its new CoA after roaming to a new foreign link. It also allows the MN to securely extend the lifetime of an existing binding that is about to expire at the CN. The deletion phase is to allow the MN to securely delete its binding at the CN.

The creation phase: The creation phase is executed only once to provide authentication of the MN and its home link to the CN, and the establishment of registration session keys. Specifically, the creation phase consists of three steps. In the first step, the CN verifies the MN's reachability at the HoA and/or the CoA. In the second step, the CN securely identifies and establishes session keys with the MN and the home link. The CN also creates a new binding between the HoA and the CoA. In the third step, the MN confirms the binding creation request to the CN and the CN acknowledges it to the MN. This extra confirmation is to assure the CN that the MN did not roam to a new CoA while the HA and the CN were running the protocol. The creation phase runs parallel to data transfer between the MN and the CN through the MN's home link; thus, a high delay is acceptable during this phase.

The update phase: The update phase is executed every time the MN needs to update its binding at the CN. It consists of three steps. In the first step, the MN sends an early binding update request to the CN. Upon receipt of this request, the CN registers the claimed CoA and sends subsequent packets destined for the MN to that CoA. In the second step, the CN verifies the request with the home link. In the third step, the CN returns an optional acknowledgement to the MN. The early binding update request, the request verification, and the acknowledgement messages are protected using the session keys established in the

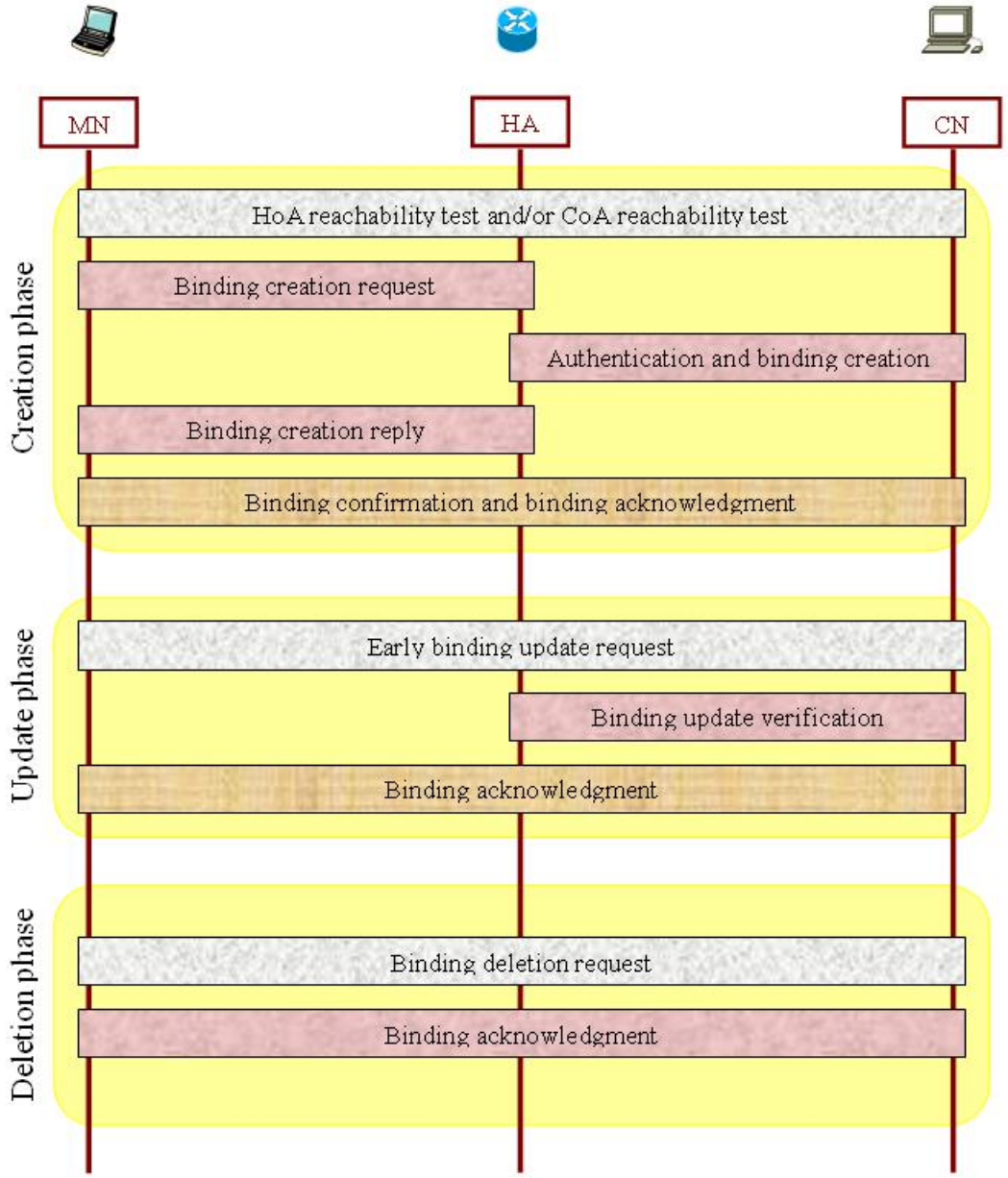

Figure 5.1: Overview of correspondent registration protocols - stationary CN case

creation phase. The communication between the MN and the CN is interrupted during the update phase (if the MN changes its CoA), and thus, the proposed correspondent registration protocols use an early binding update to minimise the delay during this phase.

The deletion phase: The deletion phase is executed every time the MN wants to remove its binding at the CN. It consists of two steps. In the first step, the MN sends a binding deletion request to the CN. In the second step, the CN returns an optional acknowledgement to the MN. The binding deletion request and the acknowledgement messages are protected using the session key established between the MN and the CN in the creation phase.

5.4 Protocols Design

This section describes the design of the three protocols. As each of the three protocols has its own creation phase, the section is divided into five subsections: (1) the creation phase for the SK-based protocol (CRE-SK phase); (2) the creation phase for the PK-based protocol (CRE-PK phase); (3) the creation phase for the INF-based protocol (CRE-INF phase); (4) the update (UPD) phase; and (5) the deletion (DEL) phase.

5.4.1 The Creation Phase for the SK-based Protocol

The following additional assumption has been used in the design of the SK-based protocol.

- **(SK1)** A mobile node's home link shares a long-term secret key with a correspondent node before an invocation of the protocol. Specifically, (1) in the stationary CN case: the HA and CN share a secret key (K_{HA-CN}); and (2) in the mobile CN case: the HA_{MN} and CN share a secret key ($\mathrm{K}_{HA_{MN}-CN}$).

The creation phase for the SK-based protocol in the stationary CN case is designed as shown in Figure 5.2. It consists of eight messages, **M1-SK** to **M8-SK**, and nine steps, **S1-SK** to **S9-SK**. Each of these messages is given a specific name:

- **M1-SK:** Care-of Test Init - denoted as **CoTI**.
- **M2-SK:** Care-of Test - denoted as **CoT**.
- **M3-SK:** Binding Request - denoted as **BReq**.
- **M4-SK:** Binding Reply - denoted as **BRep**.
- **M5-SK:** Early Binding Creation - denoted as **EBC**.
- **M6-SK:** Early Binding Acknowledgement - denoted as **EBA**.
- **M7-SK:** Binding Creation Confirmation - denoted as **BCC**.
- **M8-SK:** Binding Acknowledgement - denoted as **BA**.

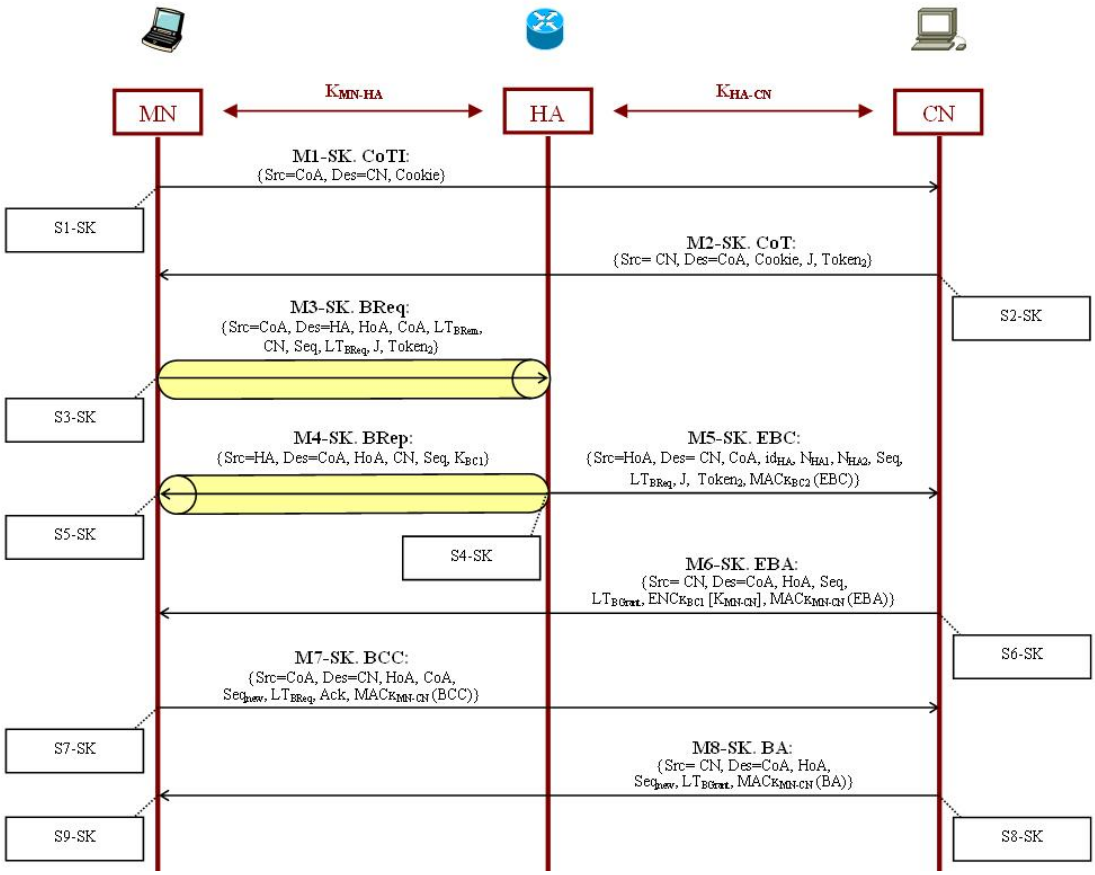

Figure 5.2: Creation phase for the SK-based protocol - stationary CN case

- **Step S1-SK:** The MN sends message **M1-SK** to initiate the creation phase and to check whether the CN supports route optimization.

- **Step S2-SK:** The CN sends message **M2-SK**, which contains a fresh token (Token_2), to the MN. Token_2 will be returned to the CN later to protect the CN against replay attacks and prove the MN's reachability at the claimed CoA.

- **Step S3-SK:** The MN securely sends message **M3-SK** to request binding creation with the CN from the HA. The MN includes the CoA, the CN's address, an initial sequence number (Seq), a binding lifetime request (LT_{BReq}), Token_2, and the remaining lifetime for the binding of HoA and CoA at both the MN and the HA (LT_{BRem}) in the message. LT_{BRem} is used as a timestamp to protect the HA against replay attacks.

- **Step S4-SK:** Upon receipt of message **M3-SK**, the HA checks the values of the LT_{BRem} and CoA to verify the freshness of the message and correctness of the claimed CoA, respectively. The HA also checks the value of LT_{BReq} to confirm that the MN is not requesting a binding lifetime that is greater than the remaining binding lifetime at the HA. Upon positive verifications, the HA generates two keys (K_{BC1} and K_{BC2}) from fresh nonces and the secret key shared with the CN. The HA then securely sends message **M4-SK** to the MN to deliver key K_{BC1}. The HA includes the Seq and CN's address in message **M4-SK**. At the same time, the HA sends message **M5-SK** to the CN to request binding creation on behalf of the MN. The HA includes the fresh nonces, the MN's HoA and CoA, the Seq, the LT_{BReq}, and Token_2 in message **M5-SK**. In addition, the HA protects message **M5-SK** using key K_{BC2}.

- **Step S5-SK:** Upon receipt of message **M4-SK**, the MN checks the value of Seq to verify freshness of the message. Upon positive verification, the MN stores key K_{BC1} in the Binding Update List entry for the CN.

- **Step S6-SK:** Upon receipt of message **M5-SK**, the CN checks the value of Token_2 to verify freshness of the message and reachability of the MN at the claimed CoA. Upon positive verification, the CN generates K_{BC1} and K_{BC2} from the secret key shared with the MN's home link and the fresh nonces enclosed in the message. The CN then verifies the integrity and authenticity of message **M5-SK** using key K_{BC2}. Upon positive verification, the CN generates a fresh key (K_{MN-CN}), creates a Binding Cache entry, and stores the binding

between the MN's HoA and CoA as well as the values of Seq, LT_{BReq}, and K_{MN-CN} at the entry. The CN also sets granted binding lifetime (LT_{BGrant}) in the entry to a MIN_BINDING_LIFETIME value to handle the case of the MN roaming to a new CoA while the HA and the CN are running the protocol. Finally, the CN sends message **M6-SK** to the MN to deliver key K_{MN-CN} and to acknowledge the binding of the CoA. The CN includes Seq, LT_{BGrant}, and the encryption of key K_{MN-CN} using key K_{BC1} in message **M6-SK**. In addition, the CN protects the message using key K_{MN-CN}.

- **Step S7-SK:** Upon receipt of message **M6-SK**, the MN checks the value of the Seq to verify the freshness of the message. Upon positive verification, the MN decrypts key K_{MN-CN} using key K_{BC1}. The MN then verifies the integrity and authenticity of message **M6-SK** using key K_{MN-CN}. Upon positive verification, the MN updates the value of K_{BC1} with the value of K_{MN-CN} and sends message **M7-SK** to the CN requesting a greater binding lifetime, confirming the receipt of K_{MN-CN} and showing the existence at the CoA. The MN includes a new sequence number (Seq_{new}) that is greater than the value of the Seq sent by the MN in message **M3-SK**. In addition, the MN protects message **M7-SK** using key K_{MN-CN}.

- **Step S8-SK:** Upon receipt of message **M7-SK**, the CN checks the value of the Seq_{new} to verify the freshness of the message. Upon positive verification, the CN verifies the integrity and authenticity of message **M7-SK** using key K_{MN-CN}. Upon positive verification, the CN updates the Binding Cache entry for the MN by storing the value of Seq_{new} enclosed in the message and by setting LT_{BGrant} to a value that is less than or equal to LT_{BReq}. Finally, the CN sends message **M8-SK** to the MN to acknowledge the binding of the CoA. The CN includes the Seq_{new} and LT_{BGrant} in message **M8-SK**. In addition, the CN protects message **M8-SK** using key K_{MN-CN}.

- **Step S9-SK:** Upon receipt of message **M8-SK**, the MN checks the value of the Seq_{new} to verify the freshness of the message. Upon positive verification, the MN verifies the integrity and authenticity of message **M8-SK** using key K_{MN-CN}. Upon positive verification, the MN updates the value of LT_{BGrant} in the Binding Update List entry for the CN.

A more detailed description of these execution steps is given in Appendix D.

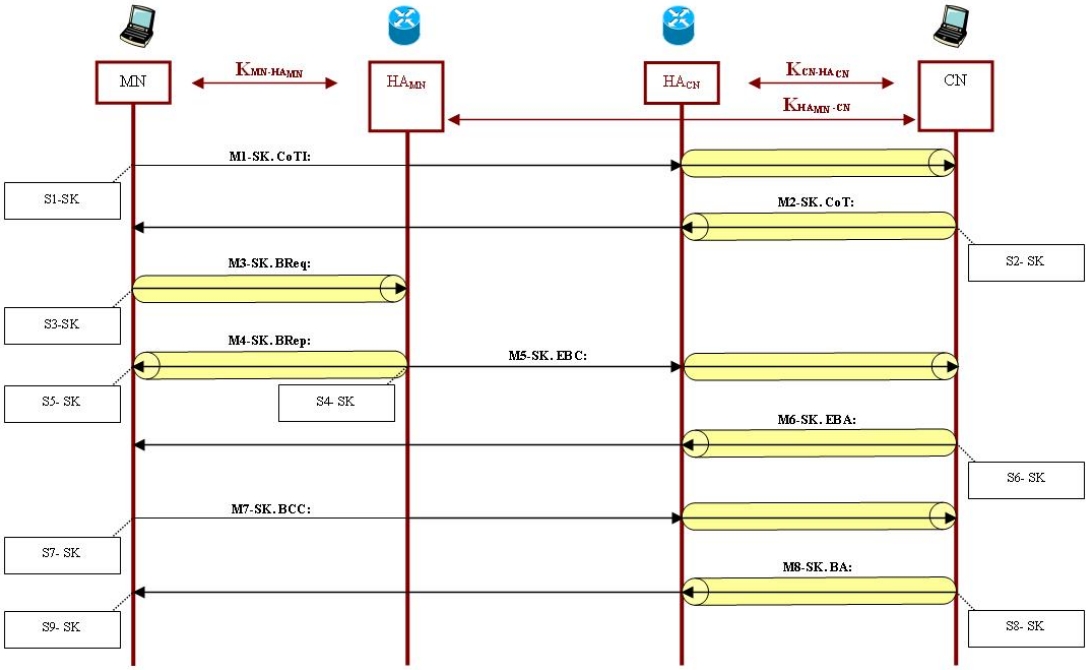

Figure 5.3: Creation phase for the SK-based protocol - mobile CN case

The creation phase for the SK-based protocol in the mobile CN case is shown in Figure 5.3. In this case, the CN is also mobile and could be away from its home link. As a result, the MN and the HA_{MN} send different binding-related messages to the CN's home address. If the CN is located in its home link, it will receive the messages directly. Otherwise, the HA_{CN} will intercept and forward the messages to the CN's current care-of address via an IPSec ESP secure tunnel. In addition, the CN sends different binding-related messages while it is away from home via HA_{CN} ('reverse tunnelling'). Apart from these points, the creation phases for SK-based protocol in the stationary and mobile CN cases are the same.

5.4.2 The Creation Phase for the PK-based Protocol

The following additional assumption has been used in the design of the PK-based protocol.

- **(PK1)** A mobile node's home link has a certified public/private key pair (PK_H, SK_H); a trusted CA has issued the home link's public key and a PKI is in place. The home link possesses the key pair before an invocation of the protocol.

The creation phase for the PK-based protocol in the stationary CN case is designed as shown in Figure 5.4. It consists of eight messages, **M1-PK** to **M8-PK**, and nine steps, **S1-PK** to **S9-PK**. Each of these messages is given a specific name:

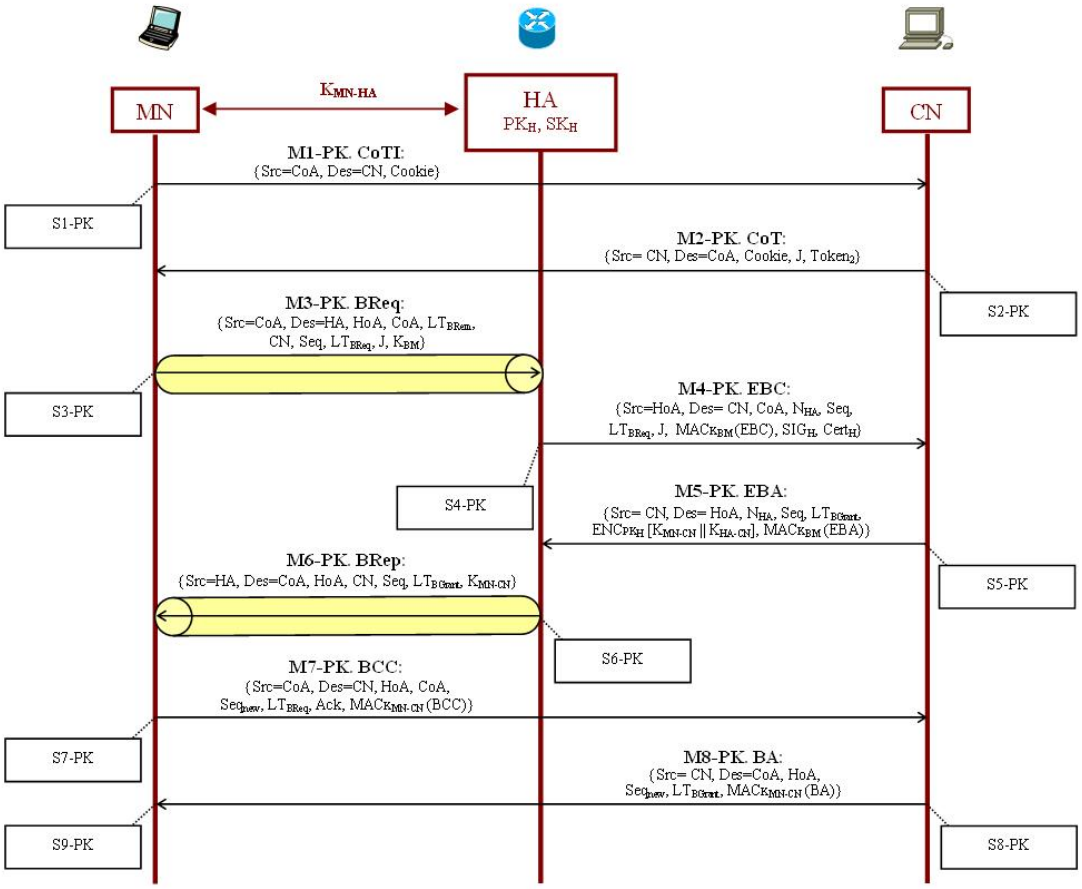

Figure 5.4: Creation phase for the PK-based protocol - stationary CN case

- **M1-PK:** Care-of Test Init - denoted as **CoTI**.
- **M2-PK:** Care-of Test - denoted as **CoT**.
- **M3-PK:** Binding Request - denoted as **BReq**.
- **M4-PK:** Early Binding Creation - denoted as **EBC**.
- **M5-PK:** Early Binding Acknowledgement - denoted as **EBA**.
- **M6-PK:** Binding Reply - denoted as **BRep**.

- **M7-PK:** Binding Creation Confirmation - denoted as **BCC**.

- **M8-PK:** Binding Acknowledgement - denoted as **BA**.

- **Steps S1-PK** and **S2-PK: Steps S1-PK** and **S2-PK** are identical, respectively, to **Steps S1-SK** and **S2-SK** mentioned in Section 5.4.1.

- **Step S3-PK: Step S3-PK** is identical to **Step S3-SK**, but the MN includes the hash value of Token_2 ($\text{K}_{BM} = \text{SHA1}\ (\text{Token}_2)$) instead of Token_2 in message **M3-PK** sent to the HA.

- **Step S4-PK:** The first part of **Step S4-PK** is identical to the first part of **Step S4-SK**. Specifically, upon receipt of message **M3-PK**, the HA checks the values of LT_{BRem}, CoA, and LT_{BReq} to verify the freshness of the message, correctness of the claimed CoA, and to confirm that the binding lifetime requested by the MN is not greater than the remaining binding lifetime at the HA, respectively. However, upon positive verifications, the HA sends message **M4-PK** to the CN to request binding creation on behalf of the MN. The HA includes a fresh nonce (N_{HA}), the MN's HoA and CoA, the Seq, the LT_{BReq}, the HA's signature on the message using the home link's private key, and a keyed hash value using key K_{BM} in message **M4-PK**. This keyed hash value is to protect the CN against replay attacks and resource exhaustion DoS attacks. The HA also temporary stores the values of N_{HA} and K_{BM} to verify the freshness, integrity, and authenticity of the response from that CN.

- **Step S5-PK:** Upon receipt of message **M4-PK**, the CN generates key K_{BM} from token Token_2, and then verifies the freshness of the message and reachability of the MN at the claimed CoA using key K_{BM}. Upon positive verification, the CN verifies the signature to confirm the integrity and authenticity of the message. Upon positive verification, the CN generates two fresh session keys (K_{MN-CN} and K_{HA-CN}), creates a Binding Cache entry, and stores the binding between the MN's HoA and CoA, as well as the values of Seq, LT_{BReq}, K_{MN-CN}, and K_{HA-CN} at the entry. The CN also sets LT_{BGrant} in the entry to a MIN_BINDING_LIFETIME value to handle the case of the MN roaming to a new CoA while the HA and the CN are running the protocol. Finally, the CN sends message **M5-PK** to the MN's HoA to deliver the two session keys and to acknowledge the binding of the CoA. The CN includes N_{HA}, Seq, LT_{BGrant},

and the encryption of the two session keys using the home link's public key in message **M5-PK**. In addition, the CN protects the message using key K_{BM}.

- **Step S6-PK:** The HA intercepts message **M5-PK** and checks the value of N_{HA} to verify the freshness of the message. Upon positive verification, the HA verifies the integrity and authenticity of the message using key K_{BM}. Upon positive verification, the HA decrypts the session keys, K_{MN-CN} and K_{HA-CN}, using the home link's private key. The HA then securely sends message **M6-PK** to the MN to deliver key K_{MN-CN} and to acknowledge the binding of the CoA. The HA also stores the value of K_{HA-CN} and discards the values of N_{HA} and K_{BM}.

- **Step S7-PK: Step S7-PK** is identical to **Step S5-SK**, but the key stored by the MN at the Binding Update List entry for the CN is K_{MN-CN} instead of K_{BC1}.

- **Steps S8-PK** and **S9-PK: Steps S8-PK** and **S9-PK** are identical, respectively, to **Steps S8-SK** and **S9-SK** mentioned in Section 5.4.1.

A more detailed description of these execution steps is given in Appendix D.

The creation phase for the PK-based protocol in the mobile CN case is shown in Figure 5.5. As in the SK-based protocol, the MN and the HA_{MN} send binding-related messages to the CN's home address. In addition, the CN sends binding-related messages while it is away from home via HA_{CN} ('reverse tunnelling').

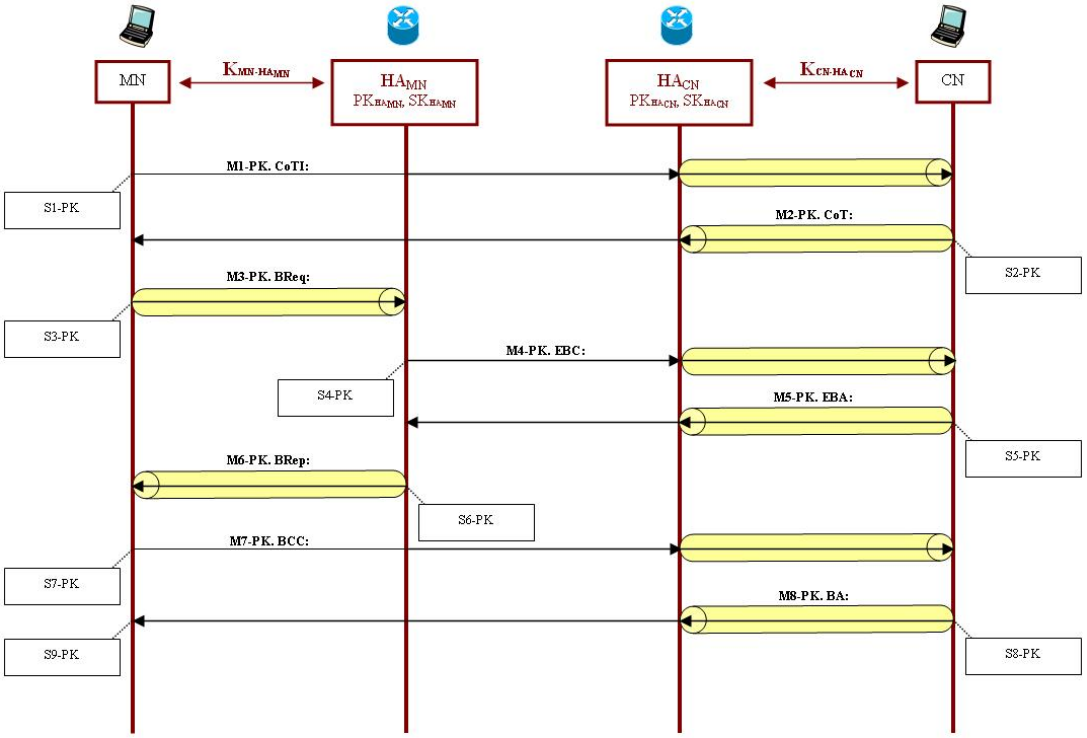

Figure 5.5: Creation phase for the PK-based protocol - mobile CN case

5.4.3 The Creation Phase for the INF-based Protocol

The following additional assumptions have been used in the design of the INF-based protocol.

- **(INF1)** A mobile node's home link has a self-generated public/private key pair (PK_H, SK_H) where the private key (SK_H) is kept securely by HAs in the home link.
- **(INF2)** A mobile node's HoA is configured as a CGA-based address using the home link's public key (PK_H). In addition, the CGA parameters (see Section 3.1) are distributed between the MN and the home link. Specifically, as the subnet prefix and the public key used are the same for all MNs belonging to the same home link, they are kept at the home link. The modifier and the collision count differ from one MN to another; thus, they are kept at the MN.

The creation phase for the INF-based protocol in the stationary CN case is designed as shown in Figure 5.6. It consists of nine messages, **M1-INF** to **M9-INF**,

and nine steps, **S1-INF** to **S9-INF**. Each of these messages is given a specific name:

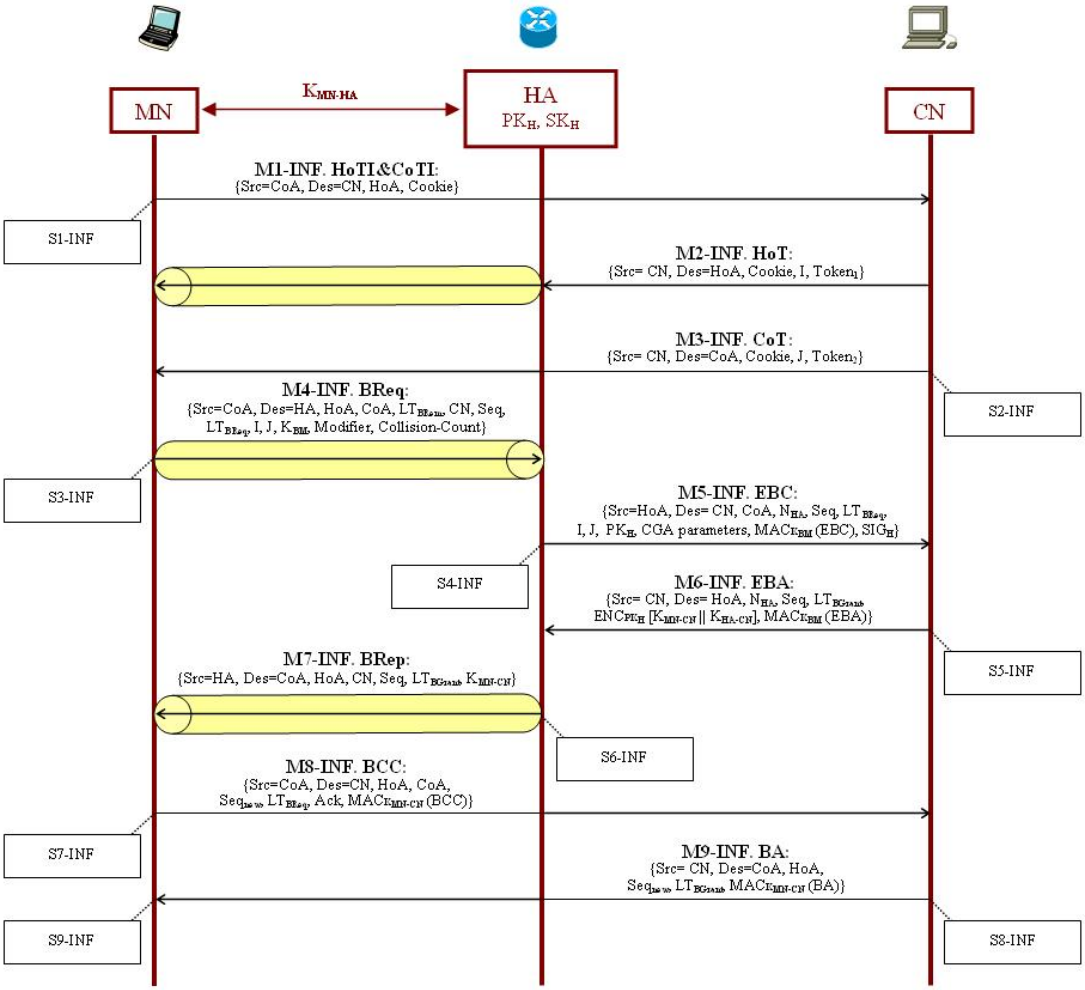

Figure 5.6: Creation phase for the INF-based protocol - stationary CN case

- **M1-INF:** Home Test Init and Care-of Test Init - denoted as **HoTI&CoTI**.
- **M2-INF:** Home Test - denoted as **HoT**.
- **M3-INF:** Care-of Test - denoted as **CoT**.
- **M4-INF:** Binding Request - denoted as **BReq**.
- **M5-INF:** Early Binding Creation - denoted as **EBC**.
- **M6-INF:** Early Binding Acknowledgement - denoted as **EBA**.

- **M7-INF:** Binding Reply - denoted as **BRep**.
- **M8-INF:** Binding Creation Confirmation - denoted as **BCC**.
- **M9-INF:** Binding Acknowledgement - denoted as **BA**.
- **Step S1-INF: Step S1-INF** is identical to **Step S1-SK** and **Step S1-PK**, but the MN's HoA is also enclosed in message **M1-INF** sent from the MN to the CN.
- **Step S2-INF: Step S2-INF** is identical to **Step S2-SK** and **Step S2-PK**, but the CN generates two tokens (Token_1 and Token_2) and sends two messages: message **M2-INF** is sent to the MN's HoA and message **M3-INF** is sent to the MN's CoA.
- **Step S3-INF: Step S3-INF** is identical to **Step S3-PK**, but the MN hashes the two tokens to generate key K_{BM}, i.e. K_{BM} = SHA1 (Token_1 || Token_2). In addition, the MN includes the values of Modifier and Collision-Count in message **M4-INF** sent to the HA.
- **Step S4-INF: Step S4-INF** is identical to **Step S4-PK**, but the HA concatenates the values of Modifier, Subnet Prefix, and Collision-Count to form the CGA parameters used in generating the MN's HoA. In addition, the HA includes the CGA parameters and the home link's self-generated public key in message **M5-INF** sent to the CN.
- **Step S5-INF: Step S5-INF** is identical to **Step S5-PK**, but the CN generates K_{BM} from tokens Token_1 and Token_2. In addition, the CN runs the CGA-based address verification algorithm (see Figures 3.2 in Section 3.1) to confirm the authenticity of HoA.
- **Steps S6-INF**, **S7-INF**, **S8-INF**, and **S9-INF: Steps S6-INF**, **S7-INF**, **S8-INF**, and **S9-INF** are identical, respectively, to **Steps S6-PK**, **S7-PK**, **S8-PK**, and **S9-PK** mentioned in Section 5.4.2.

A more detailed description of these execution steps is given in Appendix D.

The creation phase for the INF-based protocol in the mobile CN case is shown in Figure 5.7. As in the SK-based and the PK-based protocols, the MN and the

HA_{MN} send binding-related messages to the CN's home address. In addition, the CN sends binding-related messages while it is away from home via HA_{CN} ('reverse tunnelling').

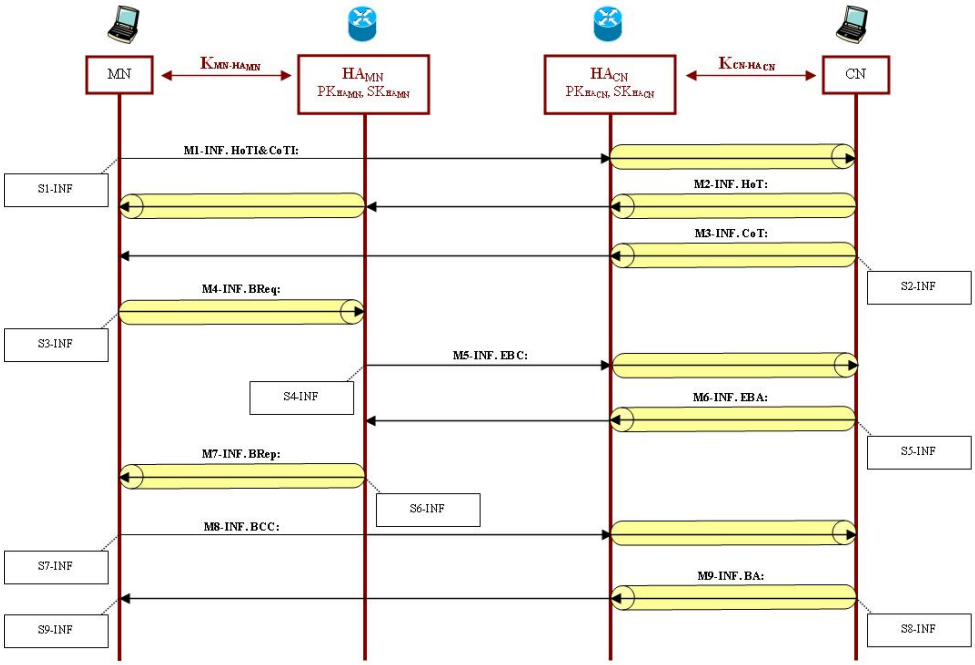

Figure 5.7: Creation phase for the INF-based protocol - mobile CN case

5.4.4 The Update Phase

The MN initiates the update phase either to inform the CN with its new CoA after roaming to a new foreign link, or to extend the lifetime of an existing binding that is about to expire at the CN. The update phase for the proposed protocols in the stationary CN case is designed as shown in Figure 5.8. It consists of four messages, **M1-UPD** to **M4-UPD**, and five steps, **S1-UPD** to **S5-UPD**. Each of these messages is given a specific name:

- **M1-UPD:** Binding Update - denoted as **BU**.
- **M2-UPD:** Binding Confirmation Request - denoted as **BCReq**.
- **M3-UPD:** Binding Confirmation Reply - denoted as **BCRep**.
- **M4-UPD:** Binding Acknowledgement - denoted as **BA**.

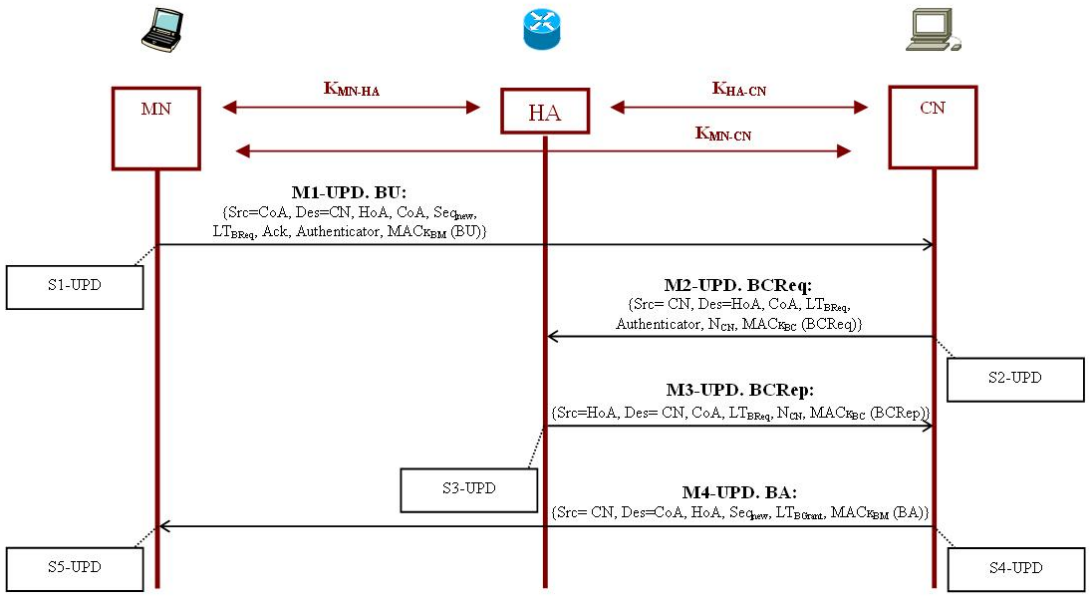

Figure 5.8: Update phase for the proposed protocols - stationary CN case

- **Step S1-UPD:** The MN generates a new sequence number (Seq_{new}) and a fresh key (K_{BM}). The key K_{BM} is generated from the shared session key established between the MN and the CN in the creation phase (K_{MN-CN}). The MN also calculates a coded value (Authenticator) by encrypting the values of its HoA and CoA, the CN's address, and the LT_{BRem} using the secret key shared with the HA (K_{MN-HA}). This Authenticator is to protect the MN's HA against replay attacks. Finally, the MN sends message **M1-UPD** to the CN to request binding update. The MN includes its HoA and CoA, the Seq_{new}, a binding lifetime request (LT_{BReq}), and the Authenticator in message **M1-UPD**. In addition, the MN protects the message using key K_{BM}.

- **Step S2-UPD:** Upon receipt of message **M1-UPD**, the CN checks the value of Seq_{new} to verify the freshness of the message. Upon positive verification, the CN generates key K_{BM}. The CN then verifies the integrity and authenticity of message **M1-UPD** using K_{BM}. Upon positive verification, the CN registers and uses the claimed CoA and concurrently requests home network's confirmation of that CoA. That is, the CN updates the Binding Cache entry for the MN with the values of CoA and Seq_{new}. The CN also sets the entry in an unconfirmed state and sets the granted binding lifetime (LT_{BGrant}) to a MIN_BINDING_LIFETIME value. The CN then generates a fresh key K_{BC}

from a fresh nonce (N_{CN}) and the secret key (K_{HA-CN}) shared with the MN's home network. Finally, the CN sends message **M2-UPD** to the MN's HoA requesting home network's confirmation of the claimed CoA. The CN includes the claimed CoA, N_{CN}, LT_{BReq}, and Authenticator in message **M2-UPD**. In addition, the CN protects the message using key K_{BC}.

- **Step S3-UPD:** The HA intercepts message **M2-UPD** and generates key K_{BC} based on key K_{HA-CN} shared with the CN and on items enclosed in the message. The HA then verifies the integrity and authenticity of the message using key K_{BC}. Upon positive verification, the HA decrypts Authenticator enclosed in the message using key K_{MN-HA} shared with the MN. The HA next checks the values of LT_{BRem} and CoA to verify the freshness of the message and correctness of the claimed CoA, respectively. The HA also checks the value of LT_{BReq} to confirm that the MN is not requesting a binding lifetime that is greater than the remaining binding lifetime at the HA. Upon positive verifications, the HA sends message **M3-UPD** to the CN to confirm the claimed CoA. The HA includes CoA, N_{CN}, and LT_{BReq} in message **M3-UPD**. In addition, the HA protects the message using key K_{BC}.

- **Step S4-UPD:** Upon receipt of message **M3-UPD**, the CN checks the value of N_{CN} to verify the freshness of the message. Upon positive verification, the CN verifies the integrity and authenticity of message **M3-UPD** using key K_{BC}. Upon positive verification, the CN updates the Binding Cache entry for the MN by changing its status to be confirmed and by setting LT_{BGrant} to a value that is less than or equal to LT_{BReq}. Finally, the CN sends message **M4-UPD** to the MN for acknowledging the binding of the CoA. The CN includes Seq_{new} and LT_{BGrant} in message **M4-UPD**. In addition, the CN protects message **M4-UPD** using key K_{BM}.

- **Step S5-UPD:** Upon receipt of message **M4-UPD**, the MN checks the value of Seq_{new} to verify the freshness of the message. Upon positive verification, the MN verifies the integrity and authenticity of message **M4-UPD** using key K_{BM}. Upon positive verification, the MN updates the value of LT_{BGrant} in the Binding Update List entry for the CN.

A more detailed description of these execution steps is given in Appendix D.

The update phase for the proposed protocols in the mobile CN case is shown in Figure 5.9. Again, the MN and the HA_{MN} send binding-related messages to the CN's home address. In addition, the CN sends binding-related messages while it is away from home via HA_{CN} ('reverse tunnelling').

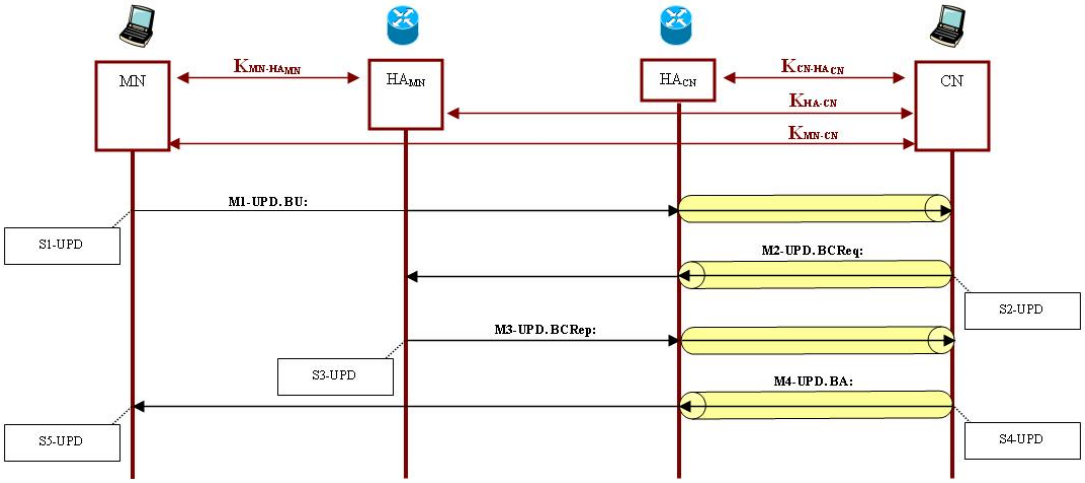

Figure 5.9: Update phase for the proposed protocols - mobile CN case

5.4.5 The Deletion Phase

The MN initiates the deletion phase to remove its binding at the CN. The deletion phase for the proposed protocols in the stationary CN case is designed as shown in Figure 5.10. It consists of two messages, **M1-DEL** and **M2-DEL**, and three steps, **S1-DEL** to **S3-DEL**. Each of the two messages is given a specific name:

- **M1-DEL:** Binding Update - denoted as **BU**.
- **M2-DEL:** Binding Acknowledgement - denoted as **BA**.
- **Step S1-DEL: Step S1-DEL** is identical to **Step S1-UPD**, but the Authenticator is not enclosed in message **M1-DEL** sent from the MN to the CN. In addition, the MN sets the CoA equal to its HoA and the binding lifetime request (LT_{BReq}) to 'zero'.
- **Step S2-DEL:** The first part of **Step S2-DEL** is identical to the first part of **Step S2-UPD**. Specifically, the CN uses the MN's HoA as an index to search its Binding Cache. Then, the CN performs **Verifications CN1-DEL**

and **CN2-DEL**, which are identical to **Verifications CN1-UPD** and **CN2-UPD**, respectively. In addition, the CN deletes the Binding Cache entry for the MN and, if requested, sends Seq_{new} and $MAC_{K_{BM}}$ (BA) to the MN in message **M2-DEL** for acknowledging the deletion of the binding, where $MAC_{K_{BM}}$ (BA) is a keyed hash value used to ensure the integrity and authenticity of message **M2-DEL**; $MAC_{K_{BM}}$ (BA) = First (96, HMAC_SHA1 (K_{BM}, (HoA || CN || Seq_{new}))).

- **Step S3-DEL: Step S3-DEL** is identical to **Step S5-UPD**, but after **Verification MN1-DEL**, which is identical to **Verification MN1-UPD**, the MN will delete the Binding Update List entry for the CN.

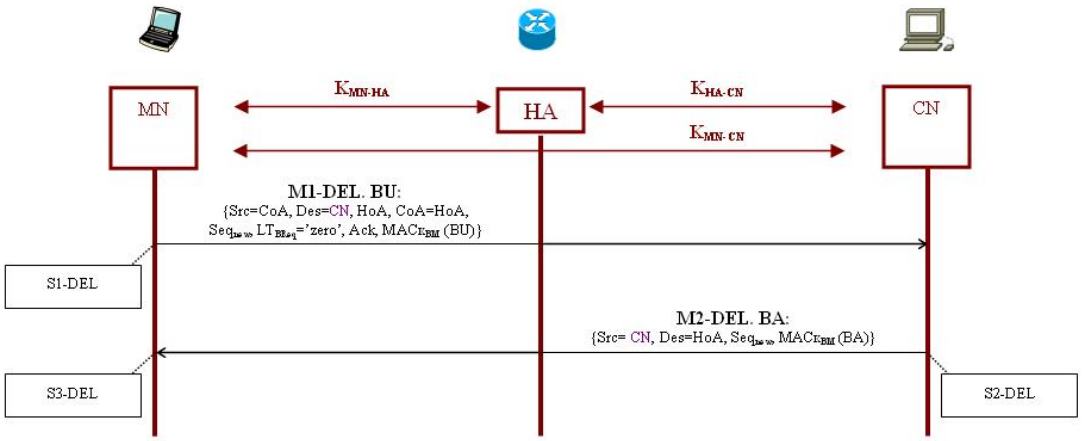

Figure 5.10: Deletion phase for the proposed protocols - stationary CN case

A more detailed description of these execution steps is given in Appendix D.

The deletion phase for the proposed protocols in the mobile CN case is shown in Figure 5.11. The MN sends message **M1-DEL** to the CN's home address. In addition, the CN sends message **M2-DEL** while it is away from its home link via HA_{CN} ('reverse tunnelling').

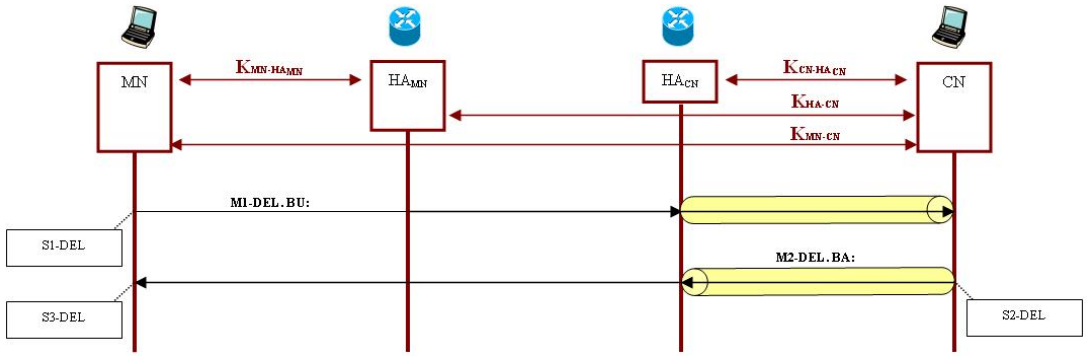

Figure 5.11: Deletion phase for the proposed protocols - mobile CN case

5.5 Chapter Summary

This chapter has presented the design of a family of correspondent registration protocols: (1) the SK-based protocol for use when a secret key is shared between the home link and the CN; (2) the PK-based protocol for use when the home link has a certified public/private key pair; and (3) the INF-based protocol for use when no prior relationship exists. In addition, the proposed protocols are designed in three phases: the creation phase, the update phase, and the deletion phase. Although the protocols have different creation phases, they all have identical update and deletion phases.

The features of the proposed correspondent registration protocols can be summarized as follows:

- They handle the case of the MN roaming to a different CoA while the HA creates a new binding with the CN on behalf of the MN. They do this by limiting the binding lifetime of the claimed CoA at the CN and by requesting the MN send a confirmation from that CoA.

- They use an early binding update to minimise the registration delay in the update phase. The CN registers the new CoA and sends subsequent packets destined for the MN to it for a limited time, while concurrently verifying the CoA with the home link.

- They use the remaining lifetime (LT_{BRem}) for the binding of the HoA and CoA at the HA's Binding Cache entry for the MN as a timestamp to protect the HA against replay attacks.
- They allow the MN's home link to verify the claimed care-of address (CoA) and the requested binding lifetime (LT_{BReq}) to prevent the MN from cheating the CN with a fake CoA and/or a long binding lifetime request.
- They increase the maximum binding lifetime to reduce the number of redundant binding refreshes and thus reduce signalling overheads.

In the following chapter, we analyse the protocols against various security attacks, prove security features, evaluate performance, and compare them with related work.

Chapter 6

Security and Performance Analyses of the Protocols

This chapter is devoted to the informal security analysis, formal security verification, and performance evaluation of the protocols presented in Chapter 5. Specifically, Section 6.1 presents informal analyses of the protocols against the security requirements specified in Section 5.1 and against the well-known security attacks specified in Section 2.2.1. To provide further proof of the security of the protocols, Section 6.2 presents formal verifications of the protocols using the framework of Protocol Composition Logic (PCL) and Casper/FDR2 model checker. Section 6.3 presents the simulation and performance evaluation of the protocols and their comparisons with the most related work. Namely, Section 6.3.1 discusses the design and construction of the simulation model, Section 6.3.2 validates the simulation model, and Section 6.3.3 presents the simulation results and draws conclusions. Finally, Section 6.4 summarises the chapter.

6.1 Informal Analysis of Protocols

In this section, we informally analyse the protocols presented in the previous chapter, and detailed in Appendix D, to demonstrate that they satisfy the security requirements specified in Section 5.1 and are resistant to the well-known security attacks specified in Section 2.2.1.

(S1) Authenticity of home address: The CN confirms whether the MN owns the claimed HoA. If not confirmed, an attacker can launch a false binding update

attack and/or a return-to-home spoofing attack. The authenticity of the HoA is assured to the CN through the following measures.

1. The CRE-SK phase achieves the authenticity of the HoA by using a keyed hash function that is generated based on the secret key shared between the home link and the CN. Specifically, **Verification CN2-SK** ensures that message **M5-SK** is indeed from the home link of the MN and has not been altered in transit. Therefore, the CN is convinced that the MN actually owns the HoA.

2. The CRE-PK phase achieves the authenticity of the HoA by using the home link's public key signature over message **M4-PK** using the home link's private key. Specifically, **Verification CN2-PK** ensures that message **M4-PK** is indeed from the home link of the MN and has not been altered in transit. Therefore, the CN is convinced that the MN actually owns the HoA.

3. The CRE-INF phase employs a CGA-based HoA together with an initial HoA reachability test to authenticate the HoA. The MN configures its HoA as a CGA-based address using the home link's self-generated public key. The MN also provides an initial HoA reachability proof. Specifically, (1) **Verification CN2-INF** ensures that message **M5-INF** is from a node that is reachable at the HoA enclosed in the message; (2) **Verification CN3-INF** ensures that the public key and the HoA enclosed in message **M5-INF** are bound; and (3) **Verification CN4-INF** ensures that message **M5-INF** is from a node that knows the private key corresponding to the public key used in generating the HoA. These verifications together prevent false binding update attacks and alleviate return-to-home spoofing attacks. As a result, the CRE-INF phase is still vulnerable to return-to-home spoofing attacks as it does not assure the CN that the public key enclosed in message **M5-INF** is authentic. An attacker, on the path between the CN and a victim network, could use its own public key and a spoofed subnet prefix to cryptographically generate a non-used (home) address with a subnet prefix from the victim network.

4. In the UPD phase, **Verification CN2-UPD** ensures that the keyed hash function enclosed in message **M1-UPD** is generated using a key (K_{BM})

that is generated from the secret key K_{MN-CN} shared between the CN and the MN. In addition, **Verification CN3-UPD** ensures that the keyed hash function enclosed in message **M3-UPD** is generated using a key (K_{BC}) that is generated from the secret key K_{HA-CN} shared between the CN and the MN's home link. These two verifications enable the CN to confirm the authenticity of the HoA.

5. In the DEL phase, the validity of the keyed hash function enclosed in message **M1-DEL** is verified in **Verification CN2-DEL** with a key (K_{BM}) that is generated from the secret key K_{MN-CN} shared between the CN and the MN. Therefore, the authenticity of the HoA enclosed in message **M1-DEL** is ensured.

(S2) Authenticity of care-of address: The CN confirms whether the MN is indeed connected to the claimed care-of address. If not confirmed, a malicious MN and an attacker can launch a malicious MN flooding attack and a false binding update attack, respectively. The CN is assured of the authenticity of the care-of address by obtaining a confirmation from the home link. This is achieved through the following measures.

1. In the CRE-SK and CRE-PK phases, the MN provides an initial CoA reachability proof. In addition, the home link confirms to the CN that the MN is connected to the CoA. Specifically, **verifications CN1-SK** and **CN1-PK** ensure that messages **M5-SK** and **M4-PK**, respectively, are from a node that is reachable at the CoA. In addition, **verifications CN2-SK** and **CN2-PK** ensure that messages **M5-SK** and **M4-PK**, respectively, are indeed from the home link of the MN and have not been altered in transit. As a result, the CN is convinced that the MN is connected to the CoA.

2. The authenticity of the CoA in the CRE-INF phase is achieved in the same way as in the CRE-SK and CRE-PK phases, i.e. by an initial CoA reachability proof and a confirmation from the home link. Therefore, the MN cannot launch a malicious MN flooding attack. In addition, an attacker cannot impersonate a legitimate MN to launch a false binding update attack. However, the attacker that had used its own public key to generate a non-used (home) address could launch a false binding update attack. This

is because the CN has not been ensured that the confirmation is indeed coming from the home link due to the use of the unauthentic public key.

3. In the UPD phase of the SK-based and PK-based protocols, the home link confirms to the CN that the MN is connected to the CoA. Specifically, **Verification CN3-UPD** ensures that message **M3-UPD** is coming from the home link and has not been altered in transit. Therefore, the CN is convinced that the CoA claimed by the MN in message **M1-UPD** matches with the CoA registered at the home link for that MN and thus the CN is assured that the MN is connected to the CoA.

4. The authenticity of the CoA in the UPD phase of the INF-based protocol is achieved in the same way as in the SK-based and PK-based protocols, i.e. by a confirmation from the home link. However, due to the use of the unauthentic public key in the CRE-INF phase, an attacker could launch a false binding update attack as the CN has not been ensured that the confirmation is indeed coming from the home link.

(S3) Integrity of binding data: The CN confirms the integrity of the binding data to detect any unauthorised modification of the home address and/or the care-of address. This requirement is achieved through the use of keyed hash functions. The analysis given for the authentication of home address and care-of address applies here.

(S4) Freshness of binding data: The CN confirms the freshness of the binding data received from the MN and home link through the use of cryptographically protected tokens, sequence numbers, and nonces. If not confirmed, an attacker can launch a replay attack by replaying a previously authenticated binding create/update request. This requirement is achieved through the following measures.

1. The CN confirms whether the tokens enclosed in the early binding creation messages, i.e. messages **M5-INF**, **M5-SK**, and **M4-PK**, match the tokens sent by the CN in the home test and/or care-of test messages, i.e. messages **M2-INF** and **M3-INF**, **M2-SK**, and **M2-PK**. Specifically, **Verification CN2-INF** ensures that the keyed hash function enclosed in message **M5-INF** is generated using a key (K_{BM}) that is generated from fresh tokens,

Token_1 and Token_2, and thus, message **M5-INF** is fresh. Also, **Verification CN1-SK** ensures that the token (Token_2) enclosed in message **M5-SK** is fresh, and thus, message **M5-SK** is fresh. In addition, **Verification CN1-PK** ensures that the keyed hash function enclosed in message **M4-PK** is generated using a key (K_{BM}) that is generated from a fresh token (Token_2), and thus, message **M4-PK** is fresh.

2. The CN confirms whether the sequence number (Seq_{new}) enclosed in messages **M8-INF**, **M7-SK**, **M7-PK**, **M1-UPD**, and **M1-DEL** is greater than the sequence number (Seq) received in the previous valid binding create/update request from the MN. The sequence number (Seq_{new}) is protected using K_{MN-CN} or K_{BM}; thus without the corresponding key, it would be difficult for an attacker to change Seq_{new} without being detected.

3. The CN confirms whether the nonce (N_{CN}) enclosed in message **M3-UPD** matches the nonce sent by the CN in message **M2-UPD**.

(S5) Protection against resource exhaustion DoS attacks: The CN does not retain any state about individual MNs until it authenticates them first, which protects the CN from memory exhaustion DoS attacks. In addition, the CN is protected against CPU exhaustion DoS attacks as following: (1) most of the verifications done by the CN are based on cheap operations; and (2) the CN performs expensive operations (if any) only after a positive outcome from cheap ones, which gives the CN some assurance about the MN before performing heavy computations.

To summarise (shown in Table 3.1), the SK-based protocol and the PK-based-protocol achieve all the security requirements stated in Section 5.1 and protect against all well-known attacks stated in Section 2.2.1. The INF-based protocol achieves the security requirements and protects against well-known attacks, except where on-path attackers that could use their own public key to generate a non-used (home) address are concerned. Considering those on-path attackers, the INF-based protocol cannot provide sufficient protection due to the use of the unauthentic public key; it fails to authenticate the HoA and CoA. As a result, the author decides to exclude the INF-based protocol from the security formal verification and the performance evaluation discussed in the rest of this chapter.

6.2 Formal Verification of Protocols

This section formally verifies the security properties of the SK-based and PK-based correspondent registration protocols presented in Chapter 5. A formal verification of a security protocol is useful for identifying security flaws that are subtle and hard to find [67]. Formal methods are more effective than informal methods. For example, the Needham-Schroeder public key protocol [68] succeeded in informal analysis, but failed in formal verification [69, 70].

Various formal methods have been successfully used to verify and debug security protocols. Generally, a formal method uses the following approach for verification of a security protocol. High-level formal notations are used in the first three steps to (1) formally model the protocol, (2) formally state the invariants and preconditions of the protocol, and (3) formally specify the security properties against which the protocol is to be checked. In step 4, the protocol is verified against its properties using either a manual tool or an automated tool that understands these notations.

In this thesis, two formal methods are used to verify the SK-based and PK-based protocols: the protocol composition logic (PCL) method and the Casper/FDR2 model checker. These formal methods have proved successful for modelling and verifying several security protocols; they have been used to verify authentication, secrecy, and other security properties [71, 72, 73, 74, 75, 76, 77, 78]. Thus, we consider them also appropriate for the verification of the SK-based and PK-based protocols. The Casper/FDR2 model checker is used to verify the security properties of the protocols. If the protocols do not satisfy the specified security properties, then the FDR2 checker shows a counterexample which represents the reason against vulnerability. Although Casper/FDR2 model checker has shown that the SK-based and PK-based protocols are not vulnerable to attacks, it does not validate the correctness of the protocols. This validation is performed using the PCL method.

6.2.1 Protocol Composition Logic (PCL)

Protocol Composition Logic (PCL) is a logic for stating and proving security properties of network protocols that use key cryptography operations [71, 72, 74]. PCL has an axiomatic system and inference rules about individual protocol actions and temporal reasoning that yield assertions about protocols composed

of multiple steps. PCL also uses a novel form of induction, called the "honesty rule", for combining facts about one role with inferred actions of other roles. For example, if Alice receives a response from a message sent to Bob. Alice may use properties of Bob's role to infer that Bob must have performed certain actions before sending his reply. PCL proofs usually use modal formulas of the form θ $[\mathrm{P}]_X$ ϕ. This form means that starting from a state where formula θ is true, after actions P are executed in thread X, the formula ϕ is true in the resulting state.

6.2.1.1 Formal Verification using PCL

This section presents a formal correctness proof of the UPD phase using the PCL method. It first models the UPD phase using a simple "protocol programming language" based on [14] and summarised in Appendix E. It then formulates the security properties that the phase ought to satisfy from the CN's point of view. Finally, it proves the correctness of the phase using a proof system based on [14, 71, 72, 73, 74] and summarised in Appendix E. The verification of the other phases can be found in Appendix F.

Modelling the UPD phase

The UPD phase is formally described by three roles (MobileNode, HomeAgent, and CorrespondentNode), each specifying a sequence of actions to be executed by an honest principal. This is illustrated in Table 6.1.

UPD : MobileNode =
(MN, $\hat{CN}$, LT_{BRem}, LT_{BReq}, HoA, CoA, CNA, Ack, seq, K_{MN-HA}, K_{MN-CN})
[new seq_{new} such that seq_{new} := succ(seq);
K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN});
authenticator := enc HoA || CoA || CNA || LT_{BRem}, K_{MN-HA};
mac_{BU} := hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack || authenticator, K_{BM});
msg_1 := HoA, CoA, seq_{new}, LT_{BReq}, Ack, authenticator, mac_{BU}; send $\hat{MN}$, $\hat{CN}$, msg_1;
receive $\hat{CN}$, $\hat{MN}$, msg_4; match msg_4 / seq_{new}, LT_{BGrant}, mac_{BA};
match seq_{new} / seq_{new};
match mac_{BA} / hash(HoA || CoA || CNA || seq_{new} || LT_{BGrant}, K_{BM});
$]_{MN}$

UPD : HomeAgent = (HA, LT_{BRem}, HoA, CoA, K_{MN-HA}, K_{HA-CN})
[receive $\hat{CN}$, $\hat{HA}$, msg_2;
match msg_2 / HoA, CoA, LT_{BReq}, authenticator, N_{CN}, mac_{BCReq};
K_{BC} := hash(HoA || CoA || CNA || N_{CN}, K_{HA-CN});
match mac_{BCReq} / hash(HoA || CoA || LT_{BReq} || authenticator || N_{CN}, K_{BC});
values := dec authenticator, K_{MN-HA}; match values as HoA || CoA || CNA || LT_{BRem};
match HoA / HoA; match CoA / CoA;

```
    match LT_BRem / LT_BRem; isLess(LT_BReq, LT_BRem);
    mac_BCRep := hash(HoA || CoA || LT_BReq || N_CN, K_BC);
    msg3 := CoA, LT_BReq, N_CN, mac_BCRep; send ĤA, ĈN, msg3;
   ]_HA

UPD : CorrespondentNode = (CN, CNA, LT_BGrant, K_MN−CN, K_HA−CN)
   [ receive M̂N, ĈN, msg1;
    match msg1 / HoA, CoA, seq_new, LT_BReq, Ack, authenticator, mac_BU;
    isLess(seq, seq_new); K_BM := hash(HoA || CoA || CNA || seq_new, K_MN−CN);
    match mac_BU / hash(HoA || CoA || seq_new || LT_BReq || Ack || authenticator, K_BM);
    new N_CN; K_BC := hash(HoA || CoA || CNA || N_CN, K_HA−CN);
    mac_BCReq := hash(HoA || CoA || LT_BReq || authenticator || N_CN, K_BC);
    msg2 := HoA, CoA, LT_BReq, authenticator, N_CN, mac_BCReq; send ĈN, ĤA, msg2;
    receive ĤA, ĈN, msg3; match msg3 / CoA, LT_BReq, N_CN, mac_BCRep;
    match N_CN / N_CN; match mac_BCRep / hash(HoA || CoA || LT_BReq || N_CN, K_BC);
    mac_BA := hash(HoA || CoA || CNA || seq_new || LT_BGrant, K_BM);
    msg4 := seq_new, LT_BGrant, mac_BA; send ĈN, M̂N, msg4;
   ]_CN
```

Table 6.1: UPD phase written in PCL language

Security Properties

The security properties that the UPD phase ought to satisfy from the CN's point of view include the following: (1) the CN agrees on the identities of the MN and the HA, i.e. session authentication; (2) the derived K_{BM} and K_{BC} keys should not be known to any principal other than the MN and the CN, and the HA and the CN, respectively, i.e. key secrecy; and (3) the HA should confirm the MN's current CoA to the CN. These properties are formulated in Definitions 1 and 2, where the predicate ActionsInOrder(a_1, a_2, ..., a_n) means that the actions a_1, a_2, ..., a_n were executed in that order.

Definition 1 (Session Authentication for the CN)

The authentication property in PCL is formulated as matching conversations [14, 72]. The basic idea of matching conversations is that after execution of the CorrespondentNode role, it is proven that there exist roles of the intended MobileNode and HomeAgent with corresponding views of the interactions. For CorrespondentNode $\hat{CN}$, communicating with MobileNode $\hat{MN}$ and HomeAgent $\hat{HA}$, matching conversations are formulated as $\phi_{UPD,CN-auth}$ defined below, where msg_1, msg_2, msg_3, and msg_4 represent the corresponding UPD messages in Table 6.1. Note that the receive action corresponding to the last message sent by the CN, i.e. msg_4, is not part of the guarantee as the CN received no acknowledgement

for this message.

$$\begin{aligned}\phi_{UPD,CN-auth} ::= &\text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \supset \\ &\exists(\hat{MN} \wedge \hat{HA}).\text{ActionsInOrder}(\\ &\quad \text{Send(MN, } \hat{MN}, \hat{CN}, \text{msg}_1), \text{Receive(CN, } \hat{MN}, \hat{CN}, \text{msg}_1), \\ &\quad \text{Send(CN, } \hat{CN}, \hat{HA}, \text{msg}_2), \text{Receive(HA, } \hat{CN}, \hat{HA}, \text{msg}_2), \\ &\quad \text{Send(HA, } \hat{HA}, \hat{CN}, \text{msg}_3), \text{Receive(CN, } \hat{HA}, \hat{CN}, \text{msg}_3), \\ &\quad \text{Send(CN, } \hat{CN}, \hat{MN}, \text{msg}_4))\end{aligned}$$

Definition 2 (Key Secrecy for the CN)
The UPD phase is said to provide key secrecy for the CN if $\phi_{UPD,CN-sec1}$ and $\phi_{UPD,CN-sec2}$ hold, where:

$$\begin{aligned}\phi_{UPD,CN-sec1} ::= &(\text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Has}(\hat{Z}, \text{K}_{BM})) \supset \\ &\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}\end{aligned}$$

$$\begin{aligned}\phi_{UPD,CN-sec2} ::= &(\text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{BC})) \supset \\ &\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}\end{aligned}$$

Security Guarantee
Theorem 1 states the security guarantee for the CN. It states that starting from a state in which the invariants and preconditions (defined in Table 6.2) of the UPD phase hold, if the CorrespondentNode role is executed, then the desired authentication and secrecy properties are guaranteed in the resulting state. Informally, the formula of theorem 1 guarantees that the CN, the MN and the HA have consistent views of protocol runs. In addition, it states that only the CN and the MN possess key K_{BM}, and only the CN and the HA possess key K_{BC}. The CN deduces its security properties based on the actions that it performs, the properties of certain cryptographic primitives, i.e. keyed hashes, and knowledge of the behaviour of honest MN and HA (by definition, an honest principal behaves in accordance with the protocol).

Theorem 1 (CN Security Guarantee)
After execution of the CorrespondentNode role, session authentication and key secrecy are guaranteed if the formulas in Table 6.2 hold. Formally,

$$\Gamma_{UPD1} \wedge \Gamma_{UPD2} \wedge \theta_{UPD1} \wedge \theta_{UPD2} \wedge \theta_{UPD3} \vdash [\text{UPD} : \text{CorrespondentNode}]_{CN}\ \text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \wedge \hat{CN} \neq \hat{MN} \neq \hat{HA} \supset \phi_{UPD,CN-auth} \wedge \phi_{UPD,CN-sec1} \wedge \phi_{UPD,CN-sec2}$$

Where:

- Γ_{UPD1} is the first invariant of the UPD. It states that the CN and the MN derive key K_{BM} locally and do not reveal it.
- Γ_{UPD2} is the second invariant of the UPD. It states that the CN and the HA derive key K_{BC} locally and do not reveal it.
- θ_{UPD1} is the first precondition of the UPD. It requires that key K_{MN-HA} is only known to the MN and the HA.
- θ_{UPD2} is the second precondition of the UPD. It requires that key K_{MN-CN} is only known to the CN and the MN.
- θ_{UPD3} is the third precondition of the UPD. It requires that key K_{HA-CN} is only known to the CN and the HA.

$$\theta_{UPD1} := (\text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{MN-HA})) \supset \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$$

$$\theta_{UPD2} := (\text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Has}(\hat{Z}, \text{K}_{MN-CN})) \supset \hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}$$

$$\theta_{UPD3} := (\text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{HA-CN})) \supset \hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}$$

$$\Gamma_{UPD1} := \text{Computes}(\hat{Z}, \text{HASH}(\hat{Z}, \text{HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{seq}_{new}, \text{K}_{MN-CN})) \supset \neg(\text{Send}(\hat{Z}, \text{m}) \wedge \text{Contains}(\text{m}, \text{HASH}(\hat{Z}, \text{HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{seq}_{new}, \text{K}_{MN-CN})))$$

$$\Gamma_{UPD2} := \text{Computes}(\hat{Z}, \text{HASH}(\hat{Z}, \text{HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{N}_{CN}, \text{K}_{HA-CN})) \supset \neg(\text{Send}(\hat{Z}, \text{m}) \wedge \text{Contains}(\text{m}, \text{HASH}(\hat{Z}, \text{HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{N}_{CN}, \text{K}_{HA-CN})))$$

Table 6.2: UPD phase preconditions and invariants

Proof of Theorem 1

The secrecy of K_{BM} ($\phi_{UPD,CN-sec1}$) and K_{BC} ($\phi_{UPD,CN-sec2}$) are proved first, and are then used in proving the authentication property ($\phi_{UPD,CN-auth}$). The secrecy properties are formalized using the Has(X, s) predicate and require that

X refers only to honest principals that shared the secret s. In $\phi_{UPD,CN-sec1}$, X is either the MN or the CN, and s is the key K_{BM}. Similarly, in $\phi_{UPD,CN-sec2}$, X is either the HA or the CN, and s is the key K_{BC}.

An induction on the basic sequences of various UPD roles is performed to show that honest principals do not perform actions that compromise the secrecy of the key K_{BM} (and K_{BC}). Informally, each induction step in the proof asserts that if, at the beginning of a basic sequence, the key K_{BM} has not already been compromised, i.e. SafeNet(K_{BM}) holds, then the basic sequence executed by a thread Z does not perform any actions that compromise the secrecy of the key K_{BM}, i.e. SendSafeMsg(Z, K_{BM}). The secrecy of the key K_{BM} is reasoned as follows:

1. Let MobileNode_1 be the first basic sequence of the MN role. Though MobileNode_1 sends out a keyed hash (recall that the K_{BM} is a hash keyed by the K_{MN-CN}), as the key hash has a structure different from that used for the K_{BM}, the **SH4** axiom can be used to argue that the send action is safe, as indicated in lines (1)-(4).

2. The basic sequence MobileNode_2 is the easy case; it contains no send action.

3. The proof for the remaining basic sequences follows a similar approach.

4. From (4), (6), (10), (14), and (18), and by application of the **NET** rule and the **HPOS** axiom, if an entity Z has K_{BM}, it implies that Z has K_{MN-CN}, as indicated in line (19).

5. Finally, according to (19) and using the secrecy of K_{MN-CN}, θ_{UPD2}, the proof of secrecy of K_{BM} is concluded, as indicated in line (20).

Let $[\text{MobileNode}_1]_{MN}$: [new seq_{new} such that $\text{seq}_{new} := \text{succ}(\text{seq})$;
$K_{BM} := \text{hash}(\text{HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{seq}_{new}, K_{MN-CN})$;
$\text{authenticator} := \text{enc HoA} \,||\, \text{CoA} \,||\, \text{CNA} \,||\, \text{LT}_{BRem}, K_{MN-HA}$;
$\text{mac}_{BU} := \text{hash}(\text{HoA} \,||\, \text{CoA} \,||\, \text{seq}_{new} \,||\, \text{LT}_{BReq} \,||\, \text{Ack} \,||\, \text{authenticator}, K_{BM})$;
$\text{msg}_1 := \text{HoA, CoA, seq}_{new}, \text{LT}_{BReq}, \text{Ack, authenticator, mac}_{BU}$;
send $\hat{MN}$, $\hat{CN}$, msg_1;$]_{MN}$ (1)

SH4 $[\text{MobileNode}_1]_{MN}$ (HoA || CoA || CNA || seq_{new}) $\neq$ (HoA || CoA || seq_{new} || LT_{BReq} || Ack || authenticator) $\supset$ Safe(mac_{BU}, K_{BM}) (2)

SH*, (2) [MobileNode$_1$]$_{MN}$ Safe($\hat{MN}$, $\hat{CN}$, msg$_1$, K$_{BM}$)

(3)

(3) SafeNet(K$_{BM}$) [MobileNode$_1$]$_{MN}$ SendsSafeMsg($\hat{MN}$, K$_{BM}$)

(4)

Let [MobileNode$_2$]$_{MN}$: [receive $\hat{CN}$, $\hat{MN}$, msg$_4$;
match msg$_4$ / seq$_{new}$, LT$_{BGrant}$, mac$_{BA}$; match seq$_{new}$ / seq$_{new}$;
match mac$_{BA}$ / hash(HoA || CoA || CNA || seq$_{new}$ || LT$_{BGrant}$, K$_{BM}$);]$_{MN}$

(5)

SafeNet(K$_{BM}$) [MobileNode$_2$]$_{MN}$ SendsSafeMsg($\hat{MN}$, K$_{BM}$)

(6)

Let [HomeAgent]$_{HA}$: [receive $\hat{CN}$, $\hat{HA}$, msg$_2$;
match msg$_2$ / HoA, CoA, LT$_{BReq}$, authenticator, N$_{CN}$, mac$_{BCReq}$;
K$_{BC}$:= hash(HoA || CoA || CNA || N$_{CN}$, K$_{HA-CN}$);
match mac$_{BCReq}$ / hash(HoA || CoA || LT$_{BReq}$ || authenticator || N$_{CN}$, K$_{BC}$); values := dec authenticator, K$_{MN-HA}$;
match values as HoA || CoA || CNA || LT$_{BRem}$;
match HoA / HoA; match CoA / CoA;
match LT$_{BRem}$ / LT$_{BRem}$; isLess(LT$_{BReq}$, LT$_{BRem}$);
mac$_{BCRep}$:= hash(HoA || CoA || LT$_{BReq}$ || N$_{CN}$, K$_{BC}$);
msg$_3$:= CoA, LT$_{BReq}$, N$_{CN}$, mac$_{BCRep}$; send $\hat{HA}$, $\hat{CN}$, msg$_3$;]$_{HA}$

(7)

SH4 [HomeAgent]$_{HA}$ (HoA || CoA || LT$_{BReq}$ || N$_{CN}$) $\neq$ (HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator) $\supset$ Safe(mac$_{BCRep}$, K$_{BM}$)

(8)

SH*, (8) [HomeAgent]$_{HA}$ Safe($\hat{HA}$, $\hat{CN}$, msg$_3$, K$_{BM}$)

(9)

(9) SafeNet(K$_{BM}$) [HomeAgent]$_{HA}$ SendsSafeMsg($\hat{HA}$, K$_{BM}$)

(10)

Let [CorrespondentNode$_1$]$_{CN}$: [receive $\hat{MN}$, $\hat{CN}$, msg$_1$;
match msg$_1$ / HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, authenticator, mac$_{BU}$; isLess(seq, seq$_{new}$);
K$_{BM}$:= hash(HoA || CoA || CNA || seq$_{new}$, K$_{MN-CN}$);
match mac$_{BU}$ / hash(HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator, K$_{BM}$);
new N$_{CN}$; K$_{BC}$:= hash(HoA || CoA || CNA || N$_{CN}$, K$_{HA-CN}$);
mac$_{BCReq}$:= hash(HoA || CoA || LT$_{BReq}$ || authenticator || N$_{CN}$, K$_{BC}$);
msg$_2$:= HoA, CoA, LT$_{BReq}$, authenticator, N$_{CN}$, mac$_{BCReq}$;
send $\hat{CN}$, $\hat{HA}$, msg$_2$;]$_{CN}$

(11)

SH4	$[\text{CorrespondentNode}_1]_{CN}$ (HoA || CoA || LT_{BReq} || authenticator || N_{CN}) $\neq$ (HoA || CoA || seq_{new} || LT_{BReq} || Ack || authenticator) $\supset$ Safe(mac_{BCReq}, K_{BM}) (12)
SH*, (12)	$[\text{CorrespondentNode}_1]_{CN}$ Safe($\hat{CN}$, $\hat{HA}$, msg_2, K_{BM}) (13)
(13)	SafeNet(K_{BM}) $[\text{CorrespondentNode}_1]_{CN}$ SendsSafeMsg($\hat{CN}$, K_{BM}) (14)
Let $[\text{CorrespondentNode}_2]_{CN}$:	[receive $\hat{HA}$, $\hat{CN}$, msg_3; match msg_3 / CoA, LT_{BReq}, N_{CN}, mac_{BCRep}; match N_{CN} / N_{CN}; match mac_{BCRep} / hash(HoA || CoA || LT_{BReq} || N_{CN}, K_{BC}); mac_{BA} := hash(HoA || CoA || CNA || seq_{new} || LT_{BGrant}, K_{BM}); msg_4 := seq_{new}, LT_{BGrant}, mac_{BA}; send $\hat{CN}$, $\hat{MN}$, msg_4;]$_{CN}$ (15)
SH4	$[\text{CorrespondentNode}_2]_{CN}$ (HoA || CoA || CNA || seq_{new} || LT_{BGrant}) $\neq$ (HoA || CoA || seq_{new} || LT_{BReq} || Ack || authenticator) $\supset$ Safe(mac_{BA}, K_{BM}) (16)
SH*, (16)	$[\text{CorrespondentNode}_2]_{CN}$ Safe($\hat{CN}$, $\hat{MN}$, msg_4, K_{BM}) (17)
(17)	SafeNet(K_{BM}) $[\text{CorrespondentNode}_2]_{CN}$ SendsSafeMsg($\hat{CN}$, K_{BM}) (18)
(4), (6), (10), (14), (18), NET, HPOS	Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has(Z, K_{BM}) $\supset$ Has(Z, K_{MN-CN}) (19)
(19), θ_{UPD2}	Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has(Z, K_{BM}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN} \supset \phi_{UPD,CN-sec1}$ (20)

The authentication property ($\phi_{UPD,CN-auth}$) is formulated as matching conversations. It can be asserted only when the secrecy of K_{BM} and K_{BC} are guaranteed and is reasoned as follows:

1. Since the CN is honest, obviously it knows that its own actions are in order, i.e. axioms **AA1, AR1, AA4** are used in line (1) to conclude that the CN has performed a certain sequence of actions.

2. Since the CN received and verified msg_1, there must be some entity $\hat{X}$ that has $\text{HASH}_{k_{BM}}$\{HoA \|\| CoA \|\| seq_{new} \|\| LT_{BReq} \|\| Ack \|\| authenticator\} and that sends out msg_1 at a previous stage, as indicated in line (2).

3. The entity $\hat{X}$ has $\text{HASH}_{k_{BM}}\{\text{HoA} \mid\mid \text{CoA} \mid\mid \text{seq}_{new} \mid\mid \text{LT}_{BReq} \mid\mid \text{Ack} \mid\mid \text{authenticator}\}$, which implies that $\hat{X}$ either computes $\text{HASH}_{k_{BM}}\{\text{HoA} \mid\mid ... \mid\mid \text{authenticator}\}$ itself, or there must be a third entity $\hat{Y}$ that computes and sends $\text{HASH}_{k_{BM}}\{\text{HoA} \mid\mid ... \mid\mid \text{authenticator}\}$ at a previous stage, as indicated in line (3).

4. For an entity $\hat{Z} \in \{\hat{X}, \hat{Y}\}$ to be able to compute $\text{HASH}_{k_{BM}}\{\text{HoA} \mid\mid ... \mid\mid \text{authenticator}\}$, it must have the key K_{BM}, as indicated in line (4).

5. From (4) and $\phi_{UPD,CN-sec1}$, $\hat{Z}$ must be either the CN itself or the MN, as indicated in line (5).

6. The CN knows that it does not send out msg_1 by itself; thus, it must be the MN that had computed and sent out msg_1. Furthermore, according to (2) and (5), the CN can conclude that the MN sends msg_1 before the CN receives it, as indicated in line (6).

7. Due to the freshness of the nonce N_{CN} generated by the CN, the HA can only receive msg_2 after the CN sends it, as indicated in lines (7)-(9).

8. The CN assumes that the honest HA acts honestly by obeying the protocol. Therefore, the HA must receive msg_2 before sending msg_3, as indicated in line (10).

9. Steps 11 to 15 are similar to steps 2 to 6, but HA, msg_3, K_{BC}, $\text{HASH}_{k_{BC}}\{\text{HoA} \mid\mid \text{CoA} \mid\mid \text{LT}_{BReq} \mid\mid \text{N}_{CN}\}$, and $\phi_{UPD,CN-sec2}$ are used instead of MN, msg_1, K_{BM}, $\text{HASH}_{k_{BM}}\{\text{HoA} \mid\mid \text{CoA} \mid\mid \text{seq}_{new} \mid\mid \text{LT}_{BReq} \mid\mid \text{Ack} \mid\mid \text{authenticator}\}$, and $\phi_{UPD,CN-sec1}$, respectively. Also, the CN can conclude that the HA sends msg_3 before the CN receives it.

10. According to (1), (6), (9), (10), and (15), all the actions are matched as in line (16). Hence, the CN can conclude that the security property of session authentication is guaranteed in the UPD phase.

AA1, AR1, AA4 [UPD : CorrespondentNode]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_1$) < Send(CN, $\hat{CN}$, $\hat{HA}$, msg$_2$) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg$_3$) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_4$)

(1)

AR1, HASH3, Γ_{UPD1} θ_{UPD2} [receive $\hat{MN}$, $\hat{CN}$, msg$_1$; match msg$_1$ / HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, authenticator, mac$_{BU}$; isLess(seq, seq$_{new}$); K$_{BM}$:= hash(HoA || CoA || CNA || seq$_{new}$, K$_{MN-CN}$); match mac$_{BU}$ / hash(HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator, K$_{BM}$);]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, authenticator, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator} ⊃ (∃X.Has(X, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) ∧ Send(X, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator})) ∧ (Send(X, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_1$))

(2)

(2), HASH4 Has(X, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) ⊃ Computes(X, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) ∨ ∃Y,m.Computes(Y, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) ∧ Send(Y, m) ∧ Contains(m, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator})

(3)

(3), HASH1 Computes(Z, HASH$_{k_{BM}}${HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator}) ⊃ Has($\hat{Z}$, K$_{BM}$) ∧ Has($\hat{Z}$, HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator)

(4)

(4), $\phi_{UPD,CN-sec1}$ θ_{UPD2} [receive $\hat{MN}$, $\hat{CN}$, msg$_1$; match msg$_1$ / HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, authenticator, mac$_{BU}$; isLess(seq, seq$_{new}$); K$_{BM}$:= hash(HoA || CoA || CNA || seq$_{new}$, K$_{MN-CN}$); match mac$_{BU}$ / hash(HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator, K$_{BM}$);]$_{CN}$ Honest($\hat{CN}$) ∧ Honest($\hat{MN}$) ∧ Has($\hat{Z}$, K$_{BM}$) ⊃ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}$

(5)

(2), (5) θ_{UPD2} [receive $\hat{MN}$, $\hat{CN}$, msg$_1$; match msg$_1$ / HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, authenticator, mac$_{BU}$; isLess(seq, seq$_{new}$); K$_{BM}$:= hash(HoA || CoA || CNA || seq$_{new}$, K$_{MN-CN}$); match mac$_{BU}$ / hash(HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack || authenticator, K$_{BM}$);]$_{CN}$ Honest($\hat{CN}$) ∧ Honest($\hat{MN}$) ⊃ Send(MN, $\hat{MN}$, $\hat{CN}$, msg$_1$) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_1$)

(6)

AN3 [new N$_{CN}$;]$_{CN}$ Fresh(CN, N$_{CN}$)

(7)

(7), FS1, P1, Γ_{UPD2} [K$_{BC}$:= hash(HoA || CoA || CNA || N$_{CN}$, K$_{HA-CN}$); mac$_{BCReq}$:= hash(HoA || CoA || LT$_{BReq}$ || authenticator || N$_{CN}$, K$_{BC}$); msg$_2$:= HoA, CoA, LT$_{BReq}$, authenticator, N$_{CN}$, mac$_{BCReq}$; send $\hat{CN}$, $\hat{HA}$, msg$_2$;]$_{CN}$ FirstSend(CN, N$_{CN}$, msg$_2$)

(8)

(1), (8), FS2, Γ_{UPD2} $\theta_{UPD2} \wedge \theta_{UPD3}$ [new N$_{CN}$; K$_{BC}$:= hash(HoA || CoA || CNA || N$_{CN}$, K$_{HA-CN}$); mac$_{BCReq}$:= hash(HoA || CoA || LT$_{BReq}$ || authenticator || N$_{CN}$, K$_{BC}$); msg$_2$:= HoA, CoA, LT$_{BReq}$, authenticator, N$_{CN}$, mac$_{BCReq}$; send $\hat{CN}$, $\hat{HA}$, msg$_2$;]$_{CN}$ Receive(HA, $\hat{CN}$, $\hat{HA}$, msg$_2$) ∧ $\hat{HA} \neq \hat{CN}$ ⊃ Send(CN, $\hat{CN}$, $\hat{HA}$, msg$_2$) < Receive(HA, $\hat{CN}$, $\hat{HA}$, msg$_2$)

(9)

(9), HON, AA4, P1 Honest($\hat{CN}$) ∧ Honest($\hat{HA}$) ⊃ Receive(HA, $\hat{CN}$, $\hat{HA}$, msg$_2$) < Send(HA, $\hat{HA}$, $\hat{CN}$, msg$_3$)

(10)

AR1, HASH3

θ_{UPD3} [receive $\hat{HA}$, $\hat{CN}$, msg_3; match msg_3 / CoA, LT_{BReq}, N_{CN}, mac_{BCRep}; match N_{CN} / N_{CN}; match mac_{BCRep} / hash(HoA || CoA || LT_{BReq} || N_{CN}, K_{BC});]$_{CN}$

Receive(CN, $\hat{HA}$, $\hat{CN}$, CoA, LT_{BReq}, N_{CN}, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}} $\supset$

($\exists$X.Has(X, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) $\wedge$ Send(X, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}})) $\wedge$ (Send(X, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_3))

(11)

(11), HASH4

Has(X, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) $\supset$ Computes(X, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) $\vee$ $\exists$Y,m.Computes(Y, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) $\wedge$ Send(Y, m) $\wedge$ Contains(m, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}})

(12)

(12), HASH1

Computes(Z, $HASH_{k_{BC}}${HoA || CoA || LT_{BReq} || N_{CN}}) $\supset$ Has($\hat{Z}$, K_{BC}) $\wedge$ Has($\hat{Z}$, HoA || CoA || LT_{BReq} || N_{CN})

(13)

(13), $\phi_{UPD,CN-sec2}$

θ_{UPD3} [receive $\hat{HA}$, $\hat{CN}$, msg_3; match msg_3 / CoA, LT_{BReq}, N_{CN}, mac_{BCRep}; match N_{CN} / N_{CN}; match mac_{BCRep} / hash(HoA || CoA || LT_{BReq} || N_{CN}, K_{BC});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has($\hat{Z}$, K_{BC}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}$

(14)

(11), (14)

θ_{UPD3} [receive $\hat{HA}$, $\hat{CN}$, msg_3; match msg_3 / CoA, LT_{BReq}, N_{CN}, mac_{BCRep}; match N_{CN} / N_{CN}; match mac_{BCRep} / hash(HoA || CoA || LT_{BReq} || N_{CN}, K_{BC});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg_3) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_3)

(15)

(1), (6), (9), (10), (14)

$\theta_{UPD2} \wedge \theta_{UPD3}$ [CorrespondentNode]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1) <

Send(CN, $\hat{CN}$, $\hat{HA}$, msg_2) < Receive(HA, $\hat{CN}$, $\hat{HA}$, msg_2) <

Send(HA, $\hat{HA}$, $\hat{CN}$, msg_3) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_3) <

Send(CN, $\hat{CN}$, $\hat{MN}$, msg_4) $\supset$ $\phi_{UPD,CN-auth}$

(16)

6.2.2 Casper Tool and FDR2 Model Checker

Casper is a compiler for the analysis of security protocols [16]. It provides a concise notation for specifying protocols and security properties. It accepts a script containing an abstract description of a protocol as well as an intruder and produces a CSP (Communicating Sequential Processes) [79] code which is verified using the FDR2 (Failure-Divergence Refinement) model checker [17]. The translation of the Casper script to the CSP code is made automatically and transparently to the user, and the results of the verification are presented in terms of the Casper model.

A Casper script defines not only the operation of a protocol but also the system to be checked. It contains a number of sections, including the following. The 'Free variable' section declares the types of variables and functions that are

used in the protocol definition. For example, in Table 6.3, the variables 'seq1' and 'ncn' have been declared as a sequence number and a nonce, respectively.

The 'Processes' section represents the roles played by the different honest principals. The parameters and the variables following the keyword 'knows' define the knowledge that the principal in question is expected to have at the beginning of the protocol run.

The 'Protocol description' section defines the sequence of messages in the protocol. For example, in Table 6.3, the notation '{hoa, coa, seq1}{kmnha} % authenticator' represents that the values of hoa, coa, and seq1 are encrypted with key kmnha. In addition, this notation also means that the recipient should not attempt to decrypt this coded value, but should instead store it in the variable 'authenticator'. Similarly, the notation 'authenticator % {hoa, coa, seq1}{kmnha}' indicates that the sender should send the value stored in the variable 'authenticator', but the recipient should expect a value of the form given by '{hoa, coa, seq1}{kmnha}'.

The 'Specification' section is used to specify security properties of the protocol. For example, the line starting with 'Agreement (mn, cn, [hoa, coa])' defines the authentication property associated with the authentication of an MN to a CN; it means that "the MN is correctly authenticated to the CN, and they agree upon the values of HoA and CoA."

The 'Actual variables' and 'System' sections represent the actual system to be checked. The type of variables to be used in the system to be checked is defined in the former section; whereas, the number and the role of the principals involved in the system are defined in the latter section. Finally, the 'Intruder Information' section defines the intruder's identity and his/her initial knowledge.

6.2.2.1 Formal Verification using Casper/FDR2

This section presents a formal verification of the update (UPD) phase using the Casper/FDR2 model checker. The verification of the other phases can be found in Appendix G. The first step of the verification is to specify the UPD phase and its security properties in a Casper script as shown in Table 6.3. The second step is to run the Casper tool to automatically translate the script to a CSP code. Finally, the CSP code is run in the FDR model checker. The result of the verification is shown in Table 6.4, which confirms that the UPD phase enables a CN to securely authenticate an MN and a home link.

 # Free variables

mn : MNode

cn : CNode

ha : HomeAgent

hoa, coa : IPv6Address

seq1 : SequenceNumber

ncn : Nonce

kmnha, khacn, kmncn : SessionKey

InverseKeys = (kmnha, kmnha),(khacn, khacn),(kmncn, kmncn)

HMAC : HashFunction

 # Processes

MOBILENODE(mn, cn, hoa, coa, seq1, kmnha, kmncn) knows HMAC

CORRESNODE(cn, ha, ncn, khacn, kmncn) knows HMAC

HOMEAGENT(ha, hoa, coa, kmnha, khacn) knows HMAC

 # Protocol description

0. → mn : cn

1. mn → cn : hoa, coa, seq1, {hoa, coa, seq1}{kmnha} % authenticator, HMAC(kmncn, hoa, coa, seq1)

2. cn → ha : hoa, coa, ncn, authenticator % {hoa, coa, seq1}{kmnha}, HMAC(khacn, hoa, coa, ncn)

3. ha → cn : hoa, coa, HMAC(khacn, hoa, coa)

4. cn → mn : hoa, coa, HMAC(kmncn, hoa, coa)

 # Specification

Agreement(mn, cn, [hoa, coa])

Agreement(ha, cn, [hoa, coa])

 # Actual variables

MN, Mallory : MNode

CN : CNode

HA : HomeAgent

HoA, CoA, MalA : IPv6Address

Seq1, Seqm : SequenceNumber

Ncn, Nm : Nonce

Kmnha, Khacn, Kmncn, Kmala : SessionKey

InverseKeys = (Kmnha, Kmnha),(Khacn, Khacn),(Kmncn, Kmncn), (Kmala, Kmala)

 # System

MOBILENODE(MN, CN, HoA, CoA, Seq1, Kmnha, Kmncn)

```
CORRESNODE(CN, HA, Ncn, Khacn, Kmncn)
HOMEAGENT(HA, HoA, CoA, Kmnha, Khacn)

# Intruder Information
Intruder = Mallory
IntruderKnowledge = {MN, CN, HA, Mallory, HoA, CoA, MalA, Seqm, Nm, Kmala}
```

Table 6.3: Casper specification of the UPD phase

```
Initialising; please wait.... Ready.
Casper version 1.8
Parsing...
Type checking...
Consistency checking...
Compiling...
Writing output...
Output written to /mnt/ntfs/Casper/UPD.csp
Done
Starting FDR
Checking /mnt/ntfs/Casper/UPD.csp
Checking assertion AUTH1_M::AuthenticateMOBILENODEToCORRESNODEAgreement_hoa_coa
T= AUTH1_M::SYSTEM_1
No attack found
Checking assertion AUTH1_M::AuthenticateHOMEAGENTToCORRESNODEAgreement_hoa_coa
T= AUTH2_M::SYSTEM_1
No attack found
Done
```

Table 6.4: Verification results of the UPD phase using Casper/FDR2

6.3 Performance Evaluation

This section reports the performance evaluation of the SK-based and PK-based protocols by comparing them to the standard correspondent registration protocol, i.e. when the return routability (RR) procedure is used to protect the registration process (see Section 2.5). In addition, the SK-based protocol is compared to the SSKv1 and SSKv2 protocols (see Sections 3.3.1.1 and 3.3.1.2). The performance is measured in terms of correspondent registration delay (CR-Delay) measured

in seconds and control signalling overhead measured in bits per second. The CR-Delay is defined as the total amount of time taken for the MN to register a new CoA with the CN. The control signalling overhead is the total amount of Mobile IPv6 signalling traffic sent and received by all involved entities, i.e. the MN, CN, and HA.

6.3.1 Simulation Modelling

OPNET^{TM} Modeler version 14.5 has been used to simulate the performance of the SK-based and PK-based protocols. In particular, the control signalling overhead is investigated at all involved entities. The CR-Delay is also investigated when varying levels of background traffic on the network are applied. The simulation results obtained are then compared to those when the RR, SSKv1, and SSKv2 protocols are run.

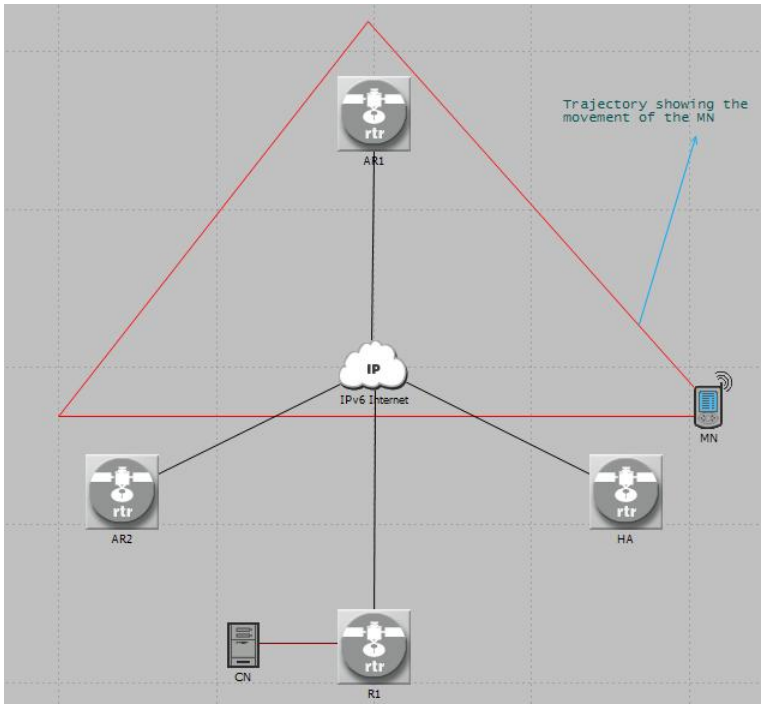

Figure 6.1: Simulation model - stationary CN case

Two network models are constructed to measure the performance in the stationary and mobile CN cases. The first model, depicted in Figure 6.1, is composed of: (1) a single stationary CN that is connected via a router (R1) to the Internet; (2) a single MN that is located at a home link; (3) an HA that represents the

home link; and (4) two access routers (AR1 and AR2) that represent foreign links. The MN moves from the home link to the first foreign link, from the first foreign link to the second foreign link, and then from the second foreign link to the home link. In this way, the MN will create a new binding, update the binding, and delete the binding with the CN, respectively.

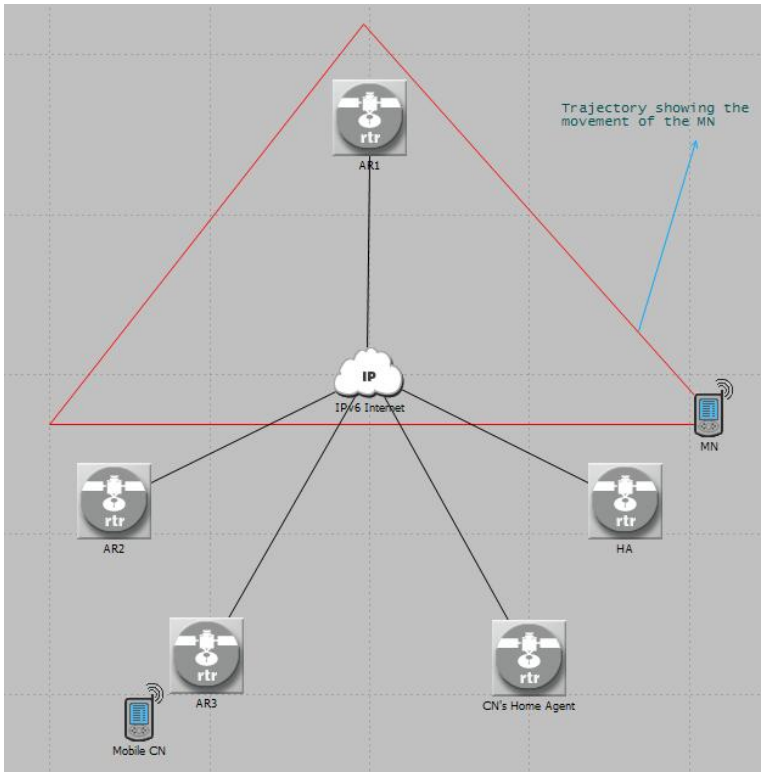

Figure 6.2: Simulation model - mobile CN case

The second model (depicted in Figure 6.2) is identical to the first model, but one additional access router (AR3) that represents a third foreign link is included. In addition, the CN is also mobile and is located at that foreign link. The router (R1) is now referred to as the CN's home agent and represents the home link of the mobile CN.

The HA and the ARs have been positioned in such a way that provide a continuous wireless coverage area for the MN. The MN performs one hundred passes (movement between HA and AR2) with three handoffs in each pass (one binding creation, one binding update, and one binding deletion). The OPNET Modeler's documentation recommends thirty as an initial rule of thumb for the number of repetitions to use. However, one hundred passes have been chosen in

the hope of averaging out any possible fluctuating factors.

Two separate simulation studies are performed. The first one is created to investigate the impact of the SK-based and PK-based protocols on CR-Delay, i.e. whether the protocols have led to any performance degradation through increasing CR-Delay. In this study, ten background traffic scenarios ranging from 0% to 90% at 10% increments are applied to investigate how different volumes of traffic may affect the results of the CR-Delay. The second simulation study is to investigate the control signalling overhead produced by all involved entities, i.e. the MN, CN, and HA, when SK-based, PK-based, RR, SSKv1, and SSKv2 protocols are executed.

6.3.2 Simulation Model Validation

This section validates the simulation models. For doing so, the OPNET debugger is used to prove that the SK-based protocol operates correctly. The OPNET debugger is applied to output the processes of generating and/or verifying session keys, cookies, tokens, and MACs values. In addition, relevant packets' information (i.e. source address, destination address, and packet size) has been inspected during runtime.

The simulation model in the stationary CN case (see Figure 6.1) has been used to validate the correctness of the SK-based protocol in the stationary CN case. The output from the OPNET debugger during simulation runtime is illustrated in Appendix C. Trace labels have been defined in the code to request output of specific information while monitoring simulation output. By observing the output, the messages exchanged were confirmed to be consistent with the expected mobility signalling exchanged among the MN, the CN, and the HA.

6.3.3 Simulation Results

This section presents and analyses simulation results obtained from the simulation study of both the CR-Delay and the control signalling overhead. It compares the results of the SK-based, PK-based, RR, SSKv1, and SSKv2 protocols.

6.3.3.1 Correspondent Registration Delay

This section presents an analysis of the CR-Delay simulation results in both the stationary and mobile CN cases. A selection of the simulation results are shown

in Figures 6.3 to 6.16. These figures illustrate the average CR-Delay in the three binding phases (i.e. creation, update, and deletion) at different network loads. As shown, the CR-Delays increase exponentially as the network load increases. This pattern is caused by the increase in the queuing delay experienced at each node. In addition, it is shown that the CR-Delay in the mobile CN case is higher than in the stationary CN case. The reason for this is due to the fact that all mobility-related messages that are sent to/from the mobile CN should be routed through the mobile CN's home link. As a result, the CR-Delay in the mobile CN case is higher by the amount of time needed to route mobility-related messages between the mobile CN's HoA and CoA. Of course, this amount depends on the current location of the mobile CN, i.e. how far the mobile CN is away from its home link.

Figures 6.3 to 6.8 show the CR-Delay in the binding creation phase for the stationary and mobile CN cases. Figures 6.3 and 6.4 illustrate the CR-Delay of the RR and SK-based protocols. These figures show that around 70% load in the stationary CN case and around 50% load in the mobile CN case, the SK-based protocol outperforms the RR protocol. This pattern could be explained as follows. At low network loads, the queuing delay experienced at each node has a less dominant impact on the CR-Delay. Therefore, the RR protocol produces a lower CR-Delay as it has shorter signalling messages and fewer processing operations. However, as the network load increases, the CR-Delay will be increasingly dominated by the queuing delays and the number of signalling messages exchanged will be a determining factor. Therefore, the SK-based protocol produces lower CR-Delay as it requires fewer number of signalling messages to create a binding.

Figures 6.5 and 6.6 illustrate the CR-Delay of the SSKv1, SSKv2, and SK-based protocols. As shown in these figures, at all loads, the SK-based protocol produces a higher CR-Delay than the SSKv1 and SSKv2 protocols. This is because the SK-based protocol requires four messages to create a new binding; whereas, the SSKv1 protocol and the SSKv2 protocol require one message and three messages, respectively, to create a new binding.

Figures 6.7 and 6.8 illustrate the CR-Delay of the RR and PK-based protocols. As shown, at all loads, the PK-based protocol produces a higher CR-Delay than the RR protocol. This is because the PK-based protocol uses computationally expensive signature generation and verification operations while creating a binding; whereas, the RR protocol uses computationally light hash operations.

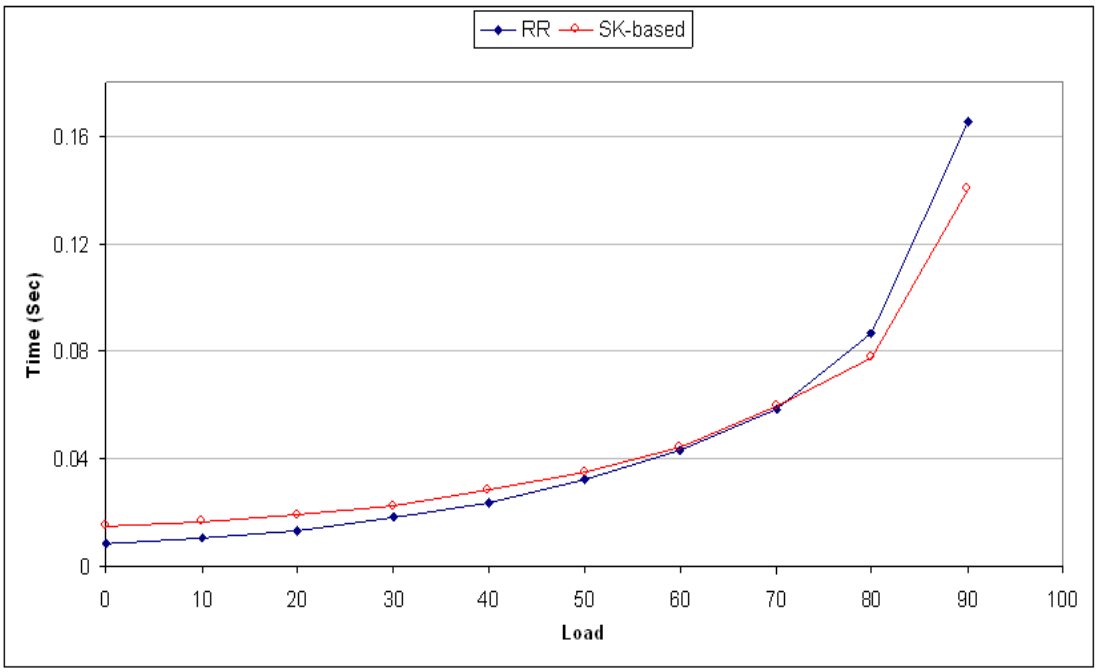

Figure 6.3: Average CR-Delay for RR and SK-based protocols vs. load (binding creation - stationary CN case)

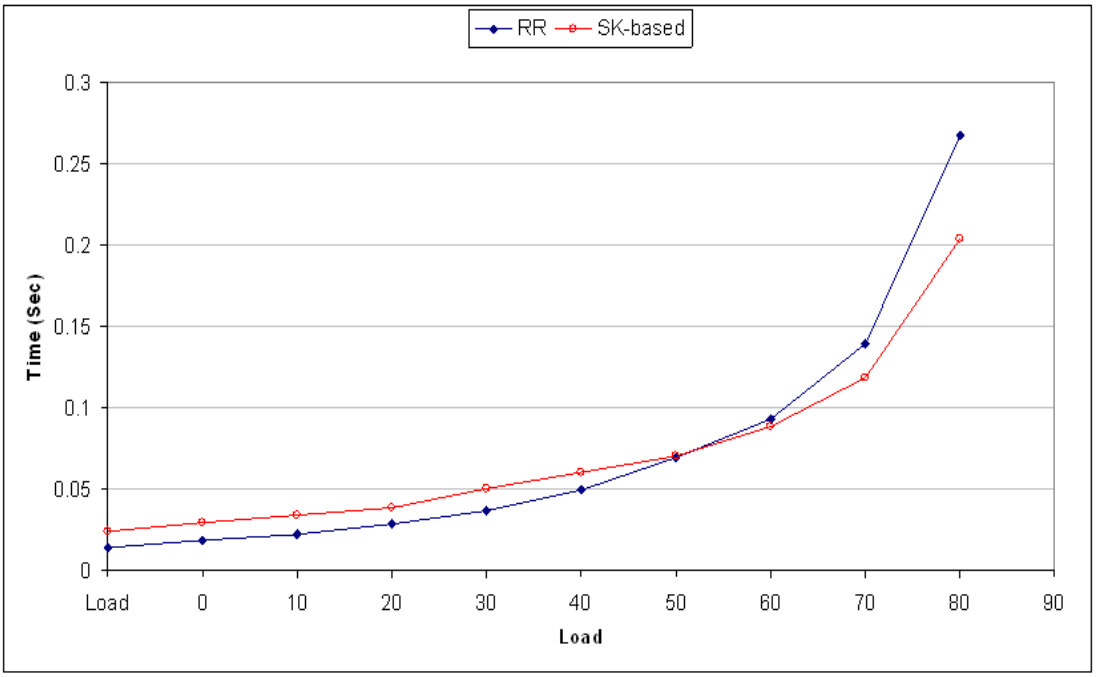

Figure 6.4: Average CR-Delay for RR and SK-based protocols vs. load (binding creation - mobile CN case)

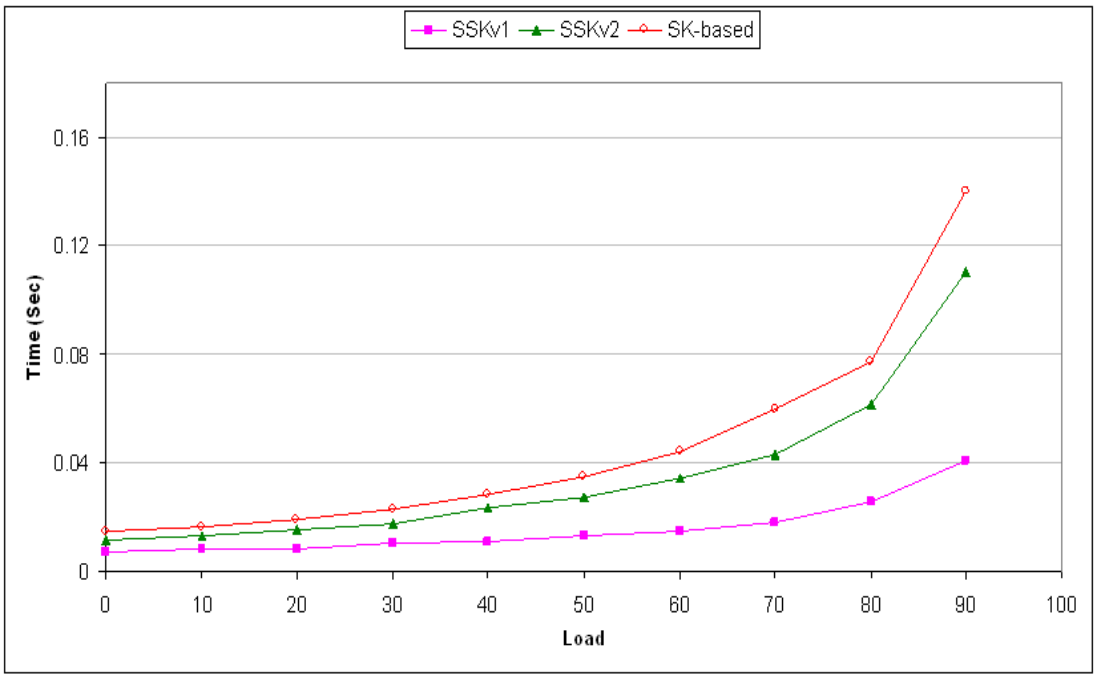

Figure 6.5: Average CR-Delay for SSKv1, SSKv2, and SK-based protocols vs. load (binding creation - stationary CN case)

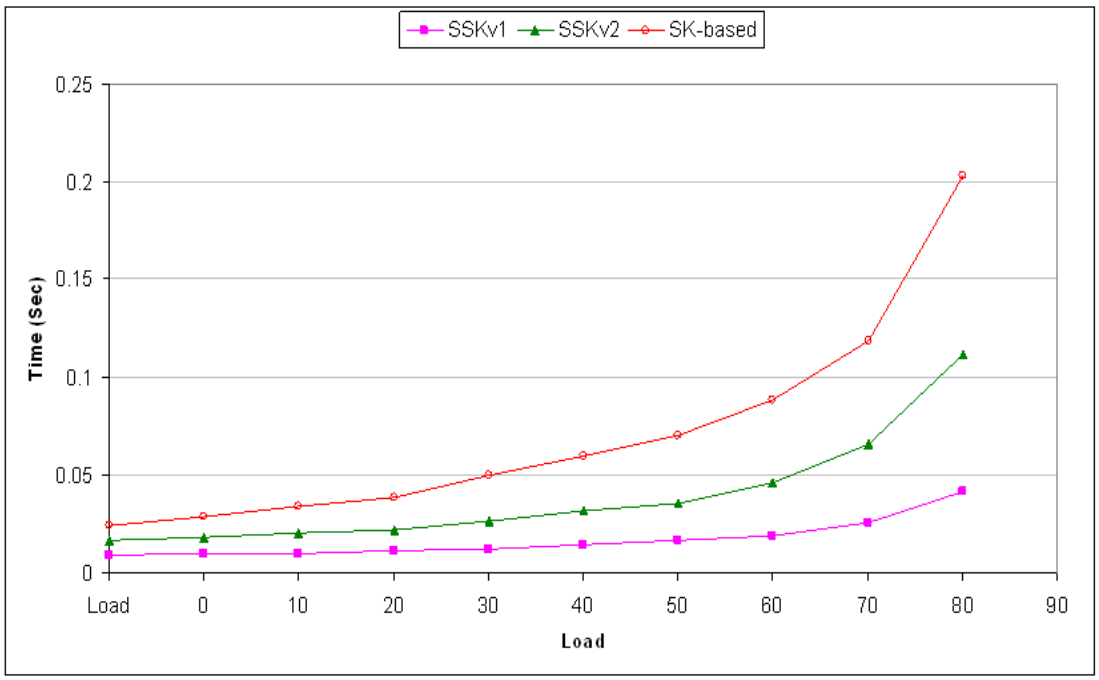

Figure 6.6: Average CR-Delay for SSKv1, SSKv2, and SK-based protocols vs. load (binding creation - mobile CN case)

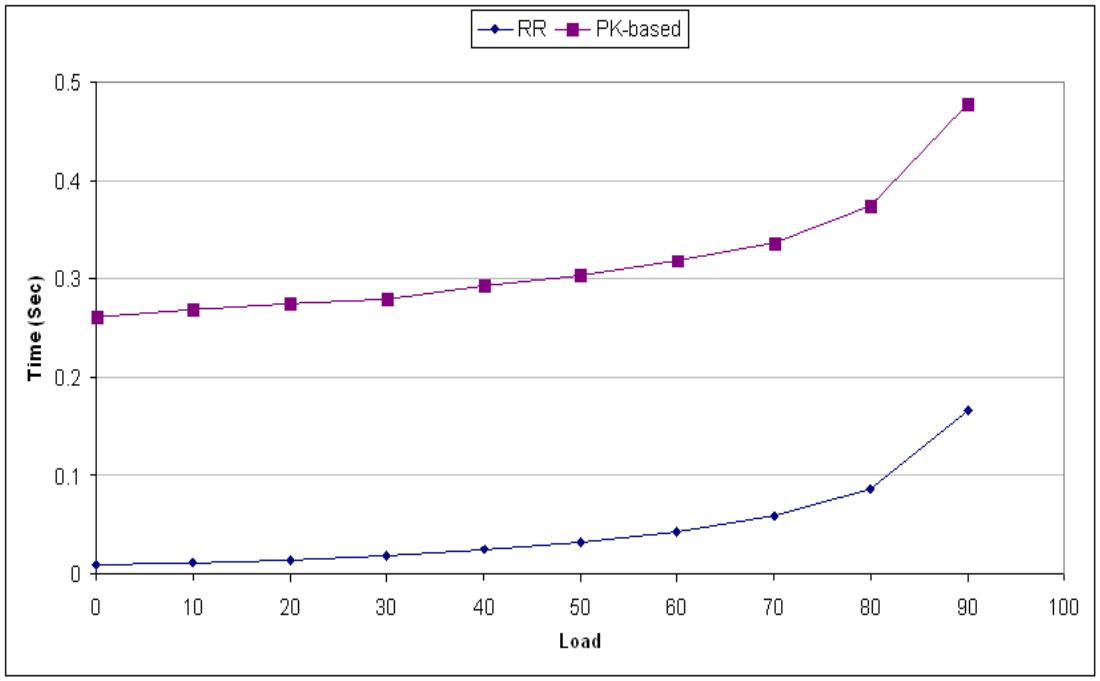

Figure 6.7: Average CR-Delay for RR and PK-based protocols vs. load (binding creation - stationary CN case)

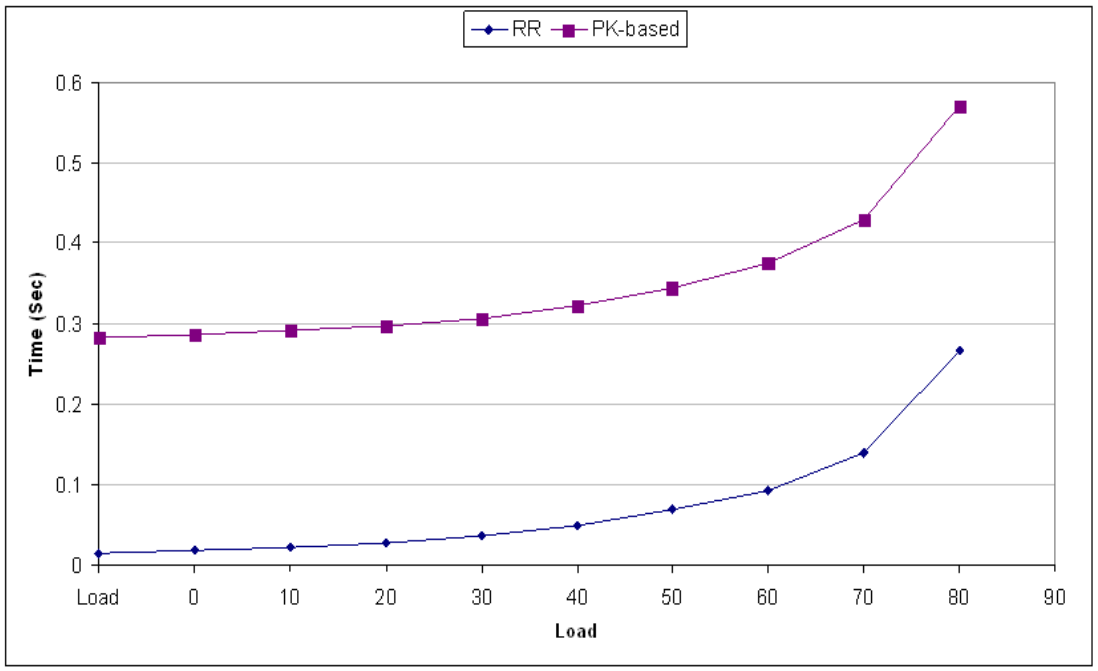

Figure 6.8: Average CR-Delay for RR and PK-based protocols vs. load (binding creation - mobile CN case)

Figures 6.9 to 6.12 show the CR-Delay in the binding update phase for the stationary and mobile CN cases. In the update phase, the SK-based and PK-based protocols are identical; thus, they will be referred to as the 'proposed protocols' in the following discussions. Figures 6.9 and 6.10 illustrate the CR-Delay of the RR protocol, the SSKv2 protocol, and the proposed protocols. As shown in these figures, at all loads, the proposed protocols produce a lower CR-Delay than the RR and SSKv2 protocols. This is because the proposed protocols require one message to update a binding; whereas, the RR and SSKv2 protocols require five messages and three messages, respectively, to update a binding.

Figures 6.11 and 6.12 illustrate the CR-Delay of the SSKv1 protocol and the proposed protocols. These figures show that the SSKv1 protocol slightly outperforms the proposed protocols. Specifically, on average, the SSKv1 protocol updates a binding about 6% and 7% faster than the proposed protocols in the stationary and mobile CN cases, respectively.

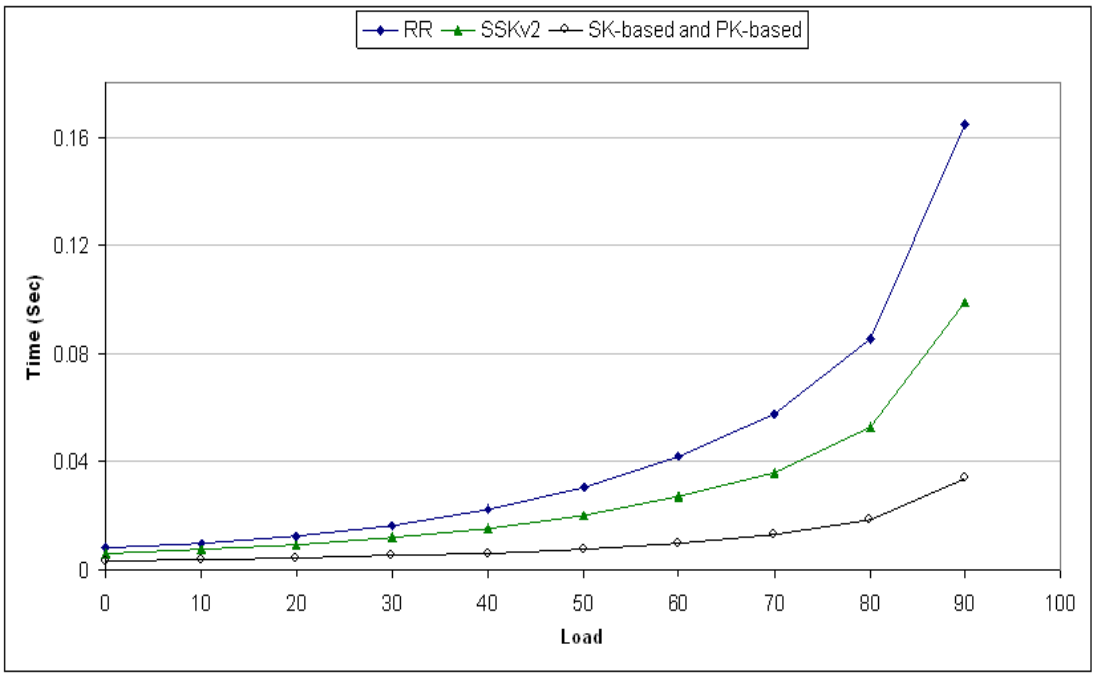

Figure 6.9: Average CR-Delay for RR, SSKv2, and SK-based and PK-based protocols vs. load (binding update - stationary CN case)

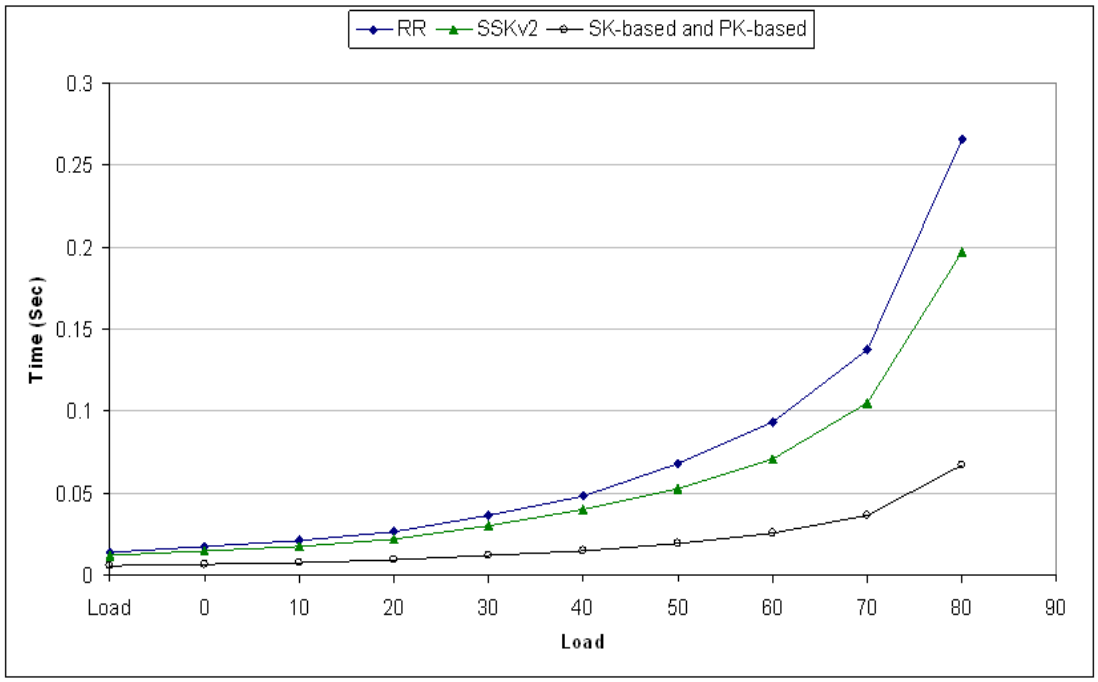

Figure 6.10: Average CR-Delay for RR, SSKv2, and SK-based and PK-based protocols vs. load (binding update - mobile CN case)

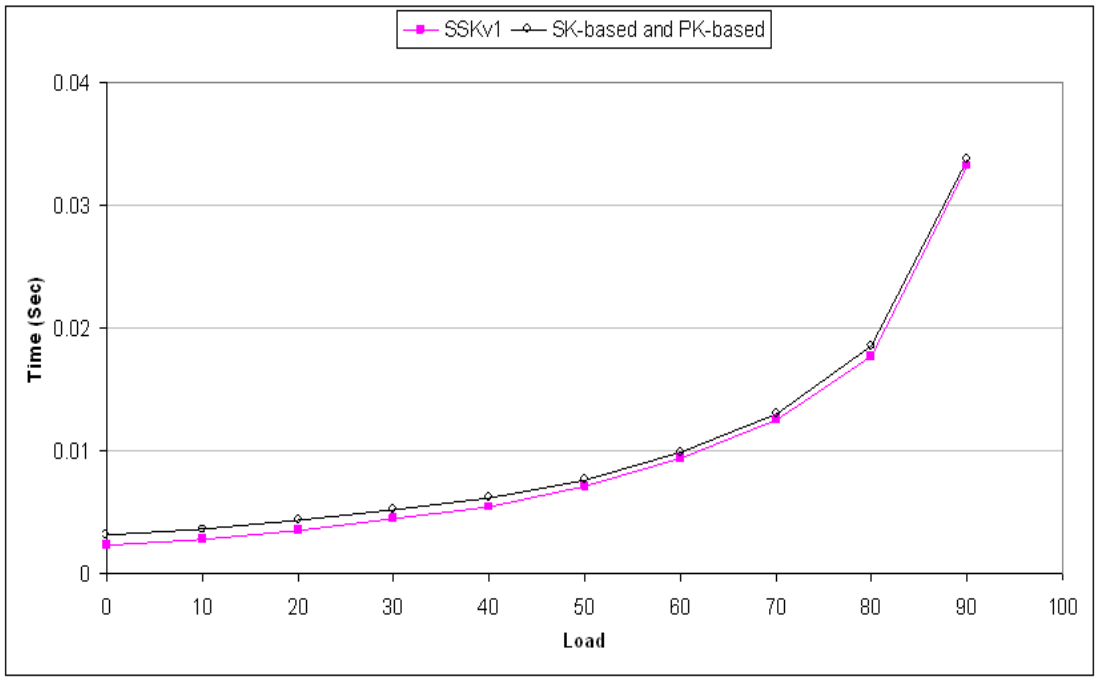

Figure 6.11: Average CR-Delay for SSKv1 protocol and SK-based and PK-based protocols vs. load (binding update - stationary CN case)

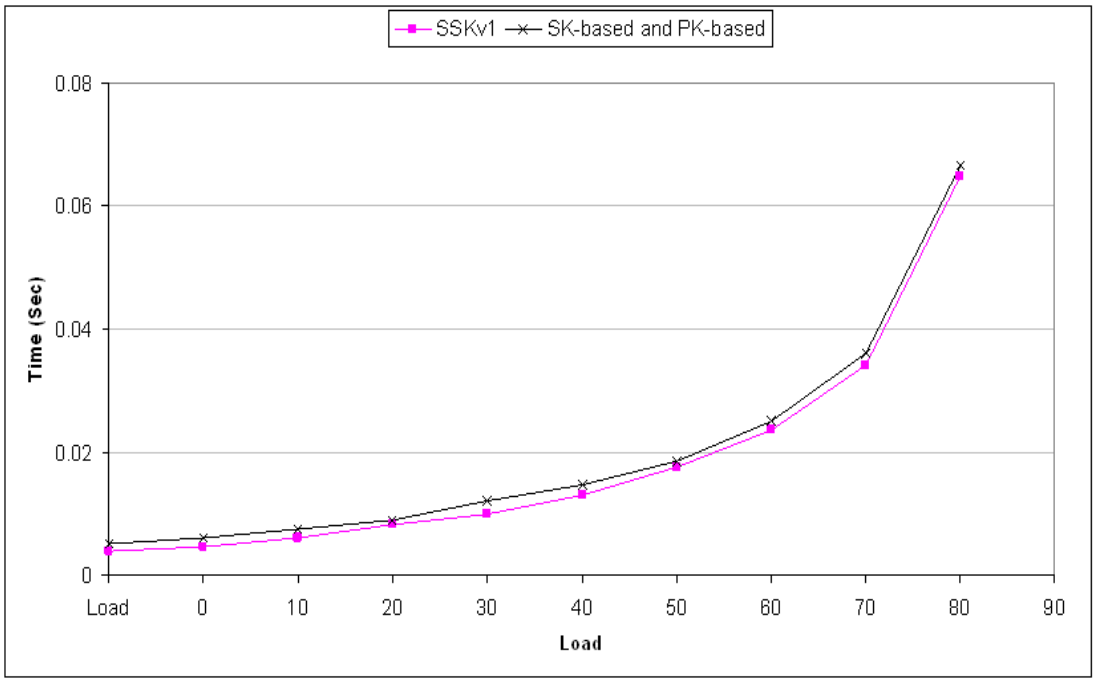

Figure 6.12: Average CR-Delay for SSKv1 protocol and SK-based and PK-based protocols vs. load (binding update - mobile CN case)

Figures 6.13 to 6.16 show the CR-Delay in the binding deletion phase for the stationary and mobile CN cases. Again, in the deletion phase, the SK-based and PK-based protocols are identical and referred to as the 'proposed protocols'. In addition, the SSKv1 and SSKv2 protocols are also identical and are thus referred to as the 'SSK protocols' in the following discussions. Figures 6.13 and 6.14 illustrate the CR-Delay of the RR protocol and the proposed protocols. As shown in these figures, at all loads, the proposed protocols produce a lower CR-Delay than the RR protocol. This is because the proposed protocols require one message to delete a binding; whereas, the RR protocol requires three messages to delete a binding.

Figures 6.15 and 6.16 illustrate the CR-Delay of the SSK protocols and the proposed protocols. These figures show that the SSK protocols slightly outperform the proposed protocols. Specifically, on average, the SSK protocols delete a binding about 4.9% and 5.2% faster than the proposed protocols in the stationary and mobile CN cases, respectively.

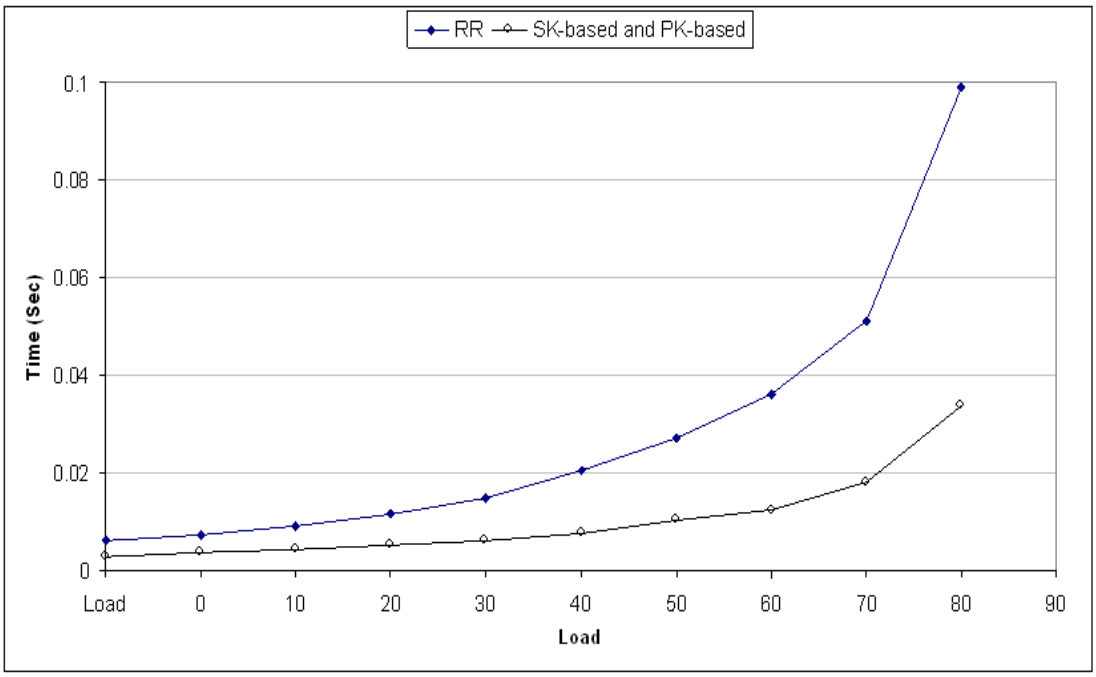

Figure 6.13: Average CR-Delay for RR protocol and SK-based and PK-based protocols vs. load (binding deletion - stationary CN case)

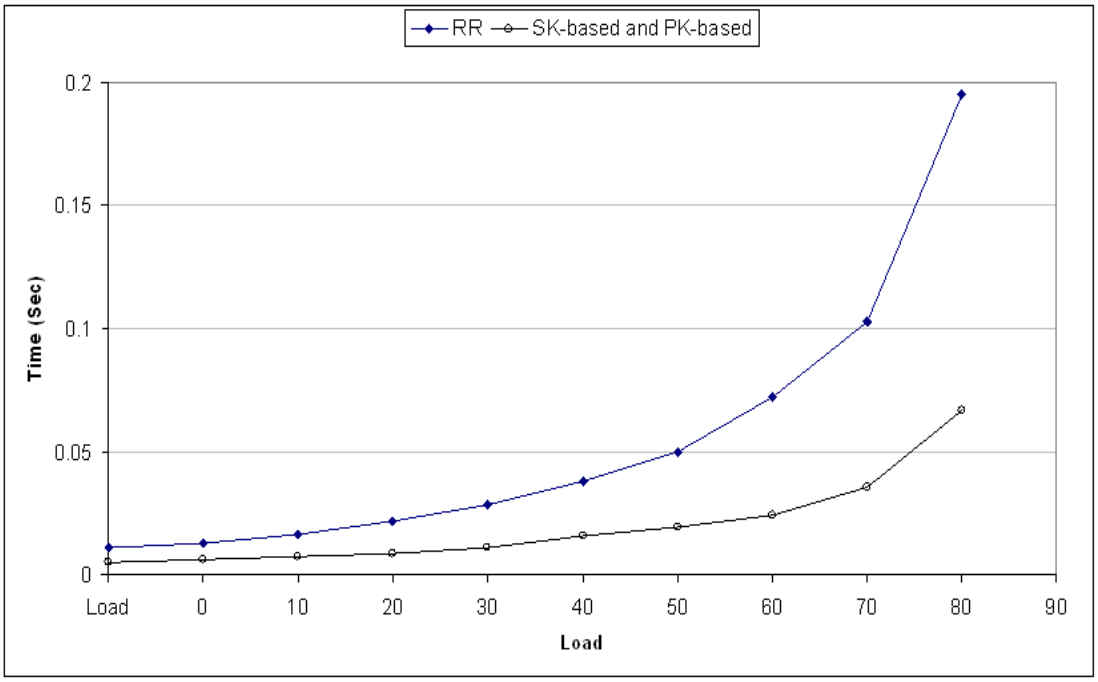

Figure 6.14: Average CR-Delay for RR protocol and SK-based and PK-based protocols vs. load (binding deletion - mobile CN case)

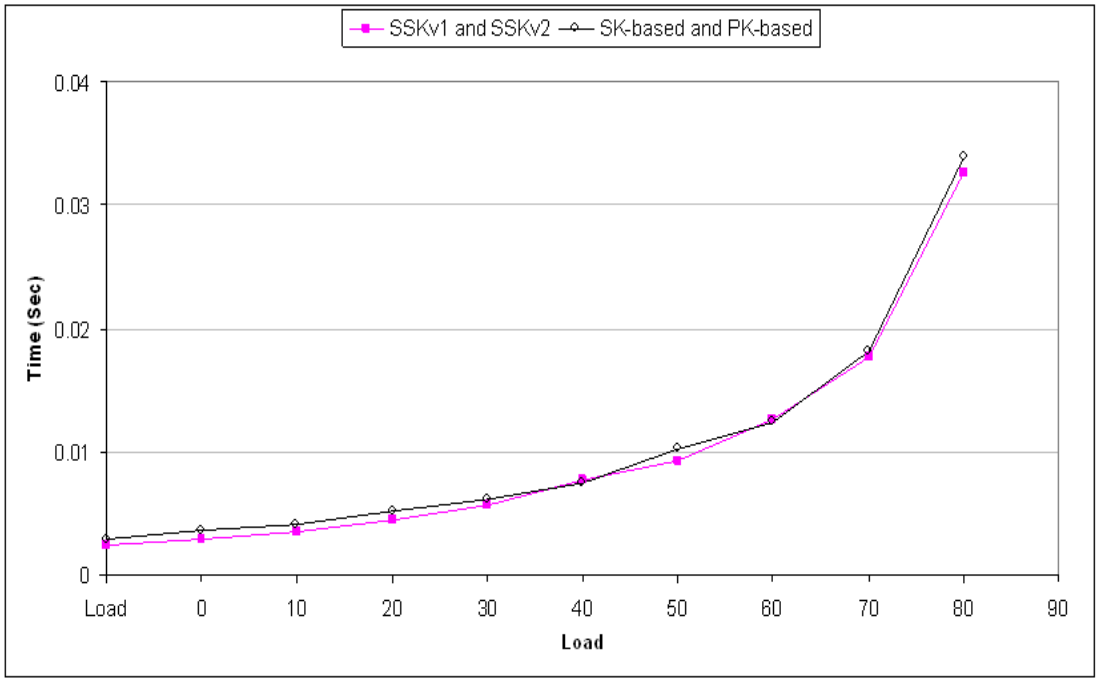

Figure 6.15: Average CR-Delay for SSKv1 and SSKv2 protocols and SK-based and PK-based protocols vs. load (binding deletion - stationary CN case)

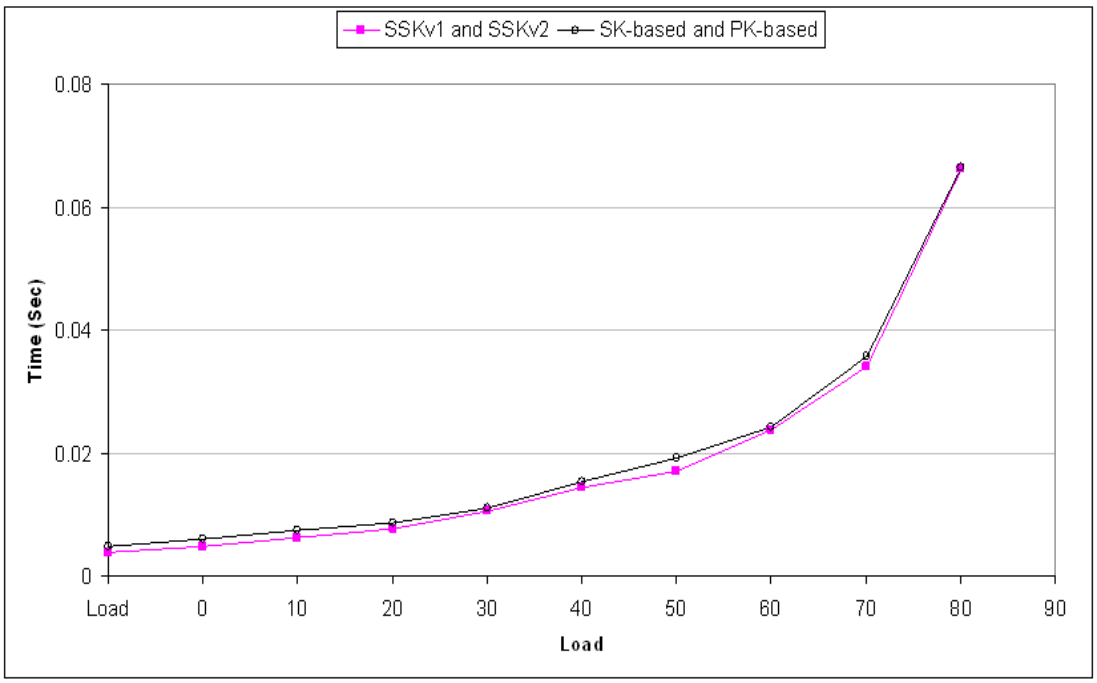

Figure 6.16: Average CR-Delay for SSKv1 and SSKv2 protocols and SK-based and PK-based protocols vs. load (binding deletion - mobile CN case)

6.3.3.2 Control Signalling Overhead

This section presents an analysis of the control signalling overhead simulation results in both the stationary and mobile CN cases for the RR, SSKv1, SSKv2, SK-based, and PK-based protocols. The control signalling overhead at an entity is the amount of Control Traffic Received (CTR) and Control Traffic Sent (CTS) by the entity. A selection of the simulation results are shown in Figures 6.17 to 6.20. These figures illustrate the control signalling overhead produced at all involved entities in the three binding phases (i.e. creation, update, and deletion) when all the simulated protocols are executed at the same rate.

The following observations can be made from Figures 6.17 to 6.20.

- While creating a new binding, the control signalling overheads in the proposed protocols, i.e. SK-based and PK-based protocols, at all involved entities are higher than in the other simulated protocols (shown in all the figures).
- While updating a binding, the control signalling overheads in the proposed protocols at the MN are lower than in the RR and SSKv2 protocols, but higher than in the SSKv1 protocol (Figure 6.17).
- While updating a binding, the control signalling overheads in the proposed protocols at the CN (and also at the CN's HA in the mobile CN case) are about 6%, 150%, and 19% higher than in the RR, SSKv1, and SSKv2 protocols, respectively (Figures 6.18 and 6.20).
- While updating a binding, the control signalling overheads in the proposed protocols at the MN's HA are about 14% lower than in the RR protocol (Figure 6.19).
- While deleting a binding, the control signalling overheads in the proposed protocols at all involved entities are lower than in the other simulated protocols (shown in all the figures).

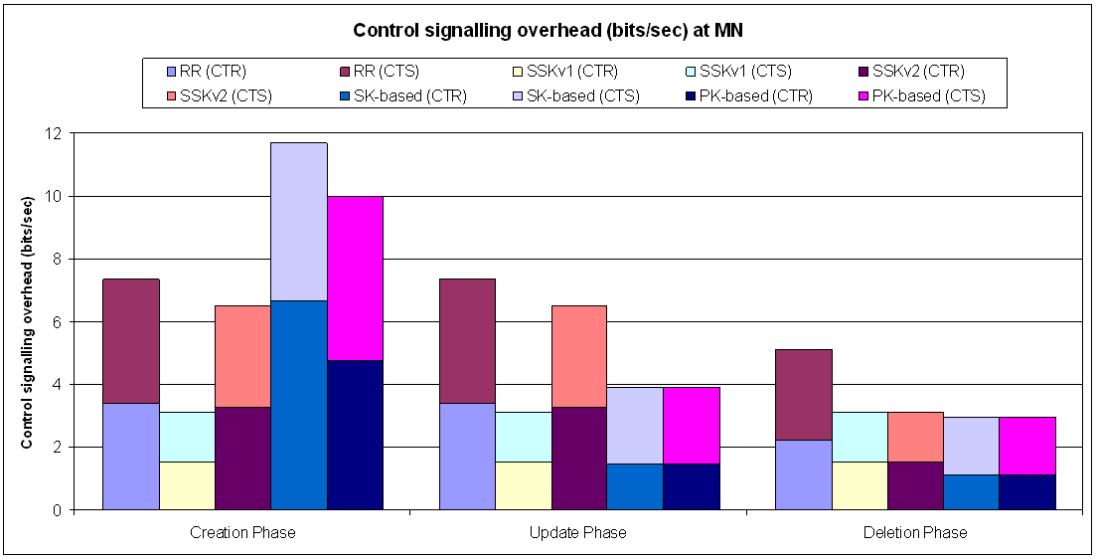

Figure 6.17: Control signalling overhead (bits/sec) at MN

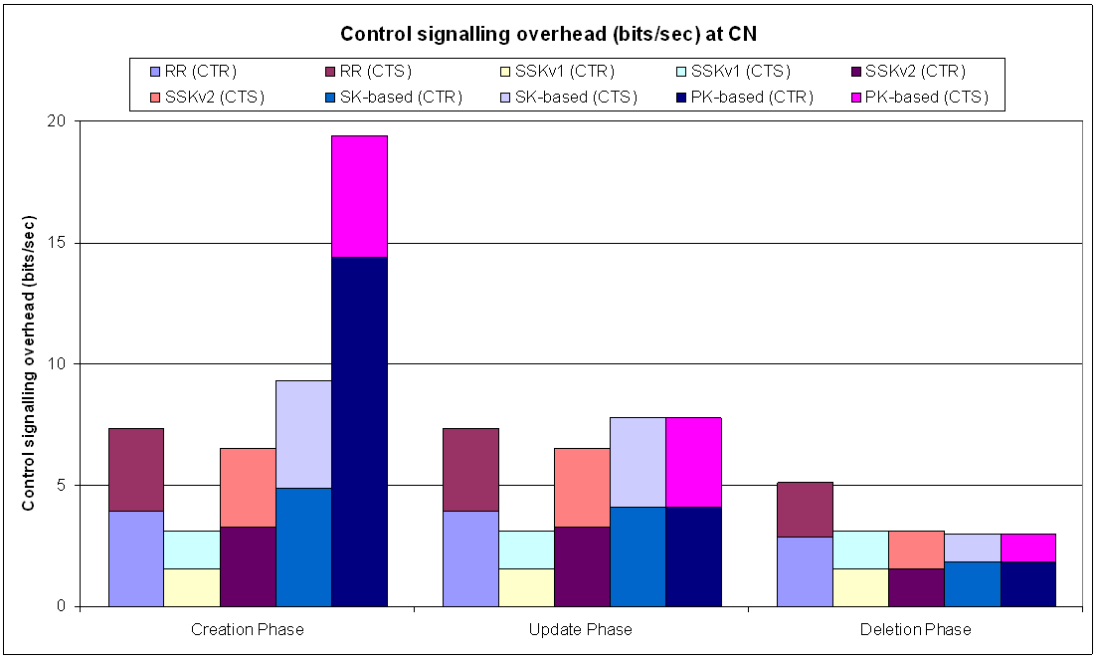

Figure 6.18: Control signalling overhead (bits/sec) at CN

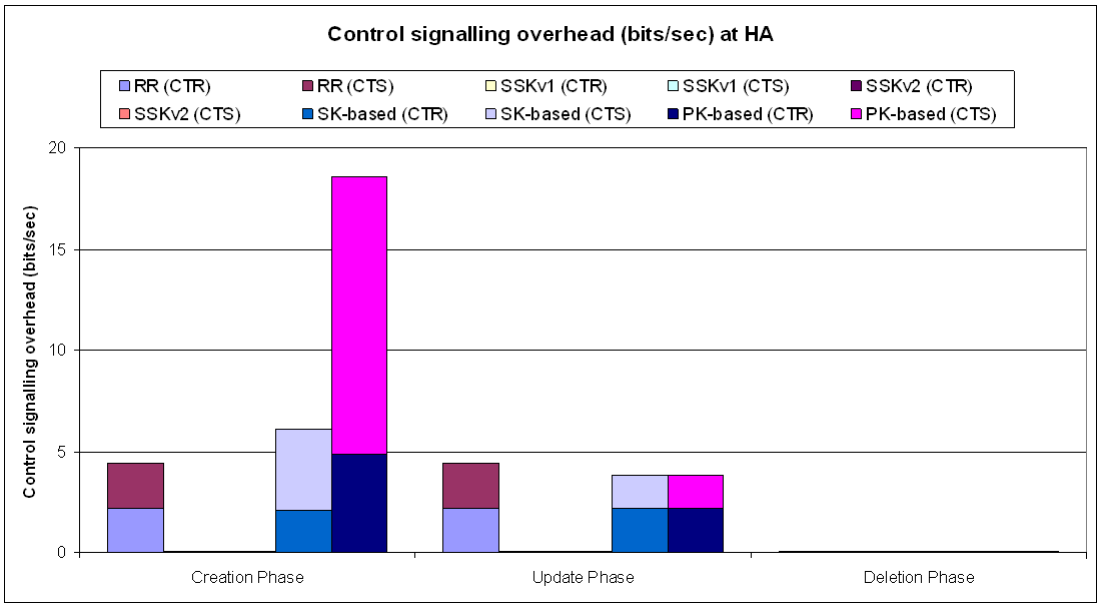

Figure 6.19: Control signalling overhead (bits/sec) at HA

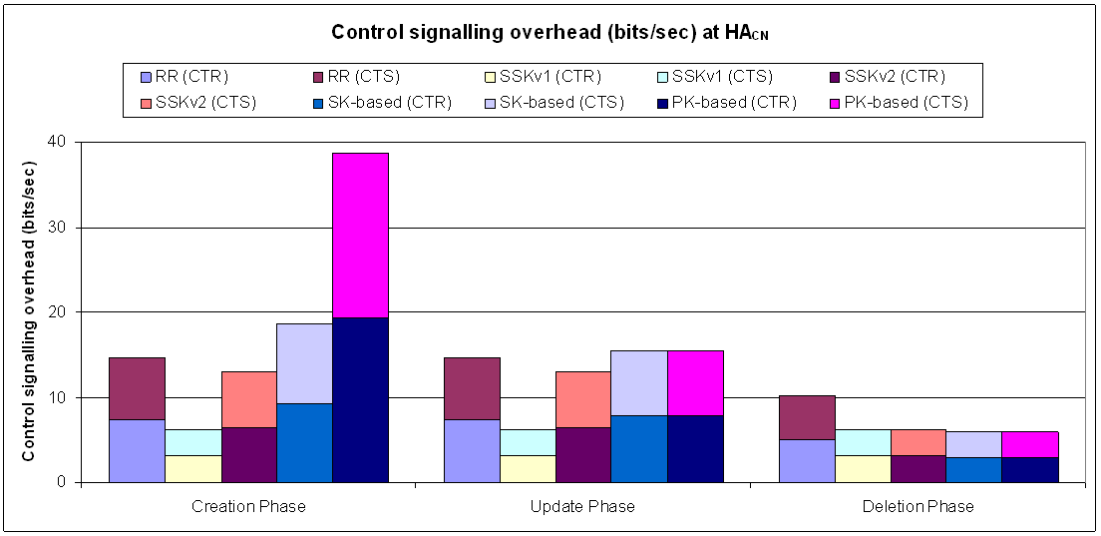

Figure 6.20: Control signalling overhead (bits/sec) at HA_{CN}

6.3.3.3 Discussions

From the simulation results presented, the following observations can be made:

- Firstly, in the creation phase, the SK-based and PK-based protocols produce a higher registration delay and control signalling overhead than all the other simulated protocols. However, this high delay is acceptable as the creation phase runs parallel to data transfer between the MN and the CN through the MN's home link.

- Secondly, in the update phase, the SK-based and PK-based protocols produce a lower registration delay than both the RR protocol and the SSKv2 protocol, but a slightly higher registration delay than the SSKv1 protocol. This is because the SK-based and PK-based protocols utilize early binding updates to minimise the registration delay. The simulation results also show that the SK-based and PK-based protocols produce a reduction in the signalling overhead compared to the RR protocol at the MN and the HA, but an increase in the signalling overhead at the CN.

- Thirdly, in the deletion phase, the SK-based and PK-based protocols produce a lower registration delay than the RR protocol, but a slightly higher registration delay than both the SSKv1 protocol and the SSKv2 protocol. The simulation results also show that the SK-based and PK-based protocols produce a reduction in the signalling overhead compared to all the other simulated protocols.

Overall, the author concludes that if a comparison between the proposed protocols and the most related work is performed on the basis of efficiency as well as security, the proposed protocols render themselves superior. Specifically, the proposed protocols offer significantly enhanced security for correspondent registrations. This is at the cost of a varying increase, depending on the protocol, in the delay and signalling overhead during the creation phase. In the update and deletion phases, the delay and signalling overhead is considerably less than at related work. It would therefore be commercially advantages to adopt the proposed protocols.

6.4 Chapter Summary

This chapter has presented the security and performance analyses of our correspondent registration protocols. The protocols are first informally analysed against the security requirements and known security attacks specified in Sections 5.1 and 2.2.1, respectively. This informal analysis has demonstrated that the INF-based protocol cannot authenticate the HoA and CoA due to the use of the unauthentic public key. It also has demonstrated that both the SK-based and PK-based protocols meet the specified security requirements and defend against the specified known attacks. The formal verification of the SK-based and PK-based protocols has further confirmed our intuitive analysis that the security requirements and known attacks are indeed satisfied and defended against, respectively, by our protocols.

Simulation models of the SK-based and PK-based protocols have been constructed using the OPNET Modeler. The model of the SK-based protocols has been validated using the OPNET Modeler. The analysis of simulation results has demonstrated that the SK-based and PK-based protocols, compared to the RR protocol, produce lower registration delay and a reduction in the signalling overhead during update and deletion phases. This is at the cost of a varying increase, depending on the protocol variant, in the registration delay and signalling overhead during the creation phase.

The next chapter concludes this thesis, and gives recommendations to further work.

Chapter 7

Conclusions and Future Work

The focus of this thesis was on improving the security and efficiency of home and correspondent registrations, i.e. the location management feature, of the MIPv6 protocol. This chapter summarises the work presented in this thesis, gives the conclusions drawn from the research findings, and recommends future work.

7.1 Thesis Summary

The work presented in this thesis can be arranged into four parts: research background, enhanced home registration protocol, family of correspondent registration protocols, and the evaluation of both protocols.

- **Research Background**

The thesis has introduced home and correspondent registrations in the MIPv6 protocol. Chapter 2 gave an overview of the MIPv6 protocol, its main components, and the two possible modes of communications allowed between an MN and a CN. This chapter also detailed the security attacks that could be launched via abuse of the location management feature of the protocol. In addition, it outlined the security services and performance requirements needed for securing home and correspondent registrations. The basic home registration protocol and the return routability procedure, which are currently used to protect home and correspondent registrations respectively, were also described. Chapter 3 surveyed other related works in protecting correspondent registrations.

• **The Enhanced Home Registration protocol**
Chapter 4 described our novel enhanced home registration (EHR) protocol. This chapter first presented an overview of the EHR protocol and detailed the design of its three novel building blocks: the symmetric CGA-based technique, the concurrent CoA reachability test, and the segmenting IPv6 address space method. The chapter next presented the description of the EHR protocol. It then informally analysed the security of the EHR protocol. Finally, Chapter 4 simulated and presented the performance comparison of the EHR protocol with the basic home registration protocol.

• **The Family of Correspondent Registration protocols**
Chapters 5 and 6 presented our novel correspondent registration protocols. Specifically, Chapter 5 first specified the design objectives for the protocols. It then gave an overview of the protocols and their three phases: the creation phase, the update phase, and the deletion phase. Finally, it presented detailed descriptions of the protocols. Chapter 6 gave the security analysis and performance evaluation of the protocols presented in Chapter 5. It first informally analysed the protocols against the requirements and the well-known attacks specified earlier. The formal verifications of the protocols using the Protocol Composition Logic (PCL) method and the Casper/FDR2 model checker were also given. Finally, the chapter simulated and presented the performance comparison of the protocols with the most related work.

• **Evaluation**
The following methods of evaluation have been used to evaluate our designs:

1. The performance (in terms of registration delay and signalling overhead) of the EHR protocol has been measured and compared with the basic home registration protocol.

2. The security of the proposed correspondent registration protocols has been informally analysed against the requirements and well-known attacks.

3. The security of the proposed correspondent registration protocols has been formally verified using the PCL method and the Casper/FRD2 model checker.

4. The performances (in terms of registration delay and signalling overhead) of the proposed correspondent registration protocols have been measured and compared with the most related work.

7.2 Contributions

The thesis has made the following contributions and discoveries:

- The symmetrical CGA-based technique is a novel lightweight CGA-based technique that makes use of a secret key shared between the participants in generating and verifying an IPv6 address. This technique reduces the computational and communication costs of the participants compared to traditional CGA-based techniques. Therefore, it is suitable for use by MNs to configure new CoAs and by HAs to verify the authenticity of those CoAs. In addition, it forces a malicious MN to attempt about (2^{61}) tries to falsely claim a third party's address as its CoA, thus reducing the likelihood of stealing third parties' addresses.

- The concurrent CoA reachability test is a novel reachability test that enables HAs to register and use MNs' new CoAs while concurrently verifying MNs' reachability at those CoAs. This test runs parallel to data transfer to and from the CoAs, thus having no effect on the registration delay.

- The segmenting IPv6 address space method is a novel method that uses two bits of the IPv6 64-bits interface identifier field to differentiate between redirectable and non-redirectable IPv6 addresses. This method reduces the number of targets that are vulnerable to DoS attacks launched via abuse of the location management feature of current, as well as future, mobility protocols; it prevents malicious MNs from impersonating either stationary nodes or other MNs located at their home links.

- The enhanced home registration (EHR) protocol extends the basic home registration protocol to support the location authentication of MNs to their HAs. The EHR protocol adds the novel symmetrical CGA-based technique, the novel concurrent CoA reachability test, and the novel segmenting IPv6 address space method mentioned above to the basic home registration protocol. Therefore, it reduces the likelihood of a malicious MN being successful in launching DoS attacks against third parties. This is done with a marginal impact on registration delay, but a significant increase in the signalling overhead.

- The SK-based, PK-based, and INF-based protocols are a family of novel correspondent registration protocols that rely on the assistance of MNs' home links

in authenticating MNs' HoAs and CoAs to CNs. These protocols offload the operational load from an MN to its home link and are designed in creation, update, and deletion phases to reduce iterating courses of the entire protocol. The creation phase runs parallel to data transfer between the MN and the CN through the MN's home link; thus, most of the authentication and keys establishment take place in it. In addition, the established keys are used to secure the update and deletion phases. Therefore, the SK-based and PK-based protocols improve the security strength and registration delay; they satisfy all the specified security requirements and reduce registration delay in the update and deletion phases. This is at the cost of an increase in the registration delay and signalling overhead during the creation phase.

- To demonstrate the efficiency and efficacy of the EHR, SK-based, and PK-based protocols, the performance of the protocols has been evaluated and compared to that of related work using the OPNET simulation package. In addition, the security of the protocols has been informally analysed against the security requirements and well-known attacks. Furthermore, the security of the SK-based and PK-based protocols has been verified using formal methods.

7.3 Future Work

The author has the following recommendations for future work:

- All of the proposed protocols, i.e. both the EHR protocol and the correspondent registration protocols, have allowed an HA and a CN to initially set the granted binding lifetime between an MN's HoA and CoA to a MIN_BINDING_LIFETIME value. This lifetime value is to limit the amount of data that the HA/CN can send to the MN's CoA while gaining confidence that the MN is indeed connected to that CoA. The optimal value for this lifetime value is an area for further research.

- The EHR protocol has prevented an MN from bypassing the concurrent CoA reachability test by limiting the number of BUs that the MN can send from an unreachable CoA to a MAX_ATTEMPT value. The optimal value for this number of attempts is a direction for further research.

- The proposed correspondent registration protocols have increased the maximum permitted binding lifetime to reduce the number of redundant binding

refreshes, thus reducing signalling overheads. The optimal value for this permitted lifetime is a point of further research.

- The symmetrical CGA-based technique has only used the compression algorithm included in the CryptoSys toolkit to try to overcome the hash-length shortage problem, i.e. at most, 62 bits of the address can be used for the hash. However, this compression algorithm cannot compress a hash value. The author needs to try further compression algorithms to investigate whether the idea of lossless compression proposed by Hwang in [58] is suited to overcome the hash-length shortage problem.

- The segmenting IPv6 address space method has required changing the way every IPv6 node chooses an IPv6 address. The resulting changes in network infrastructure and endpoint devices are points of further research

- The PK-based protocol has only been compared to the RR protocol. The protocol can be compared to other public key based protocols.

- The thesis has simulated all of the proposed protocols in an IEEE 802.11 wireless environment. The protocols can be simulated in other wireless technologies, e.g. UMTS, to evaluate the performance costs.

In conclusion, the aims of this research, to determine the security and performance needs of home and correspondent registrations, to investigate existing solutions in the context, and to design an enhanced home registration protocol as well as a family of correspondent registration security protocols with high levels of security and efficiency, have been achieved but there remain other possible avenues of future work.

Bibliography

[1] T. Aura, "Mobile IPv6 Security," in *Proc. Security Protocols, 10th International Workshop, LNCS*, vol. 2845, pp. 215–234, Springer, 2002.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11 – 33, 2004.

[3] C. Onwubiko and A. Lenaghan, "Spatio-Temporal Relationships in the Analysis of Threats for Security Monitoring Systems," *2nd International Conference on Computer Science and Information Systems*, pp. 455–471, June 2006.

[4] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, "CSI/FBI Computer Crime and Security Survey 2006," *11th Annual CSI/FBI Computer Crime and Security Survey*, 2006.

[5] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6." RFC 3775 (Proposed Standard), June 2004.

[6] R. Koodli, "Fast Handovers for Mobile IPv6." RFC 4068 (Experimental), July 2005.

[7] C. Perkins, "IP Mobility Support for IPv4." RFC 3344 (Proposed Standard), Aug. 2002. Updated by RFC 4721.

[8] H. Soliman, C. Castelluccia, K. ElMalki, and L. Bellier, "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management." RFC 5380 (Proposed Standard), Oct. 2008.

[9] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark, "Mobile IP Version 6 Route Optimization Security Design Background." RFC 4225 (Informational), Dec. 2005.

[10] S. Kent, "IP Encapsulating Security Payload (ESP)." RFC 4303, 2005.

[11] J. Arkko, V. Devarapalli, and F. Dupont, "Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents." RFC 3776 (Proposed Standard), June 2004. Updated by RFC 4877.

[12] "The Network Simulator NS-2." `http://www.isi.edu/nsnam/ns/`.

[13] O. Technologies, "OPNET University Program." `http://www.opnet.com/university_program/`.

[14] N. Durgin, J. Mitchell, and D. Pavlovic, "A Compositional Logic for Protocol Correctness," *Computer Security Foundations Workshop, IEEE*, pp. 241–255, 2001.

[15] A. Datta, A. Derek, J. C. Mitchell, and A. Roy, "Protocol Composition Logic (PCL)," *Electronic Notes in Theoretical Computer Science*, vol. 172, pp. 311–358, 2007.

[16] G. Lowe, "Casper: A compiler for the analysis of security protocols," *Journal of Computer Security*, vol. 6, no. 1-2, pp. 53–84, 1998.

[17] F. S. E. LTD. Failure-Divergences Refinement FDR2 Manual, 1997.

[18] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)." RFC 4861 (Draft Standard), Sept. 2007.

[19] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494.

[20] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration." RFC 4862 (Draft Standard), Sept. 2007.

[21] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification." RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871.

[22] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification." RFC 2473 (Proposed Standard), Dec. 1998.

[23] K. Ren, W. Lou, K. Zeng, F. Bao, J. Zhou, and R. H. Deng, "Routing optimization security in mobile IPv6," *Comput. Netw.*, vol. 50, no. 13, pp. 2401–2419, 2006.

[24] S. Song, H.-K. Choi, and J.-Y. Kim, "A Secure and Lightweight Approach for Routing Optimization in Mobile IPv6," *EURASIP J. Wirel. Commun. Netw.*, vol. 2009, pp. 1–10, 2009.

[25] S. Kent and K. Seo, "Security Architecture for the Internet Protocol." RFC 4301 (Proposed Standard), Dec. 2005.

[26] N. Doraswamy and D. Harkins, *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks*. Prentice Hall, 2003.

[27] S. Kent, "IP Authentication Header," 2005.

[28] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," 1998.

[29] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," 2005.

[30] B. Lim, C. Ng, K. Aso, and S. Krishnan, "Verification of Care-of Addresses in Multiple Bindings Registration," Expired Internet-Draft: draft-lim-mext-multiple-coa-verify-02.txt, July 2008.

[31] R. Wakikawa, V. Devarapalli, G. Tsirtsis, T. Ernst, and K. Nagami, "Multiple Care-of Addresses Registration." RFC 5648 (Proposed Standard), Oct. 2009.

[32] G. O'Shea and M. Roe, "Child-proof Authentication for MIPv6 (CAM)," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 4–8, 2001.

[33] T. Aura, "Cryptographically Generated Addresses (CGA)." RFC 3972 (Proposed Standard), 2005.

[34] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," in *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 2002.

[35] G. Montenegro and C. Castelluccia, "Crypto-Based Identifiers (CBIDs): Concepts and Applications," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 97–127, 2004.

[36] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture." RFC 4291 (Draft Standard), 2006.

[37] P. C. van Oorschot and M. J. Wiener, "Parallel Collision Search with Applications to Hash Functions and Discrete Logarithms," in *2nd Annual ACM Conference on Computer and Communications Security, Fairfax, Virginia*, pp. 210–218, ACM Press, 1994.

[38] Z. Cao, H. Deng, Y. Ma, and P. Hu, "Integrating Identity Based Cryptography with Cryptographically Generated Addresses in Mobile IPv6," in *ICCSA'07: Proceedings of the International Conference on Computational Science and Its Applications*, pp. 514–525, Springer-Verlag, 2007.

[39] C. Vogt, R. Bless, M. Doll, and T. Kuefner, "Early Binding Updates for Mobile IPv6," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, pp. 1440–1445, March 2005.

[40] C. Vogt, "Credit-Based Authorization for Concurrent IP-Address Tests," Tech. Rep. TM-2005-3, Institute of Telematics, University of Karlsruhe, Germany, June 2005.

[41] S. Bradner, A. Mankin, and J. Schiller, "A Framework for Purpose Built Keys (PBK)," Expired Internet-Draft: draft-bradner-pbk-frame-06.txt, June 2003.

[42] S. M. Faccin and F. Le, "Dynamic Diffie Hellman based Key Distribution for Mobile IPv6," Expired Internet-Draft: draft-le-mobileip-dh-01.txt, December 2001.

[43] W. Haddad, F. Dupont, L. Madour, S. Krishnan, and S. Park, "Optimizing MIPv6 (OMIPv6)," Expired Internet-Draft: draft-haddad-mipv6-omipv6-01.txt, February 2004.

[44] W. Haddad, L. Madour, J. Arkko, and F. Dupont, "Applying Cryptographically Generated Addresses to Optimize MIPv6 (CGA-OMIPv6)," Expired Internet-Draft: draft-haddad-mip6-cga-omipv6-04.txt, May 2005.

[45] J. Arkko, C. Vogt, and W. Haddad, "Enhanced Route Optimization for Mobile IPv6." RFC 4866 (Proposed Standard), May 2007.

[46] C. Perkins, "Securing Mobile IPv6 Route Optimization Using a Static Shared Key." RFC 4449 (Proposed Standard), June 2006.

[47] F. Dupont and J.-M. Combes, "Care-of Address Test for MIPv6 using a State Cookie," Expired Internet-Draft: draft-dupont-mipv6-rrcookie-06.txt, June 2008.

[48] H.-S. Yoon, R.-H. Kim, S.-B. Hong, and H.-Y. Youm, "PAK-based Binding Update Method for Mobile IPv6 route optimization," in *ICHIT '06: Proceedings of the 2006 International Conference on Hybrid Information Technology*, (Washington, DC, USA), pp. 617–623, IEEE Computer Society, 2006.

[49] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using diffie-hellman," in *EUROCRYPT'00: Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, (Berlin, Heidelberg), pp. 156–171, Springer-Verlag, 2000.

[50] J. D. Koo, J. Koo, and D. C. Lee, "A New Authentication Scheme of Binding Update Protocol on Handover in Mobile IPv6 Networks," in *Emerging Directions in Embedded and Ubiquitous Computing*, vol. 4097, pp. 972–978, Springer Berlin / Heidelberg, 2006.

[51] F. Bao, R. Deng, Y. QIU, and J. Zhou, "Certificate-based Binding Update Protocol (CBU)," Expired Internet-Draft: draft-qiu-mip6-certificated-binding-update-03.txt, March 2005.

[52] R. H. Deng, J. Zhou, and F. Bao, "Defending Against Redirect Attacks in Mobile IP," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 59–67, ACM, 2002.

[53] J.-D. Koo and D.-C. Lee, "Extended Ticket-Based Binding Update (ETBU) Protocol for Mobile IPv6 (MIPv6) Networks," *IEICE transactions on communications*, vol. 90-B, no. 4, pp. 777–787, 2007.

[54] H.-C. Chao and C.-Y. Huang, "Micro-Mobility Mechanism for Smooth Handoffs in an Integrated Ad-Hoc and Cellular IPv6 Network Under High-Speed

Movement," *IEEE Transactions Vehicular Technology*, vol. 52, pp. 1576 – 1593, November 2003.

[55] J. Arkko, T. Aura, J. Kempf, V.-M. Mäntylä, P. Nikander, and M. Roe, "Securing IPv6 Neighbor and Router Discovery," in *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, (New York, NY, USA), pp. 77–86, ACM, 2002.

[56] R. H. Deng, J. Zhou, and F. Bao, "Defending Against Redirect Attacks in Mobile IP," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 59–67, ACM, 2002.

[57] P. Nikander, "Denial of Service, Address Ownership, and Early Authentication in the IPv6 World (Transcript of Discussion)," in *Revised Papers from the 9th International Workshop on Security Protocols*, (London, UK), pp. 22–26, Springer-Verlag, 2002.

[58] M.-S. Hwang, C.-C. Lee, and S.-K. Chong, "An improved address ownership in mobile IPv6," *Computer Communications*, vol. 31, no. 14, pp. 3250–3252, 2008.

[59] S. Gunderson, "Global IPv6 Statistics - Measuring the current state of IPv6 for ordinary users." A study by Google, reported in November 2008, `http://www.ripe.net/`.

[60] K. Perset, "Internet Addressing: Measuring Deployment of IPv6," OECD Digital Economy Papers 172, OECD, Directorate for Science, Technology and Industry, 2010.

[61] P. Nikander, "An Address Ownership Problem in IPv6," Expired Internet-Draft: draft-nikander-ipng-address-ownership-00.txt, February 2001.

[62] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 power-saving mode for mobile computing in wi-fi hotspots: limitations, enhancements and open issues," *Wireless Netw.*, vol. 14, no. 6, pp. 745–768, 2008.

[63] Z. Li, L. Li, and Y. Huang, "Research on Handover in Hierarchical Mobile IPv6 Based on the Fast DAD Mechanism in Visual Domain," in *International Symposium on Computer Science and Computational Technology (ISCSCT '08)*, vol. 2, pp. 277 –280, December 2008.

[64] N. I. of Standards and Technology, "Advanced Encryption Standard (AES)," 2001. Federal Information Processing Standard Publication 197.

[65] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[66] N. I. of Standards and Technology, *Secure Hash Standard*. 2002. Federal Information Processing Standard Publication 180-2.

[67] C. A. Meadows, "Formal Verification of Cryptographic Protocols: A Survey," in *Advances in Cryptology ASIACRYPT'94*, vol. 917 of *Lecture Notes in Computer Science*, pp. 133–150, Springer Berlin / Heidelberg, 1995.

[68] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[69] G. Lowe, "An Attack on Needham-Schroeder Public-Key Authentication Protocol," *Information Processing Letters*, vol. 56, no. 3, pp. 131–133, 1995. Elsevier North-Holland, Inc., Amsterdam, The Netherlands.

[70] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR," in *Proceedings of the International Workshop on Tools and Algorithms for the Construction and Analysis of Systems - TACAS '96*, vol. 1055, pp. 147–166, LNCS, Springer-Verlag, London, UK, 1996.

[71] A. Datta, A. Derek, J. Mitchell, and D. Pavlovic, "A Derivation System for Security Protocols and its Logical Formalization," pp. 109 – 125, jun. 2003.

[72] A. Datta, A. Derek, J. Mitchell, and D. Pavlovic, "A Derivation System and Compositional Logic for Security Protocols," 2005.

[73] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A Modular Correctness Proof of IEEE 802.11i and TLS," in *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 2–15, ACM, 2005.

[74] A. Datta, A. Derek, J. Mitchell, and B. Warinschi, "Computationally sound compositional logic for key exchange protocols," pp. 321 – 334, 2006.

[75] A. W. Roscoe and P. J. Broadfoot, "Internalising Agents in CSP Protocol Models," in *Proceedings of WITS 2002*, 2002.

[76] B. Padmanabhan, "Modeling and Analysis of Security Protocols using CASPER," Master's thesis, Master's thesis, Syracuse University, August 2003.

[77] I.-G. Kim, H.-S. Kim, J.-Y. Lee, and J.-Y. Choi, "Analysis and Modification of ASK Mobile Security Protocol," pp. 79 –83, jul. 2005.

[78] H.-S. Kim, J.-H. Oh, J.-Y. Choi, and J.-W. Kim, "The Vulnerabilities Analysis and Design of the Security Protocol for RFID System," pp. 152 –157, sep. 2006.

[79] C. A. R. Hoare, *Communicating Sequential Processes*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1985.

[80] A. W. Dent and C. J. Mitchell, *User's Guide To Cryptography And Standards*. Norwood, MA, USA: Artech House, Inc., 2004.

[81] Gollmann, Dieter, *Computer Security*. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[82] Stallings, William, *Cryptography and Network Security: Principles and Practice*. Pearson Education, 2002.

[83] N. Ferguson and B. Schneier, *Practical Cryptography*. Wiley Publishing, Inc., 2003.

[84] D. Eastlake, J. Schiller, and S. Crocker, "Randomness Requirements for Security." RFC 4086 (Best Current Practice), June 2005.

[85] R. Rivest, "The MD5 Message-Digest Algorithm." RFC 1321 (Informational), Apr. 1992.

[86] N. I. of Standards and Technology, *Secure Hash Standard*. 1995. Federal Information Processing Standard Publication 180-1.

[87] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[88] N. I. of Standards and Technology, *Data Encryption Standard (DES)*. 1999. Federal Information Processing Standard Publication 46-3.

[89] X. Lai, J. L. Massey, and S. Murphy, "Markov Ciphers and Differential Cryptanalysis," in *Advances in Cryptology – EUROCRYPT'91*, pp. 17–38, Springer-Verlag, 1991.

[90] "Information Security: Advances and Remaining Challenges to Adoption of Public Key Infrastructure Technology." United States General Accounting Office report GAO-01-277, February 2001.

[91] D. O'Mahony, M. Peirce, and H. Tewar, *Electronic Payment Systems*. Artech House, Boston, 1997.

[92] A. O. Freier, P. Karlton, and P. C. Kocher, *SSL - Secure Socket Layer 3.0 Specification*. Netscape Communications Corporation - Transport Layer Security Working Group, November 1996.

[93] M. Ergen, "OPNET Tutorial." `http://wow.eecs.berkeley.edu/ergen/docs/OPNET.pdf`.

[94] CryptoSys™, *CryptoSys API and CryptoSys PKI Toolkits*. Available at http://www.cryptosys.net/.

[95] N. I. of Standards and Technology, *Data Encryption Standard*. 1977. Federal Information Processing Standard Publication 46.

[96] B. Schneier, "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)," in *Fast Software Encryption, Cambridge Security Workshop*, (London, UK), pp. 191–204, Springer-Verlag, 1994.

[97] R. L. Rivest, *The RC4 encryption algorithm*. RSA Data Security, Inc., Mar. 12, 1992.

Appendix A

Cryptographic Building Blocks

This appendix presents an overview of the basic cryptographic techniques. These techniques include random numbers, one-way hash functions, secret-key and public-key cryptosystems, Public Key Infrastructure (PKI), and Diffie-Hellman (DH) key exchange. It makes use of the following cryptographic jargon. Plaintext or cleartext denotes a message in its original form. Message encryption is a process of transforming a plaintext message into a coded message, and the resulting message is known as ciphertext. Message decryption is a reverse process of transforming a ciphertext into its original (plaintext) form. Encryption and decryption are performed through the use of a cryptographic algorithm, called a cipher, and a cryptographic key.

A.1 Basic Security Services

Security is one of the most important requirements for the exchange of users' private data over an insecure public network (e.g. Internet). A lack of security could result in an intruder being able to impersonate a user, eavesdrop on data, or change data. A security service is a processing or communication service that gives a specific kind of protection to private data. Basic security services include authentication, authorization, confidentiality, integrity, anti-replay, and non-repudiation.

- **Authentication**

Authentication is a process of verifying the identity of an entity and making sure that data originates from that entity. There are two types of authentication:

unilateral authentication and mutual authentication. The function of unilateral authentication is to assure the recipient that the data is from the source that it claims to be. Mutual authentication enables the communicating entities to verify the identity of each other. Authentication is important in preventing impersonation attacks and hence ensuring authenticity of the data. Authentication is achieved through several cryptographic techniques, such as secret-key encryptions, message authentication codes, or digital signatures [80].

- **Authorization**

Authorization is the process of specifying whether a user has the authority to conduct a particular action. Authorization is related to the existence of a security policy, which is a set of rules that specify what types of activities, resources, or services an authenticated user is permitted to perform [81].

- **Confidentiality**

Confidentiality is the prevention of the unauthorised disclosure of data, i.e. data cannot be viewed by an unauthorised entity. Confidentiality is provided by encryption, and it is important in protecting against eavesdropping attacks [81].

- **Integrity**

The integrity service is the detection of any unauthorised alteration of transmitted data, i.e. data is not alterable without detection. Alteration types include insertion, deletion, or change of transmitted data. When data authenticity and integrity are preserved, the data is called reliable. Entities can be relatively certain that they are viewing data created by a legitimate source and that no unauthorised alterations have been made. Integrity is achieved through several cryptographic techniques, such as checksums, message authentication codes, or digital signatures [81, 82].

- **Anti-Replay**

Anti-replay service is the detection of any replay of the transmitted data, i.e. the recipient can detect and reject old or duplicated data, which protects against replay attacks. Anti-replay is provided through several ways, such as using a one-time token, a random number, a sequence number, or a timestamp.

- **Non-Repudiation**

Non-repudiation is the prevention of a sender or receiver from denying transmitted data. Non-repudiation provides protection against false denials that particular data has not been sent or received by one of the participants involved in a communication. There are two types of non-repudiation: non-repudiation of origin and non-repudiation of receipt [82]. The former provides the recipient of the data with evidence of the origin. This evidence will protect against any subsequent attempt by the sender to falsely deny sending the data. The latter proves to the sender that the data was received by the specified recipient.

To provide the basic security requirements discussed above, the following sections present the essential cryptographic techniques that have been used in the design of the proposed home and correspondent registration protocols.

A.2 Random Numbers

A random number can be informally defined as a number that is "unpredictable to an attacker" [83]. Computers make use of pseudorandom number generators (PRNGs) to generate random numbers for cryptographic applications. A PRNG is a deterministic algorithm for generating a sequence of numbers that approximates the properties of random numbers.

Random numbers are used in network security protocols to provide freshness for protocol messages and to prevent replay attacks [82]. In addition, they are used for a number of computer security purposes, such as generating public-key pairs and producing secret keys [84].

A.3 One-Way Hash Functions

A hash function, also called a message digest function, is a one-way function that is relatively easy to compute but significantly hard to reverse. A hash function takes a variable-length message M as input and produces a fixed-length output, referred to as a hash value H(M). The hash value H(M) is a function of all the bits of the message M. Therefore, any change to M will result in a change to H(M). Two widely used hash functions are Message-Digest Algorithms, such as MD5 [85], and Secure Hash Algorithms, such as SHA-1 [86] and SHA-512 [66].

Hash functions are used in network security protocols to provide data integrity and data authenticity services. These services guarantee the message receiver that the message has not been tampered with in transit and that the message was originated from the claimed sender. One way of achieving this is to combine the use of a hash function with a secret key, which is known only to the sender and the receiver, to compute a message authentication code. Hash functions are also used in digital-signature generation. When digitally signing a message, instead of applying cryptographic processing to the whole message, it is applied to the message's hash value. This is much more efficient as hash values have a fixed and relatively small length.

A.4 Secret-Key (Symmetric) Cryptosystems

In a secret-key cryptosystem [82], also referred to as symmetric, the sender and the receiver use the same secret key. Secret-key cryptosystems include secret-key encryptions and message authentication codes.

A.4.1 Secret-Key Encryptions

In a secret-key encryption (also known as symmetric cipher), the sender uses a secret key to encrypt a message into ciphertext and the receiver uses the same secret key to decrypt the ciphertext into the original message. This process is illustrated in Figure A.1.

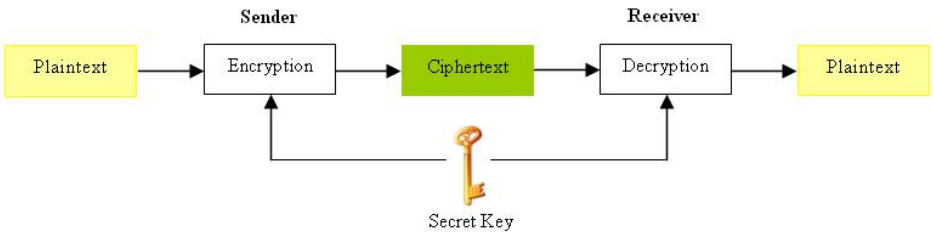

Figure A.1: Secret-key encryption/decryption scheme

Secret-key encryptions can be classified into stream and block ciphers [82], according to the way in which the plaintext is processed. A stream cipher processes the plaintext in bit-wise or byte-wise manner. A block cipher breaks up the plaintext into groups of bits (called blocks) of a fixed length and processes

one block at a time [87]. The latter is the more secure and more commonly used. Most widely used secret-key block ciphers are the Triple Data Encryption Standard (3DES [88]), the International Data Encryption Standard (IDEA [89]), and the Advanced Encryption Standard (AES [64]).

Secret-key encryptions are widely used in network security protocols to provide message confidentiality and to assure message authenticity. This is due to the fact that the only entities capable of enciphering and deciphering the message are those that know the key. Therefore, the sender keeps the message unreadable by anyone other than the intended receiver, and the receiver is assured that the message has originated from that sender.

A.4.2 Message Authentication Codes

The second type of secret-key cryptosystems is the Message Authentication Code (MAC), also referred to as the keyed one-way hash function. The MAC combines the use of a hash function with a shared secret key and can be used to provide integrity and authenticity of a message. A sender applies a hash function over a message M and a secret key K to produce a MAC value H(K, M). The sender then sends both M and H(K, M) to the receiver. Once these are received, the receiver can use the same secret key to re-compute a MAC value and compare it with the received MAC value. This operation is illustrated in Figure A.2.

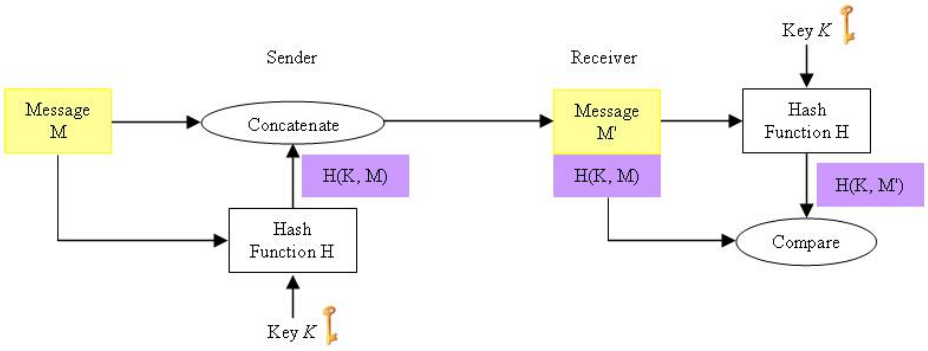

Figure A.2: The use of a MAC for message integrity and authenticity checking

Secret-key cryptosystems, i.e. secret-key encryptions and message authentication codes, are efficient and commonly used for message confidentiality, message integrity, and message authenticity. However, they need an effective and secure key management system for generating and distributing secret keys. They also

require a large number of keys; for a secret-key cryptosystem with n users, since each user has to possess n-1 keys, the required total number of keys can grow to n*(n-1)/2. Finally, secret-key cryptosystems cannot provide non-repudiation of origin or non-repudiation of receipt security services, as a message sender and receiver can later deny sending and receiving the message. These limitations have led to the advent of public-key cryptosystems [90] .

A.5 Public-Key (Asymmetric) Cryptosystems

A public-key cryptosystem, also referred to as asymmetric, uses a pair of keys; a public key, which can be known to anyone, and a private key known only to its owner. The public key and the private key are mathematically related so that data encrypted with either key can only be decrypted using the other. However, the private key cannot feasibly be derived from the public key. A public-key cryptosystem depends on one or more trusted third parties, known as certification authorities, to securely bind a public key to its rightful owner. Public-key cryptosystems include public-key encryptions and digital signatures.

A.5.1 Public-Key Encryptions

Public-key encryptions (also known as asymmetric ciphers) use public keys for encryption and private keys for decryption. As only the owner of a private key can decrypt a message, public-key encryptions can be used to provide message confidentiality. Figure A.3 shows the process of public-key encryption.

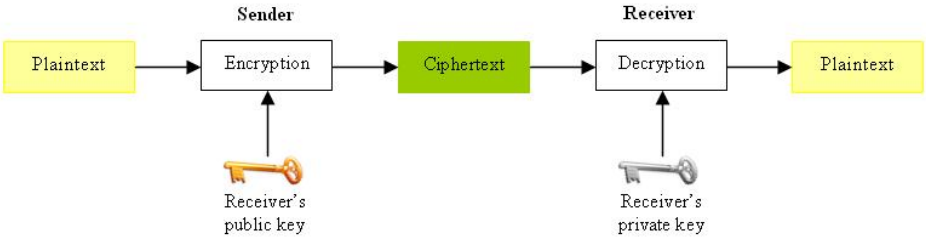

Figure A.3: Public-key encryption/decryption scheme

Public-key encryptions can provide message confidentiality, but they are not normally used for bulk-message encryption as they are much slower in comparison to secret-key encryptions (up to 100 times in software implementation and

10,000 times in hardware implementation [91]). Therefore, public-key encryption is typically combined with secret-key encryption to get the advantages of both encryptions. At the beginning of a communication session, the sender and the receiver use public-key encryption to establish a secret key. They then use the secret key to provide integrity, authenticity, and confidentiality to all subsequent messages using MAC and secret-key encryption. Well-known Internet security protocols, such as SSL [92] and IPSec [25], use this hybrid approach.

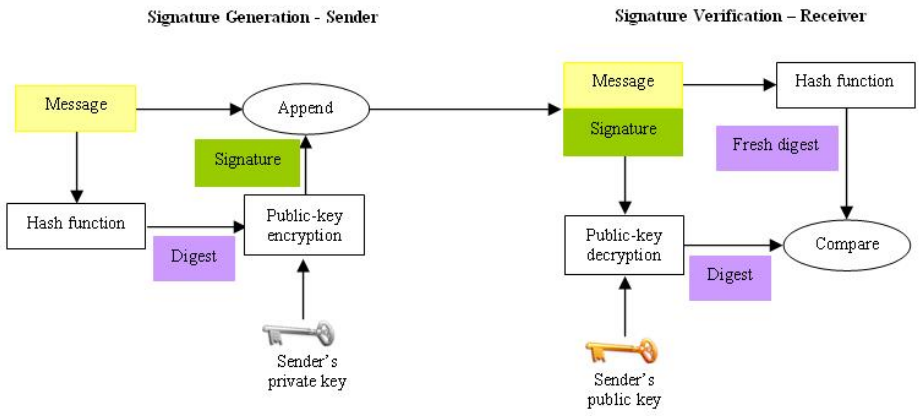

Figure A.4: Digital signature generation and verification processes

A.5.2 Digital Signatures

A digital signature of a message can be defined as a cryptographic transformation of the message using a secret known only to the signer (signer's private key). A digital signature scheme consists of two main components: namely, a signature generation algorithm and a signature verification algorithm. The signature generation algorithm takes the signing entity's private key and the message that needs to be signed as input, and then produces output as a digital signature for that message. The signature verification algorithm takes the message, the signature and the signer's public key as input, and outputs an indication as to whether or not the signature on the message is valid. For example, if a sender wished to send a digitally signed message to a receiver, the sender would first apply a hash function to the message to produce a shorter message digest. The digest would then be encrypted with the sender's private key to produce a digital signature on the message. The signature would then be appended to the original message and

sent together as a digitally signed message to the receiver. Once the signed message was received, the receiver would verify the signature by first decrypting the signature with the sender's public key to obtain a message digest. The receiver would then apply the same hash function to the received message to generate a fresh message digest. If the two digest values matched, then the signature would be positively verified. Figure A.4 shows the signature generation and verification processes.

Digital signatures can be used to provide data integrity, data authenticity, and non-repudiation of origin services [87]. Since the sender uses his/her own private key to generate the signature, a digital signature guarantees that the message must have come from the claimed sender. Thus, the sender cannot later deny that he/she generated and signed the message. The signature also assures that any changes made to data cannot go undetected.

A.6 Digital Certificates and PKI

An important security issue associated with the use of public-key cryptosystems is how to assure other entities that a public key is authentic, i.e. indeed belongs to a particular entity. A well-known way to provide such assurance is through the use of public-key certificates provided by Public-Key Infrastructure (PKI).

A public-key certificate is a structured data record issued and signed by a trusted entity, called the Certification Authority (CA). The certificate contains a number of data entries, including the public key that is being certified, the identity of the public-key owner, the validity period of the certificate, the identity of the issuer (the CA), and the CA's signature on the certificate. The CA maintains a list of all revoked certificates, known as the Certificate Revocation List (CRL). In this way, an entity can verify the authenticity of a public key by checking both the CA's signature contained in the certificate associated with the public key and the CA's most recent CRL.

PKI is a hierarchical structure of CAs that perform the issuance and management of public-key certificates for the purpose of the wide-scale deployment of public-key cryptosystems. The root CA issues public-key certificates to CAs at a level immediately below, these in turn issue certificates to CAs at the next level down, and so on, until the leaf nodes where the certificates are issued to organizations and individuals.

A.7 Diffie-Hellman (DH) Key Exchange

The Diffie-Hellman (DH) key exchange [82] is a cryptographic protocol that enables two entities without a previous security association to establish a shared secret key over an insecure public network. The shared key can then be used to authenticate and/or encrypt subsequent communications using a secret-key cryptosystem. A description of the protocol is shown in Figure A.5 and is summarised as follows:

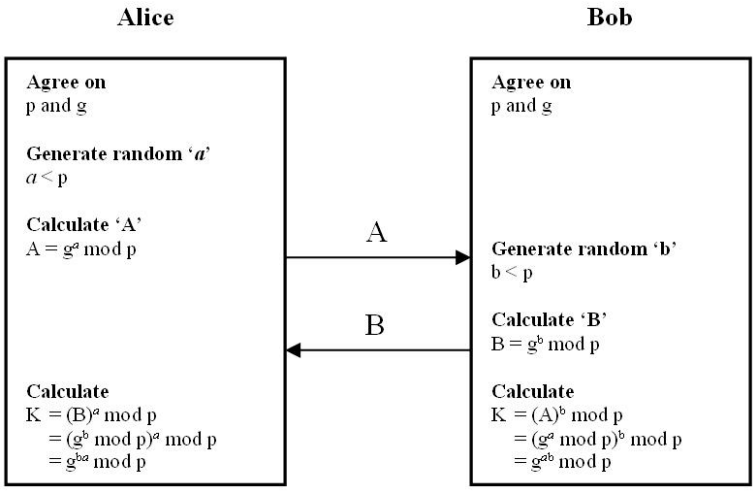

Figure A.5: A Diffie-Hellman key exchange

- Alice and Bob agree on p and g, where p is a large prime number and g is a primitive root (mod p) of the prime number p.
- Alice chooses a random private number 'a', i.e. $a < \mathrm{p}$, and computes and sends a public value 'A' to Bob, i.e. $\mathrm{A} = \mathrm{g}^a \bmod \mathrm{p}$.
- Bob chooses a random private number 'b', i.e. $\mathrm{b} < \mathrm{p}$, and computes and sends a public value 'B' to Alice, i.e. $\mathrm{B} = \mathrm{g}^{\mathrm{b}} \bmod \mathrm{p}$.
- Alice computes the key $\mathrm{K} = (\mathrm{B})^a \bmod \mathrm{p} = (\mathrm{g}^{\mathrm{b}} \bmod \mathrm{p})^a \bmod \mathrm{p} = \mathrm{g}^{\mathrm{b}a} \bmod \mathrm{p}$.
- Bob computes the key $\mathrm{K} = (\mathrm{A})^{\mathrm{b}} \bmod \mathrm{p} = (\mathrm{g}^a \bmod \mathrm{p})^{\mathrm{b}} \bmod \mathrm{p} = \mathrm{g}^{a\mathrm{b}} \bmod \mathrm{p}$.

Both Alice and Bob are now in possession of the shared secret key K, because $\mathrm{g}^{a\mathrm{b}}$ and $\mathrm{g}^{\mathrm{b}a}$ are equal mod p. In addition, perfect forward secrecy is achieved,

as the secret numbers 'a' and 'b' are discarded at the end of the session. The established shared secret key could be used to protect subsequent data exchanged between Alice and Bob.

The DH key exchange is vulnerable to a man-in-the-middle attack. In this attack, an attacker establishes two distinct DH key exchanges, masquerading as Alice to Bob and vice versa. When Alice transmits her public value 'A', the attacker intercepts it and sends his/her own public value to Bob. The attacker intercepts Bob's public value 'B' and sends his/her own public value to Alice. The attacker then decrypts any messages sent out by Alice or Bob (and reads and possibly modifies them), and re-encrypts and transmits the messages to the other entity. One possible solution to this vulnerability is to use PKI and digital signatures for the authentication of the exchanged public keys.

Appendix B

Existing Correspondent Registration Protocols

This appendix presents a more detailed description of the existing correspondent registration protocols mentioned in Chapter 3.

B.1 EBU Protocol

- The MN initiates a proactive home address test, which is technically the same as a standard home address test, by sending a HoTI message to the CN. The CN responds by returning a HoT message containing a secret home keygen token (Token_1). The MN needs to run the test at most every 3.5 minutes as this is the lifetime of a home keygen token [5]. Alternatively, the MN may be able to run the test just before the current link breaks if its link layer can trigger the indication.
- After moving to a foreign link, the MN sends an Early Binding Update (EBU) message to the CN for CoA registration. The EBU is authenticated using a binding management key (K_{BM}) that is generated based on the home keygen token received during the proactive home address test, i.e. K_{BM} = SHA1 (Token_1). The MN also initiates a concurrent care-of address test, which is technically the same as a standard care-of address test, by sending a CoTI message to the CN as soon as it has sent the EBU message.

- Upon receiving the EBU message, the CN verifies the MN's reachability at the HoA using K_{BM}, and infers the MN's new CoA. Therefore, the CN creates/updates a Binding Cache entry, stores the binding between the MN's HoA and CoA at the entry and, optionally, returns an Early Binding Acknowledgement (EBA) message for confirmation. The CN also sends subsequent packets destined for the MN to its new CoA. However, as the CN is not yet able to verify the MN's reachability at the CoA, the CN sets the entry in an unconfirmed state and limits the amount of data sent to the CoA by using a Credit-Based Authorisation technique [39, 40]. The CN also responds to the concurrent care-of address test by returning a CoT message containing a secret care-of keygen token ($Token_2$).

- When the MN receives the CoT message, it sends a BU message to the CN. The BU is authenticated using a key (K_{BM}) that is generated based on both the home and care-of keygen tokens, i.e. K_{BM} = SHA1 ($Token_1$ || $Token_2$).

- Upon receiving the BU message, the CN verifies the authenticity of the message, and then changes the status of the CoA to be confirmed. In addition, the CN optionally returns a BA message to the MN for confirmation.

B.2 PBK Protocol

Initialisation:

- The MN generates a temporary public/private key pair, which is known as a Purpose-Built Key (PBK) pair.

- The MN generates a Purpose-Built ID (PBID) by performing a cryptographic one-way hash of the public key of the PBK pair.

- When the MN initiates a new connection with a CN, it sends the PBID as well as the public key of its PBK pair along with the initial packets in the connection.

- When the CN receives the initial packets, it verifies that the received public key hashes to the received PBID. The CN then stores the PBID, the public key and the source address of the packets, i.e. the MN's HoA.

Correspondent registrations:

- The MN initiates a correspondent registration by sending a request message to the CN. The MN includes the PBID in the message and signs it with the private key of the PBK pair.
- When the CN receives the request message, it verifies the digital signature using the stored public key associated with the PBID. If the verification succeeds, the CN will generate and send a random number as a challenge to the MN's CoA.
- Upon receiving the challenge, the MN encrypts the random number and the binding information with the private key of the PBK pair to create a response. The MN then sends the response and the PBID to the CN.
- Upon receiving the response, the CN decrypts and verifies it using the stored public key. If the verification succeeds, the CN will accept the binding information and the current correspondent registration will conclude.

B.3 UDHBU Protocol

Initial correspondent registration:

- The MN initiates the protocol by sending Message_1 to the CN; Message_1 = {Src=CoA, Des=CN, HoA, N_I, g^m}, where N_I is a fresh random nonce and g^m is a DH public value from a well-known DH group. The nonce N_I is to protect the MN against replay attacks and to enable the MN to reuse g^m while ensuring that the resulting session key will be different.
- When the CN receives Message_1, it checks the DH group chosen by the MN; if it is not acceptable, the CN will reply indicating the accepted DH groups. Otherwise, the CN will reply by sending Message_2 to the MN's HoA; Message_2 = {Src=CN, Des=HoA, N_I, N_R, g^c, key-identifier, authenticator}, where N_R is a fresh random nonce; g^c is a DH public value from the same DH group chosen by the MN; key-identifier is an identifier for the public value g^c; and the authenticator is calculated using a keyed hashing MAC from a secret K_{CN} known only to the CN, i.e. authenticator = HMAC (K_{CN}, (CN || HoA || CoA || N_I || N_R || g^m || g^c)).

- When the MN receives Message_2, it verifies that the given nonce, i.e. N_I, matches the nonce sent by the MN to this CN in Message_1. The MN next computes a DH session key (K_S) based on the two nonces, its DH private value, and the CN's DH public value. The MN then sends Message_3 to the CN, which includes a BU that is authenticated using the key K_S; Message_3 = {Src=CoA, Des=CN, N_I, N_R, g^m, key-identifier, authenticator, Authenticated BU (K_S)}.

- When the CN receives Message_3, it first checks on the validity of the nonce N_R. It next validates the received authenticator; the CN calculates a fresh authenticator and compares it to the received one. If the validations succeed, the CN will compute K_S and verify the authenticity of the BU. If the BU authentication succeeds, the CN will create a Binding Cache entry and store the binding between the MN's HoA and CoA as well as the value of K_S at the entry. It also optionally sends Message_4 to the MN, where Message_4 is a BA message that is authenticated using the key K_S.

Subsequent correspondent registrations:

- In subsequent correspondent registrations, the MN and the CN use the shared session key, i.e. K_S, directly in authenticating BUs and BAs.

B.4 CGA-OMIPv6 Protocol

Initial correspondent registration:

- An MN and a CN run a modified RR procedure, which establishes a shared secret (K_{BM}) between the two nodes. The modified RR procedure combines the HoTI and the CoTI messages in one message called the HoTI&CoTI message. Apart from this, the modified and the original RR procedures are identical.

- Immediately after a successful running of the modified RR procedure, the MN sends a normal BU message, but includes the MN's public key and signature as well as the CGA auxiliary parameters that are used to generate the HoA cryptographically (see Section 3.1). The BU message is authenticated using the K_{BM} established from the modified RR procedure.

- Upon receiving the BU message, the CN verifies the authenticity of the message using K_{BM}. It next runs the CGA-based address verification algorithm (see

Figures 3.2 in Section 3.1) to verify the MN's HoA. The CN then checks on the validity of the signature. If the address and signature verifications succeed, the CN generates a fresh session key ($K_{SESSION}$) and encrypts it with the MN's public key. The CN also creates a Binding Cache entry and stores the binding between the MN's HoA and CoA as well as the value of $K_{SESSION}$ at the entry. Finally, the CN sends a BA message to the MN. The BA message includes the encrypted session key and is authenticated using the K_{BM} established from the modified RR procedure.

- Upon receiving the BA message, the MN uses K_{BM} to verify the authenticity of the message. The MN then decrypts the session key and records it for subsequent correspondent registrations.

Subsequent correspondent registrations:

- The MN initiates a subsequent correspondent registration by performing a care-of address test; the MN and the CN exchange a CoTI and a CoT message. The CoT message sent by the CN to the MN includes a secret care-of keygen token ($Token_2$).

- The MN and the CN then exchange a BU and a BA message, which they authenticate using a one-time session key (K_{BMPERM}). The two nodes generate K_{BMPERM} by hashing the shared session key with the care-of keygen token, i.e. K_{BMPERM} = HMAC_SHA1 ($K_{SESSION}$, $Token_2$).

B.5 ERO-MIPv6 Protocol

Initial correspondent registration:

- The MN initiates the initial correspondent registration by running a proactive home address test with the CN as in the EBU protocol, which results in sharing a home keygen token ($Token_1$).

- After moving to a foreign link, the MN sends an EBU&CoTI message to the CN. The EBU&CoTI message is a merging of the EBU and the CoTI messages from the EBU protocol but includes the CGA auxiliary parameters and the MN's public key and signature as in the CGA-OMIPv6 protocol. The EBU&CoTI message is authenticated using a key (K_{BM}) that is generated based on the

home keygen token received during the proactive home address test as in the EBU protocol (i.e. K_{BM} = SHA1(Token$_1$)).

- Upon receiving the EBU&CoTI message, the CN generates K_{BM} and validates the message. The CN next runs the CGA-based address verification algorithm. It then verifies the signature, generates a fresh session key ($K_{SESSION}$), and encrypts the session key with the MN's public key, as in the CGA-OMIPv6 protocol. It also utilises the Credit-Based Authorisation technique to limit the amount of data sent to the MN's unconfirmed CoA, as in the EBU protocol. Finally, the CN sends an EBA&CoT message to the MN. The EBA&CoT message is a merging of the EBA and the CoT messages from the EBU protocol but also includes the encrypted session key as in the CGA-OMIPv6 protocol. The EBA&CoT message is authenticated using the K_{BM} as in the EBU protocol.

- Upon receiving the EBA&CoT message, the MN knows the session key and the care-of keygen token. The MN and the CN next exchange normal BU and BA messages. The BU and BA messages are authenticated with a key (K_{BM-new}) that is generated based on the shared session key and the fresh care-of token.

Subsequent correspondent registrations:

- Subsequent correspondent registrations are identical to the initial correspondent registration, but the home address ownership and reachability proofs are omitted. Specifically, there is no proactive home address test, i.e. the MN and the CN do not exchange HoTI and HoT messages. In addition, the EBU&CoTI message does not contain the CGA auxiliary parameters, the MN's public key, or the MN's signature. Furthermore, the EBA&CoT message does not contain the encrypted session key.

B.6 TBU Protocol

Initial correspondent registration:

- The MN initiates a correspondent registration by sending a BU message (Message$_1$) to the HA, i.e. Message$_1$ = {Src=CoA, Des=HA, HoA, CoA, CN, N_{MN}, TS_{MN}, LT_{BReq}}, where N_{MN} is a fresh nonce; TS_{MN} is the MN's timestamp; and LT_{BReq} is a binding lifetime request. Message$_1$ is reversed tunnelled from the MN to the HA through the IPSec ESP secure tunnel.

- Upon receiving Message_1, the HA first validates timestamp TS_{MN}. It then generates a nonce (N_{HA}) and sends Message_2 to the CN on behalf of the MN. Message_2 is a request for ticket creation and correspondent registration execution between the MN and CN. Message_2 = {Src=HoA, Des=CN, CoA, N_{MN}, N_{HA}, TS_{HA}, LT_{BReq}}, where TS_{HA} is the HA's timestamp. Message_2 is also sent to the CN via the IPSec ESP secure tunnel.

- Upon receiving Message_2, the CN first validates timestamp TS_{HA}. It next generates a nonce (N_{CN}) and creates a ticket key K_{MN-CN} = HMAC (K_{HA-CN}, (N_{MN} || N_{CN} || N_{HA})), where K_{HA-CN} is the IPSec secret key shared between HA and CN. The CN also creates a ticket $\text{TcK}_{MN-CN} = \text{ENC}_{K_{CN}}$ [HoA || N_{CN} || TS_{CN} || LT_{TcK} || K_{MN-CN}], where K_{CN} is a secret random value only known by the CN; TS_{CN} is the CN's timestamp; and LT_{TcK} is the ticket's lifetime. The CN also creates a Binding Cache entry for HoA and CoA. Finally, the CN sends Message_3 to the MN's HoA, i.e. Message_3 = {Src=CN, Des=HoA, N_{HA}, N_{CN}, TS_{CN}, LT_{BGrant}, TcK_{MN-CN}}, where LT_{BGrant} is a granted binding lifetime set by CN.

- The HA intercepts Message_3 and checks on the validity of nonce N_{HA} and timestamp TS_{CN}. The HA next creates K_{MN-CN} like CN, and sends Message_4 to the MN through the IPSec ESP secure tunnel. Message_4 = {Src=HA, Des=CoA, HoA, CN, N_{MN}, TS_{HA}, LT_{BGrant}, TcK_{MN-CN}, K_{MN-CN}}.

- When the MN receives Message_4, it checks on the validity of nonce N_{MN}, and then it securely stores values of TcK_{MN-CN} and K_{MN-CN} in its Binding Update List entry for that CN. At this point, the initial correspondent registration concludes and a secret key as well as a ticket are distributed between the MN and the CN.

Subsequent correspondent registrations:

- In subsequent correspondent registrations, the MN sends a BU message directly to the CN to register its new CoA, i.e. BU = {Src=CoA, Des=CN, HoA, N_{MN}, TS_{MN}, LT_{BReq}, TcK_{MN-CN}, $\text{MAC}_{K_{MN-CN}}$ (BU)}, where N_{MN} is a fresh nonce, TS_{MN} is the MN's timestamp; and $\text{MAC}_{K_{MN-CN}}$ (BU) is a keyed hash value for the BU message using the ticket key K_{MN-CN}.

- Upon receiving the BU message, the CN checks on the validity of timestamp TS_{MN}. It next verifies the validity of the ticket TcK_{MN-CN} by decrypting and

checking the MN's HoA as well as the ticket lifetime LT_{TcK} included in the ticket. The CN then verifies the MAC using the ticket key K_{MN-CN}. Finally, the CN sends a BA message to the MN for confirmation, i.e. BA = {Src=CN, Des=CoA, HoA, N_{MN}, TS_{CN}, LT_{BGrant}, $MAC_{K_{MN-CN}}$ (BA)}.

- Upon receiving the BA message, the MN checks the validity of nonce N_{MN}, and then verifies the MAC using the ticket key K_{MN-CN}.

B.7 CBU Protocol

- The MN initiates a correspondent registration by sending a route optimization request (REQ) message to the HA, i.e. REQ = {Src=CoA, Des=HA, HoA, CoA, CN, N_0}, where N_0 is a nonce value used to match the reply message. The REQ message is 'reversed tunnelled' from the MN to the HA through the IPSec ESP secure tunnel.

- Upon receiving the REQ message, the HA generates a cookie (C_0) and sends $COOKIE_0$ message to the CN on behalf of the MN, i.e. $COOKIE_0$ = {Src=HoA, Des=CN, C_0}.

- Upon receiving the $COOKIE_0$ message, the CN generates a nonce N_1 and a cookie C_1, and sends $COOKIE_1$ message to the MN's HoA, i.e. $COOKIE_1$ = {Src=CN, Des=HoA, C_0, C_1, N_1}.

- The HA intercepts the $COOKIE_1$ message and checks on the validity of C_0. The HA next generates a nonce N_2 and a DH private value 'x', and computes its DH public value g^x. The HA then sends $EXCH_0$ message to the CN on behalf of the MN. The $EXCH_0$ = {Src=HoA, Des=CN, C_0, C_1, N_1, N_2, g^x, SIG_H [H (HoA || CN || g^x || N_1 || N_2)], $Cert_H$}, where SIG_H [] denotes the home link's signature using the private key SK_H, and $Cert_H$ is the public key certificate of the home link.

- When the CN receives the $EXCH_0$ message, it validates the cookies, the home link's public key certificate, and the signature. If all the validations are positive, the CN is confident that the MN's HoA is authorised by its home link and the DH public value g^x is freshly generated by the MN's home link. Therefore, the CN generates a DH private value 'y' and computes its DH public value g^y. The CN next computes a DH key $K_{DH} = (g^x)^y$ and a session key K_{BM}

= HMAC (K_{DH}, (N_1 || N_2)). The CN also creates a Binding Cache entry for the HoA and stores the value of K_{BM} for authenticating subsequent correspondent registrations from the MN. Finally, the HA sends $EXCH_1$ message to the MN's HoA, i.e. $EXCH_1$ = {Src=CN, Des=HoA, C_0, C_1, g^y, $MAC_{K_{BM}}$ (g^y || $EXCH_0$)}.

- Again, the HA intercepts the $EXCH_1$ message, validates the cookies, and calculates the DH key K_{DH} as well as the session key K_{BM}. The HA next validates $MAC_{K_{BM}}$ (g^y || $EXCH_0$). If all the validations are positive, the HA sends a REP message to the MN through the IPSec ESP secure tunnel, i.e. REP = {Src=HA, Des=CoA, HoA, CN, N_0, K_{BM}}.

- Upon receiving the REP message, the MN verifies that N_0 is the same as the one it sent out in the REQ message. If the verification succeeds, the MN proceeds by exchanging normal BU and BA messages with the CN, where the messages are protected using the key K_{BM}.

- In subsequent correspondent registrations, the MN and the CN use the shared session key, i.e. K_{BM}, directly in authenticating BUs and BAs.

B.8 HCBU Protocol

The protocol is divided into three steps. The first step consists of three messages and is initiated by the MN when it realises an imminent handover. The MN generates a fresh nonce N_m and sends $Message_1$ to the HA, i.e. $Message_1$ = {Src=CoA, Des=HA, HoA, CoA, CN, N_m}. The HA (on behalf of the MN) and the CN then exchange $Message_2$ and $Message_3$ to prepare for the coming binding update. $Message_2$ passes the fresh nonce N_m, MN's HoA, and a DH public value g^x to the CN, i.e. $Message_2$ = {Src=HoA, Des=CN, N_m, g^x}. In reply, $Message_3$ passes CN's fresh nonce N_c and DH public value g^y to the HA. $Message_3$ also contains a cookie $Cookie_{CN}$ that is created using CN's secret key K_{CN}. $Message_3$ = {Src=CN, Des=HoA, N_m, N_c, g^x, g^y, $Cookie_{CN}$}, where $Cookie_{CN}$ = HMAC (K_{CN}, $Message_3$). At this point, the CN does not create a state for the protocol, which protects the CN against resource exhaustion DoS attacks. However, the HA needs to record the values of N_m, N_c, g^x, g^y, and $Cookie_{CN}$.

The second step consists of three messages and is initiated by the MN when it roams to a foreign link and is configured a new CoA. The MN first obtains the

foreign link's signature on the binding of (HoA, CoA), i.e. SIG_H' [H (HoA || CoA || Validity)]. The MN then sends $Message_4$ to the HA for proving its ownership of CoA, i.e. $Message_4$ = {Src=CoA, Des=HA, HoA, CoA, Validity, CN, SIG_H' [H (HoA || CoA || Validity)], $Cert_H$'}, where $Cert_H$' is the public key certificate of the foreign link. Upon receiving $Message_4$, the HA first checks the validity of the certificate and verifies the signature contained in the message, which assures the correctness of the MN's CoA. Next, the HA computes a DH key $K_{DH} = (g^x)^y$, and a session key K_{BM} = HMAC (K_{DH}, (N_m || N_c)). The HA then sends $Message_5$ to the MN for delivering the session key, i.e. $Message_5$ = {Src=HA, Des=CoA, HoA, CN, N_m, K_{BM}}. At the same time, the HA also sends $Message_6$ to the CN for proving the MN's ownership of both the HoA and CoA. $Message_6$ = {Src=HoA, Des=CN, CoA, N_m, N_c, g^x, g^y, $Cookie_{CN}$, Validity, SIG_H [H (HoA || CoA || Validity || CN || K_{DH} || N_m || N_c)], $Cert_H$}. When the CN receives $Message_6$, it first validates the cookie $Cookie_{CN}$. It next checks the validity of the certificate. It then computes K_{DH} and verifies the signature contained in the message. If all validations succeed, the CN is confident that both the MN's HoA and CoA are indeed correct. Finally, the CN computes K_{BM} and creates a Binding Cache entry for HoA, CoA, and K_{BM}. However, the CN initialises this entry in an inactive state, indicating that a final confirmation from the CoA is required.

The third step consists of two messages, i.e. normal BU and BA messages, and is initiated immediately by the MN when it receives $Message_5$ from the HA. The MN exchanges normal BU and BA messages with the CN, where the messages are protected using the K_{BM}. As a result, the CN changes the state of the entry to active and sends subsequent traffic directly to the CoA.

In subsequent correspondent registrations, the MN obtains the foreign link's signature on the binding of (HoA, CoA). The MN then sends a BU message that contains the foreign link's signature and public key certificate to the CN. The BU message is also protected using K_{BM}.

B.9 ETBU Protocol

Initial correspondent registration:

- The MN generates a fresh nonce (N_{MN}), a fresh session key (K_{MN-CN}), and a ticket $TcK_{MN-CN} = ENC_{K_{MN}}$ [CN || TS_{MN} || LT_{TcK} || K_{MN-CN}], where K_{MN}

is a secret random value only known by MN; TS_{MN} is MN's timestamp representing the issue time of the ticket; and LT_{TcK} is the ticket's lifetime. The MN next sends a BU request message ($Message_1$) to its home agent, i.e. $Message_1$ = {Src=CoA, Des=HA, HoA, CoA, CN, N_{MN}, Seq_{MN}, LT_{BReq}, TcK_{MN-CN}, K_{MN-CN}}, where Seq_{MN} is an initial sequence number; and LT_{BReq} is a binding lifetime request. $Message_1$ is reversed tunnelled from the MN to the HA through the IPSec ESP secure tunnel.

- Upon receiving $Message_1$, the HA_{MN} generates a nonce ($N_{HA_{MN}}$) and sends a BU request forwarding message ($Message_2$) to the CN on behalf of the MN. $Message_2$ = {Src=HoA, Des=CN, $N_{HA_{MN}}$, $SIG_{HA_{MN}}$ [$N_{HA_{MN}}$ || LT_{BReq} || TcK_{MN-CN} || $TS_{HA_{MN}}$ || Seq_{MN}], $ENC_{PK_{HA_{CN}}}$ [$N_{HA_{MN}}$ || HA_{MN} || CoA || K_{MN-CN}]}, where $TS_{HA_{MN}}$ is HA_{MN}'s timestamp and $ENC_{PK_{HA_{CN}}}$ denotes encryption using the CN's home link public key.

- The CN's home agent (HA_{CN}) intercepts $Message_2$ and checks on the validity of the signature and then decrypts the message using its private key. If the validation succeeds, HA_{CN} then sends a BU request verification message ($Message_3$) to the CN through the IPSec ESP secure tunnel. $Message_3$ = {Src=HoA, Des=CN, $N_{HA_{CN}}$, CoA, Seq_{MN}, TcK_{MN-CN}, K_{MN-CN}}, where $N_{HA_{CN}}$ is a fresh nonce.

- Upon receiving $Message_3$, the CN creates a Binding Cache entry for the HoA and CoA, and stores the value of Seq_{MN} in it. It also securely stores the values of TcK_{MN-CN} and K_{MN-CN}. The CN next generates a fresh session key (K_{CN-MN}) and a ticket (TcK_{CN-MN}) as the MN, but uses its own secret value (K_{CN}). The CN then sends a BA respond message ($Message_4$) to the HA_{CN} through the IPSec ESP secure tunnel. $Message_4$ = {Src=CN, Des=HA_{CN}, HoA, Seq_{MN}, N_{CN}, TS_{CN}, LT_{BGrant}, TcK_{CN-MN}, K_{CN-MN}}.

- Upon receiving $Message_4$, the HA_{CN} sends $Message_5$ to the MN on behalf of the CN. $Message_5$ = {Src=CN, Des=HoA, $N_{HA_{MN}}$, $SIG_{HA_{CN}}$ [$N_{HA_{MN}}$ || LT_{BGrant} || TcK_{CN-MN} || $TS_{HA_{CN}}$ || Seq_{MN}], $ENC_{PK_{HA_{MN}}}$ [$N_{HA_{MN}}$ || HA_{CN} || K_{CN-MN}]}, where $TS_{HA_{CN}}$ is the CN's home link timestamp and $ENC_{PK_{HA_{MN}}}$ denotes encryption using the MN's home link public key.

- After intercepting $Message_5$, the HA_{MN} checks on the validity of the nonce $N_{HA_{MN}}$. It next decrypts the message using its private key and then checks

on the validity of the signature. If the validation succeeds, HA_{MN} will send $Message_6$ to the MN through the IPSec ESP secure tunnel. $Message_6$ = {Src=HA, Des=CoA, HoA, CN, N_{MN}, Seq_{MN}, LT_{BGrant}, TcK_{CN-MN}, K_{CN-MN}}.

- When the MN receives $Message_6$, it checks on the validity of nonce N_{MN} and sequence number Seq_{MN}. It then securely stores values of TcK_{CN-MN} and K_{CN-MN} in its Binding Update List for that CN. At this point, the initial correspondent registration concludes and two secret keys as well as two tickets are distributed between the MN and the CN.

Subsequent correspondent registrations:

- When the MN roams to a new foreign link, it first generates a fresh nonce N_{MN}. The MN then sends a BU message to the CN, i.e. BU = {Src=CoA, Des=CN, HoA, N_{MN}, Seq_{MN}, LT_{BReq}, TcK_{CN-MN}, PK_{MN}, $MAC_{K_{CN-MN}}$ (BU)}, where PK_{MN} is the MN's self-generated public key that is used in generating both the HoA and the CoA; and $MAC_{K_{CN-MN}}$ (BU) is a keyed hash value for the BU message using the ticket key K_{CN-MN}. If the CN is currently located in its home link, it will receive the BU message directly. Otherwise, the HA_{CN} will intercept and forward the messages to the CN's current CoA via a secure tunnel. If the CN simultaneously moves to a new CoA, the message is further intercepted by an HA existing in the CN's previous link and is forwarded to the CN's new CoA.

- Upon receiving the BU message, the CN verifies the validity of the MN's HoA and CoA by running the CGA-based address verification algorithm using PK_{MN} (see Figures 3.2 in Section 3.1). The CN next decrypts the ticket TcK_{CN-MN} using its secret K_{CN}, and then verifies the validation periods and the HoA included in the ticket. Finally, the CN uses the ticket key K_{CN-MN} contained in the ticket to verify the $MAC_{K_{CN-MN}}$ (BU). If all verifications succeed, the CN updates its Binding Cache entry and sends a BA message to the MN, i.e. BA = {Src=CN, Des=CoA, HoA, N_{MN}, Seq_{MN}, LT_{BGrant}, $MAC_{K_{CN-MN}}$ (BA)}.

- Upon receiving the BA message, the MN first checks the validity of nonce N_{MN} and sequence number Seq_{MN}, and then verifies the $MAC_{K_{CN-MN}}$ (BA) using the ticket key K_{CN-MN}.

Appendix C

OPNET Modeler and CryptoSys Toolkit

C.1 OPNET Modeler

Simulations modelling presented in this thesis use the OPNET^{TM} simulation package [13]. OPNET Modeler is an event driven simulator and provides a powerful graphical tool to display simulation statistics. It uses hierarchically linked domains to denote a network design, namely the process domain, the node domain, and the network domain. Figure C.1 demonstrates the model hierarchy from an example model [93].

At the lowest level, protocols and applications are defined as process models. A process model is represented as a state transition diagram (STD). Each state in the model has an Enter and Exit execution block where C/C++ coded procedures are entered. OPNET Modeler uses various pre-written kernel procedures to simplify simulation modelling by implementing key functions such as sending and receiving messages. Transitions between states have associated condition macros. The flow of control in the process is determined by interrupt events such as packet arrivals or timer expirations. There are two types of state used; the forced state and the un-forced state. The process will remain in the unforced state until the next interrupt event occurs after which it will move to the next state determined by the state-flow diagram. The process will move immediately from a forced state once the executive blocks have been processed.

The next level in OPNET Modeler's simulation model hierarchy is node modelling. A node model is built from a number of process models that are connected

together using message flows. Network nodes, such as workstations and routers, are defined as node models. Node models are used at the next level to create network models. A network model consists of a number of network nodes that are connected using network links. Sub-networks can also be used to represent an entire network topology.

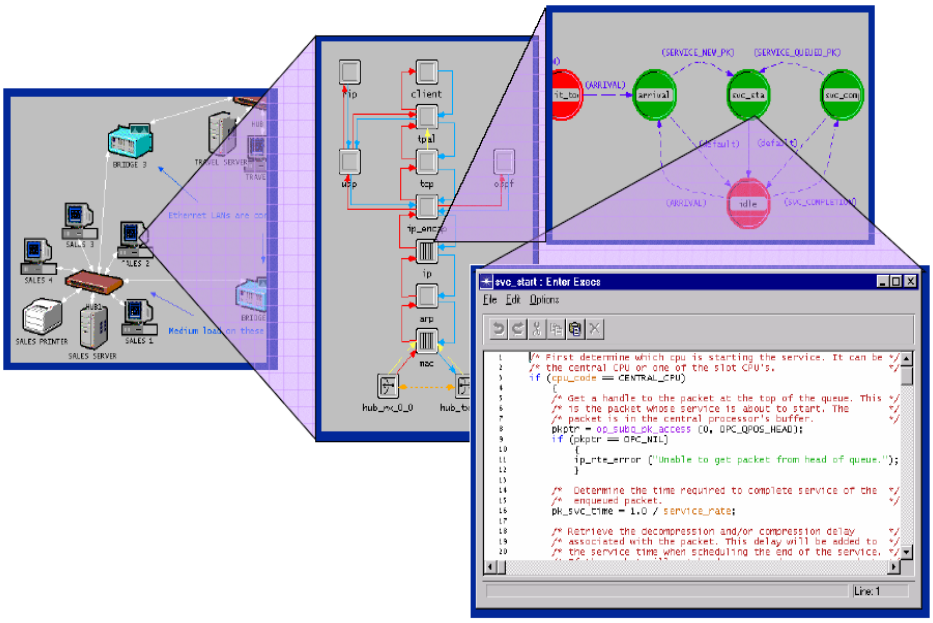

Figure C.1: OPNET modelling hierarchy [93]

OPNET Modeler has a large library of pre-implemented protocols, applications, nodes, and statistics. The basic Mobile IP protocols (MIPv4 and MIPv6) have been implemented as part of the standard models library. This is one of the main incentives for choosing OPNET as a simulator tool because the author is not required to implement the basic functionalities of MIPv6 prior to the integration of the proposed protocols.

C.2 Mobile IPv6 in OPNET Modeler

The implementation of the Mobile IPv6 protocol in the OPNET Modeler version 14.5 is based upon two process models: **mipv6_mgr** and **mipv6_mn**. The former performs the operations for home agents and correspondent nodes. The latter performs the operations for mobile nodes. Also, the former is responsible for

creating the latter; the **mipv6_mgr** process model is the parent process model that creates the **mipv6_mn** process model.

The **mipv6_mgr** process model is depicted in Figure C.2. It consists of two states: 'init' state and 'idle' state. The 'init' state initializes the **mipv6_mgr** process and reads the Mobile IPv6-related attributes such as 'node type' and 'route optimization status'. A different set of attributes is expected for every different node type. This state is also responsible for creating and invoking the child process model '**mipv6_mn** in case this node type is a mobile node. The 'idle' state is responsible for handling any incoming mobility message. This state evaluates the mobility header in the incoming message to determine the message type. Upon determining the message type, the state either processes it or sends it to the **mipv6_mn** process. An HA and/or a CN will process mobility messages in the current **mipv6_mgr** process.

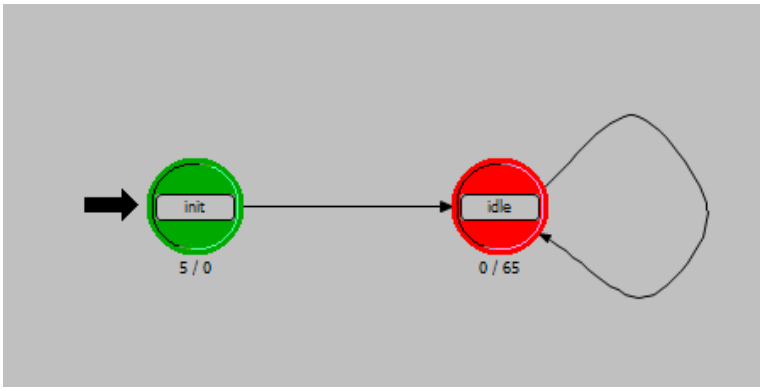

Figure C.2: mipv6_mgr process model

The **mipv6_mn** process model is depicted in Figure C.3. It consists of six states: 'init' state, 'home' state, 'ha_bind' state, 'away' state, 'route_test' state, and 'cn_bind' state. The 'init' state initializes the **mipv6_mn** process and its state variables. In addition, it reads the node's home address, sets the node's care-of address to invalid (i.e. no care-of address is set yet), and registers the node's local and global statistics.

The home state sets some flags to indicate that the node (MN) is currently located at its home link. In addition, it evaluates the current invocation of the **mipv6_mn** process by executing the **mipv6_mn_invocation_process** function (which will be explained later in greater detail).

The 'ha_bind' state is responsible for sending a BU message to an HA. This

state first checks the MN's current location, i.e. whether the MN is at the home network or at a foreign network. It then sends a BU message that requests the HA to either remove its Binding Cache entry or update it with the MN's current CoA. Finally, this state executes the **mipv6_mn_invocation_process** function to evaluate all of the invocations of the current **mipv6_mn** process.

The 'away' state sets some flags to indicate that the MN is away from its home network. The state also executes the **mipv6_mn_invocation_process** function to evaluate all of the invocations of the current **mipv6_mn** process.

The 'route_test' state is responsible for performing a route optimization procedure with CNs. This state first checks if route optimization is enabled in this MN. If enabled, it will send both of the routability test initialization messages, i.e. HoTI and CoTI messages, to all the CNs already in the MN's Binding Update List.

The 'cn_bind' state is responsible for sending a BU message to a CN. This state checks the MN's current location, and then sends a BU message that requests the CN to either remove its Binding Cache entry or update it with the MN's current CoA.

The **mipv6_mn_invocation_process** function evaluates all of the invocations of the current **mipv6_mn** process as follows. It first checks the invocation mode; if the invocation mode is direct, which means the current **mipv6_mn** process (MN) has received a mobility message, the function will determine the type of received mobility message, i.e. **BA**, **HoT**, or **CoT**. This function then processes the message as follows:

- **BA**: Search for an entry in the Binding Update List that corresponds to the source address of the BA message. If no entry is found, ignore the BA message. Otherwise, check the status and process the BA message as follows:

  1. If the status is accepted, then (1) cancel the timer associated with this entry to stop the retransmitting of BU messages; and (2) update/remove the current entry from the Binding Update List according to whether this message is a response to a registration or deregistration process.
  2. If the status is rejected and the message is coming from an HA, then retry later.
  3. If the status is rejected and the message is coming from a CN, do not try to bind again with this CN.

- **HoT** or **CoT**: check if the source of this BA message already has an entry in the Binding Update List. If no entry is found, ignore the message. Otherwise, mark the correspondent routability test as completed and wait for the second test. When both tests are completed, cancel the event associated to route test retransmission and send a BU message to the source of this message.

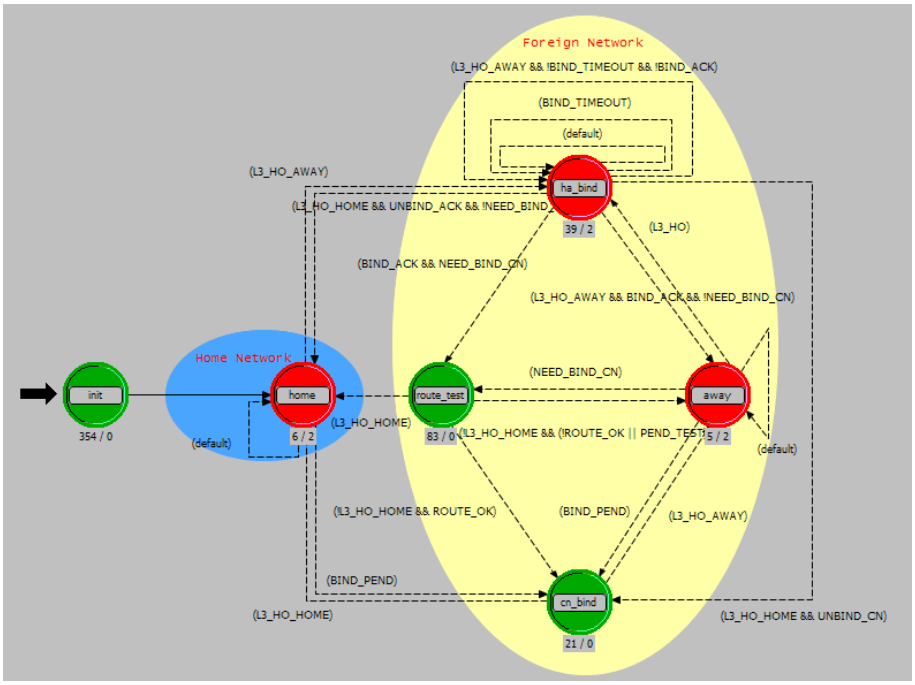

Figure C.3: mipv6_mn process model

When the invocation mode is direct, which means an interrupt has happened, the **mipv6_mn_invocation_process** function first determines the interrupt type, i.e. **layer3_handoff** interrupt, **start_binding** interrupt, **binding_update_retx_timer** interrupt, or **route_test_rtex_timer** interrupt. The function then processes the interrupt as follows:

- **layer3_handoff**: This interrupt means that the MN received a router advertisement message from a new router. The interrupt indicates that the MN has roamed to a different domain. This interrupt is processed by executing **layer3_handoff_process** function. This function first checks whether the new router is same as the current router. If they are the same, then it is a false alarm, hence ignore it. Otherwise, a new router is detected; then it checks

whether the MN returned to the home network or roamed to a foreign network. In the former case, start the deregistering procedure. In the latter case, obtain a CoA that corresponds to the new network prefix and start the registering procedure.

- **start_binding**: This interrupt means that the MN needs to start a binding procedure with a CN. The function obtains the address of the CN through the interrupt and starts the route optimization procedure with that CN.

- **binding_update_retx_timer**: This interrupt means that the binding update retransmission timer is expired; hence, the MN needs to retransmit a BU message. Therefore, the function obtains the address of the HA/CN through the interrupt and retransmits the BU message.

- **route_test_rtex_timer**: This interrupt means that the routability test retransmission timer is expired; hence, the MN needs to retransmit a HoTI and/or CoTI message. Therefore, the function obtains the address of the CN through the interrupt and retransmits the corresponding routability test initialization message.

C.3 CryptoSys Cryptography Toolkit

CryptoSys is a cryptographic toolkit that is developed by D.I. Management Services Pty Limited [94]. It provides a large library of cryptography tools for developers using VB6/VBA, C/C++, C#, VB.NET and VBScript/COM/ASP. It runs on all Win32 operating system: 95/98/NT4/2K/XP/2003/Vista/W7. A Linux and Win64 versions are also available.

CryptoSys includes two major products: CryptoSys API and CryptoSys PKI. The CryptoSys API provides symmetrical encryption. It provides: (1) four of the major block cipher algorithms: AES [64], DES [95], Triple DES [88] and Blowfish [96]; (2) a stream cipher algorithm compatible with RC4[97]; key wrap with AES and Triple DES; (3) secure message digest hash algorithms SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD2 and MD5; (4) the HMAC and CMAC message authentication algorithms; (5) data compression and decompression algorithms; (6) a secure random number generator; and (7) password-based key derivation function (PBKDF).

The CryptoSys PKI provides asymmetrical public key encryption and digital signatures. It provides: (1) RSA public and private keys generation; (2) X.509 certificates creation; (3) RSA public key encryption and signing; (4) encryption with symmetrical block ciphers Triple-DES and AES-128/192/256; (5) secure message digest hash algorithms SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD2 and MD5; (6) HMAC message authentication algorithms; and (7) a secure random number generator.

The author chooses CryptoSys because of its features as it has a large library of cryptography tools, runs on Win32 operating system, uses C/C++ programming language (compatible with OPNET), and is so easy to use.

C.4 OPNET Simulation Model Validation - EHR Model Debugging

```
HA's Address = [2005:0:0:9:0:0:0:1]
MN's HoA = [2005:0:0:9:C00:0:0:2]
-------------------------------------------------------------
Mobile IPv6 Mobility Detection: Moving away; MN performs layer-3 handoff process.
New router address = [3015:0:0:AA:0:0:0:1]

MN configures a new CoA using the symmetric CGA-based generation algorithm:
1- MN uses RNG_NonceDataHex to generate 128-bits fresh Modifier
    Modifier = 211DB539DCFDF210
2- Subnet Prefix of the foreign link = 3015:0:0:AA
3- Data1 = (Modifier || Supnet Prefix) = 211DB539DCFDF210301500AA
4- MN uses SHA1_Hmac to generate 160-bits Digest1
    Digest1 = f2723619a4def3d13ab922f9bbcb1fec376c97fe
5- First 64-bits of Digest1 = f2723619a4def3d1
6- MN sets M/S=1, H/C=0, U/L=0, and I/G=0 to perform interface identifier (ID)
    ID = f8723619a4def3d1
7- New CoA = [3015:0:0:AA:F872:3619:A4DE:F3D1]

MN sends a BU message to the HA located at address [2005:0:0:9:0:0:0:1]
    Value of Modifier included in the BU message is 211DB539DCFDF210
    Total Size of the BU message in bits is < 848 >
-------------------------------------------------------------
HA at address [2005:0:0:9:0:0:0:1] received a BU message from an MN
```

MN's HoA = [2005:0:0:9:C00:0:0:2]

MN's CoA = [3015:0:0:AA:F872:3619:A4DE:F3D1]

Value of Modifier received = 211DB539DCFDF210

HA verifies the CoA using the symmetric CGA-based verification algorithm:

1- Subnet Prefix of the received CoA = 3015:0:0:AA

2- Data1 = (Modifier || Supnet Prefix) = 211DB539DCFDF210301500AA

3- HA uses SHA1_Hmac to generate 160-bits Digest1

Digest1 = f2723619a4def3d13ab922f9bbcb1fec376c97fe

4- First 64-bits of Digest1 = f2723619a4def3d1

5- HA sets M/S=1, H/C=0 U/L=0, and I/G=0 to perform interface identifier (ID)

ID = f8723619a4def3d1

6- HA calculates the ID part of MN's CoA as [F872:3619:A4DE:F3D1]

The claimed CoA is valid, It passed the symmetric CGA-based verification algorithm.

HA uses SHA1_Hmac to generate 64-bits fresh CoT

1- Data2 = (MN's HoA || MN's CoA) = 2005009C00002301500AAF8723619A4DEF3D1

2- Digest2 = 915b8e1799890c0cdb92b64aac253c46d9809cdb

3- Generated CoT = First 64-bits of Digest2 = 915b8e1799890c0c

HA sets binding status as 'Accepted for limited period'

HA sets binding lifetime to MIN_BU_LIIFETIME

HA sends a BACoT message to MN at Address [3015:0:0:AA:F872:3619:A4DE:F3D1]

Value of CoT included in the BACoT is 915b8e1799890c0c

Value of index (I) included in the BACoT is 0

Total Size of the BACoT message in bits is < 672 >

\- -

MN at address [3015:0:0:AA:F872:3619:A4DE:F3D1] received a BACoT message from an HA

HA address [2005:0:0:9:0:0:0:1]

Status of the received BACoT messag is 'Accepted for limited period'

Value of CoT received = 915b8e1799890c0c

Value of index (I) received = 0

MN sends a BUCoT message to HA at address [2005:0:0:9:0:0:0:1]

Value of CoT included in the BUCoT is 915b8e1799890c0c

Value of index (I) included in the BUCoT is 0

Total Size of the BUCoT message in bits is < 800 >

\- -

HA at address [2005:0:0:9:0:0:0:1] received a BUCoT message from an MN

MN's HoA = [2005:0:0:9:C00:0:0:2]

MN's CoA = [3015:0:0:AA:F872:3619:A4DE:F3D1]

Value of CoT received = 915b8e1799890c0c

Value of index (I) received = 0

HA uses SHA1_Hmac to regenerate the CoT

Data2 = (MN's HoA || MN's CoA) = 2005009C00002301500AAF8723619A4DEF3D1

Digest2 = 915b8e1799890c0cdb92b64aac253c46d9809cdb

Regenerated CoT = First 64-bits of Digest2 = 915b8e1799890c0c

Regenerated CoT = Received CoT

HA sets binding status as 'Accepted'

HA sets binding lifetime to BU_LIIFETIME

HA sends a BA message to MN at Address [3015:0:0:AA:F872:3619:A4DE:F3D1]

Total Size of the BA message in bits is < 576 >

- -

MN at address [3015:0:0:AA:F872:3619:A4DE:F3D1] received a BA message from an HA

HA address [2005:0:0:9:0:0:0:1]

Status of the received BA message is 'Accepted'

Table C.1: Model debugging - EHR protocol

C.5 OPNET Simulation Model Validation - SK-based Model Debugging

HA's Address = [2005:0:0:9:0:0:0:1]

MN's HoA = [2005:0:0:9:C00:0:0:2]

CN's Address = [5035:0:0:8:0:0:0:2]

- -

Mobile IPv6 Mobility Detection: MN performs layer-3 handoff process.

New router address = [3015:0:0:AA:0:0:0:1]

- -

MN runs the enhanced home registration protocol (EHR) with HA as specified in Table C.1.

MN's current CoA = [3015:0:0:AA:C818:96A7:BC59:1ABD]

- -

MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] received traffic from CN located at address [5035:0:0:8:0:0:0:2] via HA tunnel.

1- MN uses RNG_NonceDataHex to generate 64 bit fresh cookie; cookie = 98148493.

2- MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] sends a CoTI mobility message to CN located at address [5035:0:0:8:0:0:0:2].

Value of cookie included in the message = < 98148493 >

Total Size of the message in bits = < 448 >

- -

CN at address [5035:0:0:8:0:0:0:2] received a CoTI mobility message from MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of cookie received in the message = < 98148493 >

1- CN uses SHA1_HMAC to generate 64 bit token; token = f8e9fcc1.

2- CN at address [5035:0:0:8:0:0:0:2] sends a CoT mobility message to MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of cookie included in the message = < 98148493 >

Value of token included in the message = < f8e9fcc1 >

Value of token_index included in the message = < 0 >

Total Size of the message in bits = < 512 >

- -

MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] received a CoT mobility message from CN located at address [5035:0:0:8:0:0:0:2]

Value of cookie received in the message = < 98148493 >

Value of token received in the message = < f8e9fcc1 >

Value of token_index received in the message = < 0 >

1- MN checks received cookie. The received and stored cookies are matched.

2- MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] sends a BReq mobility message to HA located at address [2005:0:0:9:0:0:0:1]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of CN's Address included in the message = < 5035:0:0:8:0:0:0:2 >

Value of token included in the message = < f8e9fcc1 >

Value of token_index included in the message = < 0 >

Value of sequence number included in the message = < 0 >

Total Size of the message in bits = < 928 >

- -

HA at address [2005:0:0:9:0:0:0:1] received a BReq mobility message from MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD].

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of CN's Address included in the message = < 5035:0:0:8:0:0:0:2 >

Value of token included in the message = < f8e9fcc1 >

Value of token_index included in the message = < 0 >

Value of sequence number included in the message = < 0 >

1- HA checks received HoA. There is a Binding Cache entry for that address, i.e. the HA is serving as a home agent for that MN's HoA.

2- HA checks received CoA. The MN claimed CoA matches with its CoA registered at the HA.

3- HA uses RNG_NonceDataHex to generate 64 bit nonce N_{HA1}; N_{HA1} = 256FCD39.

4- HA uses RNG_NonceDataHex to generate 64 bit nonce N_{HA2}; N_{HA2} = 10CC448A.

5- HA uses SHA2_HMAC to generate 256 bit K_{BC1} and K_{BC2} keys

K_{BC1} = 0B1268C5430A7A8145E17E5C21C092D9

K_{BC2} = 4498942CF8BDBCBF721259E2EAC1B9BF

6- HA at address [2005:0:0:9:0:0:0:1] sends a BRep mobility message to MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of CN's Address included in the message = < 5035:0:0:8:0:0:0:2 >

Value of sequence number included in the message = < 0 >

Value of K_{BC1} included in the message = < 0B1268C5430A7A8145E17E5C21C092D9 >

Total Size of the message in bits = < 944 >

7- HA at address [2005:0:0:9:0:0:0:1] sends an EBC mobility message to CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of N_{HA1} included in the message = < 256FCD39 >

Value of N_{HA2} included in the message = < 10CC448A >

Value of sequence number included in the message = < 0 >

Value of token included in the message = < f8e9fcc1 >

Value of token_index included in the message = < 0 >

Value of $MAC_{K_{BC2}}$ (EBC) included in the message = < 5AFC9A310F9A >

Total Size of the message in bits = < 848 >

- -

MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] received a BRep mobility message from HA located at address [2005:0:0:9:0:0:0:1]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of CN's Address included in the message = < 5035:0:0:8:0:0:0:2 >

Value of sequence number included in the message = < 0 >

Value of K_{BC1} included in the message = < 0B1268C5430A7A8145E17E5C21C092D9 >

- -

CN at address [5035:0:0:8:0:0:0:2] received an EBC mobility message from HA located at address [2005:0:0:9:0:0:0:1]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of N_{HA1} included in the message = < 256FCD39 >

Value of N_{HA2} included in the message = < 10CC448A >

Value of sequence number included in the message = < 0 >

Value of token included in the message = < f8e9fcc1 >

Value of token_index included in the message = < 0 >

Value of $MAC_{K_{BC2}}$ (EBC) included in the message = < 5AFC9A310F9A >

1- CN checks received token. The received and regenerated tokens are matched.

2- CN uses SHA2_HMAC to generate 256 bit K_{BC1} and K_{BC2} keys

K_{BC1} = 0B1268C5430A7A8145E17E5C21C092D9

K_{BC2} = 4498942CF8BDBCBF721259E2EAC1B9BF

3- CN checks received $MAC_{K_{BC2}}$ (EBC). The received and calculated MACs are matched.

4- CN uses RNG_NonceDataHex to generate 256 bit fresh session key (K_{MN-CN}).

K_{MN-CN} = 1423F477EB588C9777D69F48046E0BB4

5- CN encrypts key K_{MN-CN} using AES256_HexMode_CBC symmetric encryption with K_{BC1}.

$ENC_{K_{BC1}}$ [K_{MN-CN}] = < 6BDF4439DF4E4C250B31D03AC3AEAD1B >

6- CN at address [5035:0:0:8:0:0:0:2] sends an EBA mobility message to MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 0 >

Value of $ENC_{K_{BC1}}$ [K_{MN-CN}] included = < 6BDF4439DF4E4C250B31D03AC3AEAD1B >

Value of $MAC_{K_{MN-CN}}$ (EBA) included in the message = < 842A53F8DF4A >

Total Size of the message in bits = < 912 >

- -

MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] received an EBA mobility message from CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 0 >

Value of $ENC_{K_{BC1}}$ [K_{MN-CN}] included = < 6BDF4439DF4E4C250B31D03AC3AEAD1B >

Value of $MAC_{K_{MN-CN}}$ (EBA) included in the message = < 842A53F8DF4A >

1- MN decrypts value of $ENC_{K_{BC1}}$ [K_{MN-CN}] received using AES256_HexMode_CBC symmetric decryption with K_{BC1}. K_{MN-CN} = < 1423F477EB588C9777D69F48046E0BB4 >

2- MN checks received $MAC_{K_{MN-CN}}$ (EBA). The received and calculated MACs are matched.

3- MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] sends a BCC mobility message to CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of sequence number included in the message = < 1 >

Value of $MAC_{K_{MN-CN}}$ (BCC) included in the message = < 30A72B6C9331 >

Total Size of the message in bits = < 810 >

- -

CN at address [5035:0:0:8:0:0:0:2] received a BCC mobility message from MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 3015:0:0:AA:C818:96A7:BC59:1ABD >

Value of sequence number included in the message = < 1 >

Value of $MAC_{K_{MN-CN}}$ (BCC) included in the message = < 30A72B6C9331 >

1- CN checks received $MAC_{K_{MN-CN}}$ (BCC). The received and calculated MACs are matched

2- CN at address [5035:0:0:8:0:0:0:2] sends a BA mobility message to MN located at address [3015:0:0:AA:C818:96A7:BC59:1ABD]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 1 >

Value of $MAC_{K_{MN-CN}}$ (BA) included in the message = < 0D5D29E76552 >

Total Size of the message in bits = < 576 >

- -

MN at address [3015:0:0:AA:C818:96A7:BC59:1ABD] received a BA mobility message from CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 1 >

Value of $MAC_{K_{MN-CN}}$ (BA) included in the message = < 0D5D29E76552 >

1- MN checks received $MAC_{K_{MN-CN}}$ (BA). The received and calculated MACs are matched.

- -

- -

Mobile IPv6 Mobility Detection: MN performs layer-3 handoff process.

New router address = [4025:0:0:12:0:0:0:1]

- -

MN runs the enhanced home registration protocol (EHR) with HA as specified in Table C.1.

MN's current CoA = [4025:0:0:12:E852:5EAF:F52D:5C00]

- -

1- MN uses SHA1_HMAC to generate K_{BM} key; K_{BM} = EB3F874922BBBF9B5AB8

2- MN calculates Authenticator using AES256_HexMode_CBC symmetric encryption with K_{MN-HA}. Authenticator =

< CEFA1CC8B8F5C55A2E991C982AFF39A91953B2BEDB9E757F985F32C568AAB514 >

3- MN at address [4025:0:0:12:E852:5EAF:F52D:5C00] sends a BU mobility message to CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of sequence number included in the message = < 2 >

Value of Authenticator included in the message =

< CEFA1CC8B8F5C55A2E991C982AFF39A91953B2BEDB9E757F985F32C568AAB514 >

Value of $MAC_{K_{BM}}$ (BU) included in the message = < 8FD8FF720AC8 >

Total Size of the message in bits = < 1072 >

- -

CN at address [5035:0:0:8:0:0:0:2] received a BU mobility message from MN located at address [4025:0:0:12:E852:5EAF:F52D:5C00]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of sequence number included in the message = < 2 >

Value of Authenticator included in the message =

< CEFA1CC8B8F5C55A2E991C982AFF39A91953B2BEDB9E757F985F32C568AAB514 >

Value of $MAC_{K_{BM}}$ (BU) included in the message = < 8FD8FF720AC8 >

1- CN uses SHA1_HMAC to generate K_{BM} key; K_{BM} = EB3F874922BBBF9B5AB8

2- CN checks received $MAC_{K_{BM}}$ (BU). The received and calculated MACs are matched

3- CN uses RNG_NonceDataHex to generate 64 bit nonce N_{CN}; N_{CN} = 121B70BD

4- CN uses SHA1_HMAC to generate K_{BC} key; K_{BC} = F4E0992F4CD9482FD082

5- CN at address [5035:0:0:8:0:0:0:2] sends a BCReq mobility message to HA located at address [2005:0:0:9:0:0:0:1]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of Authenticator included in the message =

< CEFA1CC8B8F5C55A2E991C982AFF39A91953B2BEDB9E757F985F32C568AAB514 >

Value of N_{CN} included in the message = < 121B70BD >

Value of $MAC_{K_{BC}}$ (BCReq) included in the message = < 2329B8159BB2 >

Total Size of the message in bits = < 960 >

- -

HA at address [2005:0:0:9:0:0:0:1] received a BCReq mobility message from CN located at address [5035:0:0:8:0:0:0:2].

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of Authenticator included in the message =

< CEFA1CC8B8F5C55A2E991C982AFF39A91953B2BEDB9E757F985F32C568AAB514 >

Value of N_{CN} included in the message = < 121B70BD >

Value of $MAC_{K_{BC}}$ (BCReq) included in the message = < 2329B8159BB2 >

1- HA checks received HoA. There is a Binding Cache entry for that address, i.e. the HA is serving as a home agent for that MN's HoA.

2- HA checks received CoA. The MN claimed CoA matches with its CoA registered at the HA.

3- HA uses SHA1_HMAC to generate K_{BC} key; K_{BC} = F4E0992F4CD9482FD082

4- HA checks received $MAC_{K_{BC}}$ (BCReq). The received and calculated MACs are matched

5- HA decrypts value of Authenticator received using AES256_HexMode_CBC symmetric decryption with K_{MN-HA}.

Decrypted value = 2005009C0000240250012E8525EAFF52D5C005035008000208080808080808

6- HA at address [2005:0:0:9:0:0:0:1] sends a BCRep mobility message to CN located at address [5035:0:0:8:0:0:0:2].

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of N_{CN} included in the message = < 121B70BD >

Value of $MAC_{K_{BC}}$ (BCRep) included in the message = < C22BBE8A4072 >

Total Size of the message in bits = < 688 >

- -

CN at address [5035:0:0:8:0:0:0:2] received a BRep mobility message from HA located at address [2005:0:0:9:0:0:0:1]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 4025:0:0:12:E852:5EAF:F52D:5C00 >

Value of N_{CN} included in the message = < 121B70BD >

Value of $MAC_{K_{BC}}$ (BCRep) included in the message = < C22BBE8A4072 >

1- CN checks received $MAC_{K_{BC}}$ (BCRep). The received and calculated MACs are matched

2- CN at address [5035:0:0:8:0:0:0:2] sends a BA mobility message to MN located at address [4025:0:0:12:E852:5EAF:F52D:5C00]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 2 >

Value of $MAC_{K_{BM}}$ (BA) included in the message = < BED4F31F2D64 >

Total Size of the message in bits = < 640 >

- -

MN at address [4025:0:0:12:E852:5EAF:F52D:5C00] received a BA mobility message from CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 2 >

Value of $MAC_{K_{BM}}$ (BA) included in the message = < BED4F31F2D64 >

1- MN checks received $MAC_{K_{BM}}$ (BA). The received and calculated MACs are matched

- -

- -

Mobile IPv6 Mobility Detection: MN performs layer-3 handoff process.

New router address = [2005:0:0:9:0:0:0:1]

- -

MN runs the enhanced home registration protocol (EHR) with HA as specified in Table C.1.

- -

1- MN uses SHA1_HMAC to generate K_{BM} key; K_{BM} = < FBFF6D9E461A48E1D3DE >

2- MN at address [2005:0:0:9:C00:0:0:2] sends a BU mobility message to CN located at address [5035:0:0:8:0:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 3 >

Value of $MAC_{K_{BM}}$ (BU) included in the message = < 57004BE9D252 >

Total Size of the message in bits = < 800 >

- -

CN at address [5035:0:0:8:0:0:0:2] received a BU mobility message from MN located at address [2005:0:0:9:C00:0:0:2]

Value of MN's HoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of MN's CoA included in the message = < 2005:0:0:9:C00:0:0:2 >

Value of sequence number included in the message = < 3 >

Value of $MAC_{K_{BM}}$ (BU) included in the message = < 57004BE9D252 >

1- CN uses SHA1_HMAC to generate K_{BM} key; K_{BM} = < FBFF6D9E461A48E1D3DE >

2- CN checks received $MAC_{K_{BM}}$ (BU). The received and calculated MACs are matched

3- CN at address [5035:0:0:8:0:0:0:2] sends a BA mobility message to MN located at address [2005:0:0:9:C00:0:0:2]

Value of MN's HoA included in the message = $<$ 2005:0:0:9:C00:0:0:2 $>$
Value of sequence number included in the message = $<$ 3 $>$
Value of MACKBM (BA) included in the message = $<$ C7187F29D67B $>$
Total Size of the message in bits = $<$ 480 $>$
- -
MN at address [2005:0:0:9:C00:0:0:2] received a BA mobility message from CN located at address [5035:0:0:8:0:0:0:2]
Value of MN's HoA included in the message = $<$ 2005:0:0:9:C00:0:0:2 $>$
Value of sequence number included in the message = $<$ 3 $>$
Value of MACKBM (BA) included in the message = $<$ C7187F29D67B $>$
1- MN checks received $MAC_{K_{BM}}$ (BA). The received and calculated MACs are matched

Table C.2: Model debugging - SK-based protocol

Appendix D

Proposed Correspondent Registration Protocols

This appendix presents a more detailed description of the proposed correspondent registration protocols mentioned in Chapter 5.

D.1 Detailed CRE-SK Phase Description

Step S1-SK
The MN initiates the creation phase by creating a Binding Update List entry for the CN and setting the entry in a Route_Pending state. The MN then sends message **M1-SK** to the CN (as shown in Figure D.1) requesting a care-of keygen token. The MN includes a 64-bit random number (Cookie) in the message. The MN stores the Cookie at the list entry for the CN and compares it later with the response from that CN; the Cookie is used to protect the MN against replay attacks.

If the MN does not receive a matching response within a retransmission interval of one second, the MN will resend the message with a new Cookie value. The MN doubles the retransmission interval upon each retransmission in the same way, as specified in the base specification of the MIPv6 protocol ([5], see also Section 2.5).

Step S2-SK
Upon receiving message **M1-SK**, the CN uses its own secret (K_{CN}) to generate a care-of keygen token ($Token_2$) as in the RR procedure (see Section 2.5). The CN

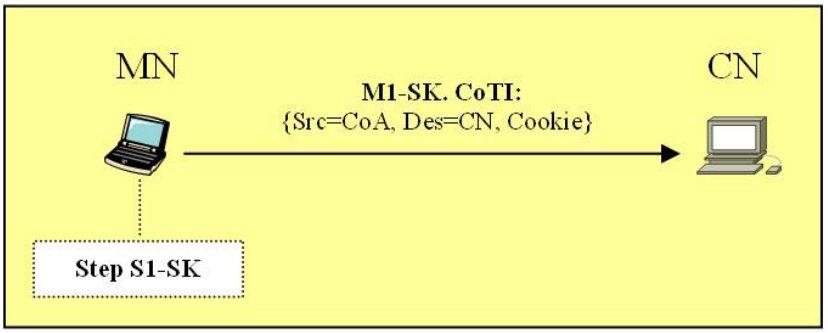

Figure D.1: Step S1-SK and message M1-SK (CoTI)

then sends the Cookie, J, and Token_2 to the MN's CoA in message **M2-SK** (as shown in Figure D.2), where J is an index that identifies the care-of nonce (N_J) used in generating Token_2. Token_2 will be returned to the CN later to prevent replay attacks. The CN generates a new nonce N_J at regular intervals, which allows the CN to check the freshness of any received token; if the token enclosed in a message is generated using a fresh nonce, then the message is fresh.

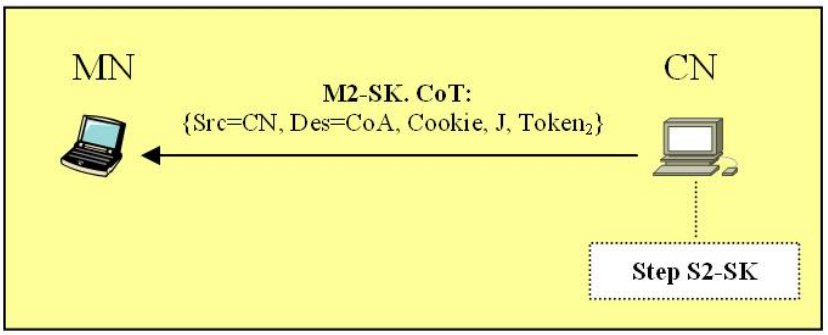

Figure D.2: Step S2-SK and message M2-SK (CoT)

Step S3-SK

When the MN receives message **M2-SK**, it uses the CN's address as an index to search its Binding Update List. If a list entry is found with a Route_Pending status, the MN will perform **Verification MN1-SK** to confirm that the received Cookie is identical to the Cookie that was sent in message **M1-SK**.

Verification MN1-SK: The MN verifies that Cookie' equals Cookie, where Cookie' is enclosed in message **M2-SK** and Cookie is stored at the matched

list entry in the MN. This verification is to detect replay attacks.

If no list entry is found, if the matched list entry is not in a Route_Pending status, or if **Verification MN1-SK** fails, the MN will discard message **M2-SK** without any further action. Otherwise, the MN will change the status of the list entry to Creation_Pending indicating that the return of a session key and an acknowledgement are expected. Then, the MN will send CoA, LT_{BRem}, CN, Seq, LT_{BReq}, J, and $Token_2$ to the HA in message **M3-SK** (as shown in Figure D.3) requesting binding creation with a specific CN. LT_{BRem} is the current remaining lifetime for the binding of HoA and CoA at both the MN and the HA. LT_{BRem} is used as a timestamp to protect the HA against replay attacks. The value of CN enclosed in the message is to inform the HA of the CN's address. Seq and LT_{BReq} are the initial sequence number and binding lifetime request, respectively. It is recommended that the MN requests the maximum permitted binding lifetime to reduce the number of redundant binding refreshes. Message **M3-SK** is sent via an IPSec ESP secure tunnel; it is encrypted using the secret key K_{MN-HA} shared between the MN and the HA.

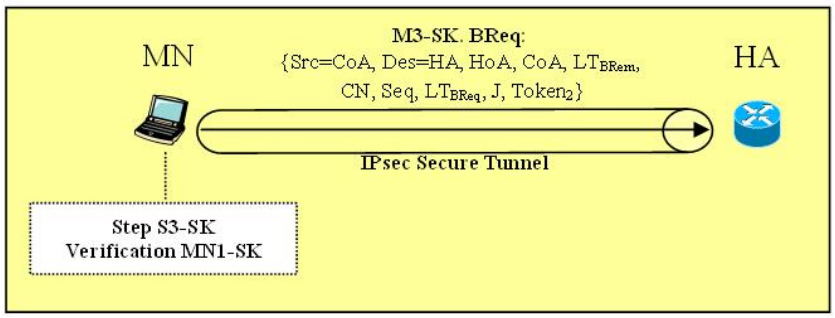

Figure D.3: Step S3-SK and message M3-SK (BReq)

Step S4-SK

Upon receiving message **M3-SK**, the HA uses the MN's HoA as an index to search its Binding Cache. If a cache entry is found, the HA will decrypt the message using key K_{MN-HA}, and then perform **Verification HA1-SK** (as shown in Figure D.4) to confirm the freshness of the message.

Verification HA1-SK: The HA checks if (LT_{BRem} - t_{valid}) $\leq LT_{BRem}$' $\leq (LT_{BRem}$

+ t_{valid}), where LT_{BRem}' is enclosed in message **M3-SK**; LT_{BRem} is the remaining binding lifetime stored locally at HA in Binding Cache entry for the HoA when message **M3-SK** is received; and t_{valid} is the validity period agreed upon priorly between the MN and HA (Assumption **A3**). This verification is to detect replay attacks.

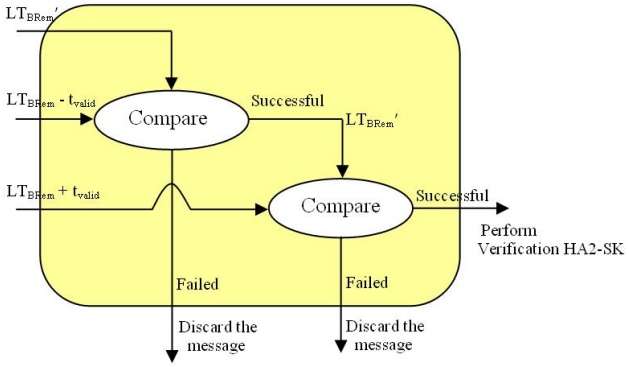

Figure D.4: Verification HA1-SK

If **Verification HA1-SK** fails (or if no cache entry is found), the HA will discard the message without any further action. Otherwise, the HA will perform **Verification HA2-SK** to confirm the correctness of the claimed care-of address.

Verification HA2-SK: The HA checks if CoA' equals CoA, where CoA' is the MN's care-of address enclosed in message **M3-SK** and CoA is the MN's care-of address stored locally at HA in the Binding Cache entry for the HoA enclosed in the message.

A positive outcome of **Verification HA2-SK** assures the HA that the MN is not cheating the CN with a fake CoA. If the verification fails, the HA will send a binding error message to the MN indicating that the claimed CoA is not yet registered at the HA, and thus, the HA cannot help registering it at the CN. If both **Verifications HA1-SK** and **HA2-SK** are successful, the HA will perform **Verification HA3-SK** to confirm that the binding lifetime requested by the MN (LT_{BReq}) is not greater than the remaining lifetime (LT_{BRem}) for the binding of HoA and CoA at the HA.

Verification HA3-SK: The HA checks if $LT_{BReq} \leq LT_{BRem}$, where LT_{BReq} is the binding lifetime request enclosed in message **M3-SK** and LT_{BRem} is the remaining binding lifetime stored locally at HA in the Binding Cache entry for the HoA when message **M3-SK** is received. This verification is to ensure that the MN is not requesting a binding lifetime greater than the remaining binding lifetime at the HA. If the verification fails, the HA will use value of LT_{BRem} instead of LT_{BReq} in message **M5-SK** sent to the CN.

After **Verification HA3-SK**, the HA generates two keys (K_{BC1} and K_{BC2}) based on the secret key shared with the CN (K_{HA-CN}) and two fresh nonces (N_{HA1} and N_{HA2}); K_{BC1} = HMAC_SHA1 (K_{HA-CN}, (HoA || CoA || CN || N_{HA1}) and K_{BC2} = HMAC_SHA1 (K_{HA-CN}, (HoA || CoA || CN || N_{HA2}). The HA then sends the CN, Seq, and K_{BC1} to the MN in message **M4-SK** via an IPSec ESP secure tunnel (as shown in Figure D.5). At the same time, the HA sends the CoA, id_{HA}, N_{HA1}, N_{HA2}, Seq, LT_{BReq}, J, $Token_2$, and $MAC_{K_{BC2}}$ (EBC) to the CN in message **M5-SK** (as shown in Figure D.6), where:

- id_{HA} is the home link's public identity, for example the URL of the home link's website or the 64-bit subnet prefix of the home link's address.
- $MAC_{K_{BC2}}$ (EBC) is a keyed hash value used to ensure the integrity and authenticity of message **M5-SK**; $MAC_{K_{BC2}}$ (EBC) = First (96, HMAC_SHA1 (K_{BC2}, (HoA || CoA || id_{HA} || N_{HA1} || N_{HA2} || Seq || LT_{BReq}))).

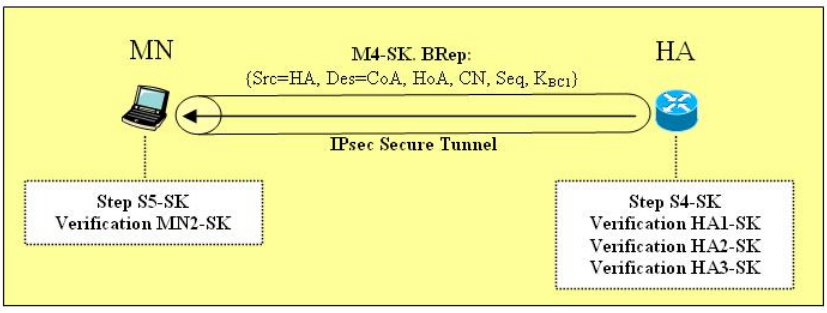

Figure D.5: Step S4-SK, message M4-SK (BRep), and Step S5-SK

Step S5-SK

When the MN receives message **M4-SK**, it decrypts the message using key K_{MN-HA}, and then uses the CN's address as an index to search its Binding

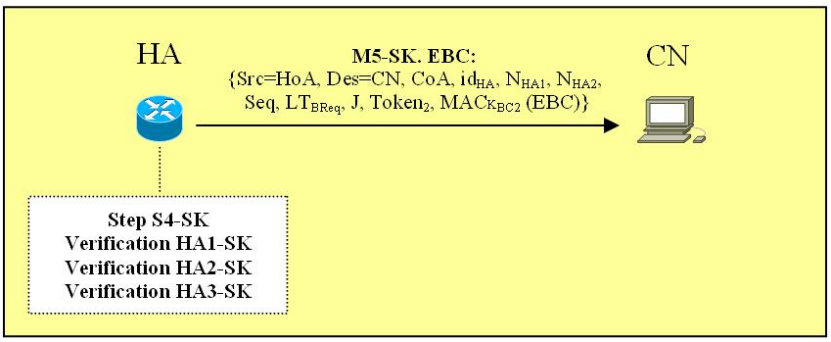

Figure D.6: Step S4-SK and message M5-SK (EBC)

Update List. If a list entry is found with a Creation_Pending status, the MN will perform **Verification MN2-SK** to confirm the freshness of the message.

Verification MN2-SK: The MN verifies that Seq' equals Seq where Seq' is the sequence number enclosed in the message and Seq is the sequence number stored at the matched list entry in the MN. This verification is to detect replay attacks.

If no list entry is found, if the matched list entry is not in a Creation_Pending status, or if **Verification MN2-SK** fails, the MN will discard message **M4-SK** without any further action. Otherwise, the MN will store key K_{BC1} enclosed in the message at the matched list entry.

Step S6-SK

When the CN receives message **M5-SK**, it performs **Verification CN1-SK** (as shown in Figure D.7) to confirm the freshness of the message and reachability of the MN at the CoA.

Verification CN1-SK: The CN verifies that $Token_2$ equals First (64, HMAC_SHA1 (K_{CN}, (CoA || N_J || 1))), where $Token_2$ and CoA are enclosed in message **M5-SK**, and the value of N_J is retrieved based on value of J enclosed in the message. This verification is to detect replay attacks.

If **Verification CN1-SK** fails, the CN will discard message **M5-SK** without any further action. Otherwise, the CN will generate K_{BC1} and K_{BC2} based on its address, the secret key shared with the home link (K_{HA-CN}), and the items

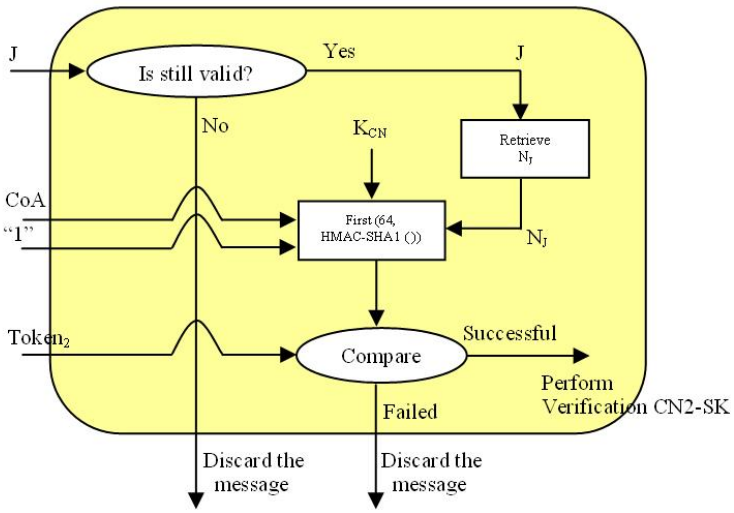

Figure D.7: Verification CN1-SK

enclosed in the message (HoA, CoA, N_{HA1}, and N_{HA2}). The CN then performs **Verification CN2-SK** (as shown in Figure D.8) to confirm the integrity and authenticity of message **M5-SK**.

Verification CN2-SK: The CN verifies that $MAC_{K_{BC2}}$ (EBC) equals First (96, HMAC_SHA1 (K_{BC2}, (HoA || CoA || id_{HA} || N_{HA1} || N_{HA2} || Seq || LT_{BReq}))), where $MAC_{K_{BC2}}$ (EBC), HoA, CoA, id_{HA}, N_{HA1}, N_{HA2}, Seq, and LT_{BReq} are enclosed in message **M5-SK**.

A positive outcome of **Verification CN2-SK** assures the CN that message **M5-SK** is coming from the home link of the MN and has not been altered in transit. Otherwise, i.e. if **Verification CN2-SK** fails, the CN will discard the message. After successful verifications, the CN first generates a fresh session key K_{MN-CN}. The CN then creates a Binding Cache entry, and stores the binding between the MN's HoA and CoA as well as the values of Seq, LT_{BReq}, and K_{MN-CN} at the entry. In addition, the CN sets the granted binding lifetime (LT_{BGrant}) between the MN's HoA and CoA to a MIN_BINDING_LIFETIME value to handle the case of the MN roaming to a new CoA while the HA and the CN are running the protocol. The CN also sends subsequent packets destined for the MN to its new CoA for as long as the lifetime is granted for the binding. Finally, the CN sends Seq, LT_{BGrant}, $ENC_{K_{BC1}}$ [K_{MN-CN}], and $MAC_{K_{MN-CN}}$ (EBA) to the MN

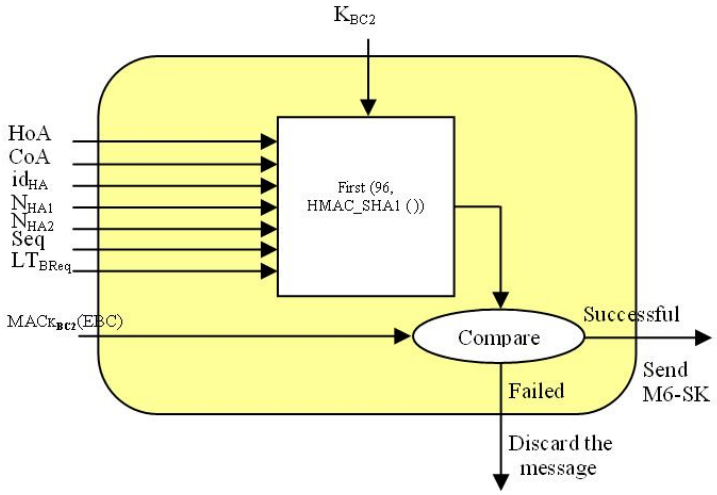

Figure D.8: Verification CN2-SK

in **message M6-SK** (as shown in Figure D.9) for acknowledging the binding of the CoA, where:

- $\text{ENC}_{K_{BC1}}$ [K_{MN-CN}] is the encryption of key K_{MN-CN} using key K_{BC1}.
- $\text{MAC}_{K_{MN-CN}}$ (EBA) is a keyed hash value used to ensure the integrity and authenticity of message **M6-SK**; $\text{MAC}_{K_{MN-CN}}$ (EBA) = First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || CN || Seq || LT_{BGrant}))).

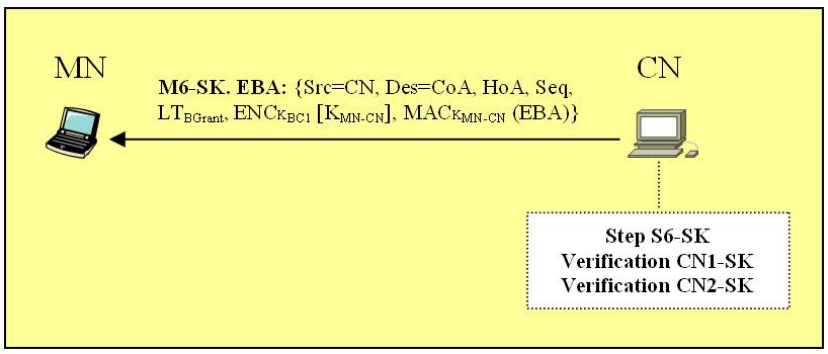

Figure D.9: Step S6-SK and message M6-SK (EBA)

Step S7-SK

When the MN receives message **M6-SK**, it uses the CN's address as an index to

search its Binding Update List. If a list entry is found, the MN will repeat **Verification MN2-SK** (see **Step S5-SK**) to confirm the freshness of the message.

After **Verification MN2-SK**, the MN will decrypt the coded session key enclosed in message **M6-SK** using K_{BC1}; $DEC_{K_{BC1}}$ [$ENC_{K_{BC1}}$ [K_{MN-CN}]] = {K_{MN-CN}}. The MN then performs **Verification MN3-SK** (as shown in Figure D.10) to confirm the integrity and authenticity of the message.

Verification MN3-SK: The MN verifies that $MAC_{K_{MN-CN}}$ (EBA) equals First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || CN || Seq || LT_{BGrant}))), where $MAC_{K_{MN-CN}}$ (EBA), HoA, CoA, CN, Seq, and LT_{BGrant} are enclosed in message **M6-SK**.

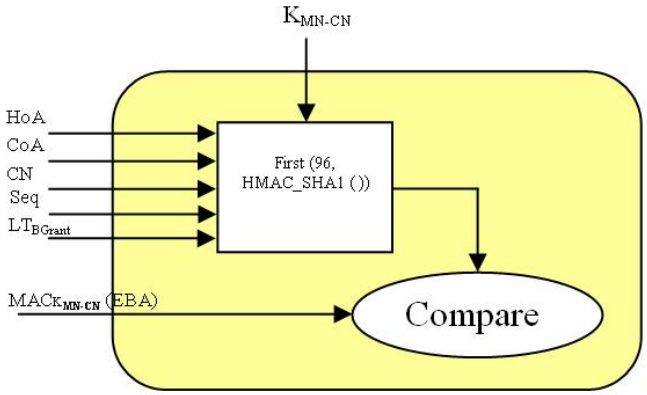

Figure D.10: Verification MN3-SK

If **Verification MN2-SK** or **MN3-SK** fails (or if no list entry is found), the MN will discard message **M6-SK** without any further action. Otherwise, the MN will update the value of K_{BC1} with the value of K_{MN-CN} and will store the value of LT_{BGrant} at the matched list entry. As LT_{BGrant} indicates a small binding lifetime, the MN sends the HoA, CoA, Seq_{new}, LT_{BReq}, Ack, and $MAC_{K_{MN-CN}}$ (BCC) to the CN in message **M7-SK** (as shown in Figure D.11), requesting a greater binding lifetime, confirming the receipt of K_{MN-CN}, and showing the existence at the CoA, where:

- Ack is an acknowledge bit that may be set to request the return of a binding acknowledgement message.

- Seq_{new} is a new sequence number that is greater than the Seq sent by the MN in message **M3-SK**.
- $\text{MAC}_{K_{MN-CN}}$ (BCC) is a keyed hash value used to ensure the integrity and authenticity of message **M7-SK**; $\text{MAC}_{K_{MN-CN}}$ (BCC) = First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || Seq_{new} || LT_{BReq} || Ack))).

The MN also changes the status of the list entry. Specifically, if the MN sets the Ack bit, then it will change the status to a Binding_Pending indicating that it is waiting for an acknowledgement; otherwise, it will change the status to a Binding_Complete indicating the binding creation is complete.

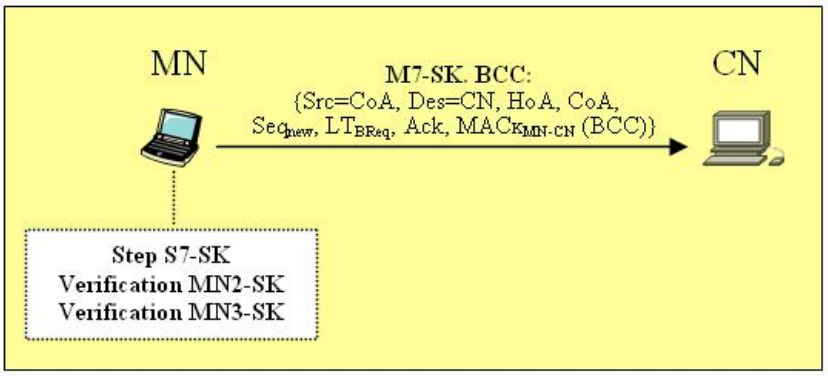

Figure D.11: Step S7-SK and message M7-SK (BCC)

Step S8-SK

Upon receiving message **M7-SK**, the CN uses the HoA enclosed in the message to find a matched cache entry. The CN then performs **Verification CN3-SK** to confirm the freshness of the message.

Verification CN3-SK: The CN verifies that Seq_{new} is greater than Seq, where Seq_{new} is the sequence number enclosed in the message and Seq is the sequence number stored at the matched cache entry in the CN (received in message **M5-SK**). This verification is to detect replay attacks.

After a successful verification, the CN next performs **Verification CN4-SK** (as shown in Figure D.12) to confirm the integrity and authenticity of the message.

Verification CN4-SK: The CN verifies that $MAC_{K_{MN-CN}}$ (BCC) equals First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || Seq_{new} || LT_{BReq} || Ack))), where $MAC_{K_{MN-CN}}$ (BCC), HoA, CoA, Seq_{new}, LT_{BReq}, and Ack are enclosed in message **M7-SK**.

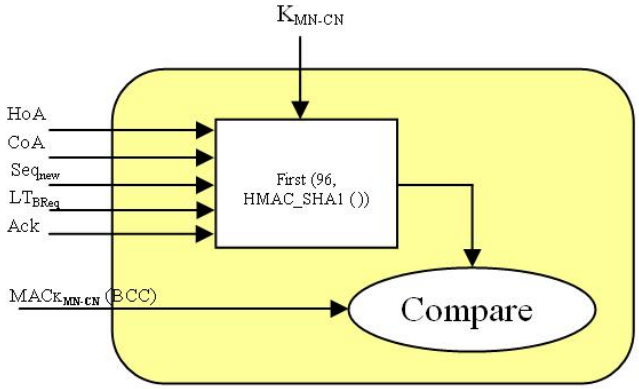

Figure D.12: Verification CN4-SK

If **Verification CN3-SK** or **CN4-SK** fails, the CN will discard the message without any further action. Otherwise, the CN will perform **Verification CN5-SK** to confirm that the binding lifetime requested by the MN (LT_{BReq}) is not greater than the binding lifetime request received in message **M5-SK**.

Verification CN5-SK: The CN checks if LT_{BReq}' $\leq LT_{BReq}$, where LT_{BReq}' is the binding lifetime request enclosed in message **M7-SK** and LT_{BReq} is the binding lifetime request stored at the CN from message **M5-SK**. If the verification fails, the CN will use the value of LT_{BReq} instead of LT_{BReq}' while setting the value of granted binding lifetime (LT_{BGrant}).

After **Verification CN5-SK**, the CN updates the matched cache entry by storing the value of Seq_{new} enclosed in the message and by setting the granted binding lifetime (LT_{BGrant}) to a value that is less than or equal to LT_{BReq}' (or LT_{BReq}). Finally, the CN checks the Ack bit enclosed in message **M7-SK**. If the MN has requested an acknowledgement, the CN will send Seq_{new}, LT_{BGrant}, and $MAC_{K_{MN-CN}}$ (BA) to the MN in message **M8-SK** (as shown in Figure D.13) to acknowledge the binding of the CoA, where:

- $MAC_{K_{MN-CN}}$ (BA) is a keyed hash value used to ensure the integrity and

authenticity of message **M8-SK**; $\text{MAC}_{K_{MN-CN}}$ (BA) = First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || CN || Seq_{new} || LT_{BGrant}))).

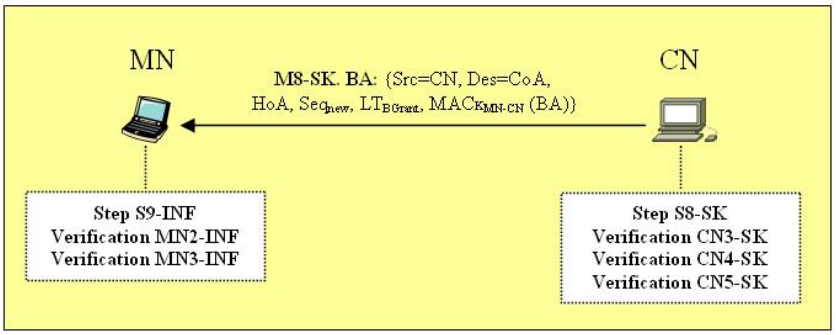

Figure D.13: Step S8-SK, message M8-SK (BA), and Step S9-SK

Step S9-SK

Upon receiving message **M8-SK**, the MN uses the CN's address to find a matched list entry, and then confirms the freshness of the message by checking the value of Seq_{new} enclosed in it, as in **Verification MN2-SK**. After a successful verification, the MN performs **Verification MN4-SK** (as shown in Figure D.14) to confirm the authenticity of the message.

Verification MN4-SK: The MN verifies that $\text{MAC}_{K_{MN-CN}}$ (BA) equals First (96, HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || CN || Seq_{new} || LT_{BGrant}))), where $\text{MAC}_{K_{MN-CN}}$ (BA), HoA, CoA, CN, Seq_{new}, and LT_{BGrant} are enclosed in message **M8-SK**.

If no matched list entry is found for that CN, or if **Verification MN2-SK** or **MN4-SK** fails, the MN will discard the message without any further action. Otherwise, the MN will update the status of the list entry to Binding_Complete indicating that the binding has been acknowledged. In addition, the MN adjusts the remaining binding lifetime depending on the given granted binding lifetime. The creation phase for the SK-based protocol in the stationary CN case is now complete.

If the MN continues to receive data from the CN via its home link, the MN will restart the creation phase by running **Step S1-SK** and sending message **M1-SK** to the CN. If the creation retries fail, the MN will stop the SK-based protocol

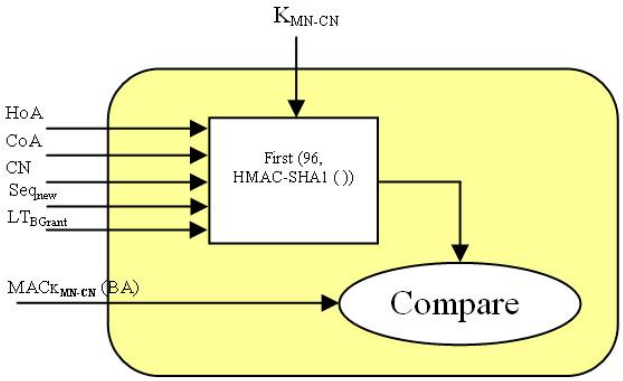

Figure D.14: Verification MN4-SK

and record in its Binding Update List entry that future binding creations should not be tried with this CN. However, such a list entry is removed after a period of time in order to allow for retrying route optimization as specified in the base specification of the MIPv6 protocol [5].

D.2 Detailed CRE-PK Phase Description

Steps S1-PK, **S2-PK**, and **S3-PK**
Steps S1-PK, **S2-PK**, and **S3-PK** are identical, respectively, to **Steps S1-SK**, **S2-SK**, and **S3-SK** mentioned in Section D (see Figures D.1 and D.2), but the MN sends the value of K_{BM} instead of $Token_2$ to the HA in message **M3-PK** (as shown in Figure D.15). Specifically, when the MN receives message **M2-PK**, it hashes $Token_2$ to generate a binding management key K_{BM}, i.e. K_{BM} = SHA1 ($Token_2$). The MN then includes K_{BM} instead of $Token_2$ in message **M3-PK** sent to the HA.

Step S4-PK
The first part of **Step S4-PK** is identical to the first part of **Step S4-SK**. Specifically, upon receiving message **M3-PK**, the HA uses the MN's HoA as an index to search its Binding Cache. If a cache entry is found, the HA will decrypt the message using key K_{MN-HA}, and then perform **Verifications HA1-PK**, **HA2-PK**, and **HA3-PK**, which are identical to **Verifications HA1-SK**, **HA2-SK**, and **HA3-SK**, respectively. **Verification HA1-PK** is to confirm the freshness of message **M3-PK**. **Verification HA2-PK** is to confirm the correctness of the

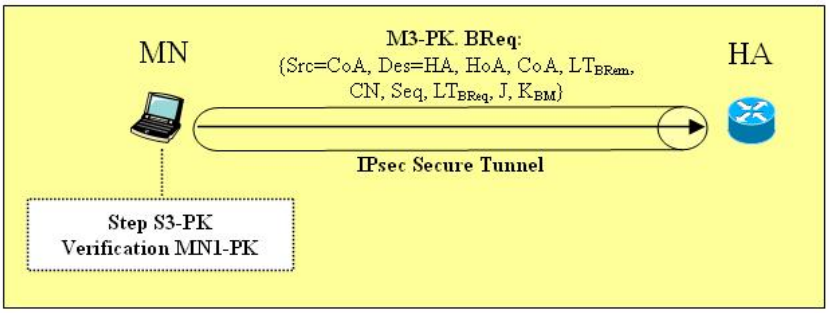

Figure D.15: Step S3-PK and message M3-PK (BReq)

claimed care-of address. **Verification HA3-PK** is to confirm that the binding lifetime requested by the MN (LT_{BReq}) is not greater than the remaining lifetime (LT_{BRem}) for the binding of HoA and CoA at the HA.

After successful verifications, the HA sends the CoA, N_{HA}, Seq, LT_{BReq}, J, $MAC_{K_{BM}}$ (EBC), SIG_H, and $Cert_H$ to the CN in message **M4-PK** (as shown in Figure D.16) where:

- N_{HA} is a fresh nonce to protect the HA against replay attacks.
- $MAC_{K_{BM}}$ (EBC) is a keyed hash value used to ensure the freshness of message **M4-PK** and the reachability of the MN at the CoA; $MAC_{K_{BM}}$ (EBC) = First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || N_{HA} || Seq || LT_{BReq}))) and K_{BM} is the key received from the MN in message **M3-PK**.
- SIG_H is the HA's digital signature, where $SIG_H = ENC_{SK_H}$ [SHA1 (HoA || CoA || N_{HA} || Seq || LT_{BReq})] and SK_H is the home link's private key.
- $Cert_H$ is the public key certificate of the home link.

The HA also temporarily stores the value of N_{HA} (generated by itself) and the value of K_{BM} (enclosed in message **M3-PK**) to verify the freshness and authenticity of the response from that CN.

Step S5-PK

When the CN receives message **M4-PK**, it performs **Verification CN1-PK** (as shown in Figure D.17) to confirm the freshness of the message and reachability

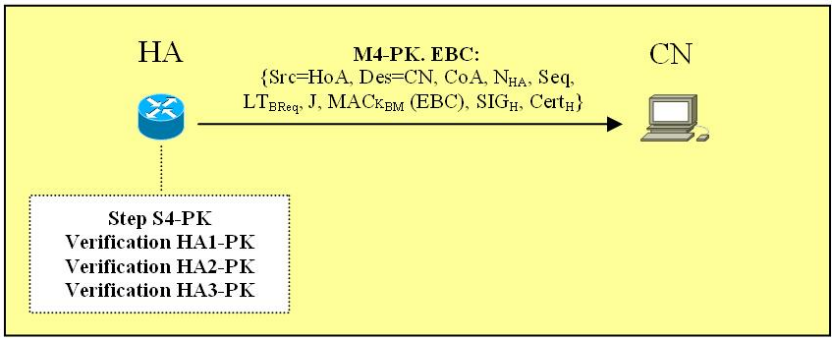

Figure D.16: Step S4-PK and message M4-PK (EBC)

of the MN at the CoA.

Verification CN1-PK: The CN uses the value of J enclosed in message **M4-PK** to generate $Token_2$, and then hashes $Token_2$ to generate K_{BM}. The CN next verifies that $MAC_{K_{BM}}$ (EBC) equals First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || N_{HA} || Seq || LT_{BReq}))), where $MAC_{K_{BM}}$ (EBC), HoA, CoA, N_{HA}, Seq, and LT_{BReq} are enclosed in message **M4-PK**.

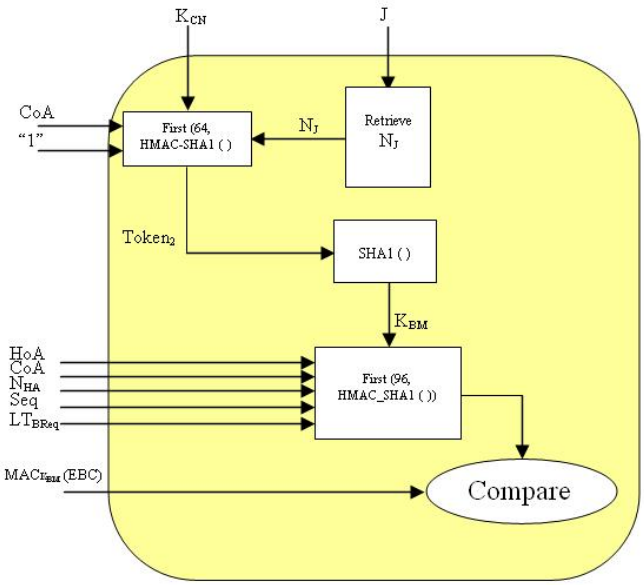

Figure D.17: Verification CN1-PK

A positive outcome of **Verification CN1-PK** assures the CN that message **M4-PK** is fresh, as the key K_{BM} is generated using unexpired token $Token_2$. It also assures the CN that message **M4-PK** is coming from a node that is reachable at the CoA, which provides some assurance of the MN's honesty before performing heavy computations. This verification is to protect the CN against replay attacks and resource exhaustion DoS attacks. After a successful verification, the CN next performs **Verification CN2-PK** to confirm the integrity and authenticity of message **M4-PK**.

Verification CN2-PK: The CN verifies the HA's signature (SIG_H) as shown in Figure D.18. The CN first decrypts SIG_H using PK_H; DEC_{PK_H} [SIG_H] = SHA1 (HoA || CoA || N_{HA} || Seq || LT_{BReq}). The CN then checks if the decrypted value equals the one freshly computed from the received items.

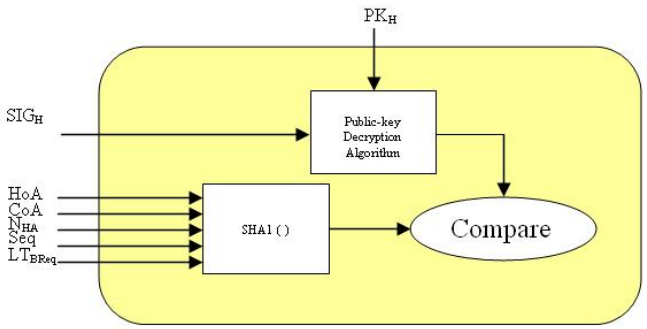

Figure D.18: Verification CN2-PK

A positive outcome of **Verification CN2-PK** assures the CN that message **M4-PK** is indeed from the home link of the MN and has not been altered in transit. If **Verification CN1-PK** or **CN2-PK** is negative, the CN will discard message **M4-PK** without any further action. Otherwise, the CN will generate fresh session keys (K_{MN-CN} and K_{HA-CN}), create a Binding Cache entry, and store the binding between the MN's HoA and CoA, as well as the values of Seq, LT_{BReq}, K_{MN-CN}, and K_{HA-CN} at the entry. In addition, the CN sets the granted binding lifetime (LT_{BGrant}) between the MN's HoA and CoA to a MIN_BINDING_LIFETIME value to handle the case of the MN roaming to a new CoA while the HA and the CN are running the protocol. The CN also sends subsequent packets destined for the MN to its new CoA for as long as the lifetime

is granted for the binding. Finally, the CN sends N_{HA}, Seq, LT_{BGrant}, ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}], and $MAC_{K_{BM}}$ (EBA) to the MN's HoA in message **M5-PK** (as shown in Figure D.19), where:

- ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}] is the encryption of K_{MN-CN} and K_{HA-CN} using public key PK_H.
- $MAC_{K_{BM}}$ (EBA) is a keyed hash value used to ensure the integrity and authenticity of message **M5-PK**; $MAC_{K_{BM}}$ (EBA) = First (96, HMAC_SHA1 (K_{BM}, (N_{HA} || Seq || LT_{BGrant} || ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}]))).

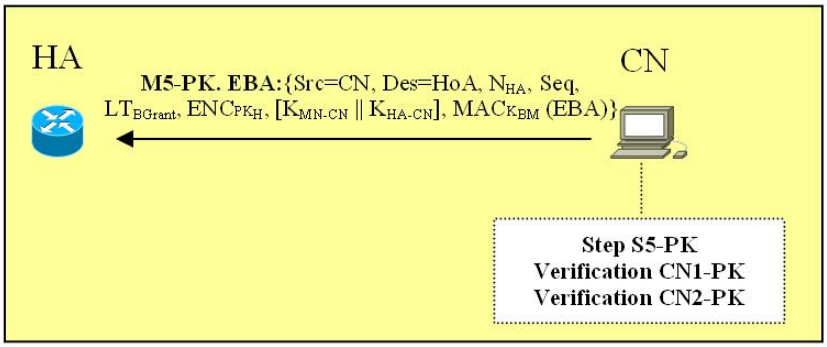

Figure D.19: Step S5-PK and message M5-PK (EBA)

Step S6-PK

The HA intercepts message **M5-PK** and performs **Verification HA4-PK** to confirm the freshness, integrity, and authenticity of the message.

Verification HA4-PK: The HA checks if N_{HA}' equals N_{HA}, where N_{HA}' is the nonce enclosed in message **M5-PK** and N_{HA} is the nonce sent by the HA in message **M4-PK**. This check is to protect the HA against replay attacks. After a successful verification, the HA uses K_{BM} to verify the integrity and authenticity of the received message (as shown in Figure D.20). Specifically, the HA verifies $MAC_{K_{BM}}$ (EBA) equals First (96, HMAC_SHA1 (K_{BM}, (N_{HA} || Seq || LT_{BGrant} || ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}]))), where $MAC_{K_{BM}}$ (EBA), N_{HA}, Seq, LT_{BGrant}, and ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}] are enclosed in message **M5-PK**. If **Verification HA4-PK** fails, the HA will discard the message without any further action.

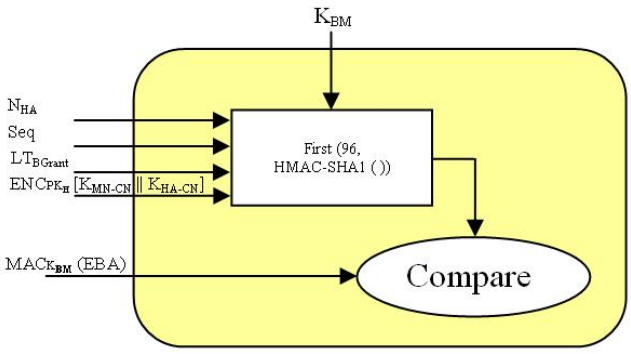

Figure D.20: Verification HA4-PK

A positive outcome of **Verification HA4-PK** assures the HA that message **M5-PK** is from a node that knows the key K_{BM} and is reachable at the CN's address, which provides some assurance of the CN's honesty before performing a heavy public-key operation. After a successful verification, the HA next decrypts the coded session keys enclosed in message **M5-PK** using the home link's private key SK_H; DEC_{SK_H} [ENC_{PK_H} [K_{MN-CN} || K_{HA-CN}]] = {K_{MN-CN} || K_{HA-CN}}. The HA then sends CN, Seq, LT_{BGrant}, and K_{MN-CN} to the MN in message **M6-PK** via an IPSec ESP secure tunnel (as shown in Figure D.21). The HA also stores the value of K_{HA-CN} and discards the values of N_{HA} and K_{BM} that it has stored while running the protocol with the CN.

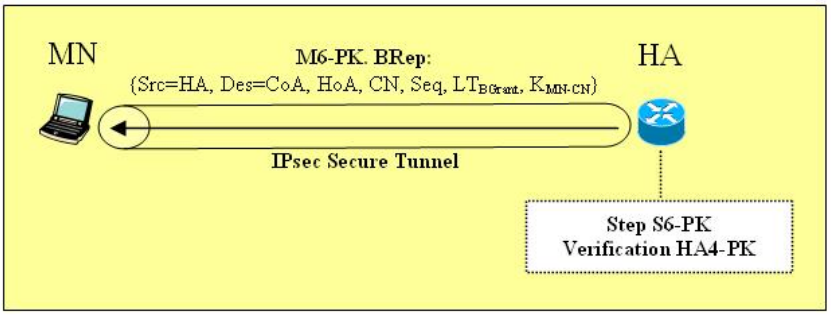

Figure D.21: Step S6-PK and message M6-PK (BRep)

Step S7-PK

When the MN receives message **M6-PK**, it decrypts the message using key K_{MN-HA}, and then uses the CN's address as an index to search its Binding Update List. If a list entry is found with a Creation_Pending status, the MN will

perform **Verification MN2-PK** to confirm the freshness of the message. Otherwise, i.e. if no entry is found (or if the matched entry is not in a Creation_Pending status), the MN will discard the received message without any further action.

Verification MN2-PK: The MN verifies that Seq' equals Seq, where Seq' is the sequence number enclosed in the message and Seq is the sequence number stored at the matched list entry in the MN. This verification is to detect replay attacks. If **Verification MN2-PK** fails, the MN will discard the message without any further action.

After **Verification MN2-PK**, the MN stores the values of K_{MN-CN} and LT_{BGrant} enclosed in the message at the matched list entry. As LT_{BGrant} indicates a small binding lifetime, the MN sends message **M7-PK** to the CN requesting a greater binding lifetime, confirming the receipt of K_{MN-CN}, and showing the existence at the CoA. Message **M7-PK** is identical to message **M7-SK** shown in Figure D.11.

Steps S8-PK and S9-PK
Steps S8-PK and **S9-PK** are identical, respectively, to **Steps S8-SK** and **S9-SK** mentioned in Section D (see Figures D.12, D.13, and D.14).

D.3 Detailed CRE-INF Phase Description

Steps S1-INF
Step **S1-INF** is identical to Step **S1-SK** and **Step S1-PK**, but the MN's HoA is enclosed in message **M1-INF** sent from the MN to the CN as shown in Figure D.22. The MN sends message **M1-INF** to request home keygen and care-of keygen tokens, and thus, the MN includes its HoA in the message.

Step S2-INF
Upon receiving message **M1-INF**, the CN performs **Verification CN1-INF** to confirm that the HoA and CoA enclosed in the message belong to different networks.

Verification CN1-INF: The CN checks that the subnet prefix part of the HoA is not equal to the subnet prefix part of the CoA.

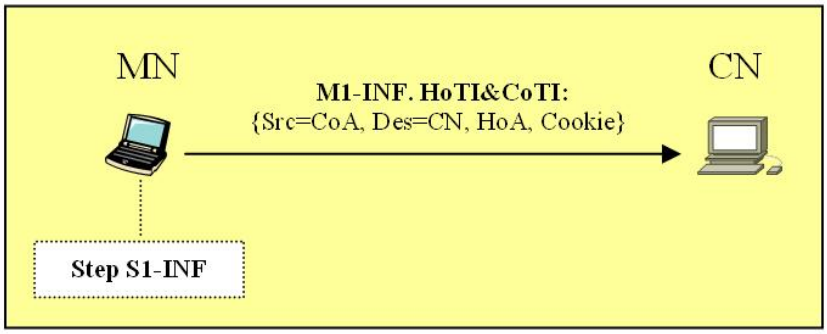

Figure D.22: Step S1-INF and message M1-INF (HoTI&CoTI)

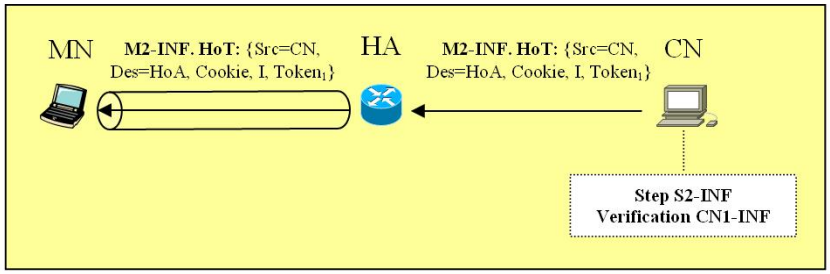

Figure D.23: Step S2-INF and message M2-INF (HoT)

A positive outcome of **Verification CN1-INF** assures the CN that the two messages it will send as a response will not be routed to the same network, and thus, the CN is not involved in an amplification attack against a particular network. If **Verification CN1-INF** fails, the CN will discard the message without any further action. Otherwise, the CN will reply to the MN's request as in the RR procedure (see Section 2.5). Specifically, the CN generates a home keygen token (Token_1) and a care-of keygen token (Token_2). The CN then sends the Cookie, I, and Token_1 to the MN's HoA in message **M2-INF**. The HA intercepts message **M2-INF** and forwards it to the MN's registered CoA via an IPSec ESP secure tunnel (as shown in Figure D.23). At the same time, the CN sends the Cookie, J, and Token_2 to the MN's CoA in message **M3-INF** (as shown in Figure D.24). I and J are indices that identify home nonce (N_I) and care-of nonce (N_J) used in generating Token_1 and Token_2, respectively.

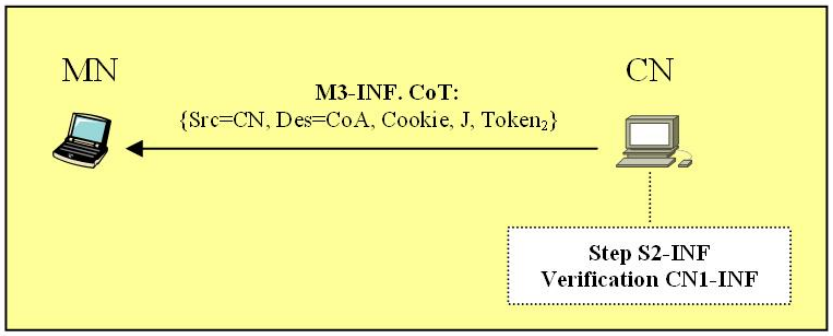

Figure D.24: Step S2-INF and message M3-INF (CoT)

Step S3-INF

When the MN receives message **M2-INF** or message **M3-INF**, it uses the CN's address as an index to search its Binding Update List. If a list entry is found with a Route_Pending status, the MN will perform **Verification MN1-INF** to confirm that the received cookie (Cookie) equals the cookie that was sent in message **M1-INF**. **Verification MN1-INF** is identical to **Verification MN1-SK** mentioned in Section D.

If no list entry is found, if the matched list entry is not in a Route_Pending status, or if **Verification MN1-INF** fails, the MN will discard the received message without any further action. Otherwise, the MN will record the given keygen token and nonce index in the list entry and wait for the second message. When the MN receives both of message **M2-INF** and message **M3-INF**, it hashes the two tokens together to form a binding management key K_{BM}; K_{BM} = SHA1 ($Token_1$ || $Token_2$). The MN then sends CoA, LT_{BRem}, CN, Seq, LT_{BReq}, I, J, K_{BM}, Modifier, and Collision-Count to the HA in message **M4-INF** (as shown in Figure D.25) requesting binding creation with a specific CN. Modifier and Collision-Count are the values of the modifier and the collision-count, respectively, which are used in generating the MN's CGA-based HoA. Message **M4-INF** is sent via an IPSec ESP secure tunnel; it is encrypted using the secret key K_{MN-HA} shared between the MN and the HA. Finally, the MN changes the status of the list entry to Creation_Pending indicating that the return of a session key and an acknowledgement are expected.

Step S4-INF

Step S4-INF is identical to **Step S4-PK**, but the HA concatenates the values of

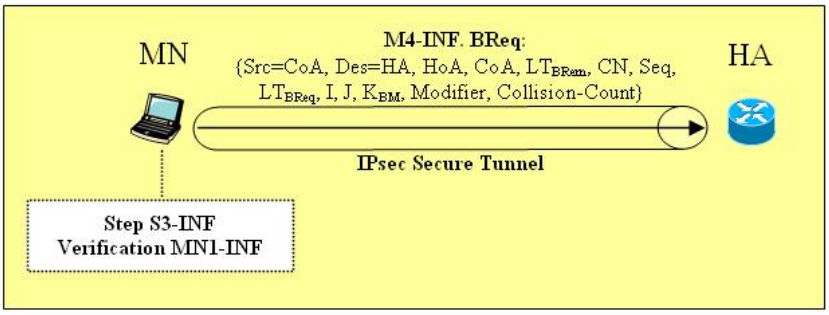

Figure D.25: Step S3-INF and message M4-INF (BReq)

Modifier, Subnet Prefix, and Collision-Count to form the CGA parameters used in generating the MN's HoA. Specifically, upon receipt of message **M4-INF**, the HA performs **Verifications HA1-INF**, **HA2-INF**, and **HA3-INF**, which are identical to **Verifications HA1-SK**, **HA2-SK**, and **HA3-SK**, respectively. If **Verification HA1-INF** or **HA2-INF** fails, the HA will discard message **M4-INF** without any further action. If **Verifications HA3-INF** fails, the HA will use the value of LT_{BRem} instead of LT_{BReq} in message **M5-INF** sent to the CN.

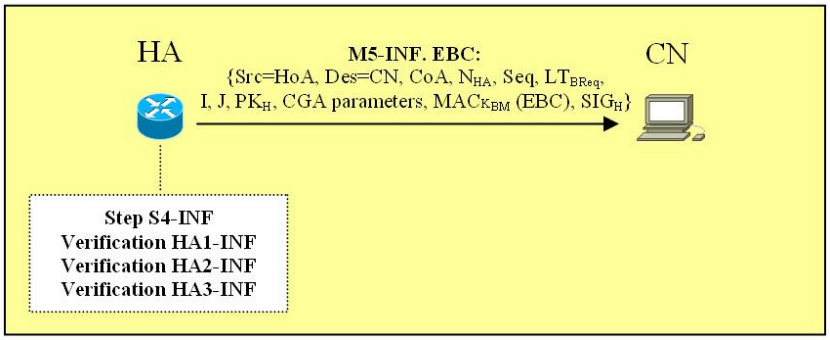

Figure D.26: Step S4-INF and message M5-INF (EBC)

After **Verification HA3-INF**, the HA concatenates from left the values of Modifier, Subnet Prefix, and Collision-Count to form the CGA parameters used in generating the MN's HoA, where Modifier and Collision-Count are enclosed in message **M4-INF**, and Subnet Prefix is the home link subnet prefix. The HA then sends message **M5-INF** (shown in Figure D.26) to the CN to request binding creation on behalf of the MN. Message **M5-INF** is identical to message **M4-PK** shown in Figure D.16, but the CGA parameters and the home link's

self-generated public key are included in the message. The HA also temporarily stores the value of K_{BM} enclosed in message **M4-INF** and the value of N_{HA} to verify the authenticity and freshness of the response from that CN, respectively.

Step S5-INF

When the CN receives message **M5-INF**, it first performs **Verification CN2-INF** (as shown in Figure D.27) to confirm the freshness of the message and reachability of the MN at both the HoA and the CoA.

Verification CN2-INF: The CN uses the values of I and J enclosed in message **M5-INF** to generate $Token_1$ and $Token_2$, respectively, and then hashes the two tokens to generate K_{BM} as in the RR procedure (see Section 2.5). The CN next verifies that $MAC_{K_{BM}}$ (EBC) equals First (96, HMAC_SHA1 (K_{BM}, (CoA || N_{HA} || Seq || LT_{BReq}))), where $MAC_{K_{BM}}$ (EBC), CoA, N_{HA}, Seq, and LT_{BReq} are enclosed in message **M5-INF**.

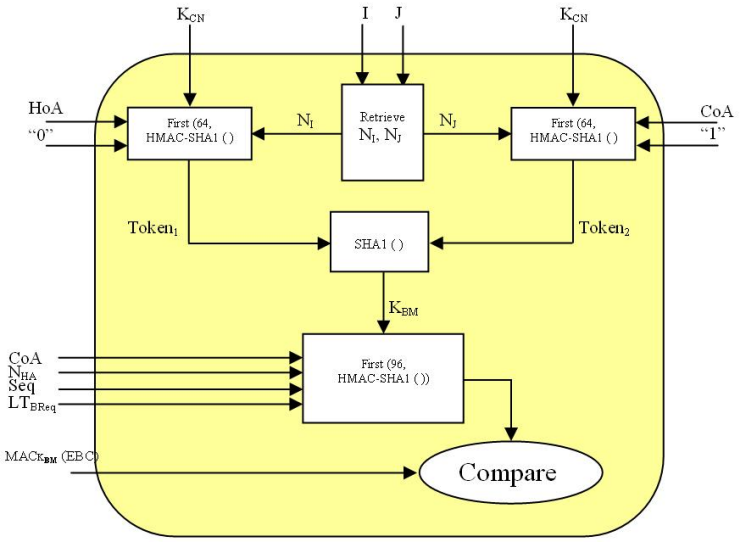

Figure D.27: Verification CN2-INF

A positive outcome of **Verification CN2-INF** assures the CN that message **M5-INF** is fresh, as the key K_{BM} is generated using unexpired tokens. It also assures the CN that message **M5-INF** is coming from a node that is reachable at both the HoA and the CoA, which provides some assurance of the MN's honesty before performing heavy computations. This verification is to protect the CN

against replay attacks and resource exhaustion DoS attacks. After a successful verification, the CN next performs **Verification CN3-INF**.

Verification CN3-INF: The CN runs the CGA-based address verification algorithm (see Figures 3.2 in Section 3.1) to confirm the authenticity of HoA.

If the outcome of **Verification CN3-INF** is positive, the CN is assured that the HoA and PK_H enclosed in message **M5-INF** are bound. The CN then performs **Verification CN4-INF** to confirm the authenticity of message **M5-INF**.

Verification CN4-INF: The CN verifies the correctness of HA's signature in message **M5-INF**. **Verification CN4-INF** is identical to **Verification CN2-PK** shown in Figure D.18.

A positive outcome of **Verification CN4-INF** assures the CN that message **M5-INF** is indeed from a node that knows the private key corresponding to the public key used in generating the HoA.

If **Verification CN2-INF**, **CN3-INF**, or **CN4-INF** is negative, the CN will discard message **M5-INF** without any further action. Otherwise, the CN will perform as in **Step S5-PK**. Specifically, the CN will generate K_{MN-CN} and K_{HA-CN}, create a Binding Cache entry, and store the binding between the MN's HoA and CoA, as well as the values of Seq, LT_{BReq}, K_{MN-CN} and K_{HA-CN} at the entry. In addition, the CN sets the granted binding lifetime LT_{BGrant} between the MN's HoA and CoA to a MIN_BINDING_LIFETIME value. Finally, the CN sends message **M6-INF** to the HA. Message **M6-INF** is identical to message **M5-PK** shown in Figure D.19.

Steps S6-INF, S7-INF, S8-INF, and S9-INF
Steps S6-INF, **S7-INF**, **S8-INF**, and **S9-INF** are identical, respectively, to **Steps S6-PK**, **S7-PK**, **S8-PK**, and **S9-PK** mentioned in Section D.

D.4 Detailed UPD Phase Description

Step S1-UPD

The MN generates a fresh key K_{BM} = HMAC_SHA1 (K_{MN-CN}, (HoA || CoA || CN || Seq_{new})), where K_{MN-CN} is the shared session key established between the MN and the CN in the creation phase, and Seq_{new} is a new sequence number that is greater than the previous sequence number sent to that CN. The MN then sends HoA, CoA, Seq_{new}, LT_{BReq}, Ack, Authenticator, and $MAC_{K_{BM}}$(BU) to the CN in message **M1-UPD** (as shown in Figure D.28), where:

- Ack is an acknowledge bit that may be set to request the return of a binding acknowledgement message.
- Authenticator is a coded value used to protect the MN's HA against replay attacks. Authenticator = $ENC_{K_{MN-HA}}$[HoA || CoA || CN || LT_{BRem}], where K_{MN-HA} is the secret key shared between the MN and the HA (Assumption **A2**), and LT_{BRem} is the current remaining lifetime for the binding of HoA and CoA at both the MN's Binding Update List entry for the HA and the HA's Binding Cache entry for the MN. Authenticator will be forwarded by the CN to the HA in message **M3-UPD** to convenience the HA that the message is fresh.
- $MAC_{K_{BM}}$(BU) is a keyed hash value used to ensure the integrity and authenticity of message **M1-UPD**; $MAC_{K_{BM}}$(BU) = First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || Seq_{new} || LT_{BReq} || Ack || Authenticator))).

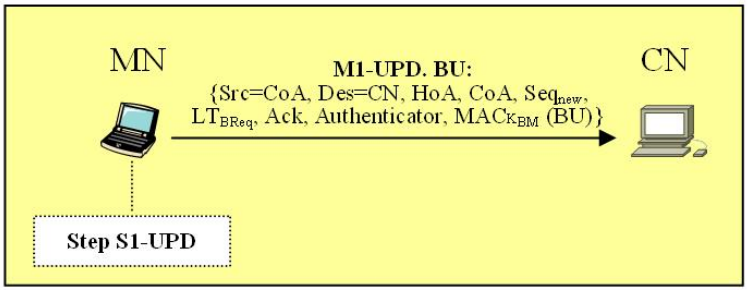

Figure D.28: Step S1-UPD and message M1-UPD (BU)

The MN also changes the status of the list entry. Specifically, if the MN sets the Ack bit, then it will change the status to a Binding_Pending indicating that

it is waiting for an acknowledgement; otherwise, it will change the status to a Binding_Complete indicating the binding update is complete.

If the MN sends message **M1-UPD** to the CN in which the Ack bit is set and does not receive a matching response within a retransmission interval of one second, the MN will resend the message with a new sequence number that is greater than the value of the sequence number used for the previous transmission of this message. The MN doubles the retransmission interval upon each retransmission in the same way, as specified in the base specification of the MIPv6 protocol ([5], see also Section 2.5).

Step S2-UPD
Upon receiving message **M1-UPD**, the CN uses HoA enclosed in the message to find a matched cache entry. The CN then performs **Verification CN1-UPD** to confirm the freshness of the message.

Verification CN1-UPD: The CN verifies that Seq_{new} is greater than Seq, where Seq_{new} is the sequence number enclosed in the message and Seq is the sequence number stored at the matched cache entry in the CN. This verification is to detect replay attacks.

After a successful verification, the CN generates key K_{BM} based on key K_{MN-CN} shared with the MN and on items enclosed in the message. The CN then performs **Verification CN2-UPD** (as shown in Figure D.29) to confirm the integrity and authenticity of the message.

Verification CN2-UPD: The CN verifies that $\text{MAC}_{K_{BM}}$(BU) equals First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || Seq_{new} || LT_{BReq} || Ack || Authenticator))), where $\text{MAC}_{K_{BM}}$(BU), HoA, CoA, Seq_{new}, LT_{BReq}, Ack, and Authenticator are enclosed in message **M1-UPD**.

If **Verification CN1-UPD** or **CN2-UPD** fails (or if no cache entry is found), the CN will discard the message without any further action. Otherwise, the CN will register and use the new CoA while concurrently requesting home network's confirmation of that CoA. Specifically, the CN updates the matched cache entry with the values of CoA, Seq_{new}, and Ack enclosed in message **M1-UPD**. The CN

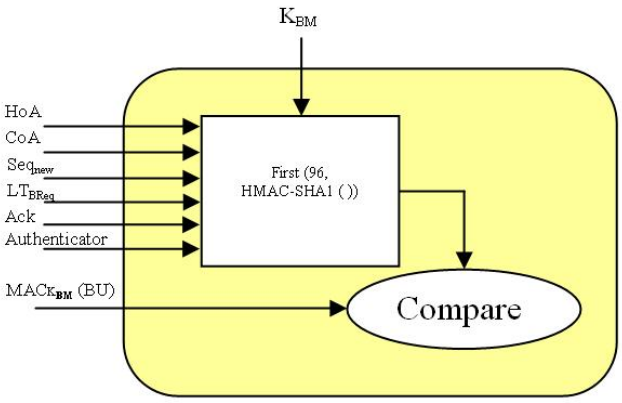

Figure D.29: Verification CN2-UPD

also sets the entry in an unconfirmed state and limits the amount of data sent to the CoA by setting the granted binding lifetime to a MIN_BINDING_LIFETIME value. In addition, the CN sends subsequent packets destined for the MN to the unconfirmed CoA for as long as the lifetime is granted for the binding. The CN then generates a fresh key K_{BC} = HMAC_SHA1 (K_{HA-CN}, (HoA || CoA || CN || N_{CN})), where K_{HA-CN} is the shared session key established between the MN's home network and the CN in the creation phase, and N_{CN} is a fresh nonce. Finally, the CN sends CoA, LT_{BReq}, Authenticator, N_{CN}, and $MAC_{K_{BC}}$ (BCReq) to the MN's HoA in message **M2-UPD** (as shown in Figure D.30) to request home network's confirmation of the MN's claimed CoA, where $MAC_{K_{BC}}$ (BCReq) is a keyed hash value used to ensure the integrity and authenticity of message **M2-UPD**; $MAC_{K_{BC}}$ (BCReq) = First (96, HMAC_SHA1 (K_{BC}, (HoA || CoA || LT_{BReq} || Authenticator || N_{CN}))).

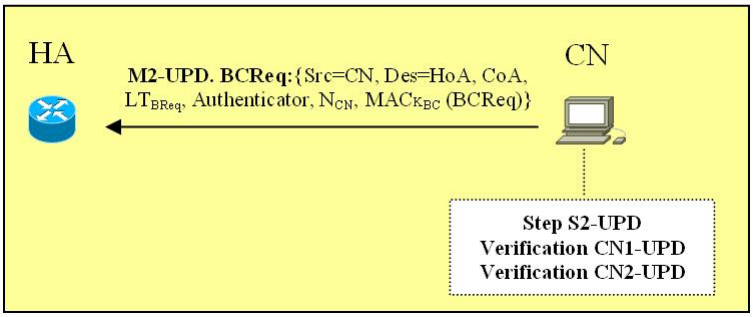

Figure D.30: Step S2-UPD and message M2-UPD (BCReq)

The CN limits the number of **M1-UPD** messages that could be received from unconfirmed CoAs. Normally, the MN's HA intercepts packets on the home link destined for the MN's HoA. Therefore, when the CN sends message **M2-UPD** to the MN's HoA, it is expecting that the message will be intercepted by the HA. However, the MN could return to its home link and cheat the CN with a fake CoA in message **M1-UPD**. In this case, the MN would receive and ignore message **M2-UPD** sent by the CN and would continue to send message **M1-UPD**. By limiting the number of **M1-UPD** messages received from unconfirmed CoAs, the CN can prevent the MN from bypassing the confirmation request by continuously sending **M1-UPD** message without involving the request.

Step S3-UPD

The HA intercepts message **M2-UPD** and generates K_{BC} based on K_{HA-CN} shared with the CN and on items enclosed in the message. The HA then performs **Verification HA1-UPD** (as shown in Figure D.31) to confirm the integrity and authenticity of the message.

Verification HA1-UPD: The HA verifies that $MAC_{K_{BC}}$ (BCReq)' equals First (96, HMAC_SHA1 (K_{BC}, (HoA' || CoA' || LT_{BReq}' || Authenticator' || N_{CN}'))), where $MAC_{K_{BC}}$ (BCReq)', HoA', CoA', LT_{BReq}', Authenticator', and N_{CN}' are enclosed in message **M2-UPD**.

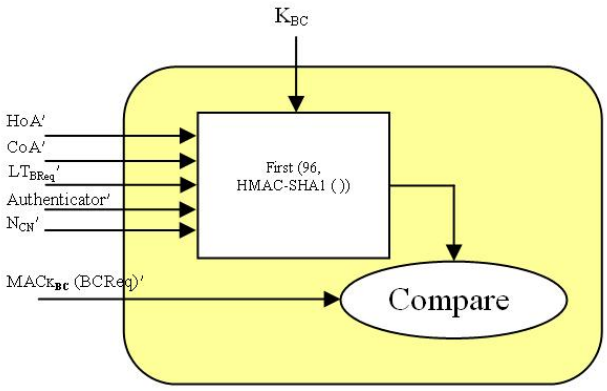

Figure D.31: Verification HA1-UPD

A positive outcome of **Verification HA1-UPD** assures the HA that message **M2-UPD** is coming from the CN and has not been altered in transit. After

a successful verification, the HA performs **Verification HA2-UPD** to confirm that the Authenticator enclosed in the message is freshly generated by the MN; **Verification HA2-UPD** is to confirm the freshness of message **M2-UPD**.

Verification HA2-UPD: The HA decrypts the Authenticator enclosed in message **M2- UPD** using key K_{MN-HA} shared with the MN; $DEC_{K_{MN-HA}}$ [Authenticator] = {HoA" || CoA" || CN" || LT_{BRem}"}. The HA then checks if the decrypted values equal the values enclosed in the message. That is to confirm that HoA" equals HoA', CoA" equals CoA', and CN" equals CN', where HoA", CoA", and CN" are the decrypted items, and HoA', CoA', and CN' are the items enclosed in the message. The HA also checks whether or not LT_{BRem}" is fresh. That is, if (LT_{BRem} - t_{valid}) $\leq$ LT_{BRem}" $\leq$ (LT_{BRem} + t_{valid}) then it is fresh, where LT_{BRem} is the remaining binding lifetime stored locally at HA in Binding Cache entry for the HoA when message **M2-UPD** is received; and t_{valid} is the validity period agreed upon priorly between MN and HA (Assumption **A3**).

A positive outcome of **Verification HA2-UPD** assures the HA that the Authenticator is generated using a fresh timestamp (LT_{BRem}") for that particular CN, and therefore message **M2-UPD** is fresh. After a successful verification, the HA performs **Verification HA3-UPD** to confirm that CoA' enclosed in the message matches the MN's current location.

Verification HA3-UPD: The HA checks if CoA' equals CoA, where CoA' is the MN's care of address enclosed in message **M2-UPD** and CoA is the MN's care-of address stored locally at HA in the Binding Cache entry for the HoA enclosed in the message.

A positive outcome of **Verification HA3-UPD** assures the HA that: (1) the CN already knows the current location of the MN; and (2) the MN is not cheating the CN with a fake CoA. After a successful verification, the HA performs **Verification HA4-UPD** to confirm that the binding lifetime requested by the MN (LT_{BReq}') is not greater than the remaining lifetime (LT_{BRem}) for the binding of HoA and CoA at the HA.

Verification HA4-UPD: The HA checks if LT_{BReq}' $\leq LT_{BRem}$, where LT_{BReq}' is the binding lifetime request enclosed in message **M2-UPD** and LT_{BRem} is the remaining binding lifetime stored locally at HA in Binding Cache entry for the HoA when message **M2-UPD** is received. This verification is to confirm that the binding lifetime requested by the MN is not greater than the remaining lifetime for the binding of HoA and CoA at the HA. If the verification fails, the HA will use the value of LT_{BRem} instead of LT_{BReq}' in message **M3-UPD** sent to the CN.

If Verification **HA1-UPD**, **HA2-UPD**, or **HA3-UPD** fails, the HA will discard message **M2-UPD** without any further action. Otherwise, the HA will send CoA, LT_{BReq}, N_{CN}, and $MAC_{K_{BC}}$ (BCRep) to the CN in message **M3-UPD** (as shown in Figure D.32) for confirming the claimed CoA, where $MAC_{K_{BC}}$ (BCRep) is a keyed hash value used to ensure the integrity and authenticity of message **M3-UPD**; $MAC_{K_{BC}}$ (BCRep) = First (96, HMAC_SHA1 (K_{BC}, (HoA || CoA || LT_{BReq} || N_{CN}))).

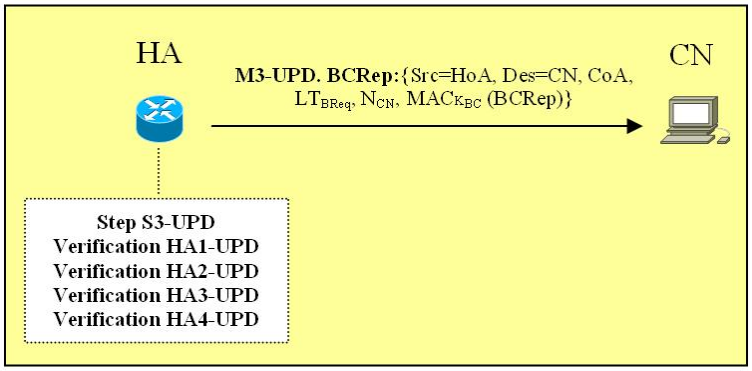

Figure D.32: Step S3-UPD and message M3-UPD (BCRep)

Step S4-UPD

Upon receiving message **M3-UPD**, the CN uses the HoA enclosed in the message to find a matched cache entry. The CN then performs **Verification CN3-UPD** to confirm the freshness, integrity, and authenticity of the message.

Verification CN3-UPD: The CN checks if N_{CN}' equals N_{CN}, where N_{CN}' is the nonce enclosed in message **M3-UPD** and N_{CN} is the nonce sent by the CN in message **M2-UPD**. This check is to protect the CN against replay attacks. After

a successful verification, the CN uses K_{BC} to verify the integrity and authenticity of the received message (as shown in Figure D.33). Specifically, the CN verifies that $MAC_{K_{BC}}$ (BCRep) = First (96, HMAC_SHA1 (K_{BC}, (HoA || CoA || LT_{BReq} || N_{CN}))), where $MAC_{K_{BC}}$ (BCRep), HoA, CoA, LT_{BReq}, and N_{CN} are enclosed in message **M3-UPD**. If **Verification CN3-UPD** fails, the CN will discard the message without any further action.

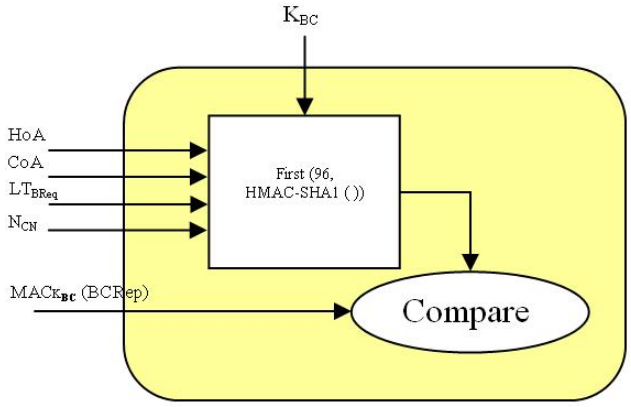

Figure D.33: Verification CN3-UPD

After a successful verification, the CN updates the matched cache entry by changing its status to be confirmed and by setting the granted binding lifetime (LT_{BGrant}) to a value that is less than or equal to LT_{BReq} enclosed in message **M3-UPD**. Finally, the CN checks the Ack bit of the matched cache entry. If the MN has requested an acknowledgement, the CN will send Seq_{new}, LT_{BGrant}, and $MAC_{K_{BM}}$ (BA) to the MN in message **M4-UPD** (as shown in Figure D.34) for acknowledging the binding of the CoA, where $MAC_{K_{BM}}$ (BA) is a keyed hash value used to ensure the integrity and authenticity of message **M4-UPD**; $MAC_{K_{BM}}$ (BA) = First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || CN || Seq_{new} || LT_{BGrant}))).

Step S5-UPD

Upon receiving message **M4-UPD**, the MN uses the CN's address to find a matched list entry, and then performs **Verification MN1-UPD** to confirm the freshness, integrity and authenticity of the message.

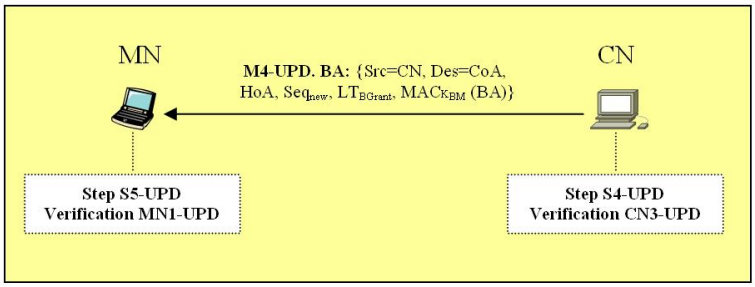

Figure D.34: Step S4-UPD, message M4-UPD (BA), and Step S5-UPD

Verification MN1-UPD: The MN verifies that Seq$_{new}$' equals Seq$_{new}$, where Seq$_{new}$' is the sequence number enclosed in message **M4-UPD** and Seq$_{new}$ is the sequence number stored at the matched list entry in the MN. This verification is to detect replay attacks. After a successful verification, the MN uses K_{BM} to verify the integrity and authenticity of the received message (as shown in Figure D.35). Specifically, the MN verifies that $MAC_{K_{BM}}$ (BA) equals First (96, HMAC_SHA1 (K_{BM}, (HoA || CoA || CN || Seq_{new} || LT_{BGrant}))), where $MAC_{K_{BM}}$ (BA), HoA, CoA, CN, Se_{qnew}, and LT_{BGrant} are enclosed in message **M4-UPD**.

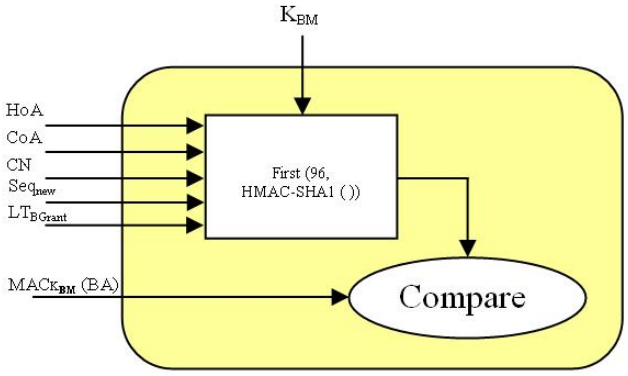

Figure D.35: Verification MN1-UPD

If no matched list entry is found for that CN, or if **Verification MN1-UPD** fails, the MN will discard the message without any further action. Otherwise, the MN will update the status of the list entry to Binding_Complete indicating that the binding has been acknowledged. In addition, the MN adjusts the remaining binding lifetime depending on the given granted binding lifetime. The update phase for the protocols in the stationary CN case is now complete.

D.5 Detailed DEL Phase Description

Step S1-DEL

Step S1-DEL is identical to **Step S1-UPD**, but the Authenticator is not enclosed in message **M1-DEL** sent from the MN to the CN. In addition, the MN sets the CoA equal to its HoA and the binding lifetime request (LT_{BReq}) to 'zero' as shown in Figure D.36.

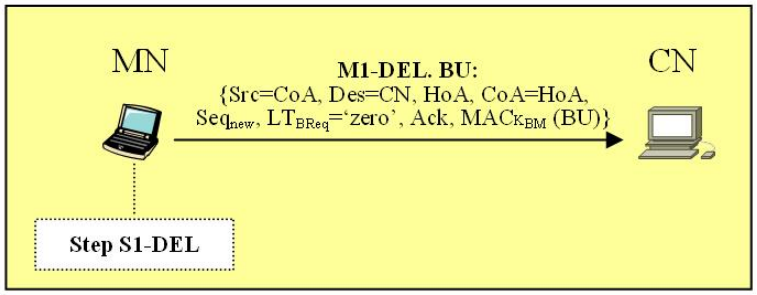

Figure D.36: Step S1-DEL and message M1-DEL (BU)

Step S2-DEL

The first part of **Step S2-DEL** is identical to the first part of **Step S2-UPD**. Specifically, the CN uses the MN's HoA as an index to search its Binding Cache. If a cache entry is found, the CN will perform **Verifications CN1-DEL** and **CN2-DEL**, which are identical to **Verifications CN1-UPD** and **CN2-UPD**, respectively. **Verification CN1-DEL** is to confirm the freshness of message **M1-DEL**; and **Verification CN2-DEL** is to confirm the authenticity of the message.

If **Verification CN1-DEL** or **CN2-DEL** fails (or if no cache entry is found), the CN will discard the message without any further action. Otherwise, the CN will delete the Binding Cache entry for the MN and, if requested, will send Seq_{new} and $MAC_{K_{BM}}$ (BA) to the MN in message **M2-DEL** (as shown in Figure D.37) for acknowledging the deletion of the binding, where $MAC_{K_{BM}}$ (BA) is a keyed hash value used to ensure the integrity and authenticity of message **M2-DEL**; $MAC_{K_{BM}}$ (BA) = First (96, HMAC_SHA1 (K_{BM}, (HoA || CN || Seq_{new}))).

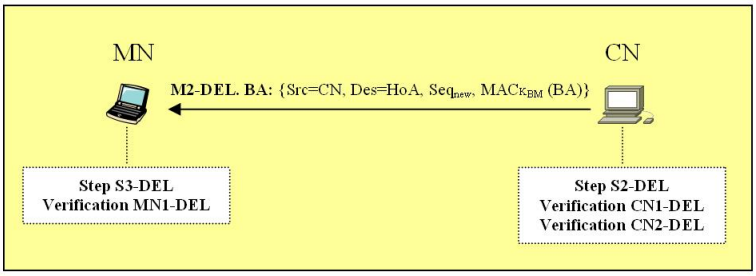

Figure D.37: Step S2-DEL, message M2-DEL (BA), and Step S3-DEL

Step S3-DEL

Step S3-DEL is identical to **Step S5-UPD**, but after **Verification MN1-DEL**, which is identical to **Verification MN1-UPD**, the MN will delete the Binding Update List entry for the CN. The deletion phase for the protocols in the stationary CN case is now complete.

Appendix E

Protocol Composition Logic (PCL)

E.1 Programming Language

names	N ::= $\hat{X}$	name.
threads	P ::= X	thread.
keys	K ::= k	basic key.
	$\overline{k}$	inverse key.
nonce	n ::= n	nonce.
numbers	i ::= i	number.
	i_1 ::= succ(i_2)	$i_1 > i_2$.
terms	t ::= x	variable term.
	name(N)	name.
	thread(P)	thread.
	key(K)	key.
	nonce(n)	nonce.
	number(i)	number.
	t_1,...,i_2	tuple of terms.
	ENC_k{t}	encrypted term.
	$HASH_k${t}	keyed hash term.
	HASH{t}	hashed term.
	$SIG_{\overline{k}}${t}	signed term.

actions	a ::=	send t	send a term t.
		receive x	receive into var x.
		new x	generate a new term.
		isLess i_1,i_2	check that $i_1 < i_2$.
		hash m	hash m.
		hash m,k	hash m using k.
		enc m,k	encrypt m using k.
		dec m,k	decrypt m using k.
		sign m,$\overline{k}$	sign m using $\overline{k}$.
		verify m,k	verify the signature.
		match t_1/t_2	match a term.

E.2 Syntax of the Logic

Action formulas:

a::=	Send(X,m)	principal $\hat{X}$ has sent message m.
	Receive(X,m)	principal $\hat{X}$ has received message m.
	New(X,t)	principal $\hat{X}$ has generated new term t.
	SymEnc(X,t,k)	principal $\hat{X}$ has computed encrypted term t using symmetric key k.
	PkEnc(X,t,k)	principal $\hat{X}$ has computed encrypted term t using public key k.
	SymDec(X,t,k)	principal $\hat{X}$ has computed decrypted term t using symmetric key k.
	PkDec(X,t,$\overline{k}$)	principal $\hat{X}$ has computed decrypted term t using its private key $\overline{k}$.
	Sign(X,t,$\overline{k}$)	principal $\hat{X}$ has signed term t using its private key $\overline{k}$.
	Verify(X,t,k)	principal $\hat{X}$ has verified signed term t using public key k.
	Hash(X,t,k)	principal $\hat{X}$ has computed hashed term t using key k.
	Hash(X,t)	principal $\hat{X}$ has computed hashed term t.
	IsLess(i_1,i_2)	number i_1 is less than number i_2.

Formulas:

ϕ::=	a	any action formula.
	$a_1 < a_2$	both actions have happened and action a_2 has happened after action a_1.
	$\phi_1 \wedge \phi_2$	both formula ϕ_1 and formula ϕ_2 hold

$\neg\ \phi$	formula ϕ is false
$\exists$x.ϕ	formula ϕ exists in thread X
$\Diamond\ \phi$	formula ϕ holds in some state in the past.
$\ominus\ \phi$	formula ϕ holds in the previous state.
Has(X,t)	principal $\hat{X}$ possesses term t. This is because it has generated t, has received t in the clear, or has received t under encryption where the decryption key is known.
Start(X)	principal $\hat{X}$ starts a new thread X; $\hat{X}$ did not execute any actions in the past in this thread.
Contains(t_1,t_2)	term t_1 contains term t_2 as a sub-term.
Fresh(X,t)	term t generated by principal $\hat{X}$ is fresh in the sense that no one else has seen any term containing t as a sub-term.
Gen(X,t)	X is the originating thread of the term t.
FirstSend(X,t,m)	principal $\hat{X}$ has sent the term t for the first time when $\hat{X}$ has sent the message m.
Computes(X,t)	principal $\hat{X}$ possesses enough information to compute term t.
Honest($\hat{X}$)	principal $\hat{X}$ assumes a particular role of the protocol and does exactly the actions prescribed by that role.
SafeMsg(m,s,k)	every occurrence of secret s in message m is protected by key k.
SendsSafeMsg(X,s,k)	$\equiv\ \forall$m.(Send(X,m) $\supset$ SafeMsg(m,s,k)), all messages sent by thread X are 'Safe'.
SafeNet(s,k)	$\equiv \forall$X.SendsSafeMsg(X,s,k), all messages sent by all threads are 'Safe'.
KeyHonest(K)	$\equiv \forall$X.$\forall$k.$\in$K.(Has(X,k)$\supset$Honest($\hat{X}$)), a thread generating a secret belong to honest principals.
OrigHonest(s)	$\equiv \forall$X.(New(X,s)$\supset$Honest($\hat{X}$)), all threads with access to a relevant key belong to honest principals.
KOHonest(s,K)	$\equiv$ KeyHonest(K) $\wedge$ OrigHonest(s), the conjunction of the two predicates.

Modal formulas:

Ψ::= θ $[a]_X$ ϕ	this means starting from a state where formula θ is true, after actions 'a' are executed by the principal $\hat{X}$, the formula ϕ is true in the resulting state.

E.3 Proof System

Axioms for protocol actions:

AA1	ϕ $[a]_X$ $\diamond$ a	if $\hat{X}$ has executed an action in some role, then the corresponding predicate asserting that the action has occurred in the past is true.
AA2	Start(X) $[a]_X$ $\neg$ a(X)	at the start of a thread any action predicate applied to the thread is false.
AA3	$\neg$Send(X,t) $[b]_X$ $\neg$Send(X,t), if σSend(X,t) $\neq$ σb for all substitutions σ	the predicate asserting thread X has not sent the term t remains false after any action that does not send a term that unified with t.
AA4	ϕ $[a_1,...,a_k]_X$ $a_1 < ... < a_k$	after thread X does actions a_1, ..., a_k in sequence, the action predicates corresponding to the actions are ordered in the same sequence.
AN1	New(X,t) $\wedge$ New(Y,t) $\supset$ $\hat{X}$=$\hat{Y}$	a particular nonce is generated by a unique thread.
AN2	ϕ [new t]$_X$ Has(Y,t) $\supset$ $\hat{Y}$=$\hat{X}$	if thread X generates a new nonce t, and does no further actions, then no one else knows t.
AN3	ϕ[new t]$_X$ Fresh(X,t)	if a thread generates a new value, then this value is fresh.
AN4	Fresh(X,t) $\supset$ Gen(X,t)	if a fresh value t is known to a thread X, then X is the originating thread of t.
AR1	Receive(X,p(t_1)) [match p(t_1) as p(t_2)]$_X$ Receive(X,p(t_2))	it is used to model obtaining information about structure of terms as they received.
AR2	a(x) [verify x,t,k]$_X$ a(SIG$_{\overline{k}}${t})	it is used to model appropriate substitution of signature verification inside the action predicate a.
AR3	a(x) [m' := dec x,t,k]$_X$ a(ENC$_k${t})	it is used to model appropriate substitution of decryption inside the action predicate a

Preservation axioms:

P1	Persist(X,t) [a]$_X$ Persist(X,t) for Persist $\in$ {Has, FirstSend, a<b, a}	certain predicates continues to hold after further actions.
P2	Fresh(X,t) [a]$_X$ Fresh(X,t) where t $\not\sqsubseteq$ a	freshness of a term holds across actions that do not send out some term containing it.

Possession axioms:

ORIG	◇New(X,t) ⊃ Has(X,t)	principal $\hat{X}$ possesses term t if $\hat{X}$ freshly generated it.
REC	◇Receive(X,t) ⊃ Has(X,t)	principal $\hat{X}$ possesses term t if $\hat{X}$ received it in some message.
TUP	Has(X,t_1) ∧ Has(X,t_1) ⊃ Has(X,(t_1,t_2))	principal $\hat{X}$ can construct a tuple if the parts are known.
PROJ	Has(X,(t_1,t_2)) ⊃ Has(X,t_1) ∧ Has(X,t_1)	principal $\hat{X}$ can decompose a tuple into its components.

Temporal ordering axioms:

FS1	Fresh(X,t) $[\text{send m}]_X$ FirstSend(X,t,m) where t ⊆ m	if $\hat{X}$ generated a fresh term t and sent it out in message m, then this is the first such send event.
FS2	FirstSend(X,t,m) ∧ a(Y,t') ⊃ Send(X,t) < a(Y,t') where X ≠ Y and t ⊆ t'	if $\hat{Y}$ did some action with a term t', which contains a term t that first sent inside a message m by $\hat{X}$ as a sub-term, then that send must have occurred before $\hat{Y}$'s action.

Hash axioms:

HASH1	Computes(X,HASH_k{t})⊃ Has(X,k) ∧ Has(X,t)	if a principal has hashed term t using key k, then the principal possesses both t and k.
HASH2	Computes(X,HASH_k{t})⊃ Has(X,HASH_k{t})	if a principal can hash term t using key k, then the principal possesses the hashed term.
HASH3	Receive(X,HASH_k{t})⊃ ∃Z,m.Computes(Z,HASH_k{t}) ∧Send(Z,m)∧ Contains(m,HASH_k{t})	if $\hat{X}$ has received a hashed term, then there must be another principal $\hat{Z}$ that hashed and sent the term in the past.
HASH4	Has(X,HASH_k{t})⊃ Computes(X,HASH_k{t})∨ ∃Z,m.Computes(Z,HASH_k{t}) ∧Send(Z,m)∧ Contains(m,HASH_k{t})	if $\hat{X}$ possesses a hashed term, then either $\hat{X}$ hashed the term or another principal $\hat{Z}$ hashed and sent the term in the past.

Encryption and signature axioms:

Where: Enc ∈ {SymEnc,PKEnc} and Dec ∈ {SymDec,PKDec}

SEC	Honest($\hat{X}$) ∧ PkDec(Y,ENC$_k${t},$\overline{k}$) ⊃ $\hat{Y}$=$\hat{X}$	a principal needs to possess the private key in order to decrypt a term encrypted with the corresponding public key.
VER	Honest($\hat{X}$) ∧ Verify(Y,SIG$_{\overline{k}}${t},k) ∧ $\hat{X} \neq \hat{Y}$ ⊃ ∃X.Send(X,m) ∧ Contains(m,SIG$_{\overline{k}}${t})	a principal can not deny signing a term (unforgeability of signatures).
ENC0	[t' := enc t,k;]$_X$ Enc(X,t,k)	if a principal has encrypted term t using key k in some role, then the corresponding predicate asserting that the encryption has occurred is true
ENC1	Has(X,t)∧Has(X,k)⊃ Computes(X,ENC$_k${t}) ⊃ Has(X,ENC$_k${t})	principal $\hat{X}$ can encrypt a term t using a key k if the term and the key are known.
ENC2	π(X,t,k) [a]$_X$ π(X,t,k), for $\pi \in$ {Enc,¬Enc}, where either a ≠ enc or a = (enc q,k') such that (q,k') ≠ (m,k)	the predicate asserting thread X has (not) encrypted term t using key k remains true after any action that does not encrypt a term using a key that unified with t and k respectively.
ENC3	Computes(X,ENC$_k${t}) ⊃ Has(X,ENC$_k${t}) ⊃ Has(X,k) ∧ Has(X,t)	if a principal has encrypted term t using key k, then the principal possesses both t and k.
ENC4	SymDec(X,ENC$_k${t},k) ⊃ ∃Y.Send(Y,m) ∧ Contains(m,ENC$_k${t})	If $\hat{X}$ can decrypt a term that is encrypted using a secret key shared with $\hat{Y}$, then it must be $\hat{Y}$ that encrypted and sent the term in the past.
PENC	PKDec(X,ENC$_k${t},$\overline{k}$) ⊃ ∃Z.Send(Z,m) ∧ Contains(m,ENC$_k${t})	if $\hat{X}$ can decrypt a term using its private key, then there must be another principal $\hat{Z}$ that encrypted and sent the term in the past.

Secrecy axioms:

SAF1 SafeMsg(m$_0$.m$_1$,s,k) $\equiv$ SafeMsg(m$_0$,s,k) $\wedge$ SafeMsg(m$_1$,s,k)

SAF2 SafeMsg(ENC$_k${m},s,k) $\equiv$ SafeMsg(m,s,k) $\vee$ k $\in$ K

SAF3 SafeMsg(ENC$_k${m},s,$\overline{k}$) $\equiv$ SafeMsg(m,s,k) $\vee$ k $\in$ K

SAF4 SafeMsg(HASH{m},s,k)

SAF5 SafeMsg(HASH$_k${m},s,k)

SAF6 SafeMsg(SIG$_{\overline{k}}${m},s,k)

SH0 Safe(m',HASH$_k${m}) , m' is an atomic term different from m

SH1 Safe(m'.m",HASH$_k${m}) $\equiv$ Safe(m',HASH$_k${m}) $\wedge$ Safe(m",HASH$_k${m})

SH2 Safe(SIG$_{k'}${m'},HASH$_k${m}) $\equiv$ Safe(m',HASH$_k${m})

SH3 Safe(ENC$_{k'}${m'},HASH$_k${m}) $\equiv$ Safe(m',HASH$_k${m})

SH4 Safe(HASH$_{k'}${m'},HASH$_k${m}) $\equiv$ k$\neq$k' $\vee$ m'$\neq$m

SH5 Safe(ENC$_{\overline{k'}}${m'},HASH$_k${m}) $\equiv$ Safe(m',HASH$_k${m})

HPOS SafeNet(HASH$_k${m}) $\wedge$ Has(X,HASH$_k${m}) $\supset$ Has(X,k)

Induction rule:

NET $\forall$p $\in$ Q.$\forall$P $\in$ BS(p).

$$\frac{\text{SafeNet(s,k) } [\text{P}]_X \text{ Honest}(\hat{X}) \wedge \phi \supset \text{SendsSafeMsg(X,s,k)}}{Q \vdash \text{KOHonest(s,k)} \wedge \phi \supset \text{SafeNet(s,k)}}$$

NET0 SafeNet(s,k) $[]_X$ SendsSafeMsg(X,s,k)

NET1 SafeNet(s,k) [receive m]$_X$ SafeMsg(m,s,k)

NET2 SendsSafeMsg(X,s,k) [a]$_X$ SendsSafeMsg(X,s,k), where a is not a send.

NET3 SendsSafeMsg(X,s,k) [send m]$_X$ SafeMsg(m,s,k) $\supset$ SendsSafeMsg(X,s,k)

POS SafeNet(s,k) $\wedge$ Has(X,m) $\wedge$ $\neg$SafeMsg(m,s,k)$\supset$ $\exists$k $\in$ K.Has(X,k) $\vee$ New(X,s)

POSL
$$\frac{\psi \wedge \text{SafeNet(s,k) } [\text{P}]_X \text{ SendsSafeMsg(X,s,k)} \wedge \text{Has(Y,m)} \wedge \neg\text{SafeMsg(m,s,k)}}{\psi \wedge \text{SafeNet(s,k) } [\text{P}]_X \text{ } \exists\text{k} \in \text{K.Has(Y,k)} \vee \text{New(Y,s)}}$$

, where P is any basic sequence of actions

Honesty rule:

HON
$$\frac{\text{Start(X) } []_X \phi \qquad \forall\ \text{p} \in Q.\forall\ \text{P} \in \text{BS(p)}.\phi\ [\text{P}]_X\ \phi}{\text{Honest}(\hat{X}) \supset \phi}$$

, where P is any basic sequence of actions

Appendix F

Formal Verification using PCL

F.1 CRE-SK Phase.

This section presents a formal correctness proof of the CRE-SK phase using the PCL method. It first models the CRE-SK phase using a simple "protocol programming language". It then formulates the security properties that the phase ought to satisfy from the CN's point of view. Finally, it proves the correctness of the phase.

Modelling the CRE-SK phase

SK : MobileNode = (MN, $\hat{HA}$, $\hat{CN}$, LT_{BRem}, LT_{BReq}, HoA, CoA, CNA, Ack, K_{MN-HA})

[new Cookie; msg_1 := CoA, Cookie; send $\hat{MN}$, $\hat{CN}$, msg_1;

receive $\hat{CN}$, $\hat{MN}$, msg_2; match msg_2/Cookie, J, Token_2; match Cookie/Cookie;

new seq; ENC_3 := enc HoA || CoA || LT_{BRem} || CNA || seq || LT_{BReq} || J || Token_2, K_{MN-HA};

msg_3 := ENC_3; send $\hat{MN}$, $\hat{HA}$, msg_3;

receive $\hat{HA}$, $\hat{MN}$, msg_4; match msg_4/ENC_4; DEC_4 := dec ENC_4, K_{MN-HA};

match DEC_4/CNA, Seq, K_{BC1}; match seq/seq;

receive $\hat{CN}$, $\hat{MN}$, msg_6; match msg_6/seq, LT_{BGrant}, ENC_6, mac_{EBA}; match seq/seq;

K_{MN-CN} := dec ENC_6, K_{BC1};

match mac_{EBA}/hash(HoA || CoA || CNA || seq || LT_{BGrant}, K_{MN-CN});

new seq_{new} such that seq_{new} := succ(seq);

mac_{BCC} := hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});

msg_7 := HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; send $\hat{MN}$, $\hat{CN}$, msg_7;

receive $\hat{CN}$, $\hat{MN}$, msg_8; match msg_8/seq_{new}, LT_{BGrant}, mac_{BA}; match seq_{new}/seq_{new};

match mac_{BA}/hash(HoA || CoA || CNA || seq_{new} || LT_{BGrant}, K_{MN-CN});

]$_{MN}$

SK : HomeAgent = (HA, id_{HA}, LT_{BRem}, HoA, CoA, K_{MN-HA}, K_{HA-CN})

[receive $\hat{MN}$, $\hat{HA}$, msg_3; match msg_3/ENC_3; DEC_3 := dec ENC_3, K_{MN-HA};

match DEC_3/HoA, CoA, LT_{BRem}, CNA, seq, LT_{BReq}, J, Token_2;

match HoA/HoA; match CoA/CoA; match LT_{BRem}/LT_{BRem}; isLess(LT_{BReq}, LT_{BRem});

new N_{HA1}; K_{BC1} := hash(HoA || CoA || CNA || N_{HA1}, K_{HA-CN});

new N_{HA2}; K_{BC2} := hash(HoA || CoA || CNA || N_{HA2}, K_{HA-CN});

ENC_4 := enc CNA || seq || K_{BC1}, K_{MN-HA}; msg_4 := ENC_4; send $\hat{HA}$, $\hat{MN}$, msg_4;

mac_{EBC} := hash(HoA || CoA || id_{HA} || N_{HA1} || N_{HA2} || seq || LT_{BReq}, K_{BC2});

msg_5 := HoA, CoA, id_{HA}, N_{HA1}, N_{HA2}, seq, LT_{BReq}, J, Token_2, mac_{EBC}; send $\hat{HA}$, $\hat{CN}$, msg_5;

$]_{HA}$

SK : CorrespondentNode = (CN, CNA, LT_{BGrant}, K_{CN}, id_{HA}, K_{HA-CN})

[receive $\hat{MN}$, $\hat{CN}$, msg_1; match msg_1/CoA, Cookie;

new J; new N_J; Token_2 := hash(CoA || N_J || 1, K_{CN});

msg_2 := Cookie, J, Token_2; send $\hat{CN}$, $\hat{MN}$, msg_2;

receive $\hat{HA}$, $\hat{CN}$, msg_5; match msg_5/HoA, CoA, id_{HA}, N_{HA1}, N_{HA2}, seq, LT_{BReq}, J, Token_2, mac_{EBC};

match Token_2/hash(CoA || N_J || 1, K_{CN});

K_{BC1} := hash(HoA || CoA || CNA || N_{HA1}, K_{HA-CN});

K_{BC2} := hash(HoA || CoA || CNA || N_{HA2}, K_{HA-CN});

match mac_{EBC}/hash(HoA || CoA || id_{HA} || N_{HA1} || N_{HA2} || seq || LT_{BReq}, K_{BC2});

new K_{MN-CN}; ENC_6 := enc K_{MN-CN}, K_{BC1};

mac_{EBA} := hash(HoA || CoA || CNA || seq || LT_{BGrant}, K_{MN-CN});

msg_6 := seq, LT_{BGrant}, ENC_6, mac_{EBA}; send $\hat{CN}$, $\hat{MN}$, msg_6;

receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC};

match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new});

match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});

mac_{BA} := hash(HoA || CoA || CNA || seq_{new} || LT_{BGrant}, K_{MN-CN});

msg_8 := seq_{new}, LT_{BGrant}, mac_{BA}; send $\hat{CN}$,$\hat{MN}$, msg_8;

$]_{CN}$

Table F.1: CRE-SK phase written in PCL language

Security Properties

(1) Session Authentication for the CN

$\phi_{SK,CN-auth}$ ::= Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$
$\exists(\hat{MN} \wedge \hat{HA})$.ActionsInOrder(
Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1), Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2),
Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2), Send(MN, $\hat{MN}$, $\hat{HA}$, msg_3),
Receive(HA, $\hat{MN}$, $\hat{HA}$, msg_3), Send(HA, $\hat{HA}$, $\hat{MN}$, msg_4),
Send(HA, $\hat{HA}$, $\hat{CN}$, msg_5),
(Receive(MN, $\hat{HA}$, $\hat{MN}$, msg_4) $\wedge$ Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_5)),
Send(CN, $\hat{CN}$, $\hat{MN}$, msg_6), Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_6),
Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7), Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7),
Send(CN, $\hat{CN}$, $\hat{MN}$, msg_8))

(2) Key Secrecy for the CN

$\phi_{SK,CN-sec}$ ::= Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has($\hat{Z}$, K_{MN-CN})) $\supset$
$\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$

CN Security Guarantee (Session Authentication and Key Secrecy)

$\Gamma_{SK1} \wedge \Gamma_{SK2} \wedge \theta_{SK1} \wedge \theta_{SK2} \vdash$[SK : CorrespondentNode]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$)
$\wedge\ \hat{CN} \neq \hat{MN} \neq \hat{HA} \supset \phi_{SK,CN-auth} \wedge \phi_{SK,CN-sec}$

θ_{SK1} := (Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has($\hat{Z}$, K_{MN-HA})) $\supset$ $\hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$
θ_{SK2} := (Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has($\hat{Z}$, K_{HA-CN})) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}$
Γ_{SK1} := Computes($\hat{Z}$, HASH($\hat{Z}$, HoA || CoA || CNA || N_{HA1}, K_{HA-CN})) $\supset$
$\neg$(Send($\hat{Z}$, m) $\wedge$ Contains(m, HASH($\hat{Z}$, HoA || CoA || CNA || N_{HA1}, K_{HA-CN})))
Γ_{SK2} := Computes($\hat{Z}$, HASH($\hat{Z}$, HoA || CoA || CNA || N_{HA2}, K_{HA-CN})) $\supset$
$\neg$(Send($\hat{Z}$, m) $\wedge$ Contains(m, HASH($\hat{Z}$, HoA || CoA || CNA || N_{HA2}, K_{HA-CN})))

Table F.2: CRE-SK phase preconditions and invariants

APPENDIX F. FORMAL VERIFICATION USING PCL

Proof of Session Authentication for the CN

AA1, AR1, AA4 [SK : CorrespondentNode]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_1$) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_2$) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg$_5$) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_6$) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_7$) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_8$)

(1)

AN3 [new N$_J$; Token$_2$:= hash(CoA || N$_J$ || 1, K$_{CN}$);]$_{CN}$ Fresh(CN, Token$_2$)

(2)

(2), FS1, P1 [new J; new N$_J$; Token$_2$:= hash(CoA || N$_J$ || 1, K$_{CN}$); msg$_2$:= Cookie, J, Token$_2$; send $\hat{CN}$, $\hat{MN}$, msg$_2$;]$_{CN}$ FirstSend(CN, Token$_2$, msg$_2$)

(3)

(1), (3), FS2 [new J; new N$_J$; Token$_2$:= hash(CoA || N$_J$ || 1, K$_{CN}$); msg$_2$:= Cookie, J, Token$_2$; send $\hat{CN}$, $\hat{MN}$, msg$_2$;]$_{CN}$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg$_2$) $\wedge$ $\hat{MN} \neq \hat{CN}$ $\supset$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_2$) < Receive(MN, $\hat{CN}$, $\hat{MN}$, msg$_2$)

(4)

AR1, HASH3, Γ_{SK2} θ_{SK2} [receive $\hat{HA}$, $\hat{CN}$, msg$_5$; match msg$_5$/HoA, CoA, id$_{HA}$, N$_{HA1}$, N$_{HA2}$, seq, LT$_{BReq}$, J, Token$_2$, mac$_{EBC}$; match Token$_2$/hash(CoA || N$_J$ || 1, K$_{CN}$); K$_{BC1}$:= hash(HoA || CoA || CNA || N$_{HA1}$, K$_{HA-CN}$); K$_{BC2}$:= hash(HoA || CoA || CNA || N$_{HA2}$, K$_{HA-CN}$); match mac$_{EBC}$/hash(HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$, K$_{BC2}$);]$_{CN}$ Receive(CN, $\hat{HA}$, $\hat{CN}$, HoA, CoA, id$_{HA}$, N$_{HA1}$, N$_{HA2}$, seq, LT$_{BReq}$, J, Token$_2$, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$} $\supset$ ($\exists$X.Has(X, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) $\wedge$ Send(X, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$})) $\wedge$ (Send(X, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg$_5$))

(5)

(5), HASH4 Has(X, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) $\supset$ Computes(X, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) $\vee$ $\exists$Y,m.Computes(Y, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) $\wedge$ Send(Y, m) $\wedge$ Contains(m, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$})

(6)

(6), HASH1 Computes(Z, HASH$_{k_{BC2}}${HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$}) $\supset$ Has($\hat{Z}$, K$_{BC2}$) $\wedge$ Has($\hat{Z}$, HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$)

(7)

(7), Γ_{SK2} θ_{SK2} [receive $\hat{HA}$, $\hat{CN}$, msg$_5$; match msg$_5$/HoA, CoA, id$_{HA}$, N$_{HA1}$, N$_{HA2}$, seq, LT$_{BReq}$, J, Token$_2$, mac$_{EBC}$; match Token$_2$/hash(CoA || N$_J$ || 1, K$_{CN}$); K$_{BC1}$:= hash(HoA || CoA || CNA || N$_{HA1}$, K$_{HA-CN}$); K$_{BC2}$:= hash(HoA || CoA || CNA || N$_{HA2}$, K$_{HA-CN}$); match mac$_{EBC}$/hash(HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$, K$_{BC2}$);]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Has($\hat{Z}$, K$_{BC2}$) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}$

(8)

(5), (8) θ_{SK2} [receive $\hat{HA}$, $\hat{CN}$, msg$_5$; match msg$_5$/HoA, CoA, id$_{HA}$, N$_{HA1}$, N$_{HA2}$, seq, LT$_{BReq}$, J, Token$_2$, mac$_{EBC}$; match Token$_2$/hash(CoA || N$_J$ || 1, K$_{CN}$); K$_{BC1}$:= hash(HoA || CoA || CNA || N$_{HA1}$, K$_{HA-CN}$); K$_{BC2}$:= hash(HoA || CoA || CNA || N$_{HA2}$, K$_{HA-CN}$); match mac$_{EBC}$/hash(HoA || CoA || id$_{HA}$ || N$_{HA1}$ || N$_{HA2}$ || seq || LT$_{BReq}$, K$_{BC2}$);]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg$_5$) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg$_5$)

(9)

AN3, Γ_{SK1} [new K_{MN-CN}; ENC_6 := enc K_{MN-CN}, K_{BC1};]$_{CN}$ Fresh(CN, ENC_6)

(10)

(10), FS1, P1 [new K_{MN-CN}; ENC_6 := enc K_{MN-CN}, K_{BC1}; mac_{EBA} := hash(HoA || CoA || CNA || seq || LT_{BGrant}, K_{MN-CN}); msg_6 := seq, LT_{BGrant}, ENC_6, mac_{EBA}; send $\hat{CN}$, $\hat{MN}$, msg_6;]$_{CN}$ FirstSend(CN, ENC_6, msg_6)

(11)

(1), (11), FS2 [new K_{MN-CN}; ENC_6 := enc K_{MN-CN}, K_{BC1}; mac_{EBA} := hash(HoA || CoA || CNA || seq || LT_{BGrant}, K_{MN-CN}); msg_6 := seq, LT_{BGrant}, ENC_6, mac_{EBA}; send $\hat{CN}$, $\hat{MN}$, msg_6;]$_{CN}$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_6) $\wedge$ $\hat{MN} \neq \hat{CN}$ $\supset$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg_6) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_6)

(12)

AR1, HASH3, Γ_{SK1} θ_{SK1} $\wedge$ θ_{SK2} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, HoA, CoA, seq_{new}, LT_{BReq}, Ack, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack} $\supset$

($\exists$X.Has(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})) $\wedge$ (Send(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7))

(13)

(13), HASH4 Has(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Computes(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\vee$ $\exists$Y,m.Computes(Y, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(Y, m) $\wedge$ Contains(m, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})

(14)

(14), HASH1 Computes(Z, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Has($\hat{Z}$, K_{MN-CN}) $\wedge$ Has($\hat{Z}$, HoA || CoA || seq_{new} || LT_{BReq} || Ack)

(15)

(15), $\phi_{SK,CN-sec}$ θ_{SK1} $\wedge$ θ_{SK2} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Has($\hat{Z}$, K_{MN-CN}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$

(16)

(13), (16) θ_{SK1} $\wedge$ θ_{SK2} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Honest($\hat{MN}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7)

(17)

(1), HON Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2) $<$ Send(MN, $\hat{MN}$, $\hat{HA}$, msg_3) $<$ Receive(HA, $\hat{MN}$, $\hat{HA}$, msg_3) $<$ Send(HA, $\hat{HA}$, $\hat{MN}$, msg_4) $<$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg_5) $<$ (Receive(MN, $\hat{HA}$, $\hat{MN}$, msg_4) $\wedge$ Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_5)) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_6) $<$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7)

(18)

(1), (4), (9), (12), (17), (18) $\theta_{SK1} \wedge \theta_{SK2}$ [CorrespondentNode]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_1$) $<$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_2$) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg$_2$) $<$ Send(MN, $\hat{MN}$, $\hat{HA}$, msg$_3$) $<$ Receive(HA, $\hat{MN}$, $\hat{HA}$, msg$_3$) $<$ Send(HA, $\hat{HA}$, $\hat{MN}$, msg$_4$) $<$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg$_5$) $<$ (Receive(MN, $\hat{HA}$, $\hat{MN}$, msg$_4$) $\wedge$ Receive(CN, $\hat{HA}$, $\hat{CN}$, msg$_5$)) $<$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_6$) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg$_6$) $<$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg$_7$) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg$_7$) $<$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg$_8$)) $\supset$ $\phi_{SK,CN-auth}$ (19)

F.2 CRE-PK Phase.

This section presents a formal correctness proof of the CRE-PK phase using the PCL method. It first models the CRE-PK phase using a simple "protocol programming language". It then formulates the security properties that the phase ought to satisfy from the CN's point of view. Finally, it proves the correctness of the phase.

Modelling the CRE-PK phase

PK : MobileNode = (MN, $\hat{HA}$, $\hat{CN}$, LT$_{BRem}$, LT$_{BReq}$, HoA, CoA, CNA, Ack, K$_{MN-HA}$)

[new Cookie; msg$_1$:= CoA, Cookie; send $\hat{MN}$,$\hat{CN}$, msg$_1$;
receive $\hat{CN}$, $\hat{MN}$, msg$_2$; match msg$_2$/Cookie, J, Token$_2$; match Cookie/Cookie;
K$_{BM}$:= hash(Token$_2$); new seq;
ENC$_3$:= enc HoA || CoA || LT$_{BRem}$ || CNA || seq || LT$_{BReq}$ || J || K$_{BM}$, K$_{MN-HA}$;
msg$_3$:= ENC$_3$; send $\hat{MN}$, $\hat{HA}$, msg$_3$;
receive $\hat{HA}$, $\hat{MN}$, msg$_6$; match msg$_6$/ENC$_6$; DEC$_6$:= dec ENC$_6$, K$_{MN-HA}$;
match DEC$_6$/CNA, seq, LT$_{BGrant}$, K$_{MN-CN}$; match seq/seq;
new seq$_{new}$ such that seq$_{new}$:= succ(seq);
mac$_{BCC}$:= hash(HoA || CoA || seq$_{new}$ || LT$_{BReq}$ || Ack, K$_{MN-CN}$);
msg$_7$:= HoA, CoA, seq$_{new}$, LT$_{BReq}$, Ack, mac$_{BCC}$; send $\hat{MN}$, $\hat{CN}$, msg$_7$;
receive $\hat{CN}$, $\hat{MN}$, msg$_8$; match msg$_8$/seq$_{new}$, LT$_{BGrant}$, mac$_{BA}$; match seq$_{new}$/seq$_{new}$;
match mac$_{BA}$/hash(HoA || CoA || CNA || seq$_{new}$ || LT$_{BGrant}$, K$_{MN-CN}$);
]$_{MN}$

PK : HomeAgent = (HA, LT$_{BRem}$, HoA, CoA, K$_{MN-HA}$, sk$_H$, Cert$_H$)

[receive $\hat{MN}$, $\hat{HA}$, msg$_3$; match msg$_3$/ENC$_3$; DEC$_3$:= dec ENC$_3$, K$_{MN-HA}$;
match DEC$_3$/HoA, CoA, LT$_{BRem}$, CNA, seq, LT$_{BReq}$, J, K$_{BM}$;
match HoA/HoA; match CoA/CoA; match LT$_{BRem}$/LT$_{BRem}$; isLess(LT$_{BReq}$, LT$_{BRem}$);
new N$_{HA}$; mac$_{EBC}$:= hash(CoA || N$_{HA}$ || seq || LT$_{BReq}$, K$_{BM}$);

SIG_{sk_H} := sign(hash(CoA || N_{HA} || seq || LT_{BReq}, sk_H);
msg_4 := HoA, CoA, N_{HA}, seq, LT_{BReq}, J, mac_{EBC}, SIG_{sk_H}, Cert_H; send $\hat{HA}$, $\hat{CN}$, msg_4;
receive $\hat{CN}$, $\hat{HA}$, msg_5; match msg_5/N_{HA}, seq, LT_{BGrant}, ENC_5,mac_{EBA};
match N_{HA}/N_{HA}; match mac_{EBA}/hash(N_{HA} || seq || LT_{BGrant} || ENC_5, K_{BM});
keys := dec ENC_5, sk_H; match keys as K_{MN-CN} || K_{HA-CN};
ENC_6 := enc CNA || seq || LT_{BGrant} || K_{MN-CN}, K_{MN-HA};
msg_6 := ENC_6; send $\hat{HA}$, $\hat{MN}$, msg_6;
$]_{HA}$

PK : CorrespondentNode = (CN, CNA, LT_{BGrant}, K_{CN})
[receive $\hat{MN}$, $\hat{CN}$, msg_1; match msg_1/CoA, Cookie;
new J; new N_J; Token_2 := hash(CoA || N_J || 1, K_{CN});
msg_2 := Cookie, J, Token_2; send $\hat{CN}$, $\hat{MN}$, msg_2;
receive $\hat{HA}$, $\hat{CN}$, msg_4; match msg_4/HoA, CoA, N_{HA}, seq, LT_{BReq}, J, mac_{EBC}, SIG_{sk_H}, Cert_H;
Token_2 := hash(CoA || N_J || 1, K_{CN}); K_{BM} := hash(Token_2);
match mac_{EBC}/hash(CoA || N_{HA} || seq || LT_{BReq}, K_{BM});
verify SIG_{sk_H}, pk_H; new K_{MN-CN}; new K_{HA-CN};
ENC_5 := enc K_{MN-CN} || K_{HA-CN}, pk_H;
mac_{EBA} := hash(N_{HA} || seq || LT_{BGrant} || ENC_5, K_{BM});
msg_5 := N_{HA}, seq, LT_{BGrant}, ENC_5, mac_{EBA}; send $\hat{CN}$, $\hat{HA}$, msg_5;
receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC};
match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new});
match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});
mac_{BA} := hash(HoA || CoA || CNA || seq_{new} || LT_{BGrant}, K_{MN-CN});
msg_8 := seq_{new}, LT_{BGrant}, mac_{BA}; send $\hat{CN}$,$\hat{MN}$, msg_8;
$]_{CN}$

Table F.3: CRE-PK phase written in PCL language

Security Properties

(1) Session Authentication for the CN

$\phi_{SK,CN-auth}$ ::= Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$
$\exists(\hat{MN} \wedge \hat{HA})$.ActionsInOrder(
Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1), Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2),
Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2), Send(MN, $\hat{MN}$, $\hat{HA}$, msg_3),
Receive(HA, $\hat{MN}$, $\hat{HA}$, msg_3), Send(HA, $\hat{HA}$, $\hat{CN}$, msg_4),
Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_4), Send(CN, $\hat{CN}$, $\hat{HA}$, msg_5),
Receive(HA, $\hat{CN}$, $\hat{HA}$, msg_5), Send(HA, $\hat{HA}$, $\hat{MN}$, msg_6),
Receive(MN, $\hat{HA}$, $\hat{MN}$, msg_6), Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7),
Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7), Send(CN, $\hat{CN}$, $\hat{MN}$, msg_8))

(2) Key Secrecy for the CN

$$\phi_{PK,CN-sec1} ::= \text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{MN-CN}) \supset \hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$$

$$\phi_{PK,CN-sec2} ::= \text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{HA-CN}) \supset \hat{Z} = \hat{CN} \vee \hat{Z} = \hat{HA}$$

CN Security Guarantee (Session Authentication and Key Secrecy)

$$\theta_{PK} \vdash [\text{PK} : \text{CorrespondentNode}]_{CN}\ \text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \wedge \hat{CN} \neq \hat{MN} \neq \hat{HA} \supset \phi_{PK,CN-auth} \wedge \phi_{PK,CN-sec1} \wedge \phi_{PK,CN-sec2}$$

$\theta_{PK} := \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \wedge \text{Has}(\hat{Z}, \text{K}_{MN-HA}) \supset \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$

Table F.4: CRE-PK phase preconditions

Proof of Session Authentication for the CN

AA1, AR1, AA4	$[\text{PK} : \text{CorrespondentNode}]_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_4) < Send(CN, $\hat{CN}$, $\hat{HA}$, msg_5) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg_8)	(1)
AN3	[new N_J; Token_2 := hash(CoA || N_J || 1, K_{CN});]$_{CN}$ Fresh(CN, Token_2)	(2)
(2), FS1, P1	[new J; new N_J; Token_2 := hash(CoA || N_J || 1, K_{CN}); msg_2 := Cookie, J, Token_2; send $\hat{CN}$, $\hat{MN}$, msg_2;]$_{CN}$ FirstSend(CN, Token_2, msg_2)	(3)
(1), (3), FS2	[new J; new N_J; Token_2 := hash(CoA || N_J || 1, K_{CN}); msg_2 := Cookie, J, Token_2; send $\hat{CN}$, $\hat{MN}$, msg_2;]$_{CN}$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2) $\wedge$ $\hat{MN} \neq \hat{CN}$ $\supset$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2) < Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2)	(4)
AR1, AR2, VER	[receive $\hat{HA}$, $\hat{CN}$, msg_4; match msg_4/HoA, CoA, N_{HA}, seq, LT_{BReq}, J, mac_{EBC}, SIG_{sk_H}, Cert_H; Token_2 := hash(CoA || N_J || 1, K_{CN}); K_{BM} := hash(Token_2); match mac_{EBC}/hash(CoA || N_{HA} || seq || LT_{BReq}, K_{BM}); verify SIG_{sk_H}, pk_H;]$_{CN}$ Receive(CN, $\hat{HA}$, $\hat{CN}$, HoA, CoA, N_{HA}, seq, LT_{BReq}, J, $\text{HASH}_{k_{BM}}$\{CoA || N_{HA} || seq || LT_{BReq}\}, SIG_{sk_H}\{CoA || N_{HA} || seq || LT_{BReq}\} $\supset$ ($\exists$X.Send(X, SIG_{sk_H}\{CoA || N_{HA} || seq || LT_{BReq}\}) $\wedge$ Send(X, SIG_{sk_H}\{CoA || N_{HA} || seq || LT_{BReq}\}) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_4))	(5)
(5)	[receive $\hat{HA}$, $\hat{CN}$, msg_4; match msg_4/HoA, CoA, N_{HA}, seq, LT_{BReq}, J, mac_{EBC}, SIG_{sk_H}, Cert_H; Token_2 := hash(CoA || N_J || 1, K_{CN}); K_{BM} := hash(Token_2); match mac_{EBC}/hash(CoA || N_{HA} || seq || LT_{BReq}, K_{BM}); verify SIG_{sk_H}, pk_H;]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg_4) < Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_4)	(6)

AN3 [new K_{MN-CN}; new K_{HA-CN}; ENC_5 := enc K_{MN-CN} || K_{HA-CN}, pk_H;]$_{CN}$ Fresh(CN, ENC_5)

(7)

(7), FS1, P1 [new K_{MN-CN}; new K_{HA-CN}; ENC_5 := enc K_{MN-CN} || K_{HA-CN}, pk_H; mac_{EBA} := hash(N_{HA} || seq || LT_{BGrant} || ENC_5, K_{BM}); msg_5 := N_{HA}, seq, LT_{BGrant}, ENC_5, mac_{EBA}; send $\hat{CN}$, $\hat{HA}$, msg_5;]$_{CN}$ FirstSend(CN, ENC_5, msg_5)

(8)

(1), (8), FS2 [new K_{MN-CN}; new K_{HA-CN}; ENC_5 := enc K_{MN-CN} || K_{HA-CN}, pk_H; mac_{EBA} := hash(N_{HA} || seq || LT_{BGrant} || ENC_5, K_{BM}); msg_5 := N_{HA}, seq, LT_{BGrant}, ENC_5, mac_{EBA}; send $\hat{CN}$, $\hat{HA}$, msg_5;]$_{CN}$ Receive(HA, $\hat{CN}$, $\hat{HA}$, msg_5) $\wedge$ $\hat{HA} \neq \hat{CN}$ $\supset$ Send(CN, $\hat{CN}$, $\hat{HA}$, msg_5) $<$ Receive(HA, $\hat{CN}$, $\hat{HA}$, msg_5)

(9)

AR1, HASH3 θ_{PK} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, HoA, CoA, seq_{new}, LT_{BReq}, Ack, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack} $\supset$

($\exists$X.Has(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})) $\wedge$ (Send(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7))

(10)

(10), HASH4 Has(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Computes(X, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\vee$ $\exists$Y,m.Computes(Y, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(Y, m) $\wedge$ Contains(m, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})

(11)

(11), HASH1 Computes(Z, $HASH_{k_{MN-CN}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Has($\hat{Z}$, K_{MN-CN}) $\wedge$ Has($\hat{Z}$, HoA || CoA || seq_{new} || LT_{BReq} || Ack)

(12)

(12), $\phi_{PK,CN-sec1}$ θ_{PK} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Has($\hat{Z}$, K_{MN-CN}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN} \vee \hat{Z} = \hat{HA}$

(13)

(10), (13) θ_{PK} [receive $\hat{MN}$, $\hat{CN}$, msg_7; match msg_7/HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BCC}; match HoA/HoA; match CoA/CoA; isLess(seq, seq_{new}); match mac_{BCC}/hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{MN-CN});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{HA}$) $\wedge$ Honest($\hat{MN}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_7)

(14)

(1), HON Honest($\hat{MN}$) $\wedge$ Honest($\hat{HA}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1) $<$ Receive(MN, $\hat{CN}$, $\hat{MN}$, msg_2) $<$ Send(MN, $\hat{MN}$, $\hat{HA}$, msg_3) $<$ Receive(HA, $\hat{MN}$, $\hat{HA}$, msg_3) $<$ Send(HA, $\hat{HA}$, $\hat{CN}$, msg_4) $<$ Receive(CN, $\hat{HA}$, $\hat{CN}$, msg_4)) $<$ Receive(HA, $\hat{CN}$, $\hat{HA}$, msg_5) $<$ Send(HA, $\hat{HA}$, $\hat{MN}$, msg_6) $<$ Receive(MN, $\hat{HA}$, $\hat{MN}$, msg_6) $<$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_7)

(15)

(1), (4), (6), (9), (14), (15)

$$\theta_{PK}\ [\text{CorrespondentNode}]_{CN}\ \text{Honest}(\hat{CN}) \wedge \text{Honest}(\hat{MN}) \wedge \text{Honest}(\hat{HA}) \supset \text{Receive}(\text{CN}, \hat{MN}, \hat{CN}, \text{msg}_1) < \text{Send}(\text{CN}, \hat{CN}, \hat{MN}, \text{msg}_2) < \text{Receive}(\text{MN}, \hat{CN}, \hat{MN}, \text{msg}_2) < \text{Send}(\text{MN}, \hat{MN}, \hat{HA}, \text{msg}_3) < \text{Receive}(\text{HA}, \hat{MN}, \hat{HA}, \text{msg}_3) < \text{Send}(\text{HA}, \hat{HA}, \hat{CN}, \text{msg}_4) < \text{Receive}(\text{CN}, \hat{HA}, \hat{CN}, \text{msg}_4) < \text{Send}(\text{CN}, \hat{CN}, \hat{HA}, \text{msg}_5) < \text{Receive}(\text{HA}, \hat{CN}, \hat{HA}, \text{msg}_5) < \text{Send}(\text{HA}, \hat{HA}, \hat{MN}, \text{msg}_6) < \text{Receive}(\text{MN}, \hat{HA}, \hat{MN}, \text{msg}_6) < \text{Send}(\text{MN}, \hat{MN}, \hat{CN}, \text{msg}_7) < \text{Receive}(\text{CN}, \hat{MN}, \hat{CN}, \text{msg}_7) < \text{Send}(\text{CN}, \hat{CN}, \hat{MN}, \text{msg}_8) \supset \phi_{PK,CN-auth} \tag{16}$$

F.3 DEL Phase.

This section presents a formal correctness proof of the DEL phase using the PCL method. It first models the DEL phase using a simple "protocol programming language". It then formulates the security properties that the phase ought to satisfy from the CN's point of view. Finally, it proves the correctness of the phase.

Modelling the DEL phase

DEL : MobileNode = (MN, $\hat{CN}$, LT_{BReq}, HoA, CoA, CNA, Ack, seq, K_{MN-CN})
[new seq_{new} such that seq_{new} := succ(seq); CoA := HoA; LT_{BReq} := 0;
K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN});
mac_{BU} := hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{BM});
msg_1 := HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BU}; send $\hat{MN}$, $\hat{CN}$, msg_1;
receive $\hat{CN}$, $\hat{MN}$, msg_2; match msg_2 / seq_{new}, mac_{BA}; match seq_{new} / seq_{new};
match mac_{BA} / hash(HoA || CNA || seq_{new}, K_{BM});
$]_{MN}$

DEL : CorrespondentNode = (CN, CNA, K_{MN-CN})
[receive $\hat{MN}$, $\hat{CN}$, msg_1;
match msg_1 / HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BU};
isLess(seq, seq_{new}); match CoA/HoA; match LT_{BReq}/0;
K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN});
match mac_{BU} / hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{BM});
mac_{BA} := hash(HoA || CNA || seq_{new}, K_{BM});
msg_2 := seq_{new}, mac_{BA}; send $\hat{CN}$, $\hat{MN}$, msg_2;
$]_{CN}$

Table F.5: DEL phase written in PCL language

APPENDIX F. FORMAL VERIFICATION USING PCL

Security Properties

(1) Session Authentication for the CN

$\phi_{DEL,CN-auth}$ ::= Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\supset$

$\exists\ \hat{MN}$.ActionsInOrder(

Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1),

Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1),

Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2))

(2) Key Secrecy for the CN

$\phi_{DEL,CN-sec}$ ::= Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Has($\hat{Z}$, K_{BM}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}$

CN Security Guarantee (Session Authentication and Key Secrecy)

$\Gamma_{DEL} \wedge \theta_{DEL1} \vdash$[DEL : CorrespondentNode]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ $\hat{CN} \neq \hat{MN}$

$\supset \phi_{DEL,CN-auth} \wedge \phi_{DEL,CN-sec}$

θ_{DEL} := (Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Has($\hat{Z}$, K_{MN-CN})) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}$

Γ_{DEL} := Computes($\hat{Z}$, HASH($\hat{Z}$, HoA || CoA || CNA || seq_{new}, K_{MN-CN})) $\supset$

$\neg$(Send($\hat{Z}$, m) $\wedge$ Contains(m, HASH($\hat{Z}$, HoA || CoA || CNA || seq_{new}, K_{MN-CN})))

Table F.6: DEL phase preconditions and invariants

Proof of Session Authentication for the CN

AA1, AR1, AA4 [DEL : CorrespondentNode]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1) < Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2)

(1)

AR1, HASH3, Γ_{DEL} θ_{DEL} [receive $\hat{MN}$, $\hat{CN}$, msg_1; match msg_1 / HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BU}; isLess(seq, seq_{new}); match CoA/HoA; match LT_{BReq}/0; K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN}); match mac_{BU} / hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{BM});]$_{CN}$ Receive(CN, $\hat{MN}$, $\hat{CN}$, HoA, CoA, seq_{new}, LT_{BReq}, Ack, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack} $\supset$

(∃X.Has(X, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(X, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})) $\wedge$ (Send(X, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) < Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1))

(2)

(2), HASH4 Has(X, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Computes(X, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\vee$ ∃Y,m.Computes(Y, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\wedge$ Send(Y, m) $\wedge$ Contains(m, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack})

(3)

(3), HASH1 Computes(Z, $\text{HASH}_{k_{BM}}${HoA || CoA || seq_{new} || LT_{BReq} || Ack}) $\supset$ Has($\hat{Z}$, K_{BM}) $\wedge$ Has($\hat{Z}$, HoA || CoA || seq_{new} || LT_{BReq} || Ack)

(4)

(4), $\phi_{DEL,CN-sec}$ θ_{DEL} [receive $\hat{MN}$, $\hat{CN}$, msg_1; match msg_1 / HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BU}; isLess(seq, seq_{new}); match CoA/HoA; match LT_{BReq}/0; K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN}); match mac_{BU} / hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{BM});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\wedge$ Has($\hat{Z}$, K_{BM}) $\supset$ $\hat{Z} = \hat{CN} \vee \hat{Z} = \hat{MN}$

(5)

(2), (5) θ_{DEL} [receive $\hat{MN}$, $\hat{CN}$, msg_1; match msg_1 / HoA, CoA, seq_{new}, LT_{BReq}, Ack, mac_{BU}; isLess(seq, seq_{new}); match CoA/HoA; match LT_{BReq}/0; K_{BM} := hash(HoA || CoA || CNA || seq_{new}, K_{MN-CN}); match mac_{BU} / hash(HoA || CoA || seq_{new} || LT_{BReq} || Ack, K_{BM});]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1)

(6)

(1), (6) θ_{DEL} [CorrespondentNode]$_{CN}$ Honest($\hat{CN}$) $\wedge$ Honest($\hat{MN}$) $\supset$ Send(MN, $\hat{MN}$, $\hat{CN}$, msg_1) $<$ Receive(CN, $\hat{MN}$, $\hat{CN}$, msg_1) $<$ Send(CN, $\hat{CN}$, $\hat{MN}$, msg_2))) $\supset$ $\phi_{DEL,CN-auth}$

(7)

Appendix G

Formal Protocol Verification using Casper/FDR2

G.1 CRE-SK Phase.

This section presents a formal verification of the CRE-SK phase using the Casper-/FDR2 model checker. It presents a Casper script that specifies the CRE-SK phase and its security properties.

 # Free variables
mn : MNode
cn : CNode
ha : HomeAgent
hoa, coa : IPv6Address
cookie, token, nha1 : Nonce
kmnha, khacn, kmncn : SessionKey
k : SessionKey x IPv6Address x IPv6Address x Nonce → Key
kbc1 : Key
InverseKeys = (kmnha, kmnha), (khacn, khacn), (kmncn, kmncn), (kbc1, kbc1), (k, k)
HMAC : HashFunction
#Processes
MOBILENODE(mn, cn, ha, hoa, coa, cookie, kmnha) knows HMAC
CORRESNODE(cn, khacn, kmncn, token, nha1) knows HMAC, k
HOMEAGENT(ha, cn, hoa, coa, nha1, kmnha, khacn) knows HMAC, k
 # Protocol description
0. → mn : cn
1. mn → cn : cookie
2. cn → mn : cookie, token
3. mn → ha : {hoa, coa}{kmnha}
<kbc1 := k(khacn, hoa, coa, nha1)>
4. ha → mn : {hoa, coa, kbc1}{kmnha}
5. ha → cn : hoa, coa, nha1, HMAC(khacn, hoa, coa, nha1)
<kbc1 := k(khacn, hoa, coa, nha1)>

6. cn → mn : hoa, coa, {kmncn}{kbc1}, HMAC(kbc1, hoa, coa)
7. mn → cn : hoa, coa, HMAC(kmncn, hoa, coa)
8. cn → mn : hoa, coa, HMAC(kmncn, hoa, coa)

# **Specification**
Agreement(mn, cn,[hoa, coa])
Agreement(ha, cn,[hoa, coa])
Secret(cn, kmncn,[mn, ha])

Actual variables
MN, Mallory : MNode
CN : CNode
HA : HomeAgent
HoA, CoA, MalA : IPv6Address
Cookie, Token, Nha1, Nm : Nonce
Kmnha, Khacn, Kmncn, Kmala : SessionKey
Kbc1 : Key
InverseKeys = (Kmnha, Kmnha), (Khacn, Khacn), (Kmncn, Kmncn), (Kbc1, Kbc1), (Kmala, Kmala)

Functions
symbolic k

# **System**
MOBILENODE(MN, CN, HA, HoA, CoA, Cookie, Kmnha)
CORRESNODE(CN, Khacn, Kmncn, Token, Nha1)
HOMEAGENT(HA, CN, HoA, CoA, Nha1, Kmnha, Khacn)

# **Intruder Information**
Intruder = Mallory
IntruderKnowledge = {MN, CN, HA, Mallory, HoA, CoA, MalA, Nm, Kmala}

G.2 CRE-PK Phase.

This section presents a formal verification of the CRE-PK phase using the Casper-/FDR2 model checker. It presents a Casper script that specifies the CRE-PK phase and its security properties.

Free variables
mn : MNode
cn : CNode
ha : HomeAgent
hoa, coa : IPv6Address
cookie, token : Nonce
pkh : PubKey
skh : SecKey
kmnha, khacn, kmncn : SessionKey
InverseKeys = (pkh, skh), (kmnha, kmnha), (khacn, khacn), (kmncn, kmncn)
HMAC : HashFunction

Processes
MOBILENODE(mn, cn, ha, hoa, coa, cookie, kmnha) knows HMAC
CORRESNODE(cn, token, khacn, kmncn, pkh) knows HMAC
HOMEAGENT(ha, hoa, coa, kmnha, pkh, skh) knows HMAC

Protocol description
0. → mn : cn

1. mn → cn : cookie
2. cn → mn : cookie, token
3. mn → ha : {hoa, coa, cn, token}{kmnha}
4. ha → cn : {hoa, coa, token}{skh}
5. cn → ha : hoa, coa, {khacn, kmncn}{pkh}
6. ha → mn : {hoa, coa, cn, kmncn}{kmnha}
7. mn → cn : hoa, coa, HMAC(kmncn, hoa, coa)
8. cn → mn : hoa, coa, HMAC(kmncn, hoa, coa)

# **Specification**
Agreement(mn, cn, [hoa, coa, kmncn])
Agreement(ha, cn, [hoa, coa, khacn])
Secret(ha, khacn, [cn])
Secret(mn, kmncn, [cn])
# **Actual variables** MN, Mallory : MNode
CN : CNode
HA : HomeAgent
HoA, CoA, MalA : IPv6Address
Cookie, Token, Nha1, Nha2, Nm : Nonce
Pkh : PubKey
Skh : SecKey
Kmnha, Khacn, Kmncn, Kmala : SessionKey
InverseKeys = (Kmnha, Kmnha),(Khacn, Khacn),(Kmncn, Kmncn),(Pkh, Skh),(Kmala, Kmala)
# **System**
MOBILENODE(MN, CN, HA, HoA, CoA, Cookie, Kmnha)
CORRESNODE(CN, Token, Khacn, Kmncn, Pkh)
HOMEAGENT(HA, HoA, CoA, Kmnha, Pkh, Skh)
# **Intruder Information**
Intruder = Mallory
IntruderKnowledge = {MN, CN, HA, Mallory, HoA, CoA, MalA, Nm, Kmala}

G.3 DEL Phase.

This section presents a formal verification of the DEL phase using the Casper-/FDR2 model checker. It presents a Casper script that specifies the DEL phase and its security properties.

Free variables
mn : MNode
cn : CNode
hoa, coa : IPv6Address
seq1 : SequenceNumber
kmncn : SessionKey
k : SessionKey x IPv6Address → Key
kbm : Key
InverseKeys = (kmncn, kmncn), (kbm, kbm)
HMAC : HashFunction
Processes
MOBILENODE(mn, cn, hoa, coa, seq1, kmncn) knows HMAC, k
CORRESNODE(cn, kmncn) knows HMAC, k

 # Protocol description
0. → mn : cn
<kbm := k(kmncn, hoa) >
1. mn → cn : hoa, coa, seq1, HMAC (kmncn, hoa, coa, seq1)
<kbm := k(kmncn, hoa) >
2. cn → mn : hoa, HMAC (kmncn, hoa)
 # Specification
Agreement(mn, cn, [hoa, coa, kbm])
Secret(mn, kbm, [cn])
 # Actual variables
MN, Mallory : MNode
CN : CNode
HoA, CoA, MalA : IPv6Address
Seq1, Seqm : SequenceNumber
Kmncn, Kmala : SessionKey
Kbm : Key
InverseKeys = (Kmncn, Kmncn), (Kbm, Kbm), (Kmala, Kmala)
 # Functions
symbolic k
 # System
MOBILENODE(MN, CN, HoA, CoA, Seq1, Kmncn)
CORRESNODE(CN, Kmncn)
 # Intruder Information
Intruder = Mallory
IntruderKnowledge = {MN, CN, Mallory, HoA, CoA, Seqm, MalA, Kmala}