EFFICIENT SECURITY PROTOCOLS FOR FAST HANDOVERS IN WIRELESS MESH NETWORKS

CELIA LI

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN
COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

JULY 2015

Abstract

Wireless mesh networks (WMNs) are gaining popularity as a flexible and inexpensive replacement for Ethernet-based infrastructures. As the use of mobile devices such as smart phones and tablets is becoming ubiquitous, mobile clients should be guaranteed uninterrupted connectivity and services as they move from one access point to another within a WMN or between networks. To that end, we propose a novel security framework that consists of a new architecture, trust models, and protocols to offer mobile clients seamless and fast handovers in WMNs. The framework provides a dynamic, flexible, resource-efficient, and secure platform for intra-network and inter-network handovers in order to support real-time mobile applications in WMNs. In particular, we propose solutions to the following problems: authentication, key management, and group key management. We propose

(1) a suite of certificate-based authentication protocols that minimize the authentication delay during handovers from one access point to another within a network (intra-network authentication).

(2) a suite of key distribution and authentication protocols that minimize the authentication delay during handovers from one network to another (inter-network authentication).

(3) a new implementation of group key management at the data link layer in order to reduce the group key update latency from linear time (as currently done in IEEE 802.11 standards) to logarithmic time. This contributes towards minimizing the latency of the handover process for mobile members in a multicast or broadcast group.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Professor Uyen Trang Nguyen for her expert supervision. I highly appreciate the valuable advice, professional guidance, and consistent encouragement from Professor Nguyen through all stages of my study and research.

I am grateful to my thesis committee members, Professors Zhen Ming Jiang, Bobby Ma, Hongmei Zhu, Suprakash Datta, and Aijun An, for reviewing my thesis and providing their valuable insights.

I would like to thank my postgraduate fellow and friend, Farzana Yasmeen, for reviewing my thesis and providing useful and detailed comments.

I would like to dedicate this thesis to my family for their deepest love, continuous encouragement, and unconditional support.

Table of Contents

List of Tables

List of Figures

Publications from the Thesis

Refereed Journal Article

1. Celia Li, Uyen Trang Nguyen, Hoang Lan Nguyen and Nurul Huda, Efficient Authentication for Fast Handover in Wireless Mesh Networks, Elsevier Journal of Computer & Security, Vol. 37, pp. 124-142, September 2013.

Refereed Conference Proceedings

2. Celia Li and Uyen Trang Nguyen, Efficient Group Key Management in Wireless Local Area Networks, International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC 2012), Vancouver, Canada, June 2012.

3. Celia Li and Uyen Trang Nguyen, Fast Authentication for Mobility Support in Wireless Mesh Networks, IEEE Wireless Communications and Networking Conference (WCNC 2011), Cancun, Mexico, March 2011.

4. Celia Li and Uyen Trang Nguyen, Fast Authentication for Mobile Clients in Wireless

Mesh Networks, IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2010), Calgary, Canada, May 2010.

5. Celia Li and Uyen Trang Nguyen, Group Key Management in Wireless Mesh Networks, International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine '07, co-sponsored by IEEE and ICST (Institute for Computer Sciences, Social Informatics and Telecommunications Engineering)), Vancouver, Canada, August 2007.

Abbreviations

AAD	Average Authentication Delay
ACK	Acknowledgement
AES	Advanced Encryption Standard
AIAD	Average Inter-network Authentication Delay
AKDD	Average Key Distribution Delay
AP-AKA	Access Point - Authentication and Key Agreement
AS	Authentication Server
BMK	Border MAC Key
BMAP	Border Mesh Access Point
BSSID	Basic Service Set Identifier
CA	Certificate Authority
CBR	Constant Bit Rate
CMK	Client MAC Key
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance

CTS	Clear to Send
DCF	Distributed Coordination Function
DEK	Data Encryption Key
DIFS	Distributed Inter-Frame Space
DoS	Denial-of-Service
DSSS	Direct Sequence Spread Spectrum
EAP	Extensible Authentication Protocol
EAP-TLS	Extensible Authentication Protocol-Transport Layer Security
GKM	Group Key Management
GKS	Group Key Server
GSM	Global System for Mobile Communications
GTK	Group Transient Key
GPRS	General Packet Radio Service
HAP	Handover Authentication Protocol
IAPP	Inter-Access Point Protocol
IP	Internet Protocol
KCK	Key Confirmation Key
KDC	Key Distribution to Client
KDN	Key Distribution to neighbors
KEK	Key Encryption Key

KM	Key Management
KPDD	Key Pre-Distribution Delay
LAN	Local Area Network
LAP	Login Authentication Protocol
LKH	Logical Key Hierarchy
MAC	Message Authentication Code
MAD	Maximum Authentication Delay
MAP	Mesh Access Point
MIAD	Maximum Inter-network Authentication Delay
MKDD	Maximum Key Distribution Delay
MP	Mesh Point
MSDU	Media Access Control Service Data Unit
NBD	Neighboring BMAP Discovery
OFT	One-way Function Tree
OFDM	Orthogonal Frequency Division Multiplexing
PANA	Protocol for Carrying Authentication for Network Access
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
PMK	Pairwise Master Key
PTK	Pairwise Transient Key

PGP	Pretty Good Privacy
RADIUS	Remote Authentication Dial In User Service
RTS	Request to Send
SHA	Secure Hash Algorithm
SIFS	Short Inter-Frame Space
SIM	Subscriber Identity Module
TK	Temporal Key
UMTS	Universal Mobile Telecommunication System
VoIP	Voice over IP
WEP	Wired Equivalent Privacy
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WMN	Wireless Mesh Network

Chapter 1

Introduction

Wireless mesh networking is an emerging technology that supports many important applications such as Internet access provisioning in rural areas, ad hoc networking for emergency and disaster recovery, security surveillance, and information services in public transportation systems [1]. The technology enables networking capability where wiring or installing cables is difficult or expensive.

A WMN is dynamically self-organized and self-configured, with nodes in the network automatically establishing and maintaining mesh connectivity among themselves. This feature brings many benefits to WMNs such as low installation cost, large-scale deployment, fault-tolerance, and self-management. WMNs are gaining popularity as a promising technology for ubiquitous high speed network access, not only for their significant cost savings but also for their flexibility and resilience.

A wireless mesh network (WMN) consists of mesh clients and mesh points (MPs). Mesh clients can be static (e.g., desktops, database servers) or mobile hosts (e.g., cellular

phones, laptops, personal digital assistants [PDAs], tablets). The mesh points form a wireless mesh backbone to provide multi-hop connectivity from one mesh client to another or to the Internet. A subset of mesh points act as mesh access points (MAPs), connecting mesh clients to the WMN. MAPs can be mobile or static. In the thesis we assume that MAPs are mostly static, such as those used for Internet access provisioning in rural areas, municipal and metropolitan networking, and security surveillance [2]. A small number of mesh points work as gateways, connecting the WMN to the Internet. Figure 1.1 shows an example of a WMN.

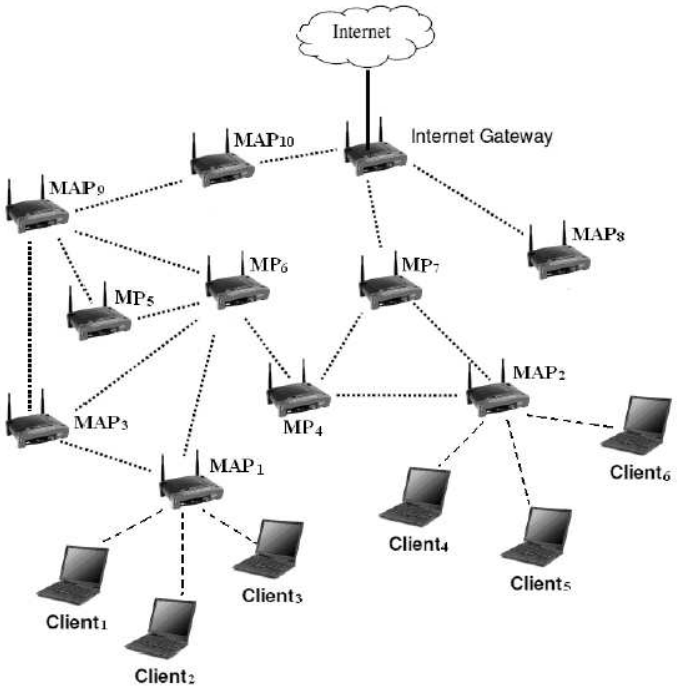


Figure 1.1: An example of a WMN

Mesh access points (MAPs) in a WMN are connected to other MAPs by one or more radio (wireless) links. Given a network of n nodes arranged in a string topology (i.e., a

straight line) and a carrier sense multiple access with collision avoidance (CSMA/CA)-based medium access control protocol such as IEEE 802.11, it has been shown through experiments that the throughput degrades approximately to $1/n$ of the raw channel bandwidth [3]. This experimental result implies that the longer a source-to-destination path (in terms of the number of wireless hops), the high the probability that a packet sent by the source to the destination will be lost or damaged. The path length thus has a significant impact on the performance of existing security protocols used in WMNs as will be discussed shortly.

In a WMN, a mobile device is connected to a MAP through a radio link. As a mobile client moves away from a MAP, it switches its connectivity to another MAP. This connectivity change involves a transition process called *handover*. When a mobile client roams from one MAP to another MAP within the same network, an *intra-network handover* takes place to provide the client with continued service. (In this dissertation, a network is defined as a set of points or nodes interconnected by communication paths and controlled by a single network operator. The term "networks" refers to different networks controlled by different network operators.) A mobile client may also switch connectivity from a MAP in one network to a MAP in another network, in which case an *inter-network handover* is required.

A handover typically involves the following operations: mobility management, authentication, authorization, key management, and accounting for billing purposes. If all these operations are performed via multi-hop wireless communications (e.g., in wireless

ad hoc networks such as WMNs), the handover latency may be unacceptably long, resulting in lost connectivity and thus service interruption. On the other hand, the current version of the wireless mesh networking standard IEEE 802.11s does not specify any mechanisms to support fast handovers for mobile clients. For example, a mesh client has to be authenticated by an authentication server via *multi-hop wireless communications*, which may result in long delay, low reliability, and thus potential service interruption.

We propose a novel security framework that consists of a new architecture, trust models, and protocols to enable seamless and fast handovers in WMNs. The framework provide a dynamic, flexible, resource-efficient and secure platform for intra-network and inter-network handovers in order to support real-time mobile applications in WMNs. In particular, we propose solutions to the following problems: authentication, key management, and group key management.

1.1 Motivations and Contributions of the Thesis

We propose the following secure and efficient solutions to support fast intra-network and inter-network handovers in WMNs:

1. We propose a novel trust model and certificates to support mobile clients moving from one MAP to another within a network. Based upon the proposed trust model, we design a suite of certificate-based authentication protocols that minimize the authentication delay during the handover process within a WMN in order to support fast intra-network handovers.

2. We design a new architecture, a trust model and certificates to support fast authentication when a mobile client moves from one network to another. Based on the proposed inter-network architecture, trust model and certificates, we propose a suite of security protocols for authentication and key distribution to minimize inter-network authentication latency.

3. We propose a new implementation of group key management at the data link layer in order to reduce the key update latency from linear time (as currently done in the IEEE 802.11s standard) to logarithmic time. This contributes towards minimizing the latency of the handover process for members in a multicast or broadcast group.

Following sections detail the motivations and contributions of our work.

1.1.1 Efficient Authentication for Fast Intra-network Handovers in a Wireless Mesh Network

With the rapid growth of mobile services for handheld devices such as smart phones, tablets and laptops, Internet connectivity anytime, anywhere has become a necessity in every day life, business, education, and entertainment. While cellular networks effectively handle the handover problem using signaling embedded in their low-level protocols, there are currently no efficient, transparent handover solutions for IEEE 802.11-based wireless networks. At the moment, these networks, even if they give the appearance of continuous connectivity to mobile clients, provide connections that are in fact often interrupted when a client transfers from one access point to the next, because the handover delay can be

as long as several seconds [4]. For some applications (e.g. transferring files), this delay is acceptable; however, it is far too long for real-time traffic such as interactive voice over IP (VoIP) or video conferencing [5].

The current version of wireless mesh networking standards IEEE 802.11s does not specify any mechanisms to support fast handovers for mobile clients. A mesh client has to be authenticated by an authentication server via *multi-hop wireless communications*, which may result in long delays, low reliability and thus potential service interruption. A performance study of message transmission delay in IEEE 802.11-based mesh networks by Srivatsa and Xie [7] shows that as the number of wireless hops between two routers increases from one to five, the delay of a message between the routers increases from 0.15 seconds to 0.8 seconds. Since the authentication process involves several messages (e.g., nine messages in the EAP-TLS protocol used by 802.11s), the handover latency may be several seconds long, which is not acceptable for real-time applications such as VoIP, newscast, and stock quote distributions.

Our work contributes towards extending the IEEE 802.11s standards to support fast handovers in a wireless mesh network. In particular, we focus on fast authentication during the handover process as well as during the initial login time. We propose a new trust model and accompanying certificates, based upon which our proposed authentication protocols are designed. We develop certificate-based authentication protocols that are resource-efficient and resilient to attacks. The authentication server does not need to be involved in the handover authentication. Instead, a new mesh access point di-

rectly authenticates a mobile client during the handover process using certificates issued by the current access point. Numerical analysis and simulation results show that our login authentication protocol improves the authentication latency of IEEE 802.11s, and our handover authentication protocol supports fast authentication during the handover process, which is lacking in IEEE 802.11s.

1.1.2 Efficient Authentication and Key Distribution for Fast Inter-Network Handovers among Multiple Wireless Mesh Networks

The home-foreign-domain model [9] provides a key distribution method and an authentication approach for inter-network handovers in the Global System for Mobile Communications (GSM) [10], Universal Mobile Telecommunication System [11], and mobile IP networks [12]. The trust between a client and a foreign network is based on a bilateral roaming agreement and real-time interactions between the client's home network and the foreign network. To access a foreign network, a client's subscriber identity module (SIM) card and the authentication center of the client's home network are pre-installed with a shared secret key K. The foreign network must communicate with the client's home network to obtain the shared key K in order to authenticate the mobile client.

In mobile IP and cellular networks, the client and the foreign base station (access point) communicate via a one-hop wireless link. Furthermore, the communication between the foreign base station and the client's home network is via high speed wired connections. Therefore, the handover latency is acceptable in these networks.

In wireless ad hoc networks such as WMNs, the connection between two access points (or between an access point and its authentication server) could all be wireless and through multiple hops. As mentioned above, the delivery latency of a message via a five-hop wireless connection can be up to 0.8 seconds. Given several messages involved in an authentication, the authentication latency incurred by the home-foreign domain handover approach can be unacceptably long in a wireless ad hoc network.

The above limitation of the home-foreign domain handover approach necessitates he development of a new security architecture and algorithms for handovers among multiple WMNs. We propose a scalable system architecture, a new trust establishment model, and certificates for inter-network handovers between WMNs. Our new architecture and trust model do not require a mobile device to be bound to its home network. A mobile client and a foreign network trust each other through via their respective certificates issued by certificate agents. Thus, a foreign network does not need to communicate with a client's home network when authenticating a client, but instead verifies the validity of an inter-network transfer certificate.

We propose a suite of fast authentication and key distribution protocols that are resource-efficient and resilient to attacks for inter-network handovers. The authentication server of a foreign network does not need to be involved in the handover authentication process. Instead, a MAP in the foreign network directly authenticates a mobile client, which avoids multi-hop wireless communications and thus minimizes the authentication latency. Numerical analysis and simulation results show that our inter-network

authentication and key distribution protocols significantly reduce the authentication latency during the inter-network handover process compared with the home-foreign domain authentication approach.

1.1.3 Efficient Group Key Management for Wireless Mesh Access Points

Multicast is a form of communication that delivers information from a source to a set of destinations simultaneously in an efficient manner; the messages are delivered over each link of the network only once and only duplicated at branch points, where the links to the destinations split. Important applications of multicast include distribution of financial data, billing records, software, newspapers, pay-per-view movies; audio/video conferencing; distance learning; and distributed online games. Although multicast is required to support many important applications, research on multicast in WMNs is still in its infancy. We address one of the most essential issues of multicast in WMNs – security. In particular, we focus on the issue of group key management in WMNs.

Given a multicast group, in order to ensure that only authorized users can access the multicast data, the data are encrypted using a cryptographic key known as the group key. The group key is known only to authenticated and authorized members of the multicast group. Every time a membership change occurs, the group key must be changed to ensure backward and forward secrecy. Backwards secrecy guarantees that a new user joining the multicast group does not have access to any old keys. This ensures that a member cannot decrypt messages sent before it joins the group. Forward secrecy requires that a member

leaving the group does not have access to any future keys. This ensures that a member cannot decrypt future messages after it leaves the group. Group key management refers to the actions taken to update and distribute the group key upon members joining and leaving a multicast group.

IEEE 802.11s, the wireless mesh network standard, employs the same security architecture as IEEE 802.11i, the standard specifying security mechanisms for wireless local area networks (WLANs). In a WMN, a mesh access point (MAP) shares a pairwise key with each mobile device it is connected to. In addition, the MAP shares a group key with all the trusted mobile devices associated with the MAP so that it can send multicast and broadcast data to this trusted group. When a client joins (or leaves) the WMNs, the MAP has to update the group key to ensure backward (or forward) secrecy. The MAP generates a new group key, encrypts it using the pairwise key the MAP shares with each trusted member, and sends the new group key to the members one by one. Thus, the communication cost of a group key update is $O(n)$, where n is the number of associated members. As the group size becomes large, the rekeying latency becomes unacceptable, especially for real-time applications.

We apply the logical key hierarchy (LKH) [13] and one-way function tree (OFT) [14] algorithms to group key management (GKM) at the data link layer to improve its rekeying performance. Specifically, we incorporate the LKH and OFT algorithms into the GKM operations of an access point and its associated mobile devices. We provide numerical analysis that shows that the LKH and OFT algorithms reduce the rekeying latency in

WMNs from linear time to logarithmic time. We also present simulation results obtained from various network conditions under realistic settings that show that the LKH and OFT algorithms dramatically improve the rekeying performance compared with the original GKM operations of IEEE 802.11.

1.2 Thesis Organization

The remainder of the thesis is organized as follows. We provide a review of related work in Chapter 2. In Chapter 3, we present a suite of authentication protocols that support fast intra-network handovers in a WMN. In Chapter 4, we present a suite of new security protocols to minimize the inter-network handover latency. In Chapter 5, we present a new implementation of group key management at the data link layer to minimize the key update latency. We conclude the thesis, and outline future research directions in Chapter 6.

Chapter 2

Literature Review

This chapter begin with background information on wireless communication, followed by a review of cryptographic techniques and security threats. We then provide a review of security models and trust management in wireless networks, followed by an overview of the IEEE 802.11i security standard. Lastly, we present a survey of security protocols for intra-network and inter-network handovers, followed by a literature review of group key management.

2.1 Overview of Wireless Communications

In this section, we provide an overview of wireless communications with an emphasis on transmission and interference range, hidden terminal and exposed terminal problems, medium access mechanisms and spread spectrum technologies.

2.1.1 Transmission Range and Interference Range

Successful transmission and reception of a signal mainly depends on the transmission range and the interference range [27]. Assuming a concentric dissemination, a transmission range constitutes the range where a signal can be correctly received. An interference range is the range within which stations in receive mode can be "interfered with" by an unrelated transmitter and thus suffer a loss of packets.

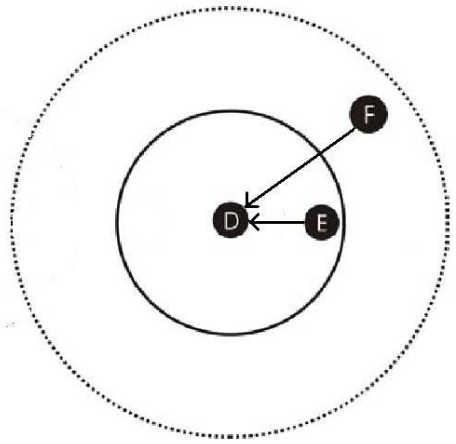


Figure 2.1: Transmission range and interference range

Figure 2.1 shows an example of the transmission and interference ranges of a node D where the solid line denotes its transmission range, and the dotted line depicts its interference range. Assume that node D and E are within each other's transmission range and node F is between D's transmission and interference range. If node E and node F send packets at the same time to node D, node F can disturb the transmission from node E to node D. The interference range of node D therefore includes all positions of node F that can damage E's packets being sent to node D.

While the transmission range of a node mainly depends on its signal propagation function, its interference range may not to be able to be foreseen, and depends on parameters such as signal strength or reflection, and may change frequently. This effect is known from practical implementations on data transmission.

2.1.2 Hidden Terminal and Exposed Terminal Problems

In wireless communications, a channel represents a band of frequencies over which signals are carried from sources to destinations. A radio antenna is capable of sending or receiving signals on a specific channel. Wireless is a shared medium and wireless devices share the same air space. This can lead to collisions if more than one device tries to send to the same receiver on the same channel.

Consider the scenario with three wireless mobile devices, as shown in Figure 2.2. The transmission range of A reaches B, but not C. The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches both A and C.

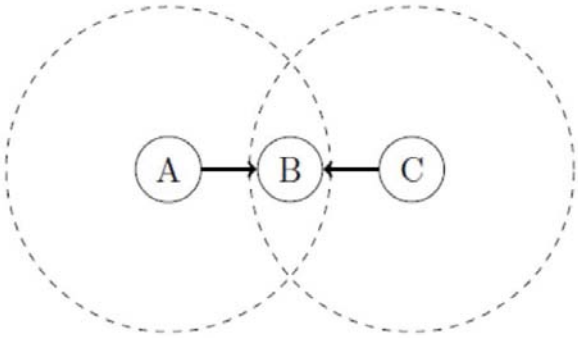


Figure 2.2: An example of hidden terminal problem

In the scenario shown in Figure 2.2, A starts sending packets to B. C does not receive this transmission. C also wants to send packets to B and senses the medium. The medium appears to be free as A is out of C's range and C cannot sense the packets being sent from A to B. Hence, C also starts sending packets simultaneously to B, causing a collision at B. A cannot detect this collision at B and continues with its transmission. A is *hidden* from C and vice versa.

While hidden terminals may cause collisions, the following problem may cause unnecessary delay. Assuming the same orientation as shown in Figure 2.2, consider the situation that B sends packets to A and C wants to transmit data to some other wireless mobile device outside the interference range of A and B. C senses the carrier and detects that the carrier is busy (B's signal). C postpones its transmission until it detects the medium is idle again. However, as A is outside the interference range of C, waiting is not necessary. A 'collision' at B from C's transmission to another device is too weak to propagate to A, and will not interfere with B's transmission to A. In this situation, C is *exposed* to B.

2.1.3 Medium Access Mechanisms

The IEEE 802.11 standard defines two methods in which an IEEE 802.11 radio card may gain control of the half-duplex medium. The default method, Distributed Coordination Function (DCF), is a random access method determining who gets to transmit on the wireless medium next. The other medium access control method called Point Coordi-

nation Function (PCF), where the access point briefly takes control of the medium and polls the stations served by the AP. DCF uses a contention-based algorithm to provide access to all traffic, which includes two methods, CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) and DCF with RTS/CTS (Request to Send/Clear to Send). The PCF and DCF sub-layers are shown in Figure 2.3.

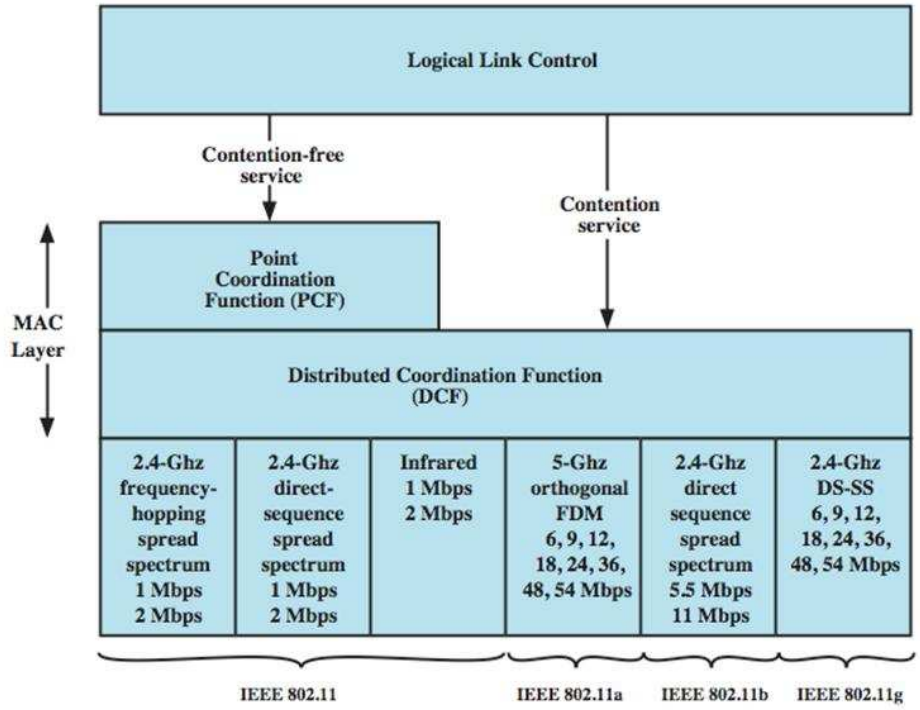


Figure 2.3: IEEE 802.11 protocol architecture

In IEEE 802.11 standards, carrier sensing is the primary method used to avoid collisions. Carrier sensing is accomplished by simply measuring the amount of energy received on the channel. If that energy is above a certain threshold, the sensing node determines

that another node is currently transmitting and that it must remain wait for the channel to become free/available.

Along with carrier sensing, inter-frame space (IFS) is primarily used to ensure that the channel is truly free. When a node is sensing the channel, the channel must be free for the length of the distributed coordination function IFS (DIFS) period. The DIFS is the longest inter-frame space, used as a minimum delay for asynchronous frames contending for access. The SIFS is the shortest waiting time for medium access and used as the wait time between the Request to Send (RTS), Clear to Send (CTS), DATA, and acknowledgement (ACK) frames. Since the SIFS is always shorter than the DIFS, this ensures that another node does not incorrectly determine that the channel is idle during the handshake and that priority is given to the transmission in progress.

In the following subsections, we discuss two medium access mechanisms in DCF, CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) and RTS/CTS/Data/ACK.

2.1.3.1 CSMA/CA

Carrier sense with multiple access and collision avoidance (CSMA/CA) is a media access control (MAC) layer mechanism used by IEEE 802.11 wireless local area networks (WLANs). CSMA/CA is a random access scheme with carrier sense and collision avoidance through random backoff. The basic CSMA/CA mechanism is shown in Figure 2.4. If the medium is idle for at least the duration of the DCF (distributed coordination function) inter-frame spacing (DIFS), a node can access the medium right away.

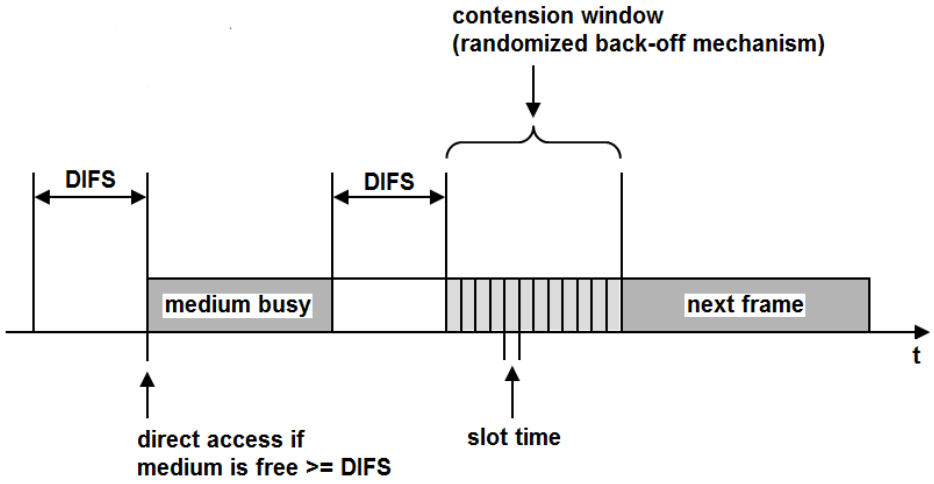


Figure 2.4: CSMA/CA

If the medium is busy, nodes have to wait for the duration of the DIFS, entering a contention phase afterwards. Each node chooses a random backoff time within a contention window and starts counting down its backoff timer. The backoff timer stores a node's residual backoff times. The node continues to sense the medium. If a node senses the channel is busy, it has to wait for the next chance until the medium is idle again. If a certain station does not get access to the medium in the first cycle, it stops its backoff timer, waits for the channel to be idle again for the DIFS duration and starts the counter again. As soon as the counter expires, the node accesses the medium. This means that deferred stations do not choose a randomized backoff time again, but continue to count down. Stations that have waited longer have an advantage over stations that have just entered, in that they only have to wait for the remainder of their backoff timer from the previous cycles. Thus, this randomly distributed delay helps to avoid collisions; other-

wise all stations would try to transmit data after waiting for the medium to become idle again. CSMA/CA reduces the probability of collision among nodes, but it cannot avoid the hidden terminal problem.

2.1.3.2 RTS/CTS/Data/ACK

To solve the hidden terminal problem discussed in Section 2.1.2, the IEEE 802.11 WLAN standard defines a medium access control mechanism using two control packets RTS and CTS. Acknowledgements are added for enhanced reliability. Figure 2.5 illustrates the use of RTS, CTS, data and acknowledgement (ACK).

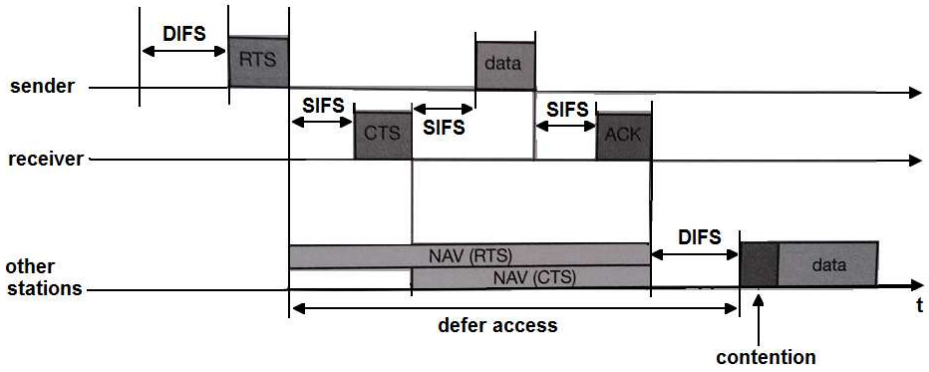


Figure 2.5: DCF with RTS/CTS/Data/ACK

After waiting for the DIFS duration (plus a random backoff time if the medium was busy), the sender issues a RTS control packet. The RTS packet includes the receiver's address of the data transmission to come and the duration of the whole data transmission. This duration specifies the time interval necessary to transmit the whole data frame and the acknowledgement (ACK) related to it. Every node receiving this RTS has to set its

network allocation vector (NAV) in accordance with the duration field. The NAV then specifies the earliest point at which the station can try to access the medium again.

If the receiver of the data transmission receives the RTS, it answers with a CTS message after waiting for a short inter-frame spacing (SIFS) interval. The CTS packet contains the duration field. All stations receiving this packet from the receiver of the intended data transmission have to adjust their NAV. The latter set of receivers need not be the same as the first set receiving the RTS packet. All nodes have to wait before accessing the medium. Thus, this mechanism reserves the medium for one sender exclusively.

Finally, the sender can send the data after a SIFS interval. The receiver waits for a SIFS period of time after receiving the data packet and then acknowledges whether the transfer was correct. When the transmission completes, the NAV in each node marks the medium as free and the standard cycle can start again.

When using RTS and CTS to avoid the hidden terminal problem, collisions can only occur at the beginning while the RTS is sent. This is because two or more stations may start sending at the same time (RTS or other data packet).

The above RTS/CTS mechanism is only used in unicast communications, but not used in multicast communications. There currently does not exist an effective algorithm for implementing RTS/CTS mechanism in multicast communications for the following two reasons. First, CTS packets sent by the multicast neighbors of a transmitter have a very high probability of colliding at the transmitter. More importantly, it may not be possible

for all the multicast neighbors to agree on a common time slot for the transmission of a packet, or the delay would be very long to reach such an agreement. Therefore, all multicast implementations in wireless networks so far have used only CSMA/CA without RTS/CTS.

2.1.4 Overview of DSSS and OFDM

The IEEE 802.11 standard specifies two modulation methods, which are direct-sequence spread spectrum (DSSS) and orthogonal frequency-division multiplexing (OFDM) [28]. IEEE 802.11 introduces DSSS as a technology to minimize the interference for the 2.4 GHz frequency band, while IEEE 802.11a uses OFDM modulation to control interference in the 5 GHz frequency band. DSSS supports the transmission rate up to 11Mbit/s at the physical layer, while OFDM supports the transmission rate up to 54Mbits/s.

2.1.4.1 Direct Sequence Spread Spectrum (DSSS)

Direct sequence spread spectrum (DSSS) is a spread spectrum modulation technique. In DSSS, the original data signal is multiplied with a pseudorandom noise spreading code. DSSS provides stronger protection against interfering signals, and makes the interfering signal less perceptible. DSSS also provides security of transmission if the code is not known to the public, which makes it very popular in military applications.

When transmitting data using DSSS, the required data signal is multiplied with what is known as a spreading or chip code data stream. Figure 2.6 shows spreading with

DSSS. The resulting data stream has a higher data rate than the data itself. The data is multiplied using the XOR (exclusive OR) function.

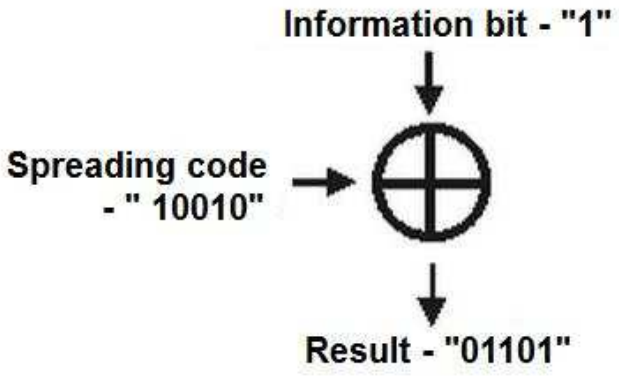


Figure 2.6: Spreading with DSSS

Each bit in the spreading sequence is called a chip, and this is much shorter than each information bit. The spreading sequence or chip sequence has the same data rate as the final output from the spreading multiplier. The rate is called the chip rate, and this is often measured in terms of a number of M chips/second.

The baseband data stream is then modulated onto a carrier. Figure 2.7 shows the process of DSSS generation. In this way the overall signal is spread over a much wider bandwidth than if the data had been simply modulated onto the carrier. The reason is that signals with high data rates occupy wider signal bandwidths than those with low data rates.

To decode the signal and receive the original data, the signal is first demodulated from the carrier to reconstitute the high speed data stream. Demodulating the received signal

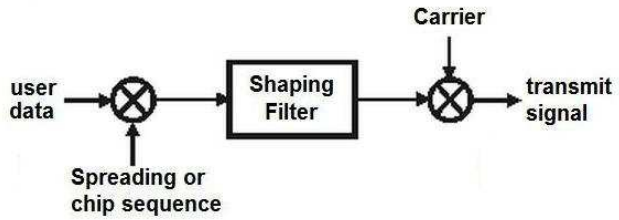


Figure 2.7: DSSS Generation

is achieved using the same carrier as the transmitter and reversing the modulation. The result is a signal with approximately the same bandwidth as the original spread spectrum signal. Additional filtering can be applied to generate the original signal. The receiver then uses the same chip sequence to reconstruct the original data. Figure 2.8 shows the process of DSSS decoding.

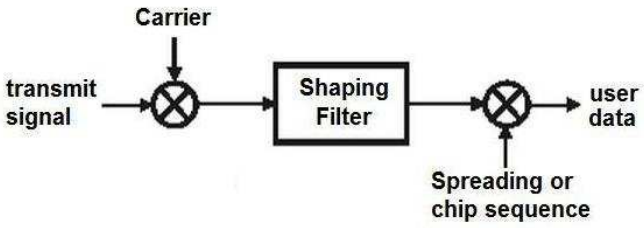


Figure 2.8: DSSS decoding

It is possible to transmit several sets of data independently on the same carrier and then reconstitute them at the receiver without mutual interference. This way a base station can send data to several mobile devices on a single channel. Similarly several mobile devices can send data to a single base station, provided that in each case an independent spreading code is used.

2.1.4.2 Orthogonal Frequency Division Multiplexing (OFDM)

Orthogonal Frequency Division Multiplexing or OFDM is a modulation format that is being used for many of the latest wireless and telecommunications standards, such as IEEE 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac and more.

OFDM is a form of multi-carrier modulation. OFDM works by splitting the radio signal into multiple smaller sub-signals that are then transmitted simultaneously at different frequencies to the receiver. It is necessary for a receiver to be able to receive the whole signal to be able to successfully demodulate the data. As a result, when signals are transmitted close to one another, they must be spaced so that the receiver can separate them using a filter, and there must be a guard band between them. Figure 2.9 shows the traditional view of receiving signals carrying modulation. On the other hand, the sidebands overlap from each carrier in OFDM as shown in Figure 2.10. The sidebands in OFDM can still be received without the interference because they are orthogonal to each another. This is achieved by having the carrier spacing equal to the reciprocal of the symbol period.

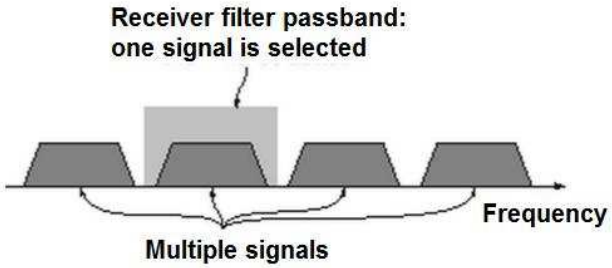


Figure 2.9: Traditional view of receiving signals carrying modulation

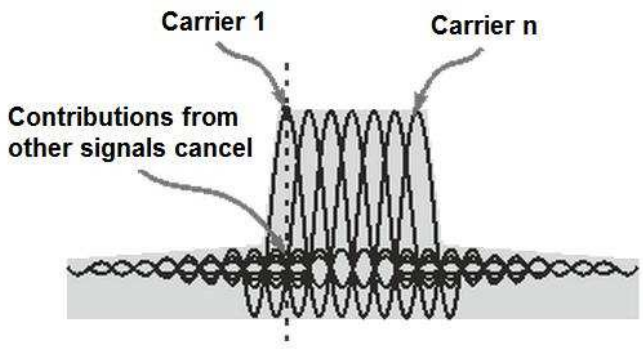


Figure 2.10: OFDM Spectrum

The data to be transmitted on an OFDM signal is divided over a large number of radio frequencies. Each radio frequency carries only a small portion of the total amount of data. This reduces the data rate taken by each carrier. The lower data rate has the advantage that interference from reflections is much less critical. This is achieved by adding a guard band time or guard interval into the system. This ensures that the data is only sampled when the signal is stable and no new delayed signals arrive that would alter the timing and phase of the signal.

The distribution of the data across a large number of carriers in the OFDM signal has some further advantages. Nulls caused by multi-path effects or interference on a given frequency only affect a small number of the carriers, the remaining ones being received correctly. By using error-coding techniques, which does mean adding further data to the transmitted signal, it enables many or all of the corrupted data to be reconstructed within the receiver. This can be done because the error correction code is transmitted in a different part of the signal.

Thus, using OFDM makes the transmitted signal robust against frequency selective interference and fading, such as multipath fading. If frequency selective fading occurs on the radio channel, only a small portion of the data is affected, while in a broadband transmission with all data on a single carrier the complete radio-frequency signal would be affected.

One requirement of the OFDM transmitting and receiving systems is that they must be linear. Any non-linearity will cause interference between the carriers as a result of inter-modulation distortion. This will introduce unwanted signals that would cause interference and impair the orthogonality of the transmission.

2.2 Overview of Cryptography

Cryptography is defined as the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, and authentication [15]. The use of cryptographic techniques offers:

- *confidentiality*: ensuring that no one can read the message except the intended receiver;
- *data integrity*: assuring the receiver that the received message has not been altered;
- *authentication*: proving the identity of the user;
- *non-repudiation*: assuring that a sender cannot deny having sent the message.

A goal of cryptography is to adequately address the above four criteria in both theory and practice. Cryptography is about the prevention and detection of any malicious activity. In this section, we will discuss major cryptographic techniques, namely, encryption and decryption, digital signatures, and message authentication codes.

2.2.1 Encryption Schemes

In cryptography, encryption is the process of encoding messages in such a way that others cannot read it, but authorized parties can. An encryption scheme is an effective approach to achieve confidentiality. In an encryption scheme, a message is encrypted using an encryption algorithm, turning it into an unreadable ciphertext. This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the ciphertext should not be able to determine anything about the original message. An authorized party, however, is able to decode the ciphertext using a decryption algorithm that usually requires a decryption key. There are two types of encryption schemes: symmetric-key encryption and public-key encryption [16].

2.2.1.1 Symmetric-key Cryptography

Symmetric-key algorithms use a single secret key to encrypt and decrypt messages, as shown in Figure 2.11. Thus communicating parties must agree on the same secret key before they wish to communicate with each other.

A symmetric encryption scheme $SE=(G, E, D)$ consists of three algorithms [16], as

follows:

(1) The key generation algorithm G is a randomized algorithm that returns a string K. Let $Keys(SE)$ denote a set of all strings that have non-zero probability of being output by G. The members of this set are called keys. We write $K \xleftarrow{R} G$ for the operation of executing G and let K denote the key returned.

(2) The encryption algorithm E takes the key $K \in Keys(SE)$ and a plaintext $M \in \{0,1\}^*$ to return a *ciphertext* $C \in \{0,1\}^*$ denoted as $C \xleftarrow{R} E_K(M)$.

(3) The decryption algorithm D takes a key $K \in Keys(SE)$ and a ciphertext $C \in \{0,1\}^*$ to return the plaintext $M \in \{0,1\}^*$ denoted as $M \leftarrow D_K(C)$.

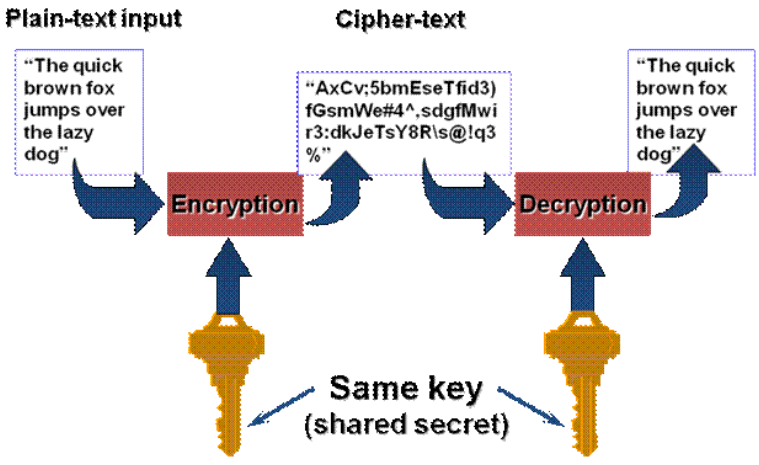


Figure 2.11: Symmetric key encryption and decryption

2.2.1.2 Public-key Cryptography

Public-key encryption algorithms use a private key that is known only to its owner and a public key that can be made known to anyone. The public key and the private key are mathematically linked. Data encrypted with a public key can be decrypted only with its corresponding private key and vice versa.

Each user u in the network has a pair of keys $\langle P_u, S_u \rangle$ associated with him. The public key P_u is accessible to everyone, and the private-key S_u is known only to user u. A public and private key pair is generated by running a key-generation algorithm. To send a secret message M to user u, the sender first encrypts message M into a cipher text $C = E_{P_u}(M)$ using u's public key P_u and a public encryption algorithm E, and then sends the cipher text C to user u. E is a public encryption algorithm. Upon receiving cipher text C, user u can decrypt the message by using his private key S_u and computing $D_{S_u}(C)$, where D is a decryption algorithm. Clearly, for this to work we need that $D_{S_u}(E_{P_u}(M)) = M$.

Figure 2.12 shows the public-key encryption and decryption process. Two parties (sender and recipient) use public-key encryption as follows. If the sender wants to send to the recipient an encrypted message, he uses the recipient's public key to encrypt the message. For example, Bob wants to send a message to Alice. Bob first encrypts the message using Alice's public key and then sends the encrypted message to Alice. When receiving the encrypted message, Alice decrypts the encrypted message using her private key.

During the transmission of the encrypted message, an unauthorized user might inter-

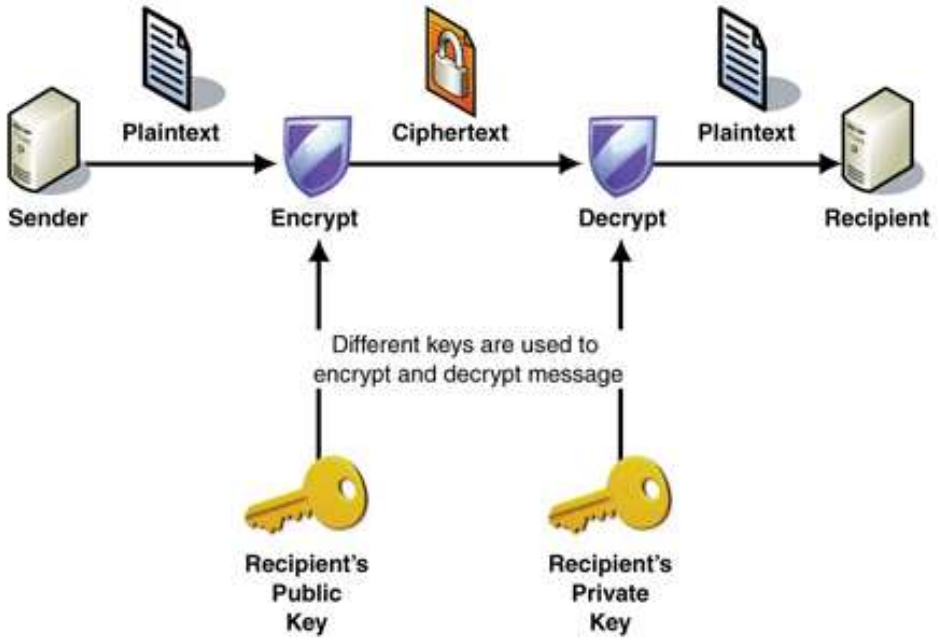


Figure 2.12: Public key encryption and decryption

cept the encrypted message from Bob. However, the unauthorized user cannot retrieve the original message since the encrypted message can only be decrypted with Alice's private key, which is known only to Alice. If Alice wants to send a message back to Bob, she encrypts her message using Bob's public key. Bob then decrypts the message using his private key.

A public-key encryption scheme is a triplet (G, E, D) of algorithms (key generation, encryption, and decryption) which are required to satisfy the following conditions [16]:

(1) Key generation algorithm: an algorithm G produces a pair (P_u, S_u), where P_u is called the public key, and S_u is the corresponding private key. We also refer to (P_u, S_u) as a pair of encryption/decryption keys.

(2) Encryption algorithm: an algorithm E takes as inputs a public-key P_u and a string $M \in \{0,1\}^k$ called the message, and produces as output a string $C \in \{0,1\}^*$ called the ciphertext. The notation $C \in E_{P_u}(M)$ denotes C being an encryption of message M using key P_u.

(3) Decryption algorithm: an algorithm D takes as inputs a private-key S_u and a ciphertext C from the range of $E_{P_u}(M)$, and produces as output a string $M' \in \{0,1\}^*$, such that for every pair (P_u, S_u), for every M, for every $C \in D(P_u, M)$, the probability for the decryption result $D_{S_u}(C)$ being not equal to the output string M' is negligible.

2.2.1.3 Symmetric-key vs. Public-key Cryptography

Symmetric-key cryptography uses a single key for both encryption and decryption. It is easier to implement, and generally requires less processing power. On the other hand, public-key cryptography uses different keys for encryption and decryption. The decryption key cannot be calculated from the encryption key. Public-key encryption is normally used to encrypt other keys for subsequent communications. Symmetric-key cryptograph is well suited for performing cryptographic transformations on large streams of data because symmetric key encryption is computationally less expensive than public-key encryption given equivalent levels of security.

Symmetric-key cryptography requires a sender and a receiver to agree on a key before data transmission. The security of the cryptographic algorithm lies solely with the key. Symmetric-key cryptography incurs high costs for key creation and maintenance. For

example, given M people, the total of M^2 keys have to be created/maintained. On the other hand, the public-key cryptography only requires to maintain M pair of keys. The drawbacks of public-key cryptography is that it is more computationally expensive compared with most symmetric-key algorithms of equivalent security, and also requires the use of large keys. These drawbacks makes it cost prohibitive to send large amounts of data using public-key encryption.

Since both symmetric and public key cryptography have their own advantages, file transfer systems typically employ a hybrid of the two, such as SSL (secure socket layer) used in FTPS (file transfer protocol secure) and HTTPS (hypertext transfer protocol secure), or SSH (secure shell) used in SFTP (secure file transfer protocol). Hybrid cryptosystems employed in an FTPS or SFTP server use public keys to initially encrypt symmetric keys known as session keys. The session keys are then used to encrypt the actual data. A session key is only used in one session. After the session, the key is simply discarded. If a session key is compromised, only the data sent within that particular session will be at risk.

2.2.2 Public Key Cryptography

Public key cryptography can be used to provide a secure method for exchanging secret keys. Two of the most common key exchange algorithms are the Diffie-Hellman key agreement algorithm and RSA key exchange process. Diffie-Hellman and RSA are two most popular public key algorithms in use today.

2.2.2.1 Diffie-Hellman Key Agreement

In 1976, Whitfield Diffie and Martin Hellman published the Diffie-Hellman algorithm for key exchange [96]. This was the first published use of public key cryptography. The Diffie-Hellman key exchange algorithm was the first practical method for establishing a shared secret over an unsecured communication channel.

The effectiveness of the Diffie-Hellman key agreement algorithm depends on the difficulty of computing discrete logarithms. Figure 2.13 shows the basic Diffie-Hellman Key Agreement process.

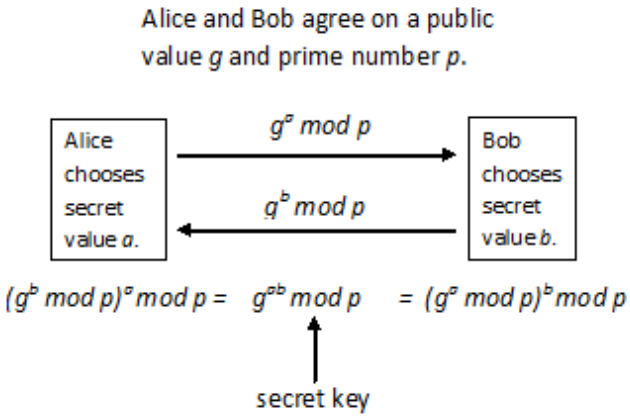


Figure 2.13: Diffie-Hellman key agreement

- Alice and Bob agree on a prime number p and a base g, which is a primitive root modulo p.
- Alice chooses a secret number a, and sends Bob g^a mod p.

- Bob chooses a secret number b, and sends Alice $g^b \bmod p$.
- Alice computes $(g^b \bmod p)^a \bmod p = g^{ab} \bmod p$.
- Bob computes $(g^a \bmod p)^a \bmod p = g^{ab} \bmod p$.

Both Alice and Bob then use the number $g^{ab} \bmod p$ as their shared key for confidential communications.

However, the Diffie-Hellman algorithm is vulnerable to man-in-the-middle attacks as there is no authentication in place before keys are exchanged. Therefore, the algorithm is usually used in combination with an additional authentication method, such as digital signatures. For example, IPsec uses the Diffie-Hellman algorithm in conjunction with RSA authentication to exchange a session key that is used for encrypting all traffic that crosses an IPSec tunnel.

The Diffie-Hellman algorithm is not intended for use as a general encryption scheme. Its purpose is to transmit a shared key across an insecure medium.

2.2.2.2 RSA Key Exchange

In the year following the Diffie-Hellman proposal, Ron Rivest, Adi Shamir, and Leonard Adleman proposed another public key encryption system, which is now known as the Rivest-Shamir-Adleman (RSA) algorithm [97]. The RSA algorithm shares many similarities with the Diffie-Hellman algorithm in that RSA is also based on multiplying and factoring large integers. However, RSA is significantly faster than Diffie-Hellman algorithm [98]. The RSA algorithm can be used for digital signatures, key exchanges and

encryption. It is built into software such as Microsoft, Apple and Novell products. It has also been implemented into hardware such as network interface cards and smart cards.

For RSA key exchange, secret keys are exchanged securely online by encrypting a secret key with the intended recipient's public key. Only the intended recipient can decrypt the message to retrieve the secret key because the decryption requires the use of the recipient's private key. Therefore, a third party who intercepts the encrypted shared secret key cannot decrypt the message without the knowledge of the intended recipient's private key. Figure 2.14 illustrates the basic RSA key exchange process.

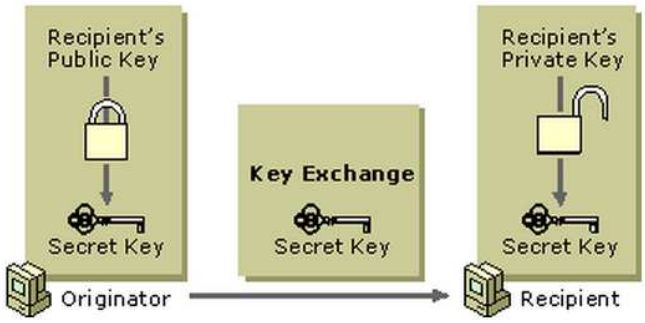


Figure 2.14: Basic RSA Key Exchange

2.2.2.3 Diffie-Hellman vs. RSA Key Exchange

Diffie-Hellman and RSA are both based on supposedly intractable problems. While Diffie-Hellman is based on the difficulty of exponentiation and modular arithmetics, RSA is based on the difficulty of factoring large numbers. With keys of 1,024 bits long, Diffie-Hellman and RSA give comparable levels of security [98].

In terms of computation time, the RSA encryption using public keys is substantially faster than any Diffie-Hellman operation. The RSA decryption using private keys entails more or less the same amount of work as the Diffie-Hellman key exchange with similar resistance [98]. Therefore, we choose RSA rather than Diffie-Hellman for key exchanges in our proposed authentication protocols to support fast intra-network and inter-network handovers.

2.2.3 Digital Signatures

A digital signature gives a recipient reasons to believe that the message was created by a known sender and that the message was not altered in transit [16]. Digital signatures ensure the integrity, authentication, and non-repudiation of a message.

A digital signature is a hash value that has been encrypted with the sender's private key. A hash value (often called a message digest) is derived from the original text of the message. The act of signing just means the hash value was encrypted with a private key. The hashing function ensures the integrity of the message, and the signing of the hash value provides authentication and non-repudiation.

Following are the available cryptographic methods to provide different types of security services:

- A message can be encrypted, which provides confidentiality.
- A message can be hashed, which provides integrity.
- A message can be digitally signed, which provides integrity, authentication, and

non-repudiation.

- A message can be encrypted and digitally signed, which provides confidentiality, integrity, authentication, and non-repudiation.

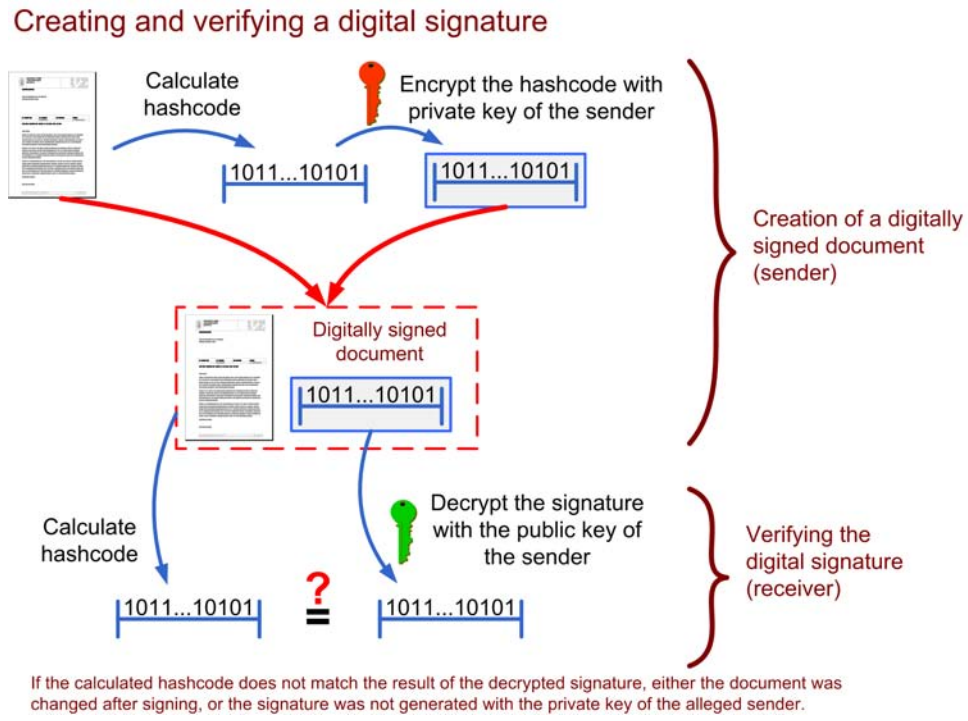


Figure 2.15: Digital signature generation and verification

Figure 2.15 shows a digital signature generation and verification process. Assume two users have agreed upon a hash function and a key for the signature verification process in advance. If Alice wants to send a digitally signed message to Bob, Alice will perform the following steps:

- Generating a message digest of the original plain-text message

- Encrypting the message digest using her private key (This encrypted message digest is the digital signature.)
- Transmiting the message and the digital signature to Bob

Upon receiving the digitally signed message, Bob will:

- generate a new digest for the received message
- decrypt the attached digital signature using Alice's public key
- compare the value he calculated with the value he decrypted

If Alice's digital signature can be verified, the following can be assumed:

- Bob is assured that the message was not modified. If even one bit of the original message was changed, the digest generated using the received message would differ from the decrypted value, and cause the signature verification process to fail.
- Bob is assured that Alice sent the message. Public key transformation functions cannot be duplicated by any practical means; therefore, only a signature generated by the originator's private key can be correctly decrypted using the originator's public key.

Although the integrity of the message is assured, there is no authentication supporting non-repudiation unless Alice's public key can be proven to belong to Alice only.

For example, if Sue were able to establish an alias for herself as Alice, she might masquerade as Bob's friend Alice. This problem is resolved by using an authentication

service. An authentication server is a third party who vouches for the identity of a public key owner using a public key infrastructure (PKI). A PKI provides the public-key encryption and digital signature services necessary to verify, enroll, and certify users of a secure application.

2.2.4 Message Authentication Code

Another technique to ensure data integrity is message authentication codes (MACs). Unlike digital signatures, MAC values are both generated and verified using the same key. This implies that the sender and receiver of a message must agree on a shared key before initiating communication.

The process of MAC generation and verification is shown in Figure 2.16. The sender of a message executes a MAC algorithm to produce a MAC value. The message and a key shared by the sender and receiver are the inputs to the algorithm. The message and the MAC value are then sent to the receiver. The receiver in turn runs the message of the transmission through the same MAC algorithm using the same key, producing a second MAC value. The receiver then compares the first MAC received to the second generated MAC. If they are identical, the receiver can safely assume that while in transit the message was not altered.

A message authentication code $MA = (G, \tau, \nu)$ results from three algorithms, as follows [16]:

(1) The key generation algorithm G is a randomized algorithm that returns a key K;

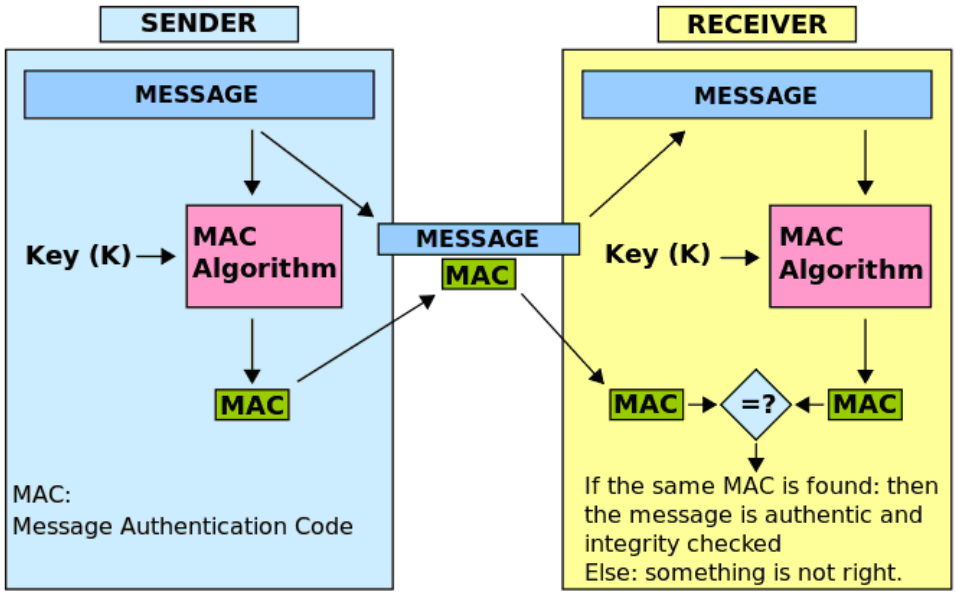


Figure 2.16: The use of MAC algorithm

denoted as $K \xleftarrow{R} G$.

(2) The MAC algorithm τ is an algorithm that takes the key K and a message M to return a MAC value σ; denoted as $\sigma \leftarrow \tau_K(M)$.

(3) The verification algorithm ν is an algorithm that takes the key K, a message M, and a MAC value τ for M to return a verification result d, which is a one-bit message indicating if the verification is successful and denoted as $d \leftarrow \nu_K(M, \tau)$.

The cryptographic security goal of a MAC is to ensure integrity and authentication. Digital signatures provide non-repudiation property in addition to integrity and authentication. Digital signatures are usually slower than MACs, and thus used only when there is not yet a shared secret or the non-repudiation property is required. Thus, we

choose MACs in our proposed authentication protocols to support fast intra-network and inter-network handovers.

2.3 Security Threats

In this section, we review major security threats encountered in WMNs [18, 19].

(1) Physical Threats

Routers of a WMN are usually deployed outdoors, such as the roofs of buildings, or on street lamps. They are thus vulnerable to attacks, such as tampering with information inside routers, stealing private keys stored in routers, or even replacing a router with a rogue router. Therefore, physical protection of routers in a WMN is very important.

(2) Jamming Attacks

Jamming is a type of attack which interferes with the radio frequencies that a node uses for communications. A jamming source could be powerful enough to disrupt communication in the entire network. Even with less powerful jamming sources, an adversary can potentially disrupt communication in the entire network by strategically distributing the jamming sources. Jamming attacks in WMNs may happen very often since this type of attack can be launched without much effort and sophistication.

(3) Threats to Routing Protocols

WMNs may be susceptible to routing protocol threats and route disruption attacks.

Many of these threats require packet injection with specialized knowledge of the routing protocol. We summarize these threats below.

- In a black-hole attack, an attacker creates forged packets to impersonate a valid router and subsequently drops packets resulting in high packet loss rates.
- In a grey-hole attack, an attacker creates forged packets to selectively drop routes or inspect network traffic.
- In a worm-hole attack, routing control messages are replayed from one network location to another, which can severely disrupt routing.
- In a route error injection attack, an attacker disrupts routing by injecting forged route error messages to break mesh links. Relative to other routing attacks, this attack conceivably has higher exploitability because it does not require detailed knowledge of the routing protocol state model.

(4) Identity Privacy Attack

Users would like to remain anonymous while roaming in different parts of the network for privacy reasons. To protect clients' privacy, client IDs are numbers or strings that should not relate to the clients' real identities, much like bank account numbers or social security numbers.

(5) Forgery Attack

Forgery usually describes a message-related attack against a cryptographic digital signature scheme. An attacker tries to fabricate a digital signature for a message without

having access to the respective signer's private signing key, which is called a forgery attack. There are three types of forgery: existential, selective, and universal [17].

Existential forgery is the creation (by an attacker) of at least one message/signature pair where the signature was not produced by the legitimate signer. Existential forgery is essentially the weakest adversarial goal. As long as the pair of message and signature is valid, the attacker has succeeded in constructing an existential forgery.

Selective forgery is the creation (by an attacker) of a message and signature pair where the message has been chosen by the attacker prior to the attack. The message may be chosen to have interesting mathematical properties with respect to the signature algorithm. However, in selective forgery, the message must be fixed before the start of the attack.

Universal forgery is the creation (by an attacker) of a valid signature for any given message. An attacker capable of universal forgery is able to sign messages he chose himself (as in selective forgery), messages chosen at random, or even specific messages provided by an opponent.

Besides the above forgery attacks, there is also a *total break*: when an attacker can compute the signer's private key and therefore forge any possible signature on any message.

(6) Space-time Trade-off Attack

Space-time trade-off is a situation in which memory usage can be reduced at the cost

of slower program execution (and, conversely, the computation time can be reduced at the cost of increased memory usage). A space-time trade-off attack is a type of cryptographic attack where an attacker tries to recover a key when the plaintext and the ciphertext are known. Attackers can use the space-time trade-off method to try to break the data encryption keys used in the data encryption algorithms [109].

This attack can also be used against hash-based MAC algorithms where attackers attempt to recover a MAC key of a hash-based MAC algorithm. A space-time trade-off attack has two phases: pre-computation phase and online phase. In the pre-computation phase, the attacker executes an exhaustive search and stores the hashing results. This is done offline and may take a long time. Once this pre-computation is done, the attacker could recover a key almost instantaneously by using the results that are pre-computed and saved in the memory. The time taken in the online phase is shortened thanks to the pre-computation results stored in memory.

(7) Replay Attack

An attacker records messages of an ongoing authentication session and replays these messages in the future in an attempt to be successfully authenticated and possibly gain access to the network as a client. An attacker may replay a client's messages to gain access to the network, or an access point's messages in order to impersonate the access point.

(8) Denial-of-Service (DoS) Attack

A denial-of-service (Dos) attack is an attempt to make a machine or network resource unavailable to its intended users. A DoS attack generally consists of efforts to temporarily or indefinitely interrupt or suspend services of a host connected to the network. In an authentication protocol, an attacker may send bogus messages or replay past valid messages repeatedly to force a router to use up its resources to process a large amount of these DoS attack messages.

(9) Impersonation Attack

In an impersonation attack, an attacker masquerades as a trusted node. IP address spoofing is a form of impersonation attack. IP spoofing refers to the creation of IP packets with a forged source IP address, with the purpose of concealing the identity of the actual sender.

The attacker can use such IP spoofing attacks to defeat IP address-based authentication. This process is primarily used when trust relationships are already established between devices on a network and internal system. For example, on some corporate networks, the internal systems trust each other through IP addresses. Users can log in without a username and password if they are already connecting from another device already accepted by the internal network.By spoofing the IP address from a trusted device, an attacker may be able to impersonate the target device without authentication.

Among the security threats discussed above, identity privacy attack, forgery attack, time-memory trade-off attack, replay attack, DoS attack, and impersonation attacks are the main security concerns to authentication and key management in WMNs, which are the focus of our research. Although physical attacks, routing protocol attacks and jamming attacks may happen in WMNs due to the nature of wireless communications, defending against these threats belong to other security solutions such as intrusion detection, prediction and prevention, as well as secure routing.

2.4 Security Models and Trust Management in Wireless Networks

In this section, we provide a review of security models and trust management in wireless networks.

2.4.1 Security Models for Handovers in Wireless Networks

Several security models have been proposed to address the issue of secure handovers in wireless networks. The home-foreign-domain model in [9, 20, 21] provides a security approach for handovers among multiple wireless networks. The home-foreign-domain model is usually observed in the global system for mobile (GSM) communications, the universal mobile telecommunication system (UMTS), or in mobile IP networks. During a handover in this model, the home domain of a user is involved, where the user is registered and its account information is kept. The home domain is contacted by a foreign domain every time the user roams to the foreign domain and needs to be authenticated.

Using the home-foreign domain model, mobile IP allows users to keep the same IP address, stay connected, and maintain ongoing applications while roaming between IP networks [29]. Agent discovery, registration and tunneling are three phases performed in a mobile IP process.

- Agent discovery: Mobility agents (home and foreign agents) advertise their availability on each link on which they provide service.
- Registration: When the mobile node is away from its home network, it registers its care-of address with its home agent. A care-of address is a temporary IP address assigned by a foreign network to a mobile device when the device is connecting to that network. The home network forwards messages to the mobile devices using the current care-of address assigned to the device.
- Tunneling: In order for datagrams to be delivered to the mobile node when it is away from the home network, the home agent has to tunnel the datagrams to the care-of address.

Figure 2.17 illustrates the routing of datagrams to and from a mobile node away from home, once the mobile node has registered with its home agent. The mobile node is presumed to be using a care-of address provided by a foreign agent.

- A datagram to the mobile node arrives on the home network via standard IP routing.
- The datagram is intercepted by the home agent and is tunneled to the care-of address, as depicted by the arrow going through the tube.

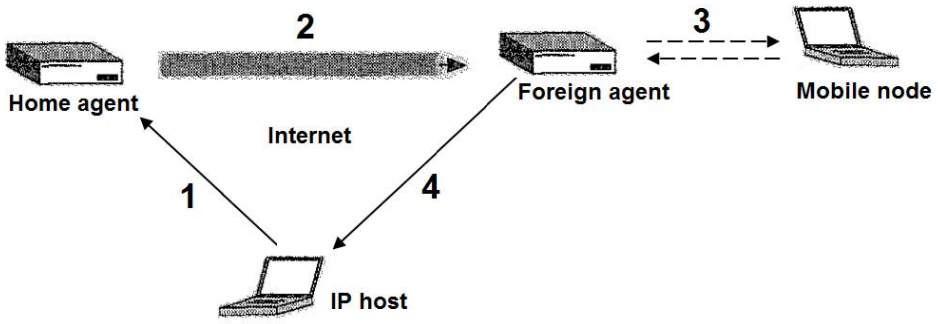


Figure 2.17: Mobile IP datagram flow

- The datagram is "detunneled" and delivered to the mobile node.
- For datagrams sent by the mobile node, standard IP routing delivers each datagram to its destination. In the figure, the foreign agent is the mobile node's default router.

Leu [22] proposed another model supporting secure handovers among different domains in cellular networks. With this approach, a client is registered with a home domain. To be authenticated, a client in a foreign domain needs to be authenticated by the authentication server of its home domain. All authentication messages are forwarded by a broker. The broker is trusted by all the domains to facilitate clients roaming among different domains. Comparing to the home-foreign-domain model, the benefit of this model is that bi-lateral roaming agreements among different domains are not needed. However, to authenticate a visitor, a foreign domain needs to contact the visitor's home network, which is similar to the home-foreign-domain model.

If applying either of the above two models to WMNs, the latency of inter-network handovers could be very long due to multi-hop wireless communications, and thus may

result in potential service interruption.

Cellular networks and IEEE 802.11 networks employ handover mechanisms for handover within the same network type. Mobile IP provides triggers or other services to accelerate mobile IP based handovers. Existing IEEE 802 standards provide mechanisms for detecting and selecting network access points, but do not allow for detection and selection of network access points in a way that is independent of the network type. Furthermore, multi-technology enabled terminals have also become available. Such multi-mode terminals pose new challenges to mobility management [56]. To address some of these challenges, the IEEE 802.21 standard, also called media independent handover (MIH), provides different handover mechanisms to enable handovers between networks of the same type as well as handovers between heterogeneous technologies (including IEEE 802 and cellular technologies) while optimizing session continuity.

Our work in this thesis addresses one of the most important issues to support seamless and fast handovers - security. In particularly, we propose solutions for authentication, key management and group key management in WMNs. As part of our future work, we will extend our proposed security solutions to support handovers between heterogeneous networks without incurring service interruptions, hence improving the user experience with mobile terminals.

2.4.2 Trust Management

As an important concept in network security, trust is interpreted as a set of relations among entities participating in the network activities. In commercially deployed wireless networks, such as WMNs, unauthorized clients should not be allowed to access network resources. Trust is fundamental in such networks, and any security mechanism requires trust as its underlying component.

Trust management in the Internet can be broadly classified into two models: hierarchical public key infrastructure (PKI) and web of trust [23]. In the hierarchical PKI model, there is a certificate authority (CA) at the top level and trust flows from the top to bottom, down to end users. This hierarchical trust model does not burden end users to prove their identity. IBM research laboratories developed a trust establishment framework [24] allowing the "bottom-up" emergence of a PKI through the exchange of certificates.

In comparison with a centralized PKI, a web-of-trust model is based on the idea of decentralized trust and social networks, which leaves trust decisions in the hands of individual users, not centralized certificate authorities. Pretty good privacy (PGP) [25] implements a web of trust model for establishing trust relationships between users. This could allow a user to indirectly and unknowingly trust other people that the user does not know.

In PGP, each user maintains a list of other users' public key certificates. This list is called a key ring. When a new user's public-key certificate is inserted to user U's key

ring, user U assigns a level of trust to this certificate. PGP defines trust levels and allows a user to assign three levels of trustworthiness to another user's public-key certificate. The three levels of trust are "complete trust", "marginal trust", or "no trust". This trust level tells PGP how much a user trust another user's public-key certificate. PGP allows a user to sign other users' public-key certificates. Therefore, a user's public key certificate could contain a number of other users' signatures. If a user receives a sender's certificate that contains a number of other users' signatures, the user will verify all signatures using the corresponding public keys from his key ring. After verifying all signatures from the sender's public key certificate, the user will trust the sender if the user determines that at least one of the signatures in the sender's public key certificate has been signed by another user he/she completely or marginally trusts.

Trust management in distributed and resource-constraint networks, such as mobile ad hoc networks (MANETs), is more difficult than that in the Internet. Generally, this type of distributed network has neither pre-established infrastructure, nor centralized control servers/trusted third parties. The trust information or evidence used to evaluate trustworthiness is provided by peers, i.e., the nodes that form the network. Yi [26] used a PGP-like mechanism to initialize an ad hoc system. Trust between nodes is established through secure side channel communications (e.g., physical contact, infrared communication). They assume that social relationships among mobile ad hoc network members are essentially the same as those in a PGP system, where the trust establishment is based on social relationships in a real society or community.

For intra-network or inter-network handover operations, clients and destination access points need to authenticate each other. The trust between clients and destination access points in wireless networks such as WLAN, GSM, or mobile IP networks is based on the trust between the clients and their home networks. A client must register with its home network. To authenticate a client, the destination access point has to communicate with the client's home network. However, the binding of a client with a home network may result in potential service interruption sin WMNs due to multi-hop wireless communications. To develop an efficient solution for fast intra-network and inter-network handovers in WMNs, we propose a new architecture, trust models and certificates, which do not require a client to be bound to a home network. Thus, the destination access points do not need to communicate with a client's home network for authentication. To the best of our knowledge, no such trust model or certificates exist in literature to support a scalable architecture and to provide a dynamic and flexible handover process for intra-network and inter-network handovers.

2.5 Overview of IEEE 802.11i Security Standard

The security solutions employed by the IEEE 802.11 standard include wired equivalent privacy (WEP), Wi-Fi protected access (WPA), and Wi-Fi protected access 2 (WPA2).

WEP is the first security solution in IEEE 802.11. Figure 2.18 shows the authentication steps of WEP. An access point and a user are both configured with a shared encryption key. The access point issues a random challenge to the user. The user then

returns the challenge encrypted using the shared key. The access point decrypts this message. If the decrypted message matches the original challenge, the access point is considered to successfully authenticate the user.

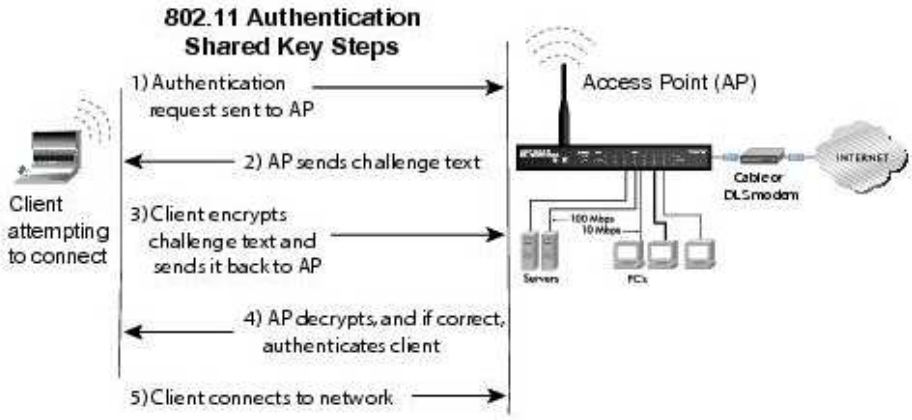


Figure 2.18: WEP Authentication

WEP is specified in the IEEE 802.11b standard. Although its name implies that wireless connections are as secure as wired connections, WEP has been demonstrated to have numerous flaws. For example, the key in WEP is static, which means that with sufficient time and the right tool, a hacker could use reverse-engineering to derive the encryption key. There exist programs such as Aircrack that can recover a 128-bit WEP key in 2-3 minutes [104].

WPA supersedes WEP as a security technology for IEEE 802.11i wireless networks. WPA improves WEP in terms of authentication and encryption. WPA supports the advanced encryption standard (AES) algorithm, which is a stronger encryption algorithm

than RC4 used in WEP. WPA provides mutual authentication, which WEP does not offer.

IEEE 802.11i is an IEEE 802.11 amendment used to facilitate secure communication for WLANs. IEEE 802.11i, implemented with WPA, forms a complete wireless security standard for defining authentication, key management, and group key management.

WPA was designed to be used with the temporal key integrity protocol (TKIP). TKIP is an encryption protocol included as part of the IEEE 802.11i standard for wireless LANs (WLANs). It was designed to provide more secure encryption than the notoriously weak WEP, the original WLAN security protocol. TKIP is the encryption method used in Wi-Fi Protected Access (WPA), which replaced WEP in WLAN products.

WPA2 replaced the original WPA technology on all certified Wi-Fi hardware and is based on the IEEE 802.11i standard for data encryption. WPA2 improves the security of Wi-Fi connections by requiring the use of stronger wireless encryption techniques than WPA requires. Specifically, WPA2 replaces TKIP in WPA with a stronger data encryption method called CCMP (counter mode cipher block chaining message authentication code protocol). CCMP is an AES-based encryption protocol that forms part of the IEEE 802.11i standard for wireless local area networks, particulary those using WiMax technology. CCMP employs 128-bit keys and a 48-bit initialization vector that minimize vulnerability to reply attacks. The pre-shared keys used in WPA-2 are automatically changed and authenticated at devices after a specified period of time, known as the rekey interval, has elapsed. This encryption is so complex that it requires special hardware to be added to access points for the purpose of encryption and decryption.

2.5.1 Authentication in IEEE 802.11i

The model of authentication in IEEE 802.11i was borrowed from the IEEE 802.1X standard [30] as shown in Figure 2.19. IEEE 802.1X was originally intended for wired local area networks (LANs), but it turned out that the same concepts can be used in wireless LANs. The IEEE 802.1X model involves three entities in the authentication procedure: a supplicant, an authenticator, and an authentication server. The supplicant is a client device (e.g., a laptop) that wishes to connect to the network. The authenticator is a network device, such as a wireless access point. The authentication server is a process, which can run on the access point in smaller networks, or on a dedicated server machine in larger networks.

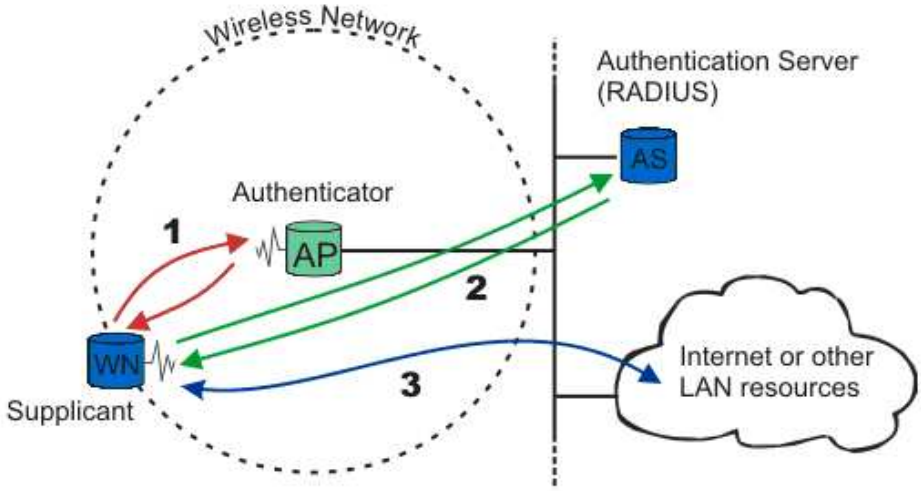


Figure 2.19: 802.1x

The authenticator acts like a security guard to a protected network. The supplicant

is not allowed access through the authenticator to the protected side of the network until the supplicants identity has been validated and authorized. With 802.1X port-based authentication, the supplicant provides credentials, such as a digital certificate, to the authenticator. The authenticator forwards the user's credentials to the authentication server for verification. If the authentication server determines that the credentials are valid, the supplicant is allowed to access resources located on the protected side of the network.

Thus, a client must be authenticated before it can gain access to a network, which involves three steps as shown in Figure 2.20.

(1) When a client requests access to a network, an access point requests the client's identity. The client responds to the authenticator (access point) with the client's identity, which is forwarded to the authentication server.

(2) The authentication process is in step (2). The complete process of an IEEE 802.11i authentication consists of messages exchanged between the client and the authentication server.

EAP-TLS [31] is a default authentication standard adopted in IEEE 802.11i WLANs. EAP-TLS is a certificate-based mutual authentication protocol. An authentication server provides its certificate to a user and requests the user's certificate. The user validates the authentication server's certificate to authenticate the authentication server. The user then responds to the server with its certificate. After verifying the user's certificate, the authentication server successfully authenticates the client.

During the authentication process, the access point relays packets between the client and the authentication server.

(3) When the authentication process finishes, the authentication server sends a successful message (or, a failure message if the authentication has failed) to the user through the access point. The client is then granted access to the network.

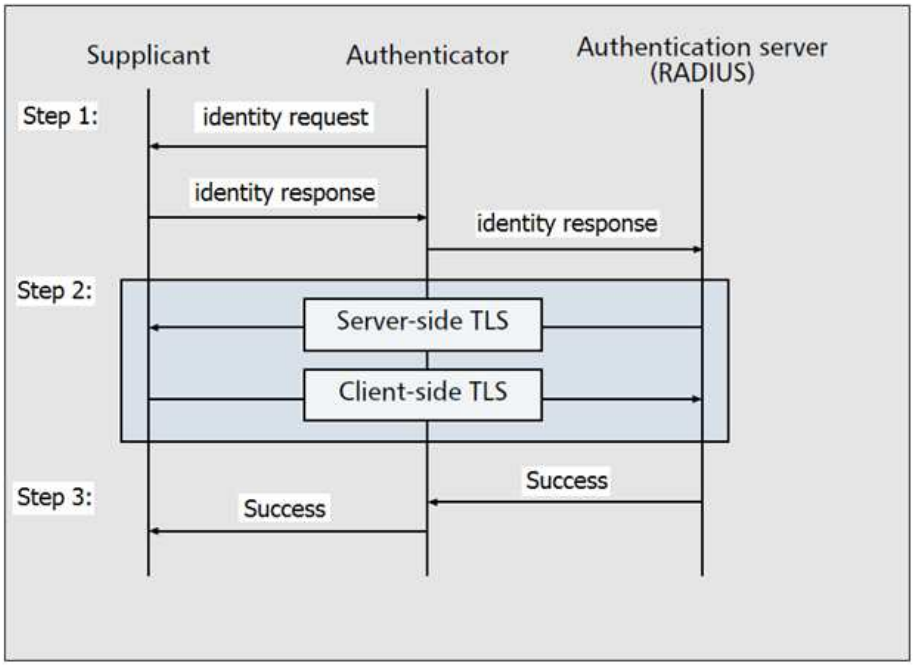


Figure 2.20: EAP-TLS

2.5.2 Key Management and Group Key Management in IEEE 802.11i

The result of the authentication process in IEEE 802.11i not only allows a client to connect to an access point, but also provides several keys for further communication between the

client and the access point. There are three types of keys generated in the IEEE 802.11i key management scheme as discussed below.

- Pairwise Master Key (PMK): PMKs are the highest order keys in the IEEE 802.11i standard. A PMK is computed by a client and an authentication server through the IEEE 802.11i authentication protocol. The PMK is encrypted using a key shared by the authentication server and the access point. The authentication server then sends the encrypted PMK to the authenticator (the access point). The PMK is known only to the client and the authenticator. It is a master key, because it is not used directly for encryption or integrity protection of messages, but rather used to derive encryption keys.

- Pairwise Transient Key (PTK): PTKs are derived from a PMK, generated and updated through a 4-way handshake protocol [31]. After confirming the existence of a PMK and the liveness of a client and an authentication server, the client and the authentication server generates a pairwise transient key (PTK) for each subsequent communication, and synchronizes the installation of the PTK on both machines.

- Group Temporal Key (GTK): GTK is used to protect multicast data.

Figure 2.21 shows the messages exchanged through a 4-way handshake protocol.

- An access point sends a nonce value ANonce to a client. The client generates a nonce SNonce, and computes a PTK using the PMK shared with the authenticator as follows.

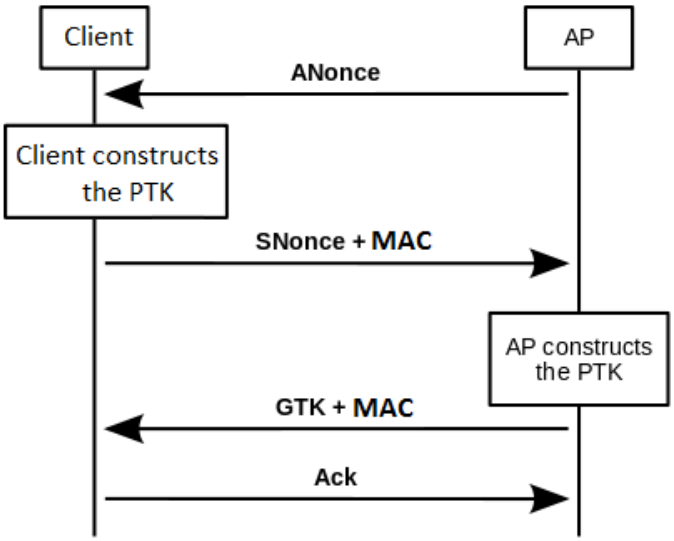


Figure 2.21: The 4-way handshake protocol

$$PTK = f(PMK, ANonce \parallel SNonce \parallel M_A \parallel M_C),$$

where M_C and M_A are the physical addresses of the client and the AP, respectively. The operator || denotes a concatenation. f is a pseudo-random number generation function.

As soon as the PTK is generated, it is divided into three separate keys as shown in Figure 2.22: the key encryption key (KEK), the key conformation key (KCK), and the temporal encryption key (TEK).

- Key encryption key (KEK): Used to encrypt the GTK as discussed below;
- Key confirmation key (KCK): Used to compute a message authentication code as discussed below;

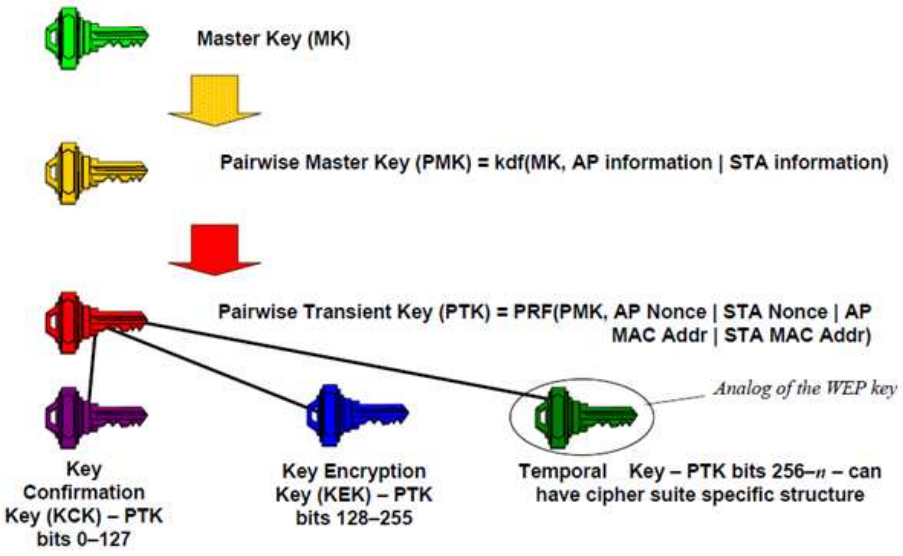


Figure 2.22: 802.11i key hierarchy

  - Temporal encryption key (TEK): Used to encrypt or decrypt unicast data packets as shown below.

- The client sends a nonce value SNonce and a MAC value of SNonce to the access point. KCK is the MAC key used to generate the MAC value. The access point then computes the same PTK according to the above formula.

- The access point uses the KCK to verify the MAC value sent by the client. If the verification is successful, the access point generates a GTK and encrypts it using the KEK. The access point then sends the encrypted GTK and the MAC value of GTK to the client.

- The client decrypts the message to get the GTK. After verifying that the MAC

value in the message is valid, the client sends a confirmation message to the access point.

Both PMKs and PTKs need to be updated after their lifetimes expire. The GTK needs to be updated periodically. In addition, when a device leaves the network, the GTK needs to be updated as well. This is to prevent the device from receiving further multicast or broadcast messages from the access point.

To update the group key update, IEEE 802.11i defines a two-way handshake protocol as follows [31]. First, an access point generates a new GTK, encrypts the GTK using a KEK assigned to each client, then unicasts each client the encrypted KEK along with a MAC value derived from the KCK. The MAC value in the message protects the data from modification. Second, after receiving the new GTK, each client replies to the access point with an acknowledgment.

2.6 Security Protocols for Intra-network and Inter-network Handovers

In this section, we first describe the handover process in wireless networks. We then review authentication protocols and key distribution schemes for intra-network and inter-network handovers, respectively.

2.6.1 Handover Operations in Wireless Networks

A handover procedure consists of several phases [32]. In the first phase, a handover impetus is detected. The obvious reason to initiate a handover procedure is that a mobile

client is about to move out of the transmission range of the currently serving access point. An alternative handover occurs if a mobile device moves into the range of a network access point that is preferable to the current one because of a stronger signal, while the currently serving access point is still available. Another reason to initiate a handover is load balancing. A client is in the transmission range of more than one access point and the currently serving access point is overloaded. Detection of a handover is based on a so-called handover algorithm. A handover algorithm takes collected measurement data as input, and determines whether or not a handover should take place. The measurement data typically include the currently received signal strength and/or the current load on the serving access point among other factors.

Once a handover impetus is detected, a new access point (the destination access point) is selected in the next phase of the handover procedure. The choice of a destination access point typically depends on the signal strength of all access points, and whether or not these access points have the capacity to serve the client after the handover.

In the last phase, the execution phase, the client disconnects itself from its currently serving access point (the source access point of the handover) and connects to the destination access point. The execution phase also includes mobility management on the network side that guarantees the re-routing of incoming and outgoing data traffic over the new access point.

When a client roams to a new access point, the client needs to be authenticated by the new access point. Besides authentication, another critical issue is key distribu-

tion. A shared key needs to be distributed to the client and the new access point before the authentication process. In practice, authentication and key distribution always go together. In the next section, we review existing authentication and key distribution schemes proposed for handover operations in wireless networks.

2.6.2 Authentication and Key Distribution Schemes for Intra-network Handovers

Due to resource constraints on mobile devices, security mechanisms for handovers should be designed to avoid negative impacts on performance and energy consumption.

We first identify the requirements of authentication and key distribution protocols designed specifically for the intra-network handover operation in WMNs.

- The authentication and key distribution protocols must incur low computation costs due to mobile devices' limited computational capabilities, storage and/or power supply. The number of messages to be exchanged should be minimized due to wireless channels having much lower bandwidth than wired networks.
- The delay of re-authentication during the handover process should be sufficiently short to avoid service interruption.
- The authentication protocol must support mutual authentication between a client and an access point. It must also provide the protection of client identity, and must be resilient to various types of attacks [33] such as forgery, replay attack, denial of service, space-timee trade-off attack, and identity privacy attack.

- The amount of control traffic generated by mobility management mechanisms, such as handover authentication and key distribution, has a significant impact on the overall network performance. Network operators are interested in reducing the amount of control traffic in their networks (possibly at the expense of higher server loads or lower handover performance [34]).

We broadly divide authentication and key distribution protocols for wireless networks into three categories: multi-hop authentication and key distribution, proactive authentication and key distribution, and ticket-based authentication and key distribution.

- In multi-hop authentication and key distribution protocols [36, 37, 38, 39], when a mobile client moves from one access point to another, it has to be re-authenticated by the authentication server (home network) which may be located many hops away from the client. Multi-hop wireless communications incur long latency and may lead to service interruptions.

- Proactive authentication and key distribution protocols [40, 41] attempt to minimize the authentication and key distribution latency during the handover process by distributing pairwise master keys (PMK) as proof of successful initial authentications to potential target access points of a mobile client before the client moves in contact with another access point.

- Ticket-based authentication and key distribution protocols [34, 42] try to minimize authentication and key distribution latency during the handover process by using

tickets as proof of successful initial authentications.

In the following subsections, we discuss the authentication and key distribution protocols in each category.

2.6.2.1 Multi-hop Authentication and Key Distribution

The current wireless mesh networking standard IEEE 802.11s [43, 44] uses IEEE 802.11i security standards [35]. Using IEEE 802.11i authentication protocols, such as EAP-TLS, a client is authenticated by an authentication server (AS), which may be many hops away from the client. When the client transfers from one MAP to another, the client has to be re-authenticated by the authentication server, which can incur long latencies.

IEEE 802.11F or inter-access point protocol (IAPP) is an optional extension to IEEE 802.11 that provides wireless access point communications among multi-vendor systems. When a client moves away from its current access point, it may start to search for a new access point. If a new access point is located, the client will send a re-association request frame to the new access point. The request contains the client's MAC address and the basic service set identifier (BSSID) of the old access point. Upon receiving the re-association request frame, the new access point sends an access-request message to the authentication server to verify the BSSID of the old access point. If that BSSID is valid, the authentication server will send an access-accept message to the new access point which contains security information for handover communication between the old and new access points. IAPP supports secure exchanges of clients' keys between the

current access point and the new access point during the handover process. However, IAPP does not effectively reduce the handover latency because both the current and new access points have to communicate with a remote authentication dial in user service (RADIUS) server during the handover process [45, 46].

Protocol for carrying authentication for network access (PANA) [36] is a network-layer transport for the extensible authentication protocol (EAP) defined in IEEE 802.11 standards that enables authentication between clients and access networks. PANA runs between a client and a server in order to perform authentication and authorization for the network access service. PANA does not define any new authentication mechanisms, but carries the EAP payload instead, which performs the authentication. Therefore, authentication during the handover still has to be performed via the multi-hop wireless communication mechanism of EAP.

Our proposed login authentication protocol (LAP) and handover authentication protocol (HAP) (described in section 3.2) only require one-hop communications during the intra-network handover authentication and key distribution process instead of multi-hop communication, thus significantly minimizing the handover latency and service interruption.

2.6.2.2 Proactive Authentication and Key Distribution

In the handover authentication and key distribution protocol of IEEE 802.11i, after the authentication server successfully authenticates a mobile client, it sends a key called pair-

wise master key (PMK) to the access point associated with the client. The client then performs the same calculation as the authentication server to obtain the same PMK. The access point and the client then use the PMK to derive a pair-wise transient key (PTK) for encrypting future packets exchanged between them [35]. The authentication server then sends the PMK to the neighbors of the current access point one-by-one. The PMK serves as proof of the client's successful authentication performed by the authentication server. By letting the authentication server pre-distribute the PMK to the neighbors of the current access point, the client does not need to be authenticated by the authentication server when it moves to another access point. However, the pre-distribution of keys by the authentication server incurs extra traffic overhead within the backhaul network. In addition, if the distance between the authentication server and a neighbor access point is long, the PMK may not arrive in time at the neighbor access point before the client moves and connects to that neighbor access point, thus causing service interruption. Consider the mesh network shown in Figure 2.23. Client C is authenticated successfully and connected to MAP M. The authentication server then sends a PMK to the neighbors of M, namely MAP N, R and P. When client C connects to a neighbor MAP in the future, the neighbor MAP will use the PMK to authenticate C. The authentication server distributes the PMK to N, R and P via three, four and five hops, respectively. This incurs traffic in the backhaul network. When the traffic load in the network is heavy, it may take longer for the PMK to reach the neighbor MAPs, and can increase the chance of service interruption if the client moves faster. Although our proposed

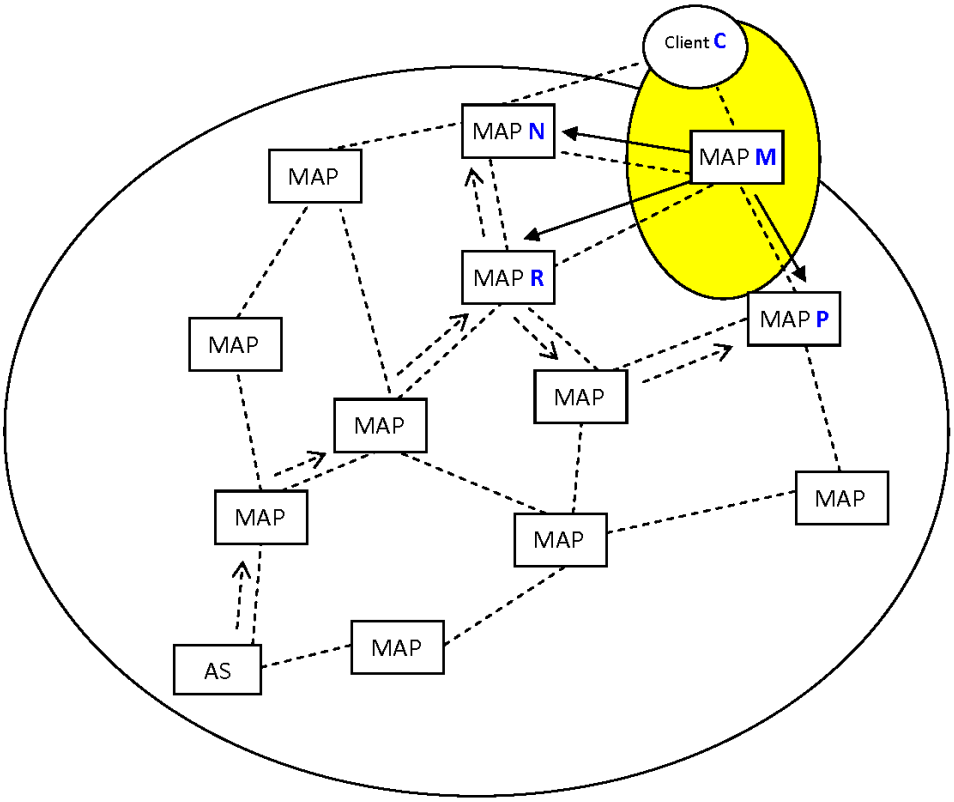


Figure 2.23: Backhaul overhead

handover authentication also employs pre-distribution of keys, it requires only one local transmission (one broadcast) from the current access point to its neighbors as opposed to multi-hop key pre-distributions in other protocols [40, 41], which impose extra traffic overhead in the backhaul network.

The scheme proposed by Mish et al. [40] pre-distributes PMKs using neighbor graphs. Once the mobile station A completes an initial full EAP-TLS authentication with an access point AP_i, the authentication server determines the neighbors of AP_i using the neighbor graph. The authentication server then sends a PMK to each neighboring node

N of access point AP_i. The client A also receives all the PMKs the authentication server sends to AP_i's neighbors. If A requests to re-associate with N in the future, N will use the PMK to authenticate A. When A roams and connects itself to a new access point, AP_j, the authentication server will in turn distribute a PMK to each of AP_j's neighbors. There are two major issues with this scheme. First, as the number of mobile clients in the network increases and as they move around the network, the PMK distribution task is a burden on the authentication server. Second, the PMK distribution consumes network bandwidth on a global scale.

Park et al. [41] also use neighbor graphs for pre-distribute keys. However, the authentication server does not distribute the PMKs. Instead, the authentication server distributes a set of matrices, which are then used by access points and mobile clients in combination with the key generation process proposed by Du et al. [47] to generate PMKs. This protocol proposed by Park in [41] works as follows. After the authentication server successfully authenticates a client C using EAP-TLS, it generates two matrices for C: a matrix M of size $h \times N$ and a matrix A of size $N \times h$, where N is the number of access points in the network, and $h < N$ is a random number chosen by the authentication server. Let i and j denote the identification numbers of client C and the associated access point, respectively. The authentication server then sends row $A(i)$ of matrix A and column $M(i)$ of matrix M to client C, and row $A(j)$ of matrix A and column $M(j)$ of matrix M to the access point. The matrix information is encrypted using the private key shared by the client (or the access point) and the authentication server. The client

and its associated access point then exchange columns $M(i)$ and $M(j)$, which serve as proofs of their initial successful authentications. They compute $K_{ij} = A(i) \times M(j)$ and $K_{ji} = A(j) \times M(i)$, respectively. K_{ij} and K_{ji} have the same value because matrix K is symmetric, which is the PMK shared by the client and the acess point. This scheme has the same drawbacks as the algorithm proposed by Mish et al. [40].

Our proposed handover authentication protocol (HAP) (described in section 3.2.2) distributes a shared key between neighboring MAPs (i.e., local traffic) and does not require the involvement of an authentication server, which significantly minimizes global traffic overheads and key pre-distribution latency.

2.6.2.3 Ticket-based Authentication and Key Distribution

Li [42] proposed a ticket-based authentication and key distribution protocol to support fast handover in WLANs, which is a proactive key distribution approach. After the authentication server successfully authenticates a mobile client C, it sends a set of tickets to C, one for each neighbor access point of the current access point C connects to. A ticket for a neighbor N contains the encrypted PMK to be shared by C and N later, when C moves to the service area of N. The authentication server distributes the PMKs, which are stored in the set of tickets C owns, to the neighbor access points in preparation for C's roaming. The major drawback of this scheme is the distribution of PMKs to the neighbor access points, which is acceptable in the wired backbone of a WLAN, but is bandwidth-consuming in the wireless backbone of a WMN. In addition, the authentication server

has to generate a large number of tickets, one for each client-AP pair in the network. However, our proposed protocols only require to generate one ticket per client.

The protocol proposed by Kassab [34] is very similar to that by Mish et al. [40] discussed earlier. After the authentication server successfully authenticates a mobile client C, it sends a set of PMKs to C and to the neighbor access points of the current access point C that is associated with, one PMK per client-access point pair. When C roams to a neighbor access point N, it generates a ticket that is encrypted with the PMK shared by C and N. N uses the shared PMK to verify the ticket and authenticates C. This protocol has the same disadvantages as the proactive key distribution scheme proposed by Mish et al. [40].

Shames Qazi [48] proposed a ticket-based authentication scheme for WMNs. The authentication server assigns tickets to registered mesh clients so that they can communicate with each other. The scheme is designed for authentication between mesh clients, and not between MAPs and clients. In addition, it does not provide any solution for fast authentication and key distribution during intra-network handover.

Anmin Fu [49] proposed a handover authentication and key distribution mechanism based on tickets for IEEE 802.16m (mobile WiMAX). In this scheme, all the access points and clients in a network are considered as a group and share a group key. After the authentication server successfully authenticates a client C, it generates a ticket for C, which is encrypted with the group key, and sends the ticket to C. When C moves to another access point N, it submits the ticket to N, who verifies the ticket using the group

key. In large mesh networks, using a single group key for the whole network is neither secure nor scalable.

A qualitative comparison of our proposed login authentication protocol (LAP) and handover authentication protocol (HAP) with other protocols is given in Table 2.1, where n denotes the number of neighbor MAPs of the current MAP to which a client is currently connected.

The major difference between our handover authentication scheme (discussed in section 3.2.2) and the other ticket-based protocols is that in our scheme keys and certificates needed for a handover are distributed by a MAP to its one-hop neighbors, while in the other protocols they are distributed by the authentication servers which are typically multiple hops away from the neighbor MAPs.

2.6.3 Authentication and Key Distribution Schemes for Inter-network Handovers

In this section, we provide a literature review of authentication and key distribution protocols for secure inter-network handovers in wireless networks. We broadly divide authentication and key distribution protocols for inter-network handovers into four categories: 3GPP authentication and key agreement protocols, authentication protocols for secure roaming service in GLOMONET, key distribution approach for inter-network handovers in heterogeneous wireless networks, and EAP-based inter-domain authentication protocols.

Table 2.1: Comparison of authentication approaches in wireless networks

Protocol	Type of Authentication	Login or Handover?	# of Hops[a]	Authentication Sever Involved?	Backhaul Overhead[b]	Neighbor Graph Required?
EAP-TLS [43, 44]	Multi-hop	Login	Multiple	Yes	9	No
IAPP [45, 46]	Multi-hop	Handover	Multiple	Yes	2	No
PANA [36]	Multi-hop	Login	Multiple	Yes	7	No
Mobile IP [37, 38, 39]	Multi-hop	Handover	Multiple	Yes	2	No
LAP (Chapter 3)	Certificate	Login	One	No	0	No
802.11i handover [35]	Pro-active	Handover	Multiple	Yes	n	No
Mish et al. [40]	Pro-active	Handover	Multiple	Yes	max 3n	Yes
Park et al. [41]	Pro-active	Handover	Multiple	Yes	2n + 1	Yes
Kassab et al. [34]	Ticket-based	Handover	Multiple	Yes	n + 1	Yes
Anmin Fu [49]	Ticket-based	Handover	Multiple	Yes	n	No
HAP (Chapter 3)	Certificate	Handover	One	No	0	No

[a] Number of hops between a client and the authenticator.

[b] Number of messages exchanged between MAPs and the authentication server to prepare for an intra-network handover.

2.6.3.1 3GPP Authentication and Key Agreement Protocols

The universal mobile telecommunication system (UMTS) is a prominent standard in 3G wireless systems. UMTS adopts an enhanced authentication and key agreement from the third-generation partnership project (3GPP), also referred to as 3GPP AKA [57]. 3GPP AKA greatly enhances the authentication framework of the global system for mobile communications (GSM) system with some added security features such as the integrity check feature and strong encryption algorithms.

In the 3GPP AKA protocol, besides authenticating each other, a user and a network agree on the cipher and integrity keys, CK and IK, respectively. These keys are derived from the user's secret key K and revealing them may disclose information about the user's secret key. Therefore, if there exists any vulnerability in the 3GPP AKA protocol, the subscriber's key K may be compromised.

The two main design goals of 3GPP AKA are mutual authentication between a user and a foreign network, and the establishment of a new pair of cipher and integrity keys (CK and IK) after a successful mutual authentication. Since CK and IK are derived from the user's secret key K, the user's secret key may be compromised if CK and IK are known to other users. Hence, a secure connection, via mutual authentication, needs to be established between a user and a foreign network before session keys (CK and IK) can be exchanged. The user maintains a counter SQN_{MS} to verify the freshness of CK and IK. So does the home network using a counter SQN_{HN}.

Figure 2.24 illustrates the steps of the 3GPP AKA protocol.

- A user MS sends a request including its ID $IMSI$ to a foreign network FN via the home network HN. The foreign network FN identifies user MS by its ID $IMSI$, and then sends the authentication data request including the user's ID $IMSI$ to the home network HN.

- Upon the receipt of the authentication data request, the home network sends an authentication data response back to the foreign network FN that contains an ordered array of n authentication vectors $AV(1...n)$. An authentication vector is a concatenation of the following fields/variables: RAND, XRES, CK, IK, and AUTN. The generation of an authentication vector is shown in Figure 2.25.

- The foreign network FN selects the next unused authentication vector from the ordered array of AVs in its database. FN then sends the random challenge RAND and the authentication token AUTN extracted from the AV to the user MS.

- MS computes the anonymity key AK as $AK = f_5(K, RAND)$, where K is the user's secret. MS also computes the sequence number SQN as $SQN = (SQN \oplus AK) \oplus AK$.

  - If $SQN \neq (SQN \oplus AK) \oplus AK$, user MS sends a user authentication reject message to FN and aborts the procedure.

  - If $SQN = (SQN \oplus AK) \oplus AK$, user MS verifies if the received SQN is in the correct range.

    * If $SQN >= SQN_{MS}$, the authentication of the FN is successful. User

MS computes $XRES$ and sends it to foreign network FN after verifying that the received SQN is in the correct range, (e.g., SQN is in the range of 1 to n).

* If $SQN < SQN_{MS}$, MS sends a synchronization failure message to FN. After receiving the synchronization failure message from the MS, FN sends a message to the home network HN, which contains RAND and AUTN. The home network HN then checks if the SQN_{HN} is indeed out of range. If so, it sets its own SQN to that of the received SQN_{MS} and generates a new batch of AVs which are sent back to the FN. After receiving the new batch of AVs the FN deletes the old ones and replaces them with the new AVs.

FN compares the received $XRES$ sent from MS with the one in the AV. If they match, the authentication of MS is successful. After successfully authenticating each other, MS and FN obtains CK and IK from the AV.

An adaptive protocol for authentication and key agreement (AP-AKA) [50] was developed based on the framework of 3GPP AKA to enhance the security of the 3GPP AKA. Both 3GPP AKA and AP-AKA follow the home-foreign-domain model and provide enhancement to achieve mutual authentication. Compared to 3GPP AKA, the home network HN in AP-AKA does not maintain a counter for each user. This modification eliminates the operational overhead of re-synchronization. The analysis in [50] shows that 3GPP AKA is weak towards a special case of redirection attacks while AP-AKA is

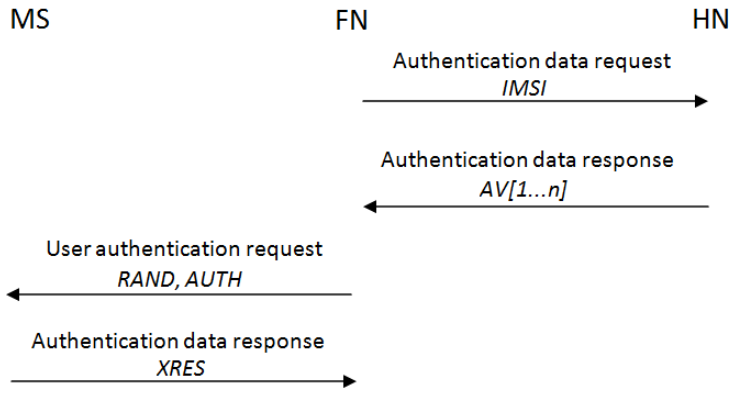


Figure 2.24: 3GPP AKA

robust against it. However, in AP-AKA the user's traffic redirection via a virtual relay to a neighboring network could cause the user to be charged more than usual because the location of the user has been virtually changed. Also, AP-AKA introduces two additional message exchanges between a user and a foreign network, which adds some signaling cost. The first step of the AP-AKA is not integrity protected, so it could be forged. Also, a man-in-the-middle attack can be executed on AP-AKA while inter-networking with a GSM, because the foreign network initiates the AKA procedure without an integrity check.

2.6.3.2 Authentication Protocols for Secure Roaming Service in GLOMONET

GLOBAL mobility network (GLOMONET) [58], such as GSM and CDMA (code division multiple access) networks, offers effective global roaming service for a legitimate user between a home network and a visited network. However, it also increases the possibility of

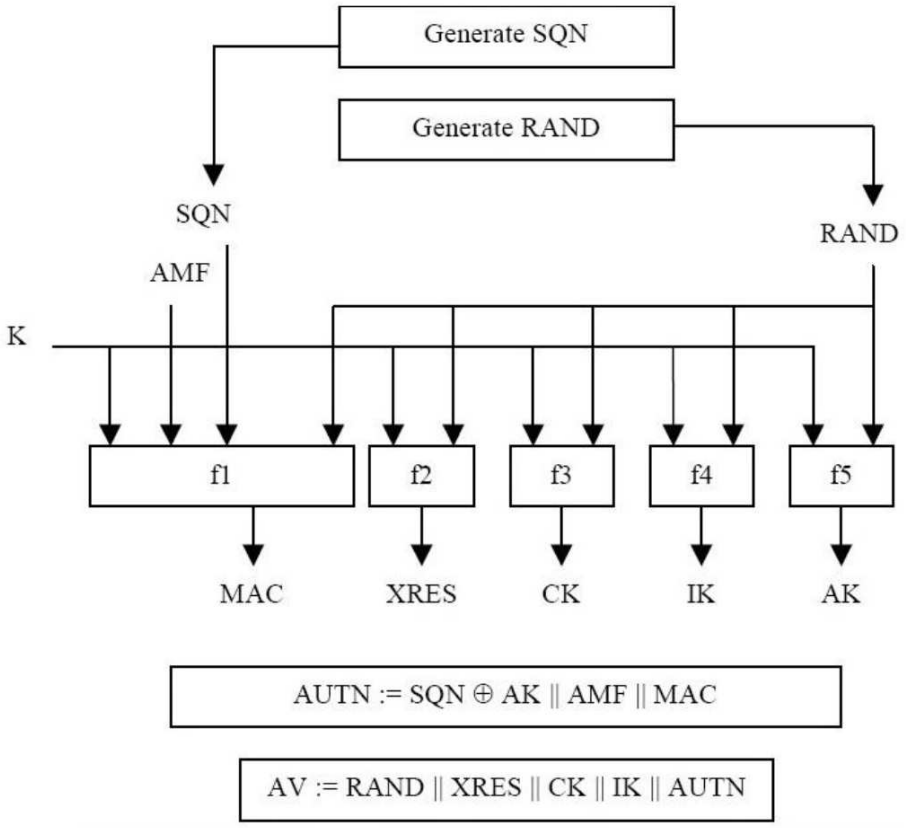


Figure 2.25: Generation of authentication vectors

illegal access from malicious intruders. Several authentication protocols [9, 58, 59, 60] for global roaming services have been developed in the GLOMONET. Suzuki et al developed an authentication protocol for roaming services [58]. They introduced a challenge/response interactive authentication mechanism with a symmetric cryptosystem to construct their authentication protocol. Buttyan et al. pointed out some potential attacks to the authentication protocol in [58], and further proposed an improved protocol and made it resistant against the presented attacks [59]. Subsequently, Hwang et al. [60] introduced a new self-encryption mechanism to simplify the protocol in [59].

Two protocols proposed in [9] further simplify the authentication protocols proposed in [60] and aim at providing identity anonymity for roaming users in the GLOMONET environment.

The first protocol in [9] hides the real user's identity via a prearranged pseudonym identity PID based on the secret splitting principle [61]. Secret splitting is a type of information-hidden technique that divides a message into pieces. Each piece by itself has no meaning, but when these pieces are put together, the original message can be restored. In this protocol, a foreign network F authenticates a roaming user M through M's home network H. After the authentication, an authentication key is established between M and F. In subsequent communications, F can directly authenticate M by using the authentication key rather than authenticating it through H. The proposed protocol uses a symmetric-key algorithm and can be applied when a foreign network and a home network have a pre-established shared secret.

The second protocol in [9] hides the real user's identity by encrypting the real identity with a shared key based on a self-certified scheme [62]. This protocol considers home network H as a temporary trusted third party (TTY) for roaming services. When M visits F, both of them initialize a registration procedure with H (F acts as an access agent for M). If M and F successfully register with H, they both obtain a witness from H, and the trust relations between M and F are established. In subsequent communications, M can directly negotiate a session key with F without accessing H. The protocol uses a public-key algorithm and can be applied when a foreign network and a home network do not have a pre-established shared secret.

All of these authentication protocols are based on a home-foreign-domain model, which requires multi-hop wireless communications to a client's home network and results in longer delays for inter-network handovers.

2.6.3.3 Key Distribution Approach for Inter-network Handovers in Heterogeneous Wireless Networks

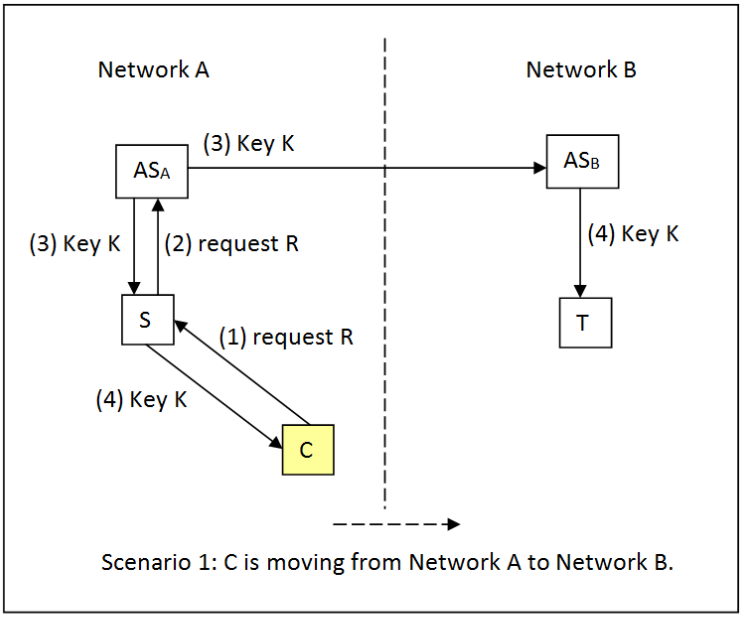
For a seamless handover, keys must be available at the target network at the time of the handover. Non-cellular wireless access networks, such as wireless mesh networks, do not have a dedicated handover infrastructure. As a result, no special entities are available to perform key distributions.

Hoeper [51] proposed a key distribution approach for inter-network handovers in heterogeneous wireless networks. A heterogeneous wireless network consists of devices using

different underlying radio access technologies such as Wi-Fi, 3G, or 4G. Suppose a client C moves from a router S of network A to a router T of network B, where network A and B have roaming agreements for supporting inter-network handovers. Client C and network B need to establish a shared secret key before C moves to B. Authentication server AS_A of network A and authentication server AS_B of network B serve as key distributors. Multi-hop wireless communications are required between router S and authentication server AS_A in network A, and also between router T and authentication server AS_B in network B.

Suppose C is moving from router S of network A to router T of network B. Key distribution protocols are triggered by client C's handover request R in two ways as shown in Figure 2.26. In scenario 1, a shared key K is distributed through C's currently connected router S. In this case, S forwards C's request R to its authentication server AS_A for moving to network B. Upon receiving the request, AS_A generates a key K and sends it to AS_B. AS_B then forwards key K to T. AS_A also sends key K to C through S. Both C and target router T then have a shared key K. In scenario 2, a shared key K is distributed through C's target router T. In this case, C sends a key distribution request R to T, which forwards the request R to its key distributor AS_B. AS_B then requests key K from AS_A who returns key K to T.

This key distribution solution requires server access during a handover process. It imposes communication delays and additional network traffic which significantly slows down the handover process.



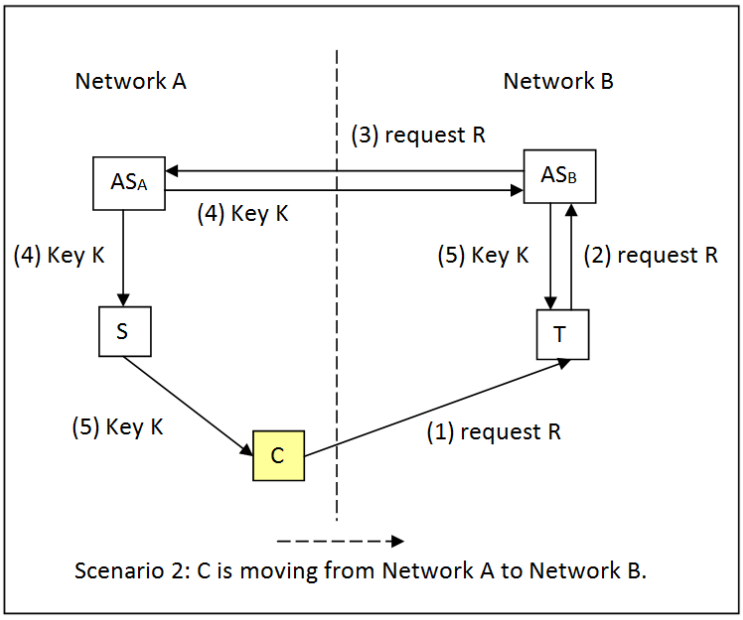


Figure 2.26: An example of key distribution methods proposed by Hoeper

2.6.3.4 EAP-based Inter-domain Authentication Protocols

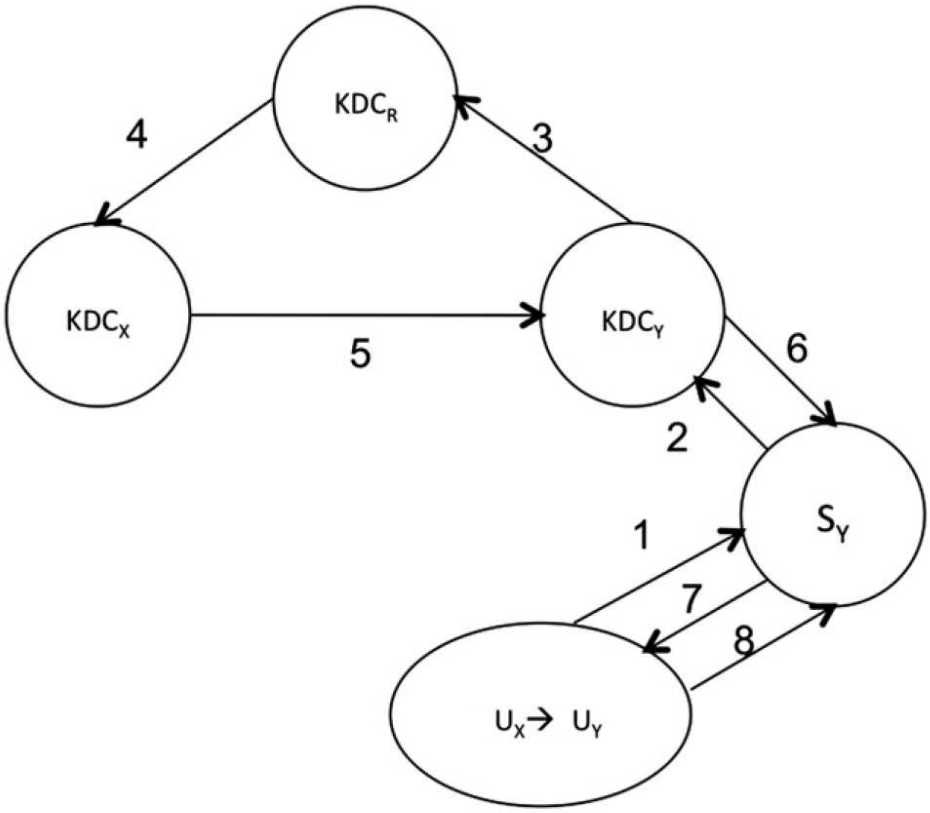
We discuss two inter-domain authentication protocols that uses the extensible authentication protocol (EAP) framework, namely, the one-time key secure network protocol (ONSP) [63] and the inter-domain authentication protocol proposed by Heba K. Aslan in [53].

The one-time key secure network protocol (ONSP) [63] proposed for WiFi (802.11) networks is a modified version of the Kerberos authentication protocol within the existing extensible authentication protocol (EAP) framework. ONSP provides inter-domain authentication to a user that already has a security association with the home domain. ONSP requires a key distribution center (KDC) to manage the authentication between users and servers.

For a very large network, it is impractical to register all the users in a single domain. In ONSP, users and servers register with their own KDCs in a hierarchical structure. Each node in the hierarchy represents a domain, and a parent domains administers all its child domains. Each domain has one KDC to manage the authentication of its users and servers. Every KDC shares a different secret key with each descendant KDC to execute inter-domain authentication. As a result, the root KDC has to save the keys shared with all descendant KDCs, one key per KDC.

Figure 2.27 illustrates the authentication flow in ONSP for an inter-domain authentication process.

1. A user U_X roaming from domain X wants to access a server S_Y in a foreign domain



1. $U_X \rightarrow S_Y : AURQ_{U_X}$
2. $S_Y \rightarrow KDC_Y : AURQ_{U_X} || AURQ_{S_Y}$
3. $KDC_Y \rightarrow KDC_R : AURQ_{U_X} || AURQ_{KDC_Y}$
4. $KDC_R \rightarrow KDC_X : AUFW_{KDC_X} || AURS_{KDC_Y}$
5. $KDC_X \rightarrow KDC_Y : AURS_{U_X} || AURS_{KDC_Y}$
6. $KDC_Y \rightarrow S_Y : SID || AURS_{U_X} || AURS_{S_Y}$
7. $S_Y \rightarrow U_X : AURS_{U_X} || CH_{S_Y} || ST_{S_Y}$
8. $U_X \rightarrow S_Y : RES_{S_Y} || A_{U_Y}$

Figure 2.27: The ONSP inter-domain authentication process

Y. The user U_X sends an authentication request message $AURQ_{U_X}$ to the server S_Y.

2. S_Y generates an authentication request $AURQ_{S_Y}$, and sends $AURQ_{S_Y}$ and the user's authentication request $AURQ_{U_X}$ to the KDC of domain Y, KDC_Y.

3. KDC_Y creates an authentication request $AURQ_{KDC_Y}$, forwards the user's authentication request $AURQ_{U_X}$ and KDC_Y's authentication request $AURQ_{KDC_Y}$ to key distribute center KDC_R, the parent KDC of KDC_X and KDC_Y.

4. KDC_R generates an authentication request $AUFW_{KDC_X}$ which contains the user's authentication request $AURQ_{U_X}$, a new user identity U_Y, a new temporary user key K_{U_Y} and a new session key K_{SS}.KDC_R encrypts the authentication request $AUFW_{KDC_X}$ using a key K_{KDC_X} shared by KDC_R and KDC_X. In addition, KDC_R creates an authentication response $AURS_{KDC_Y}$ to KDC_Y after verifying KDC_Y's authentication request $AURQ_{KDC_Y}$. KDC_R then sends the authentication request $AUFW_{KDC_X}$ and the authentication response $AURS_{KDC_Y}$ to KDC_X.

5. KDC_X decrypts the authentication request $AUFW_{KDC_X}$ using the shared key K_{KDC_X} to obtain U_Y, K_{U_Y} and K_{SS}. KDC_X then generates an authentication response $AURS_{U_X}$ for the user. KDC_X sends two authentication responses to KDC_Y: one is the authentication response $AURS_{KDC_Y}$ for KDC_Y from KDC_R; the other is the authentication response $AURS_{U_X}$ for U_X from KDC_X.

6. After verifying the authentication response $AURS_{KDC_Y}$ from KDC_R, KDC_Y is

considered to successfully authenticate KDC_R. KDC_Y obtains the user identity U_Y and temporary key K_{U_Y} generated by KDC_R. KDC_Y then generates a new session identity SID and an authentication response $AURS_{S_Y}$ to server S_Y. KDC_Y sends to server S_Y the new session identity SID and two authentication responses: one is the authentication response $AURS_{U_X}$ from KDC_X to the user, and the other is the authentication response $AURS_{S_Y}$ from KD_Y to S_Y.

7. The server S_Y extracts the session key K_{SS} from the authentication response $AURS_{S_Y}$ of KDC_Y. S_Y creates a nonce NN_{S_Y} and then generates a challenge CH_{S_Y} using K_{SS}. S_Y also generates a service ticket ST_{S_Y} for the user using K_{SS}. S_Y sends to the user the challenge CH_{S_Y}, the service ticket ST_{S_Y}, and the authentication response $AURS_{U_X}$ of KDC_X.

8. The user U_X obtains the session key K_{SS} and the new identity U_Y from the authentication response $AURS_{U_X}$. U_X creates a response RES_{S_Y} to CH_{S_Y} with the nonce NN_{S_Y} after decrypting CH_{S_Y} using key K_{SS}. In addition, the user generates a temporary authenticator A_{U_Y} to be used for subsequent authentication.

In order to allow a user to access service on a foreign server S_Y, several authentication rounds are required to be performed, such as between the nearest common KDC and the foreign KDC, between the previously visited KDC and the user, between the foreign KDC and the foreign server, and between the foreign server and the user. Thus, ONSP inter-domain authentication suffers from high authentication delay due to multiple

authentications performed between several network entities.

The inter-domain authentication protocol proposed in [53] also uses the EAP protocol for authentication and key distribution. The proposed protocol is based on the use of hash functions and the Diffie-Hellman algorithm to distributes authentication keys between mobile stations and base stations. In order to avoid the domino effect, the Diffie-Hellman components are distributed instead of the authentication keys themselves. However, the use of the Diffie-Hellman algorithm increases the authentication latency due to expensive exponentiation and modular arithmetic operations that both the MS and the BS have to perform. Similar to ONSP [63], an authentication server is involved in the authentication process to accept a user's handover request.

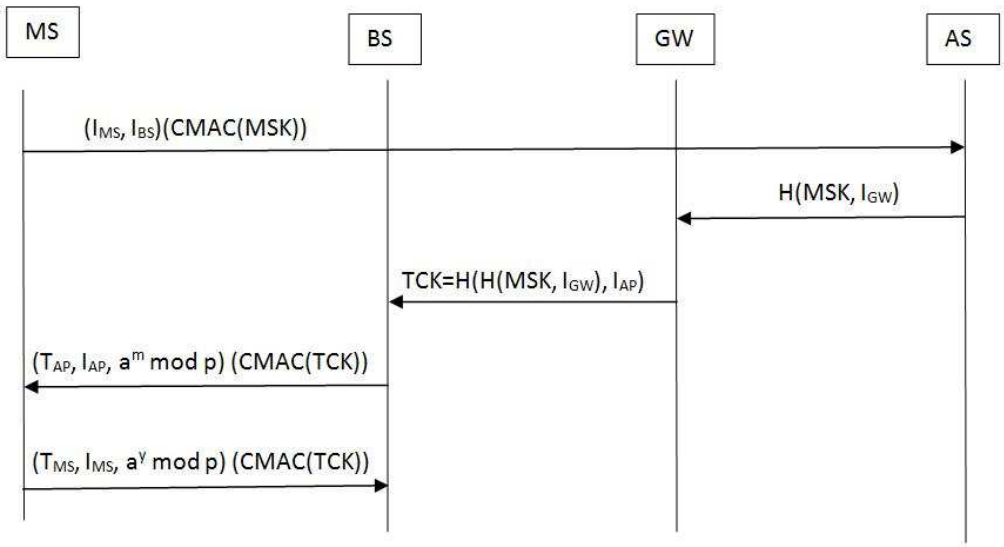


Figure 2.28: Inter-domain handover authentication process

Figure 2.28 illustrates the steps of the inter-domain handover authentication protocol.

- When a mobile station (MS) moves from one base station (BS) to anther in a different region, it sends to the authentication server (AS) a request containing its identity I_{MS}, the identity of the target base station I_{BS}, and a cipher-based message authentication code (CMAC) value of the message computed using a master session key (MSK). A master session key is shared by a mobile station and the authentication server. A MSK is unique for each mobile station.

- The MS and the AS calculate a hash value $H(MSK, I_{GW})$ using the MSK and the identity of the foreign network gateway (I_{GW}). The AS forwards the hashed value $H(MSK, I_{GW})$ to the GW.

- The MS and the GW compute a temporary CMAC key $TCK = H(H(MSK, I_{GW}), I_{BS})$. The GW then forwards the temporary key TCK to the BS.

- The BS sends to the MS a message $\{T_{BS}, I_{BS}, a^m \bmod p, CMAC(TCK)\}$, which contains a timestamp T_{BS}, the identity of BS (I_{BS}), its Diffie-Hellman component $a^m \bmod p$, where m is the Diffie-Hellman secret of the BS, and a CMAC value computed using TCK. After receiving the message, the MS verifies the freshness of the message using T_{BS}, and then uses TCK to validate the CMAC value. If the computed CMAC value is same as the one in the massage, the MS is considered to successfully authenticate the BS.

- The MS sends to the BS a message $\{T_{MS}, I_{MS}, a^y \bmod p, CMAC(TCK)\}$ containing a timestamp T_{MS}, its identity I_{MS}, its Diffie-Hellman component $a^y \bmod p$,

where y is the Diffie-Hellman secret of the MS, and a CMAC value computed using TCK. The BS is considered to successfully authenticate the MS if the timestamp T_{MS} is valid, and the computed CMAC value $CMAC(TCK)$ matches the MAC value the BS previously sent to the MS. Both the MS and the BS then calculate the authentication key $AK = a^{ym} \bmod p$, which will be used to derive the traffic encryption key for subsequent communications.

All the protocols discussed above require the involvement of a client's home network or a third party entity during the handover. Consider the scenario in Figure 2.29 where a client roams from WMN_A to WMN_B. If using one of the above authentication and key distribution schemes in WMNs, a client needs to communicate with its home network or a third party entity via the wired backhaul of WMN_B. The messages exchanged between the client and the wired backhaul of WMN_B are via multi-hop wireless communications. As the number of wireless hops increases from one to five, the delay of a message between the routers increases from 0.15 seconds to 0.8 seconds [7]. Since the authentication process involves several messages, the handover latency may be several seconds long. The latency of multi-hop wireless communications during the inter-network handover process may result in longer delay, thus leading to potential service disruptions. This limitation of the above protocols motivates us to propose a new authentication and key distribution solution for fast inter-network handovers.

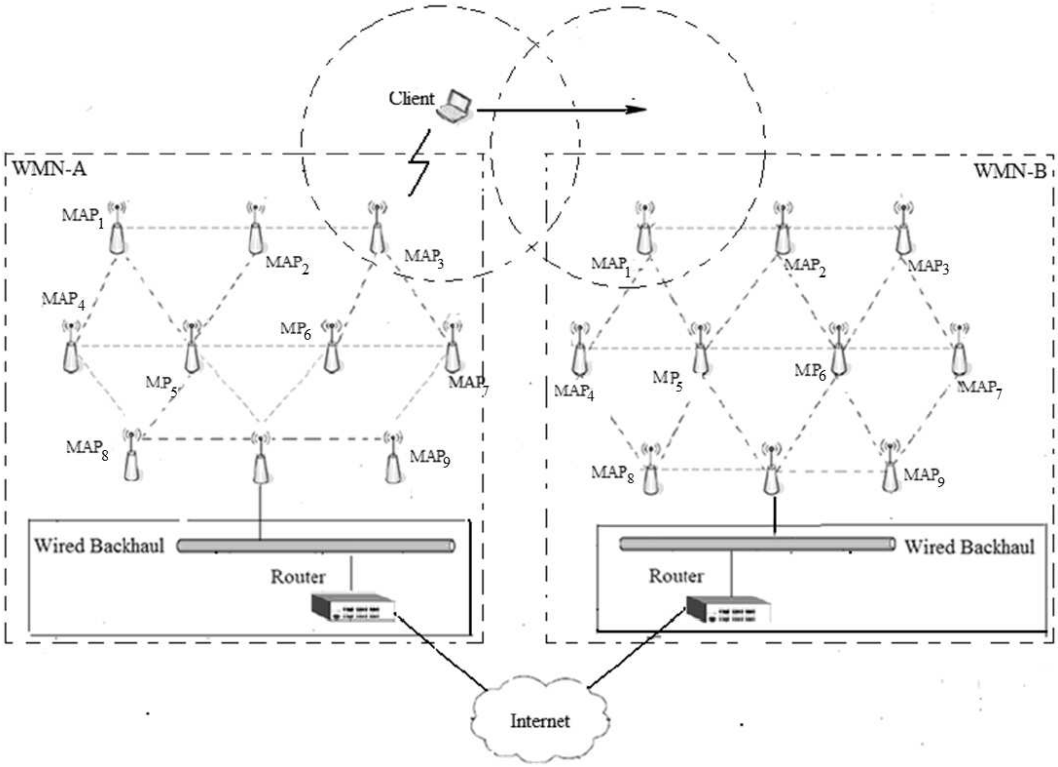


Figure 2.29: Inter-network handover scenario

2.7 Group Key Management

Group key management (GKM) refers to the actions taken to update and distribute the group key upon members joining and leaving a multicast group. When an existing member leaves (or a new member joins) the multicast group, the group key must be updated accordingly to achieve *forward secrecy* (or *backward secrecy*). Forward secrecy ensures that after an existing member leaves the group, he/she will not have access to future keys. Backward secrecy ensures that after a new member joins the group, he/she will not have access to past keys.

Existing work on GKM has been proposed for group communications at the application layer. To the best of our knowledge, no work has addressed the issue of GKM at the data link layer for multicast/broadcast. (We often use the term "data link" instead of "medium access control" in this dissertation because the former is shorter, and the abbreviation "MAC" is used to denote "message authentication code".) In this section, we first review existing work on GKM at the application layer. We then discuss the applicability of the reviewed works to GKM at the data link layer in WMNs.

GKM for groups of small to medium sizes can be classified into two approaches: centralized and contributory.

2.7.1 Centralized GKM

In the *centralized approach* [13, 14, 64, 65, 66, 118, 121, 122, 123, 124], a central key server (CKS) is responsible for generating, updating and distributing the group key of a

multicast group. Every time a member joins (leaves) the group, the CKS generates and distributes a new group key to the whole group in order to ensure backward (forward) secrecy. For the purpose of key distribution, each member i of the group shares a secret key k_i with the CKS.

The simplest key distribution mechanism is based on a flat structure [122, 123, 124]. When a new member u joins the group, the CKS generates a new group key K'. It then encrypts the new group key K' using the current group key K and sends the encrypted message to the whole group. The current members of the group will decrypt the message using the current group key K to obtain the new group key K'. Because the new member u does not know the current group key K, the CKS has to encrypt the new group key K' using the secret key k_u it shares with the new member u, and sends the encrypted message to u. Member u will decrypt the message using key k_u to obtain the new group key K'.

When a member v leaves a group, the CKS generates a new group key K''. Unlike the above GKM procedure for a join operation, a leave operation does not allow the CKS to encrypt the new group key K'' using the current group key K and then send the encrypted message to the whole group. If the CKS did, the leaving member v would know the new group key K'' since v knows the current group key K; that would violate the forward secrecy requirement. Therefore, the CKS has to encrypt the new group key K'' using each member i's individual shared key k_i (excluding member v), resulting in $O(n)$ encrypted messages, where n is the number of members in the multicast group. As

the group size becomes large, GKM based on the flat structure is not scalable, especially when members leave the group often.

To improve the scalability of GKM for large groups, logical key trees have been used in many GKM algorithms [13, 14, 64, 65, 66, 118, 121]. Thanks to the use of logical key trees, the number of encrypted messages required for a leave operation that the CKS has to send is reduced from $O(n)$ to $O(\log n)$. A detailed description of the logical key tree GKM approach can be found in Section 5.1. Representatives of GKM algorithms in this category are the logical key hierarchy (LKH) [13] and one-way function tree (OFT) [14] algorithms, which will be described in detail in Sections 5.1.2 and 5.1.3, respectively.

2.7.2 Contributory GKM

The centralized GKM approach assumes that there is a secure channel between each member and the CKS so that they can establish a shared key before GKM operations start. In some types of networks such as mobile ad hoc networks or sensor networks, such pre-established secure channels are not readily available. This limitation of these networks calls for a different approach to GKM called contributory GKM.

In contrast to the centralized GKM approach, the contributory GKM approach does not use a central key server. Instead, each group member contributes an equal share to the common group key (which is computed as a function of all members' contributions). Figure 2.30 shows an example of contributory GKM. Four members contribute four keys, K_1, K_2, K_3 and K_4. The group key K is a function of the four members' keys: $K =$

$f(K_1, K_2, K_3, K_4)$.

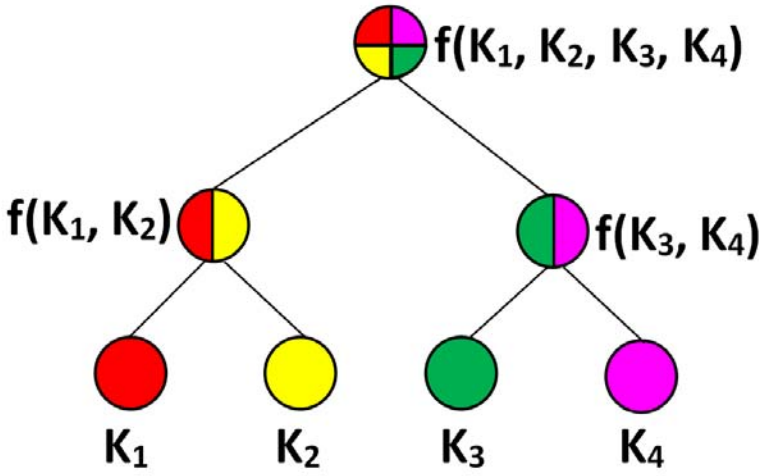


Figure 2.30: An example of a contributory approach

Due to the lack of pre-established secure channels, many GKM algorithms proposed for wireless ad-hoc networks fall into this category [127, 128, 129, 130].

Existing contributory GKM are generally based on the Diffie-Hellman key exchange algorithm [118]. The Diffie-Hellman algorithm involves several exponentiation operations, which can be computationally expensive to be executed on resource-constrained mobile devices. (This high computational cost is a trade-off for the unavailability of pre-established secure communication channels.)

For very large multicast groups distributed over vast geographical areas, centralized and contributory GKM schemes may not be able to handle such groups. In this case, decentralized GKM algorithms should be used instead.

2.7.3 Decentralized GKM

Using the *decentralized approach* [67, 68, 119, 120, 125, 126], a multicast group is organized into smaller subgroups, each having its own subgroup key. Each subgroup is managed by a subgroup controller which is in charge of key computation and distribution within its subgroup. Subgroup controllers and subgroups can exist on multiple levels, and lower-level subgroup controllers are clients of higher-level subgroup controllers. The absence of a general group key means membership changes in a subgroup are treated locally. Thus, a membership change impacts only the subgroup of the member: only the subgroup key needs to be updated, independently of the keys of the other subgroups. Different subgroups may use different GKM protocols. This approach can allow more entities to fail before the whole group is affected, minimizing the problem of concentrating the work on a single centralized key server. Thus, it reduces the risk of total system failure.

Figure 2.31 illustrates a decentralized GKM architecture with six subgroups. Each of subgroups has its own subgroup key. A main group controller maintains control of the top-level subgroup. There are no clients in the top-level subgroup, only the group controller and its subgroup controllers being part of this subgroup. As shown in Figure 2.31, GC, SGC_2, SGC_3 and GSC_4 form the top-level subgroup. GC generates a subgroup key K_1, encrypts it with each subgroup controller's individual key and sends to SGC_2, SGC_3 and GSC_4. The subgroup controller (e.g., SGC_2, SGC_3, GSC_4) then decrypts it, re-encrypts it with its own subgroup key (e.g., K_2, K_3, K_4), and then multicasts it to its own subgroup. When a membership change occurs in a subgroup, only that subgroup

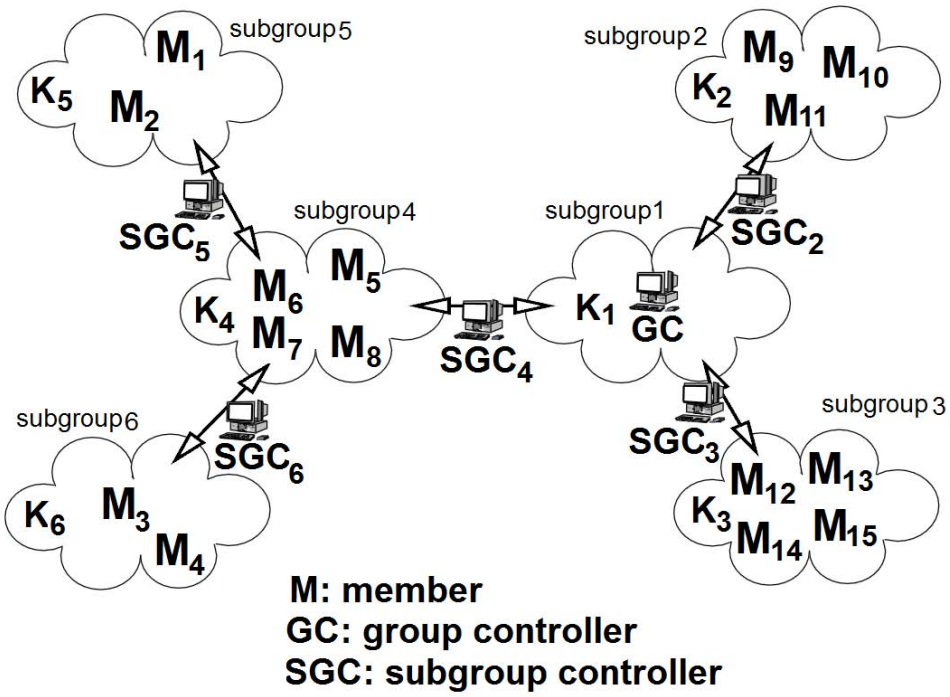


Figure 2.31: An example of a decentralized architecture

involved in a rekeying process. For example, when member M_1 or M_2 leaves subgroup 5, only the subgroup K_5 needs to be updated. The subgroup controller SGC_5 generates a new subgroup key K_5' and distributes to the members in the subgroup 5. Other subgroup keys are not affected by the membership changes of the subgroup 5.

Using this approach, sending data is not as simple as multicasting the data to the group encrypted with a subgroup key because such a multicast will only reach the local subgroup. Two methods can be used for data transmission. We use Figure 2.31 as an example to illustrate the two data transmission methods.

The first method is simple and completely transparent to a sender.

- The source encrypts a multicast packet p using the group key K_1 as $E_{K_1}(p)$ and multicasts the encrypted packet $E_{K_1}(p)$ to SGC_2, SGC_3 and GSC_4.

- Subgroup key controllers SGC_2, SGC_3 and GSC_4 decrypt the packet $E_{K_1}(p)$ using key K_1, and re-encrypt the multicast packet p with the corresponding subgroup keys as $E_{K_2}(p)$, $E_{K_3}(p)$ and $E_{K_4}(p)$, respectively. Each subgroup controller then multicasts its encrypted message to its own subgroup.

- Members (e.g., M_5, M_6, M_7, M_8) in subgroup 4 decrypt the encrypted packet $E_{K_4}(p)$ using subgroup key K_4 to obtain the multicast packet p.

- Subgroup controllers SGC_5 and SGC_6 are also members of subgroup 4. After decrypting the encrypted message $E_{K_4}(p)$ using subgroup key K_4, SGC_5 and SGC_6 re-encrypt the multicast packet p using subgroup keys K_5 and K_6 as $E_{K_5}(p)$ and

$E_{K_6}(p)$, respectively.

- Subgroup controller SGC_5 then multicasts $E_{K_5}(p)$ to subgroup 5. Members in subgroup 5 decrypts message $E_{K_5}(p)$ using subgroup key K_5 to obtain the multicast packet p. (Similar operations are performed in subgroup 6.)

In the first method, to decrypt and re-encrypt multicast data would require an enormous amount of computational overhead on subgroup controllers.

In the second method, the source generates a random number as a data encryption key (DEK) on a per-transmission basis (i.e., each multicast packet will be encrypted with a different DEK). The sender uses a DEK to encrypt the multicast packet p as $E_{DEK}(p)$. In order to decrypt the encrypted message $E_{DEK}(p)$, members need to know the DEK. The source uses the first method to encrypt and distribute the DEK to all group members along with the encrypted packet $E_{DEK}(p)$. Subgroup controllers only need to forward each encrypted multicast packet $E_{DEK}(p)$ to members without decrypting and re-encrypting the multicast packet. In this method, the computational cost for decrypting and re-encrypting a multicast packet is reduced to the cost for decrypting and re-encrypting a randomly generated number, the DEK.

The decentralized architecture ensures scalability for GKM in very large scale networks such as cellular wireless network, WiMax, and 4G systems, where the members of a group may be distributed over different and vast geographical areas.

2.7.4 Applicability to GKM at the Data Link Layer

Among the above GKM approaches, the centralized approach is the most efficient and cost-effective for GKM at the data link layer in WMNs for the following reasons. First, there already exists a central controller that can generate and distribute the group key to the members, which is the access point of a basic service set. Second, the members of the group − the mobile devices connected to the access point − are physically located close to the central controller. Thus, there is no need for a decentralized scheme. Third, there is no need to use the contributory approach because there already exists a secure communication channel between the access point and each mobile device. The secure channel is provided by a shared key between the access point and and each device called a *pairwise transient key* (PTK) in IEEE 802.11 standards [35]. The PTK is generated through the 4-way handshake protocol of the IEEE 802.11i standard, and is used to encrypt unicast data between a mobile device and an access point.

As a result, we use the centralized approach for GKM at the data link layer in WMNs. Instead of the flat structure used in IEEE 802.11 GKM, we use the logical key tree structure for GKM at the data link layer in order to reduce the rekeying latency from $O(n)$ to $O(\log n)$.

2.8 Chapter Summary

In this chapter, we have reviewed wireless communications, cryptography, security threats, security models and trust management in wireless networks, and IEEE 802.11i security

standard. We present a literature review of existing security protocols for intra-network and inter-network handovers, as well as group key management schemes for fast rekeying implementation.

Chapter 3

Efficient Authentication for Fast Intra-Network Handovers in a Wireless Mesh Network

Authentication is essential in any service-oriented communication networks to identify and reject any unauthorized network access. Designs and implementations of authentication protocols, or any security protocols in general, for WMNs are challenging due to bandwidth-limited wireless channels; reduced throughput caused by wireless multi-hop routing [55]; vulnerable shared broadcast medium; distributed network architectures and operations; and resource-constrained mobile devices (e.g., cellular phones, PDAs). On the other hand, clients' mobility requires efficient, fast yet secure handover mechanisms.

Existing authentication protocols employed for wireless networks such as those in IEEE 802.11i and 802.11s standards do not meet the above needs and challenges. For

instance, the authentication protocol in IEEE 802.11i is a centralized scheme (intended for use in wireless local area networks) and requires an authentication server, which is not efficient for WMNs. First, multi-hop routing between an access point and the authentication server via *wireless* links would result in long delay, low reliability and thus potential service interruption. Second, a central authentication server impedes distributed operations and thus affects scalability. The IEEE 802.11s standard [43] defined for wireless mesh networks uses the same the same security architecture as IEEE 802.11i, and hence inherits the above drawbacks of IEEE 802.11i. Moreover, the current version of the IEEE 802.11s standard does not specify any mechanism to support fast handovers for mobile clients running real-time applications such as voice over IP (VoIP), newscast and tele-conferencing.

Our work in this chapter contributes towards extending the IEEE 802.11s standards to support fast handovers for mobile clients. In particular, we focus on fast authentication during the handover process as well as during the initial login time. We propose a new trust model and certificates for a WMN, based upon which our proposed authentication protocols are designed. Our proposed certificate-based authentication protocols are resource-efficient and resilient to attacks. No central authentication server is needed. Instead, mobile clients and mesh access points directly authenticate each other, avoiding wireless multi-hop communications. Fast authentication from one MAP to another in the same network during the handover process is supported using certificates. Performance analysis and simulation results show that our login authentication protocol improves the

latency of 802.11s login authentication, and our handover authentication protocol supports fast authentication during the handover process.

In Section 3.1, we describe the proposed certificates and trust model for intra-network handovers in detail. In Section 3.2, we present our login and handover authentication protocols. We discuss the security analysis in Section 3.3. We present the performance evaluations of the proposed protocols in Section 3.4. We summarize this chapter in Section 3.5.

3.1 The Proposed Trust Model and Certificates for Intra-network Handovers

In this section, we present a trust model and certificates upon which our authentication protocols are built. We describe in detail the different types of certificates used in the proposed authentication protocols to support fast authentication during the handover process in a WMN.

3.1.1 The Proposed Trust Model

We propose a trust model built upon the existing WMN architecture to offer mobile clients' seamless, fast handovers from one MAP to another in the same network.

Trust relationships among entities in a WMN are the basis for designing authentication protocols for intra-network handovers. The proposed trust model (shown in Figure 3.1) is built upon the concept of "certificate" and "certificate agent". A certificate is used to

establish the trust relationships among various entities in WMNs. A certificate agent is a trusted third party who issues and manages various types of certificates and trusted by the entities in a mesh network. A certificate agent's role can be compared to public-key certificate authorities or credit card issuers.

Following are the trust relationships among the network entities shown in Figure 3.1:

(1) certificate agent − client: The mutual trust is based on the public key certificates issued by the certificate authority and is established when a client applies for a client certificate from a certificate agent.

(2) certificate agent − mesh access points (MAPs): The mutual trust between a MAP and its certificate agent is established via the public key certificates issued by a public certificate authority (e.g., Comodo, Symantec, Godaddy, GlobeSign). The trust is established when a MAP applies for a MAP certificate from a certificate agent.

(3) MAP − client: The mutual trust relationship between a client and its home MAP is established via their respective client certificate and MAP certificate, which are described in Sections 3.1.2.1 and 3.1.2.2.

(4) MAP − MAP: Any two neighboring MAPs trust each other via their MAP certificates. This trust allows a client to roam among different MAPs in a mesh network.

Obtaining a client certificate or a MAP certificate is done offline before a client joins a network, and is not part of the authentication process. Thus, the public key operations for

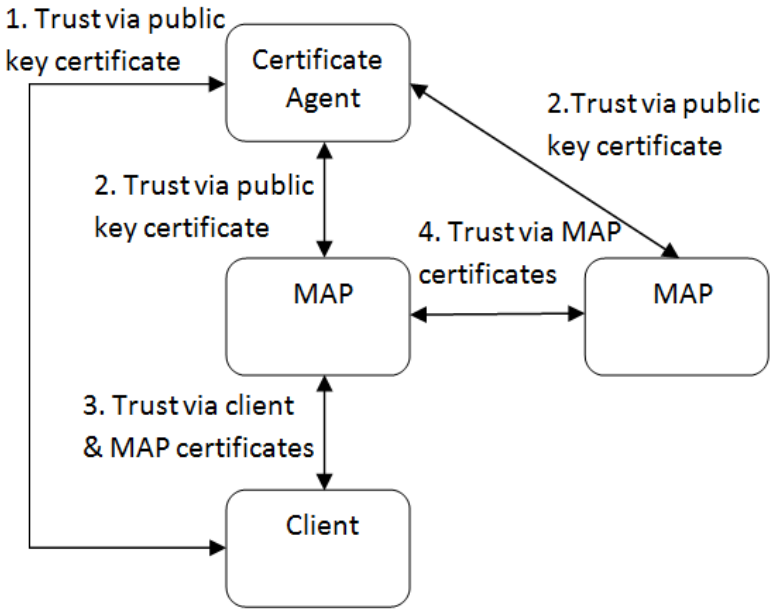


Figure 3.1: Trust model of a WMN

obtaining certificates do not affect the efficiency of our authentication protocols presented in Section 3.2.

3.1.2 Certificates Used in the Proposed Authentication Protocols

Certificates are issued and managed by certificate agents who are trusted by all entities of WMNs to perform such tasks. There can be several certificate agents serving a network. Certificates are used to establish the trust between a certificate agent and a MAP, a certificate agent and a client, a MAP and a client, and between a MAP and another MAP (see Figure 3.1). The lifetime of a certificate is determined by its issuer's policy.

Three types of certificates are used in our authentication protocols: client certificate, MAP certificate, and intra-network transfer certificate. They are needed for mutual

authentication between a client and a MAP when the client logs in to the network, or roams to another MAP in a single WMN.

We will use the notations listed in Table 3.1 throughout the chapter to facilitate the discussions.

3.1.2.1 Client Certificates

A client applies for a client certificate from a certificate agent. The trust between a client and a certificate agent is established through their public key certificates issued by a central authority.

Following is the structure of a client certificate:

$$\boxed{T_C = \{I_C, I_A, \tau_{exp}, P_C, Sig_A\}}$$

- T_C: client certificate issued by certificate agent A whose ID is I_A.
- I_C: ID of the client who has been given this certificate.
- I_A: ID of the certificate agent who issued the certificate T_C.
- τ_{exp}: expiry date and time of certificate T_C. The certificate agent will re-issue a new certificate for the client if the certificate has expired.
- P_C: public key of client I_C, which is used by a MAP to verify the signature signed by the client in the login authentication protocol (see Section 3.2.1). The certificate agent obtains the public key from the client's public key certificate. We assume that

Table 3.1: Notations

Notation	Description
C	Client
R	Mesh access point (MAP)
A	Certificate agent
I_x	ID of entity x
Θ_C	Intra-network transfer certificate issued to a client
P_x	Public key issued to x
T_x	Certificate issued to x
τ_{exp}	Expiry date and time of a certificate
N_x	A nonce generated by x
Sig_x	Digital signature of entity x
MAC_{alg}	Type of MAC algorithm
$E_{P_x}(m)$	Encryption of message, m using x's public key
$D_{P_x}(m)$	decryption of message, m using x's public key
$E_{K_{MAC}}(m)$	Encryption of message, m using MAC key K_{MAC}
K_{MAC}	The key used to produce a message authentication code (Section 3.1.2.3)
$V_k(m)$	Message authentication code (MAC) resulting from the application of a MAC algorithm and a MAC key k on a message m

the agent is a trusted party and has access to public key certificates of all clients and MAPs.

- Sig_A: digital signature of certificate agent I_A, which gives a recipient reason to believe that the certificate was created by certificate agent I_A, and that it was not altered in any way.

3.1.2.2 MAP Certificates

The operator of a mesh network applies for MAP certificates, one per MAP, and distributes them to the MAPs in the network. The operator is also responsible for requesting and distributing a new MAP certificate before the current MAP certificate expires. Following is the structure of a MAP certificate:

$$\boxed{T_R = \{I_R, I_A, \tau_{exp}, P_R, Sig_A\}}$$

- T_R: MAP certificate issued by certificate agent A whose ID is I_A.
- I_R: ID of the MAP that is given this certificate.
- I_A: ID of the certificate agent who issued certificate T_R to MAP R.
- τ_{exp}: expiry date and time of certificate T_R. The certificate agent will re-issue a new certificate for the MAP once the current certificate expires.
- P_R: public key of MAP R, which will be used by clients to verify the signature of MAP R in messages R sends. The certificate agent obtains the public key from the

MAP's public key certificate.

- Sig_A: digital signature of certificate agent I_A.

3.1.2.3 Intra-network Transfer Certificates

An intra-network transfer certificate is used to establish the trust relationship between a MAP and a client when a client roams from one MAP to another in a single WMN. When a client C first logs into the network, it sends its client certificate to a nearby MAP M_1, which will authenticate the client. If authentication succeeds, M_1 will issue to C an intra-network transfer certificate and become the home MAP of C. (We borrow the terminology from mobile IP.) When C roams to a foreign MAP M_2, it submits the certificate to M_2 for authentication. The intra-network transfer certificate proves to the foreign MAP that client C has been successfully authenticated by its home MAP.

The structure of an intra-network transfer certificate Θ_C is as follows:

$$\Theta_C = \{\mu, V_{K_{MAC}}(\mu)\}, \text{ where}$$
$$\mu = \{I_{cert}, I_R, I_C, P_C, \tau_{exp}, MAC_{alg}\}$$

Message μ stores the information of the client and home MAP as follows:

- I_{cert}: ID of the intra-network transfer certificate. The combination of I_{cert}, I_R and I_C uniquely identifies a transfer certificate in the network.

- I_R: ID of the MAP who issues this intra-network transfer certificate.

- I_C: ID of the client who owns this intra-network transfer certificate.

- P_C: public key of the client. The client's home MAP obtains the client's public key from C's client certificate.

- τ_{exp}: expiry date and time of this certificate.

- MAC_{alg}: message authentication code algorithm. (The inclusion of the type of MAC algorithm in an intra-network transfer certificate is optional. It is not required if the parties agree on an algorithm in advance.)

We now discuss about the value $V_{K_{MAC}}(\mu)$ stored in the intra-network transfer certificate and the use of the MAC algorithm. During the authentication between client C and its home MAP M_1 (step (1) in Figure 3.2), they exchange two partial keys (also called *nonces*[1]) N_{C1} and N_{R1} (see Section 3.2.1 for details of the authentication procedure). They will both then compute a shared key $K_{MAC} = N_{C1}||N_{R1}$, where $||$ denotes a concatenation. M_1 subsequently applies the MAC algorithm and key K_{MAC} to message μ to produce a MAC value $V_{K_{MAC}}(\mu)$, which will protect message μ, and thus the intra-network transfer certificate against forgery and unauthorized modifications. M_1 combines message μ and $V_{K_{MAC}}(\mu)$ to form the certificate to be sent to C.

M_1 also sends a message $r = \{I_{cert}, I_R, I_C, K_{MAC}\}$ to M_2, which contains the ID of this intra-network transfer certificate, M_1's ID I_R, C's ID I_C and key K_{MAC} to be used with this intra-network transfer certificate. The combination of I_{cert}, I_R and I_C is used to identify the association of a key K_{MAC} with the corresponding intra-network transfer

[1]Such a partial key is used only once and cannot be re-used by the party that created it in the first place. In this article, we call these partial keys *nonces* to simplify the presentation.

certificate.

When client C moves into contact with a foreign MAP (e.g. M_2) to prepare for a handover to the new MAP, C submits the intra-network transfer certificate issued by M_1 to the foreign MAP (e.g. M_2) for authentication (step (3) in Figure 3.2).

In order to allow a foreign MAP (e.g. M_2) to process the intra-network transfer certificate and authenticate C, the home MAP M_1 is required to securely send the key $K_{MAC} = N_{C1}||N_{R1}$ to the foreign MAP (e.g. M_2) in advance. (We describe in Section 3.2.2 how to deliver key K_{MAC} from the home MAP to any of its neighbor in a timely, secure, and efficient manner.)

The foreign MAP (e.g. M_2) will use key K_{MAC} and the MAC algorithm to verify the authenticity and data integrity of the intra-network transfer certificate Θ_C submitted by client C. (M_2 will also verify the identity of C in the handover authentication protocol described in Section 3.2.2, and illustrated by steps (4) and (5) in Figure 3.2.)

It should be noted that each certificate has its own expiration date. The synchronization of certificate updates follows the timing synchronization function of the 802.11s standard [72]. The lifetime of a key K_{MAC} is the same as that of the intra-network transfer certificate associated with it. A foreign MAP in the network can re-issue a new intra-network transfer certificate for the certificate owner if the current intra-network transfer certificate is about to expire.

Readers may note that the formats of the above certificates are similar to that of X.509 certificates. However, our certificates contain extra information that cannot be

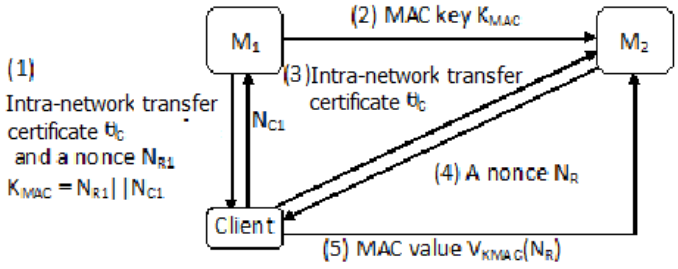


Figure 3.2: Information exchange between a client and MAPs

accommodated by the X.509 format, e.g., client ID in a client certificate, MAP ID in a MAP certificate, and the MAC value in an intra-network transfer certificate.

3.2 The Proposed Authentication Protocols

Built upon the above proposed trust model and certificates, we propose two authentication protocols, one for the initial login into a network and the other for subsequent roaming (handovers). Our authentication protocols follow a key hierarchical structure similar to that of IEEE 802.11i [35]. That is, a pairwise master key (PMK) is created during the authentication process, and a pairwise transient key (PTK) and a group transient key (GTK) are derived from the PMK subsequently. The two parties involved in the authentication will used the PTK for point-to-point communications and the GTK for group communications (broadcast, multicast) between them.

Public key operations are computationally intensive. Mobile devices, on the other hand, have limited computing capability and power resources. Therefore, our design of the proposed authentication protocols aim to minimize:

- the number of message exchanges between a mobile client and MAPs, or the authentication server; thus minimizing the authentication latency and resource consumption by the mobile device.

- the number of public key operations performed by mobile devices; thus minimizing resource consumption by mobile devices.

In addition, we aim to minimize the number of multi-hop communications, thus minimizing the authentication latency and traffic in the backhaul network. At the same time, we ensure that the protocols are secure and scalable. Note that MAPs are not computationally constrained and typically have constant power supplies; thus we are not concerned about them regarding the computation power required for performing public key operations.

3.2.1 The Login Authentication Protocol (LAP)

The trust between a client and a MAP is established via the client certificate and the MAP certificate. Since an agent is a *trusted* authority, a client certificate (or a MAP certificate) issued in advance by the agent is the proof of the authentication between the agent and the corresponding client (or MAP).

Following are the order of the messages to be exchanged in the protocol:

(1) $C \longrightarrow R$: I_C

(2) $R \longrightarrow C$: T_R

(3) $C \longrightarrow R$: $T_C, E_{P_R}(N_{C1}, N_{C2}, N_{C3})$

(4) $R \longrightarrow C$: $E_{P_C}(N_{R1}, N_{R2}, N_{R3})$

(5) $C \longrightarrow R$: N_{R2}

(6) $R \longrightarrow C$: N_{C2}, Θ_C

(1) A client C requests to join a network and associate with a MAP. C sends a request message containing its ID to MAP R.

(2) A MAP R replies with a message which contains its MAP certificate to inform mesh clients and neighboring MAPs of its presence and ID. Client C verifies the digital signature of the certificate agent A who issued the MAP certificate T_R using A's public key. (We assume that client C and MAP R have the public key certificate of the certificate agent.) C also verifies other information in the MAP certificate such as the ID of the certificate agent and the certificate expiry date.

(3) If the above verifications are successful, C extracts the MAP's public key from the MAP certificate T_R (see Section 3.1.2.2) and generates three nonces N_{C1}, N_{C2} and N_{C3}. C then encrypts the nonces N_{C1}, N_{C2} and N_{C3} using MAP R's public key P_R, and sends the encrypted message $E_{P_R}(N_{R1}, N_{R2}, N_{R3})$ and C's client certificate T_C to MAP R. Upon receiving the message, R decrypts N_{C1}, N_{C2} and N_{C3} using its private key, and verifies the digital signature of the certificate agent who issued the client certificate T_C (using the certificate agent's public key). R then verifies other

information recorded in the client certificate T_C such as the ID of the certificate agent who issued T_C and the certificate expiry date.

(4) If the above verifications succeed, MAP R retrieves the client's public key from certificate T_C (see Section 3.1.2.1), and generates a message containing three nonces N_{R1}, N_{R2} and N_{R3}. R then encrypts three nonces N_{R1}, N_{R2} and N_{R3} using the client's public key P_C, and sends the encrypted message $E_{P_C}(N_{R1}, N_{R2}, N_{R3})$ to client C. C will decrypt the message using its private key to get N_{R1}, N_{R2} and N_{R3}. Both the client and the MAP then calculate their shared MAC key $K_{MAC} = N_{C1}||N_{R1}$, where the operator $||$ denotes a concatenation, and N_{C1} and N_{R1} are the nonces generated in steps (3) and (4). (The security of nonces N_{C1} and N_{R1}, and thus inclusively key K_{MAC}, is ensured by the MAP's and client's public-private keys.)

(5) Client C then sends N_{R2} to the MAP R. Upon receiving this message, MAP R has successfully authenticated the client C, because only C has the knowledge of N_{R2}.

(6) To allow the client to authenticate the MAP, R sends N_{C2} (generated by C in step (2)) to client C. The MAP also creates an intra-network transfer certificate Θ_C for C, and subsequently sends a message containing both N_{C2} and the intra-network transfer certificate to C. After client C receives N_{C2} correctly, it is considered to have successfully authenticated the MAP because only R has the knowledge of N_{C2}. C will use the intra-network transfer certificate Θ_C to roam from one MAP

to another in the network.

Following are additional discussions of the above protocol.

(a) Although other clients could see and may attempt to use the intra-network transfer certificate, only the rightful owner of the certificate will be able to use it to pass the handover authentication procedure. The certificate has to be used in conjunction with the key K_{MAC}, which only the client owning the intra-network transfer certificate knows (see Section 3.2.2).

(b) We recommend SHA-2 hash functions for use in the hash-based MAC algorithm because they are employed in several widely-used security applications and protocols. SHA-2 is considered collision resistant [73].

If the size of the MAC output is L bits, the size k of the MAC key K_{MAC} should be longer than $L/2$ bits. Key sizes of less than $L/2$ bits would decrease the security strength of the function. Keys longer than L bits are acceptable but the extra length would not significantly increase the function strength [74]. Therefore, we recommend a key size of 160 bits, the same size as that of the SHA-2 outputs. As a result, the size of the nonces N_{C1} and N_{R1} (and of the other nonces) is 80 bits.

(c) Key management between a MAP and a client allows the MAP and the client to derive a shared key to be used after the authentication for secure data exchanges. We follow the framework of key management defined in IEEE 802.11i security standards [35]. That is, right after step (4) of the authentication procedure, both parties compute

a shared pairwise master key as follows:

$$PMK = N_{C3}||N_{R3} \tag{3.1}$$

After the login authentication is completed, the two parties use the pairwise master key PMK to compute a shared key called pairwise transient key (PTK) as specified by the IEEE 802.11i security standards (see Section 3.2.3). The PTK will be used to encrypt packets exchanged between the client and the MAP. The generation and computation of the PTK is discussed in Section 3.2.3.

3.2.2 The Handover Authentication Protocol (HAP)

To support fast handovers for clients roaming from one MAP to another, we propose a method of key pre-distribution among neighboring MAPs. After a home MAP M_1 successfully authenticates a client C through the login authentication protocol, it generates a message containing the ID of the intra-network transfer certificate, M_1's ID, C's ID, key K_{MAC} associated with C's intra-network transfer certificate. The MAP then encrypts the message using the public key P_x of a neighboring MAP M_x, and sends the encrypted message to M_x. (We assume that each MAP has the public key certificates of its neighboring MAPs.) The neighbor MAP M_x decrypts the message using its private key to extract key K_{MAC} to prepare for future authentications of client C. The above public key operations are performed by MAPs, which are not constrained in terms of computing capability or power supply.

Since the client may move in any direction, the home MAP should send key K_{MAC} to all of its neighbors in anticipation of client Cs mobility. The home MAP can combine several encrypted messages (each containing the ID of the intra-network transfer certificate, client ID, client's public key and K_{MAC}) into one packet and transmit the packet to all neighbors using a broadcast in order to save bandwidth. After a neighboring MAP M_2 receives keys K_{MAC} and a request for connection from client C, it executes the following handover authentication protocol (presented in the order of the messages exchanged):

(1) $C \longrightarrow M_2$: $\Theta_C, N_C, V_{K_{MAC}}(N_C)$
(2) $M_2 \longrightarrow C$: $N_R, V_{K_{MAC}}(N_C, N_R)$
(3) $C \longrightarrow M_2$: $N_R, V_{K_{MAC}}(N_R)$

(1) Client C submits its intra-network transfer certificate Θ_C to the foreign MAP M_2, along with a nonce N_C, and a message authentication code $V_{K_{MAC}}(N_C)$. The message authentication code is the result of applying the MAC algorithm and secret key K_{MAC} to nonce N_C.

When M_2 receives this message, it first verifies the correctness of $V_{K_{MAC}}(N_C)$ using the MAC key K_{MAC} it received from the home MAP M_1. If the computed MAC value matches $V_{K_{MAC}}(N_C)$, M_2 can confirm that message (1) of HAP is valid. Next, M_2 verifies the validity of the intra-network transfer certificate. It checks the content of the intra-network transfer certificate, especially the ID of the client's certificate agent and the certificate expiry date. It then applies the MAC algorithm and the secret key K_{MAC} received from M_1 to message μ to output a message authentica-

tion code $V'_{K_{MAC}}(\mu)$. (Recall from Section 3.1.2.3 that an intra-network transfer certificate consists of two parts: the relevant information stored in a message μ and a message authentication code $V_{K_{MAC}}(\mu)$, which is the result of applying a MAC algorithm and a MAC key to message μ.) If $V'_{K_{MAC}}(\mu) = V_{K_{MAC}}(\mu)$, M_2 can confirm that the intra-network transfer certificate is valid (i.e., C was successfully authenticated by its home MAP).

Note that an attacker may capture the intra-network transfer certificate and attempt to use it, but will not pass the MAP's authentication, because the attacker cannot produce a valid pair $(N_C, V_{K_{MAC}}(N_C))$ without the knowledge of key K_{MAC}. Furthermore, the pair $(N_C, V_{K_{MAC}}(N_C))$ enables the protocol to resist denial-of-service attacks (see Section 3.3.6).

(2) M_2 generates a nonce N_R, and computes a message authentication code $V_{K_{MAC}}(N_C, N_R)$, which are sent to client C. When C receives this message, it computes a MAC value $V'_{K_{MAC}}(N_C, N_R)$, using nonces N_C and N_R. If $V'_{K_{MAC}}(N_C, N_R) = V_{K_{MAC}}(N_C, N_R)$, the client has successfully authenticated the foreign MAP M_2. Nonce N_C serves as a challenge which C presents to M_2. The inclusion of N_C in the MAC computation is the response of M_2 to the challenge. (We also include nonce N_R in the MAC computation so that the recipient of the message can detect unauthorized changes to the nonce.)

(3) Client C then executes the MAC algorithm using the MAC key K_{MAC} it computed in step (4) of the login authentication (Section 3.2.1), and the nonce N_R as input.

The result is a message authentication code $V_{K_{MAC}}(N_R)$, which C will send to M_2 along with N_R, which was the challenge from M_2. Upon receiving $N_R, V_{K_{MAC}}(N_R)$, M_2 repeats the same MAC calculation on N_R. If it obtains the same message authentication code as $V_{K_{MAC}}(N_R)$, then this proves C's identity since C is the only client who has the knowledge of the key K_{MAC}.

Following are additional implementation issues and discussions.

(a) If the foreign MAP M_2 receives the intra-network transfer certificate Θ_C before the message $r = \{I_{cert}, I_R, I_C, K_{MAC}\}$ from the home MAP (Section 3.1.2.3), M_2 will not be able to verify the validity of the intra-network transfer certificate because it does not have the MAC key K_{MAC} in order to apply the MAC algorithm to the certificate. In that case, M_2 sends back an error message to C and C will initiate a login authentication instead of a handover authentication.

M_2 will issue a new intra-network transfer certificate to C through the login authentication protocol. Client C will use the new intra-network transfer certificate issued by M_2 for its subsequence roaming instead of using C's previous intra-network transfer certificate issued by M_1. If MAP M_3 is a neighboring MAP of both M_1 and M_2, M_3 may receive two messages $r_1 = \{I_{cert_{M_1}}, I_{M_1}, I_C, K_{MAC_1}\}$ and $r_2 = \{I_{cert_{M_2}}, I_{M_2}, I_C, K_{MAC_2}\}$ after C performs the login authentication protocol with M_1 and M_2, respectively. When C moves from MAP M_2 to the next MAP M_3, C submits the new intra-network transfer certificate generated by M_2 to M_3. The combination of $I_{cert_{M_2}}$, I_{M_1} and I_C in the message r_2 indicates the

association of key K_{MAC_2} with the corresponding intra-network transfer certificate Θ_{2_C}. M_3 will use the key K_{MAC_2} and Θ_{2_C} to authenticate C through the handover authentication protocol.

In this worst-case scenario, the handover authentication reverts back to the current practice in WMNs, i.e., repeating the login authentication with the foreign MAP. However, with low to moderate mobility speeds, we expect that this worst-case scenario will not happen often, and the handover authentication protocol will be used in most cases.

(b) After M_2 receives message $r = \{I_{cert}, I_R, I_C, K_{MAC}\}$ from the home MAP, it also propagates this message to its neighbors to prepare for client C's future move to another MAP, say M_3. M_3 will use message r and the intra-network transfer certificate submitted by C to authenticate C as described above.

(c) The MAC key K_{MAC} has to be updated periodically to maintain its security. When it is updated, the intra-network transfer certificate associated with it has to be renewed as well. The MAP R currently serving the client (either a foreign MAP or its home MAP) is responsible for generating a new intra-network transfer certificate and a new MAC key. The MAP then encrypts them using the shared key PTK and sends the encrypted message to the client.

(d) After a successful handover authentication, the foreign MAP will generate a new pairwise master key PMK, and encrypt the PMK using client's public key P_C.

The foreign MAP then sends the encrypted PMK to the client. The client will use his private key to decrypt the message to obtain PMK. The foreign MAP and the client will use the PMK to compute a shared key (pairwise transient key PTK) for their subsequent secure communications, which will be discussed in the next subsection.

(e) The handover processing delay should be as short as possible to ensure quality of service for real-time applications, such as voice over IP [111].

The International Telecommunication Union defines network delay for voice applications in Recommendation G.114 [95]. This recommendation specifies the standards for one-way delay limits as shown in Table 3.2.

Table 3.2: Standards for one-way delay limits

Range in milliseconds	Description
0-150	acceptable for most user applications
150-400	acceptable provided that administrators are aware of the transmission time and the impact it has on the transmission quality of user application
above 400	unacceptable for general network planning purpose

The handover authentication protocol (HAP) does not use digital signatures (public key cryptography), but rather a MAC algorithm (symmetric key cryptography), to

minimize authentication latency during the handover process.

The computation cost of a MAC operation is 0.015ms [74]. Three MAC values are generated in the HAP. To verify them, a receiver also needs to perform three MAC operations. Thus, there is a total of six MAC operations executed in the HAP, which requires 0.09ms for the computation cost.

The computation costs for generating and verifying a digital signature are 11.6ms and 17.2ms [100], respectively. If using digital signatures in the HAP instead of the MACs, three digital signature generations and three digital signature verifications are required. The total computation cost of all digital signature operations in the HAP will be 86.4ms. Thus, the computation cost of the MAC operations used in the HAP is only about 1% of that of digital signatures.

Given the acceptable delay limit of 150ms specified by International Telecommunication Union (see Table 3.2), if digital signatures are used in the HAP, the computation cost of the digital signatures would consume 57.6% of the total delay in addition to other types of delay such as transmission, packet processing and queuing delay. The total delay would be unacceptable for handover authentication of real-time applications. On the other hand, the computation cost of the MAC operations used in the proposed HAP results in only 0.06% of the total delay. Therefore, we use a MAC algorithm in the HAP instead of digital signatures.

(f) Fast handover authentication requires the deployment of certificate agents, and generation and maintenance of certificates. The offline deployment of certificate agents

compensates for the elimination of an authentication server during the handover authentication process. (The HAP does not involve an authentication server.) Fast handover authentication also requires MAPs to execute more complex operations of the handover authentication protocol in order to directly authenticate mobile clients.

3.2.3 Key Generation

We briefly describe the procedure for generating PTKs after a successful authentication between a client C and a MAP R. The PTK generation procedure follows the four-way handshake protocol defined in IEEE 802.11i [35], as follows.

(1) $R \longrightarrow C$: M_R, N_R, T_1
(2) $C \longrightarrow R$: $M_C, N_C, T_2, V_{PTK}(M_C, N_C, T_2)$
(3) $R \longrightarrow C$: $M_R, N_R, T_3, V_{PTK}(M_R, N_R, T_3)$
(4) $C \longrightarrow R$: $M_C, T_4, V_{PTK}(M_C, T_4)$

In the above procedure,

- M_C and M_R denote the physical addresses of C and R, respectively.
- N_C and N_R are nonces generated by C and R, respectively.
- T_1, T_2, T_3 and T_4 indicate the message types.

The four-way handshake protocol starts with MAP R generating a nonce N_R and sending it to the client C. Client C receives message (1) of the 4-way handshake protocol,

generates a nonce N_C, and computes a PTK using the PMK it shares with MAP R as follows where the operator $||$ denotes a concatenation.

$$PTK = f(PMK, \min(M_C, M_R) \parallel \max(M_C, M_R) \parallel \min(N_R, N_C) \parallel \max(N_R, N_C)) \tag{3.2}$$

C then sends a message to R that contains nonce N_C and a message authentication code (MAC) $V_{PTK}(N_C, N_C, MT_2)$. The MAC serves as proof of C's possession of the PMK, because the PTK is the key for generating the MAC and the PTK is computed using the PMK.

Upon receiving message (2), MAP R computes the PTK using Eq. (3.2), and uses the PTK to verify the MAC sent by C. If the verification is successful, R generates a message authentication code $V_{PTK}(M_R, N_R, MT_3)$ and sends it to C in message (3) so that C can verify R's possession of the PMK. Message (3) of the 4-way handshake protocol also includes a group transient key (GTK) for multicast applications. We omit the GTK in the message because it is unrelated to the PTK generation procedure.

After C successfully verifies the MAC sent by R, it sends a confirmation to R, which is message (4) shown above.

The PTK is updated periodically using the above four-way handshake protocol. The PMK is also updated periodically (but at a much less frequent rate than the PTK) by the login authentication protocol presented in Section 3.2.1.

3.3 Security Analysis of the Proposed Authentication Protocols

In this section, we identify the security threats [33] relevant to our proposed protocols and discuss counter-measures against them.

3.3.1 Overview

The proposed protocols are protected against various security threats, thanks to the following security features:

- Digital signatures of certificate agents in client and MAP certificates: to prevent forgery of and unauthorized modifications to these certificates.
- Public-key cryptography: to protect messages (3) and (4) of the login authentication protocol (Section 3.2.1).
- Nonces (used-only-once partial keys): to combat replay attacks and denial-of-service attacks, as will be discussed shortly.
- MAC algorithm and MAC keys: to enable a receiver to verify that a message or an information unit (e.g., a nonce) in a message has not been altered in an unauthorized manner. They also provide assurances that a message has been originated by an entity in possession of the MAC key.

The following rules apply to both login and handover authentication protocols:

(R1) A new message with nonces intended for a specific recipient must use newly generated nonces and not those previously sent to the recipient. If a message with nonces was lost or damaged and the message is retransmitted, the retransmitted message must use newly generated nonces.

(R2) Each message is associated with a timer. If the timer expires before the sender receives a response from the intended recipient of the message, the sender assumes that the message has been lost or damaged.

(R3) If the authentication procedure fails after a pre-determined number of tries, the MAP will give up and send the diagnostic information to the network administrator, which will initiate an investigation to determine the cause of the failure.

In addition, a client and a MAP involved in a login authentication session are required to follow the following rule:

(R4) If any of the messages (3) to (6) of the login authentication protocol (Section 3.2.1) is lost, the login authentication protocol will restart from message (3).

Similarly, the following rules are required by the handover authentication protocol:

(R5) When a receiver receives a message with a nonce and a corresponding MAC value, it performs the MAC computation. If the resulting MAC value does not match the MAC value in the message, the receiver assumes that this is a message from an attacker.

(R6) If any message of the handover authentication protocol is lost, the protocol will restart from message (1).

Note that message losses and retransmissions discussed in this chapter are meant to be associated with the transport layer. (Loss detections and retransmissions may be done at the data link layer [e.g., by the RTS/CTS/DATA/ACK exchange of the IEEE 802.11 medium access control protocol], but are transparent to the authentication protocols and do not follow the above rules.)

In the following sub-sections, we describe the countermeasures implemented in the proposed authentication protocols against the attacks listed in [33] that are relevant to our protocols.

3.3.2 Identity Privacy Attack

Most people would like to remain anonymous while roaming in different parts of a network for privacy reasons. To protect clients' privacy, client IDs in certificates are numbers or strings that are not related to the clients' real identities, much like bank account numbers or social security numbers. Only the certificate agents know the mapping between clients' real identities and client IDs recorded in the certificates they issue.

3.3.3 Forgery Attack

A certificate agent's digital signature ensures that the client certificates it issues are protected against modifications and that counterfeit certificates are infeasible to fabricate.

The integrity of an intra-network transfer certificate $\Theta_C = \{\mu, V_{K_{MAC}}(\mu)\}$ is ensured by the accompanying MAC value $V_{K_{MAC}}(\mu)$. Any unauthorized changes to the content of an intra-network transfer certificate will result in an incorrect MAC value because the attacker does not know the MAC key shared between the client and its home MAP. Similarly, a counterfeit intra-network transfer certificate will not be paired with a correct MAC value due to the counterfeiter's lack of knowledge of the MAC key.

3.3.4 Time-memory Trade-off Attack

The simplest form of attack against hash-based MAC algorithms is to use brute force to uncover the secret key. An attacker would use a given input and the corresponding MAC output value (e.g., N_C and $V_{K_{MAC}}(N_C)$ in message (1) of the handover authentication protocol, Section 3.2.2) to figure out the MAC key using brute force. With pre-computation done offline, the time taken in the online stage is shortened at the expense of more memory. This is called a time-memory trade-off attack. To combat this type of attack, we use current state-of-the-art MAC algorithms, SHA-2, in the proposed protocols, and periodically update MAC keys.

3.3.5 Replay Attack

An attacker records messages of an ongoing authentication session and replays these messages in the future in an attempt to be successfully authenticated and possibly gain access to the network as a client. An attacker may replay a client's messages to gain

access to the network, or a MAP's messages in order to impersonate the MAP. We prevent this type of attack by using message encryption, nonces, and the security rules listed in Section 3.3.1.

3.3.5.1 Replaying Client Messages

We consider possible replay attacks on messages generated by the proposed authentication protocols.

In the *login* authentication protocol described in Section 3.2.1, an attacker A overhears and replays a message (3) sent earlier by a client C.

- If the MAP had successfully received the original message (3) from C, it would have saved the nonces N_{C1}, N_{C2} and N_{C3}. When the MAP receives the replayed message, it compares the nonces in the message against the saved nonces N_{C1}, N_{C2} and N_{C3}, and can detect that this is a replayed message because a new message is supposed to have new nonces and not repeated nonces (rules (R1) and (R2) in Section 3.3.1).

- If the MAP did not receive the original message (3) from C, the MAP may accept the replayed message as a valid message (if the timer associated with the sent message (2) has not expired yet) and reply with a message (4). However, the attacker will not be able to decrypt message (4) because he does not know the private key of client C, and thus fails to proceed to step (5) of the login authentication protocol in Section 3.2.1.(Client C will also see message (4) sent by the MAP, assuming that

it has not timed out on the lost message, and proceed to step (5) of the protocol. In this case, the attacker actually helps instead of harming.)

In the *login* authentication protocol, an attacker may replay a message (5) of a client C.

- If the MAP had successfully received the original message (5) from C, it can detect that the current message is a replayed message thanks to the repetition of nonce N_{R2} in the message.
- If the MAP did not receive the original message (5) from C, the MAP may accept the replayed message as a valid message and reply with a message (6). (Again, the attacker helps the client "retransmit" the lost message (5), assuming that the MAP has not timed out due to the lost message from C.) Note that although the attacker will also receive message (6) it will not be able to access network services because that requires the knowledge of the pairwise master key (PMK) described in Section 3.2.1. The attacker does not have that knowledge because it does not possess the necessary private keys to decrypt messages (3) and (4) in order to obtain the nonces needed to compute the PMK.

In the *handover* authentication protocol presented in Section 3.2.2, an attacker captures and replays a message (1) sent earlier by a client C.

- If the MAP had successfully received the original message (1) from C, it saved the nonce N_C. When the MAP receives the replayed message, it compares the nonce

in the message against the saved nonces, and can detect that this is a replayed message because a new message is supposed to have a newly generated nonce.

- If the MAP did not receive the original message (1) from C, it may accept the replayed message as a valid message and reply with a message (2). However, the attacker will not be able to compute the correct MAC value $V_{K_{MAC}}(N_R)$ because it does not know the MAC key K_{MAC}, and thus fails the authentication by the MAP in step (3).

  Note that client C may also receive message (2) from the MAP, if it has not timed out on the lost message, and responds with a message (3). If C does not receive message (2) before a timer expires, it re-sends a new message (1) with a different nonce $N_{C'}$.

Also in the *handover* authentication protocol, an attacker may replay a message (3) of a client C.

- If the MAP had successfully received that message (3) from C earlier, it can detect that this is a replayed message thanks to the repetition of nonce N_R in the message.

- If the MAP did not receive the original message (3) from C, it may accept the replayed message as a valid message. The client is then considered successfully authenticated by the MAP, assuming that the MAP receives the replayed message before it times out on the lost message. Since only the client and the MAP know the shared key K_{MAC}, by computing the MAC value $V'_{K_{MAC}}(N_R)$ using K_{MAC} and

comparing it with the one in the message (3), M_2 can confirm that message (3) is originated from C, not from the attacker. Thus, M_2 has successfully authenticated client C as C is the only client who knows K_{MAC}.

3.3.5.2 Replaying MAP Messages

We examine possible attack scenarios aimed at replaying MAP messages.

In the *login* authentication protocol described in Section 3.2.1, an attacker overhears and replays a message (4) sent earlier by a MAP R.

- If the client had successfully received the message (4) from R, it saved the nonces N_{R1}, N_{R2} and N_{R3}. When the client receives the replayed message, it compares the nonces in the message against the saved nonces, and can detect that this is a replayed message because a new message is supposed to have newly generated nonces (rules (R1) and (R2) in Section 3.3.1).

- If the client did not receive the original message (4) from R, the client may accept the replayed message as a valid message (if the timer on the sent message (3) has not expired yet), and reply with a message (5). However, the attacker will not be able to generate the MAC value $V_{K_{MAC}}(N_{C2})$ because he does not know the MAC key K_{MAC}, and thus fails the authentication by the client in step (6).

  Note that the MAP may also receive the replayed message correctly and proceed to step (6) of the protocol. In this case, the attacker actually helps to "retransmit" the message (4) that the MAP lost in the first place. (If R does not receive the

replayed message (5) , client C will time out on waiting for message (6) from R and restart the authentication procedure.)

Also in the *login* authentication protocol described in Section 3.2.1, an attacker may replay a message (6) sent earlier by a MAP R.

- If client C had successfully received message (6) from R earlier, it can detect that this is a replayed message because it had received the same transfer ticket earlier.
- If client C did not receive the original message (6) from the MAP, C will accept the replayed message and consider the authentication successful (assuming that C receives the replayed messages before it times out on the lost message). However, the attacker will not be able to impersonate the MAP because it does not know the PMK shared by the client and the MAP, which is required for subsequent communications between the client and the MAP.

In the *handover* authentication protocol presented in Section 3.2.2, an attacker overhears and replays a message (2) sent earlier by a MAP R. If the client had successfully received the original message (2) from R earlier, it can detect that this is a replayed message thanks to the repetition of nonce N_R in the message. If the client did not receive the original message (2), it may accept the replayed message as a valid message and reply with a message (3) (before it times out on the lost message). After C receives the replayed message (2), it computes a MAC value $V'_{K_{MAC}}(N_C, N_R)$ using K_{MAC} (a shared key between C and M_2). If $V'_{K_{MAC}}(N_C, N_R) = V_{K_{MAC}}(N_C, N_R)$, C confirms that message (2)

is originated from M_2, not from the attacker. The protocol prevents the attacker from impersonating M_2 if the attacker does not know the key K_{MAC} shared only between C and M_2. Thus, client C has succesfully authenticated M_2, who has the knowledge of the shared key K_{MAC}, not the attacker.

3.3.6 Denial-of-Service (DoS) Attack

An attacker may send bogus messages or replay past valid messages repeatedly to force a MAP to spend resources on processing a large amount of these DoS attack messages. To combat a DoS attack, the proposed authentication protocols rely on the security features and rules stated in Section 3.3.1.

3.3.6.1 Analysis of the Login Authentication Protocol

In the login authentication protocol described in Section 3.2.1, an attacker may repeatedly send copies of message (1) to a MAP. The MAP will interpret the duplicates of this message as the losses of messages (2) it has sent. The MAP will stop the authentication procedure after a pre-determined number of failed attempts, according to rule (R3) stated in Section 3.3.1, to save resources. Note that this type of attack can happen to any protocol, and not just specifically to authentication.

An attacker may sniff valid message (3) and message (5) from a successful login authentication and replay the message repeatedly to the involved MAP in order to overwhelm it. The MAP can detect that this is a replayed attack because a new message (5)

is supposed to have a new nonce. If the MAP receives the replayed message several times, it can infer that it is under a DoS attack and can take appropriate actions to thwart the attack [78, 79, 80, 81, 82, 83, 85, 86, 87].

Note that an attacker may flood a MAP with bogus copies of message (3) that it creates by itself, but those bogus messages will be detected by the MAP because the attacker could not possess a valid client certificate T_C. After processing a number of such bogus messages, the MAP can infer that it is under a DoS attack and take appropriate actions. (If an attacker possesses a valid client certificate, this can be categorized as an insider attack, which is much harder to detect. This requires human interventions, e.g., checking if the mobile device was stolen; verifying the client's background.).

3.3.6.2 Analysis of the Handover Authentication Protocol

All messages of the handover authentication protocol are protected against forgery and unauthorized modifications by the MAC algorithm. An attacker cannot generate a valid message in the handover authentication protocol, without the knowledge of the MAC key, which is shared only with the client, its home MAP, and the foreign MAP (Rule (R5) stated in Section 3.3.1).

On the other hand, an attacker may repeatedly replay message (1) (or message (3)) originated earlier by a client C. The MAP can detect that these are replayed messages because the attacker will not be able to compute the correct MAC value $V_{K_{MAC}}(N_R)$ and thus fails the authentication by the MAP in step (3). If the MAP receives the replayed

message several times, it can conclude that it is under a DoS attack and can take necessary counter-attack measures [78, 79, 80, 81, 82, 83, 85, 86, 87].

3.3.7 Compromised MAPs

An attacker may compromise a MAP by: (1) dropping valid authentication messages to prevent clients from joining the network, or (2) granting access to unauthorized or non-paying users. Following are effective counter-measures against these attacks.

(1) Dropping valid messages deviates from the normal procedure of the authentication protocol, which requires the attacker to modify the authentication code. Software-based attestation techniques such as SWATT [89] and Pioneer [90] can be used to externally verify the contents of the memory of an embedded device (SWATT) or a CPU (Pioneer) in order to detect changes to the original code. An external verifier can detect with high probability if a single byte of the memory deviates from the expected value [89]. These techniques allow a network operator to periodically verify the routers in its network and detect compromised nodes. Note that this attack can happen to any protocol (e.g., routing) and not just authentication. From a client's point of view, the consequence of the attack is similar to that of a router failure: the client times out on the authentication request, and will look for another MAP nearby to join. This type of router placement redundancy should be implemented regardless of security issues: if a MAP fails or malfunctions, nearby MAPs should be able to support the failed MAP's clients.

(2) To grant access to users that do not own valid certificates, the attacker would need to modify the authentication code. One countermeasure is to use attestation techniques such as SWATT and Pioneer to detect changes in the authentication code, as discussed above. An alternative we propose is to use a dual authentication process. The authentications described in Section 3.2, if successful, give the client only short-term access to network services. The client will subsequently be authenticated by an authentication server (via multi-hop communications), while enjoying network services using the short-term access permission. After the server successfully authenticates the client, it will issue to the client a service certificate [8] that serves as a pass for the client to access network services on a long-term basis. An illegitimate or non-paying user will not be issued such a service certificate, and will not be able to continue to use network services after the short-term access privilege expires. The dual authentication process allows for both fast authentication during the handover process, and for stronger security provisions by an authentication server.

3.4 Performance Evaluation

We compare the performance of our proposed authentication protocols with existing protocols using both numerical analysis and simulations. The protocols to be compared include EAP-TLS and the algorithm proposed by Kassab et al. [34]. A detailed description of Kassab's algorithm is given in Appendix A. EAP-TLS is the most commonly

used authentication protocol for IEEE 802.11-based wireless networks and represents the multi-hop handover authentication approach. Kassab's [34] and Li's [42] algorithms are representatives of the certificate-based approach and are the closest to ours. Kassab's and Li's algorithms work in a similar manner. The major difference between them is that the authentication server (AS) in Kassab's distributes PMKs to the MAPs neighboring to the home MAP, while the AS in Li's distributes certificates. The size of a certificate is bigger than that of a PMK. Thus the traffic overhead incurred by Li's algorithm is higher than that by Kassab's. Therefore we chose to compare our handover authentication protocol (HAP) with the more efficient algorithm, Kassab's.

3.4.1 Numerical Analysis

The numerical analysis demonstrates the theoretical gain of our proposed protocols over EAP-TLS and Kassab's scheme. The performance of the protocols is measured in terms of

- *communication costs*, which indicate the number of messages exchanged between a MAP and a client to complete an authentication session.

- *computation costs*, which are the latencies (in milliseconds) incurred by the following security operations: encryption using public key (E_{pub}); decryption using public key (D_{pub}); encryption using shared key (E_K); decryption using shared key (D_K); generation of a digital signature (G_{sig}); verification of a digital signature (V_{sig}); computation/verification of a message authentication code (MAC); and hashing.

Table 3.3 lists the above operations, the current state-of-the-art algorithms implementing the operations, and the computation time each of these algorithms incurs [91] (the first, second, and third columns, respectively). The fourth, fifth, and sixth columns of Table 3.3 lists the numbers of security operations the proposed login and handover authentication protocols, Kassab's scheme and EAP-TLS perform, respectively. By multiplying the computation cost of each operation (from the third column) and the number of times it is executed, and summing up the costs of all operations executed by a protocol, we obtain its total computation cost as shown in the third to last row of Table 3.3. The computation cost of the login authentication protocol (97.935 ms) is slightly less than that of EAP-TLS (97.962ms). But more importantly, the computation cost of the handover authentication protocol (0.105 ms) is 2.45% of the Kasssab's scheme (4.3 ms) and is three orders of magnitude lower than that of the login authentication and EAP-TLS protocols.

The second to last row of Table 3.3 lists the number of messages exchanged in each protocol. The authentication latencies shown in the last row are the sums of computation costs and communication delays, where d is the average delay of a one-hop transmission incurred by a message, and h is the number of hops between the client and the home authentication server. (Parameter h is applicable to only EAP-TLS as our handover protocol and Kassab's handover scheme does not require a client to communicate with the home MAP during the handover process.) The average delay of a one-hop transmission d includes the backoff time, RTS/CTS/DATA/ACK exchange, DIFS and SIFS values,

transmission time, propagation time, and processing time as shown in Figure 3.3. The results show that the larger the number of hops between a client's home MAP and a foreign MAP, the lower the authentication latency our protocols incur compared with EAP-TLS.

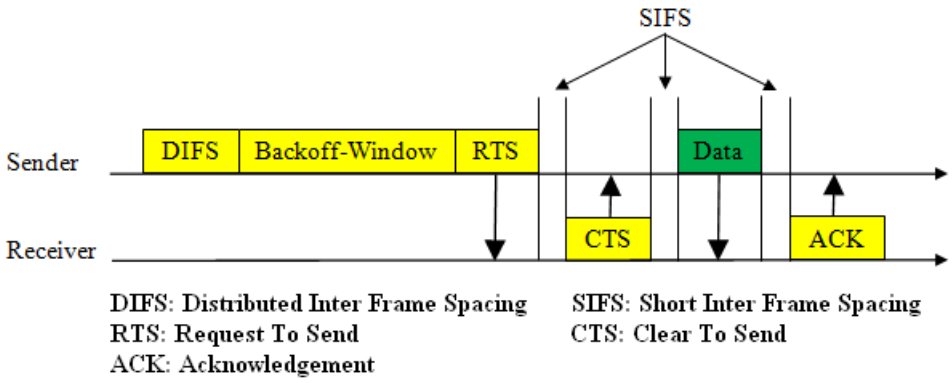


Figure 3.3: Delay incurred by a one-hop transmission

In particular, the gain of the login authentication protocol over EAP-TLS is due to

- a reduction in the number of messages exchanged, six vs. nine;
- one-hop communication between the client and the MAP vs. multi-hop communication between the client and the authentication server (captured by parameter h).

The gain of the handover authentication protocol over EAP-TLS is also due to the above two reasons, plus the elimination of public key operations during the handover authentication. The gain of the HAP over Kassab's protocol results from less cryptographic operations, and one less message, three vs. four.

Table 3.3: Computation and communication costs of authentication protocols

Op.	Algorithm	Time (ms)	Login see 3.2.1	Handover see 3.2.2	EAP-TLS	Kassab's
E_{pub}	RSA [99]	1.42	1	0	1	0
D_{pub}	RSA	33.3	1	0	1	0
G_{sig}	ECDSA [100]	11.6	1	0	1	0
V_{sig}	ECDSA	17.2	3	0	3	0
E_K	AES [101]	2.1	0	0	0	1
D_K	AES	2.2	0	0	0	1
MAC	HMAC [74]	0.015	1	7	0	2
Hash	SHA-2 [73]	0.009	0	0	3	0
Total computation cost (ms)			97.935	0.105	97.962	4.3
Number of messages			6	3	9	4
Authentication latency (ms)			97.935+6d	0.105+3d	97.962+9dh	4.3+4d

Table 3.4: Common simulation parameters

Parameter	Value
Movement model	Random way point
Speed	0-30m/s
Propagation fading model	none
Transmission range of MAPs	315m
Transmission range of mesh clients	304m
Transmission rate at physical layer	2 Mbits/s
Physical layer protocol	PHY802.11b
Number of runs per data point	10
Confidence interval	95%

Table 3.5: Simulation parameters for different experiments

Experiment	Figure	Network	Clients, Mobility Speed
1.	Figure 3.6(a), LAP	400m x 400m, one MAP	20-60 nodes, 0-30m/s
2.	Figure 3.6(b), LAP vs. EAP-TLS		10-60 nodes, 20m/s
3.	Figure 3.6(c), HAP	600m x 600m, four MAPs	20-60 nodes, 5-30m/s
4.	Figure 3.6(d), HAP		10-60 nodes, 10-20m/s
5 and 6.	Figure 3.6(e) - 3.6(h), HAP vs. Kassab, and EAP-TLS	600m x 600m, five MAPs, each MAP is six hops away from the AS	10-60 nodes, 20m/s
7 and 8.	Figure 3.6(i) and 3.6(j), HAP vs. Kassab		10-60 nodes, 20m/s

3.4.2 Simulation Results

We use QualNet (version 5.2), a commercial software that provides scalable simulations of wireless networks [102], for our experiments.

3.4.2.1 Performance Metrics

One performance metric is *authentication delay* (latency), which is measured as the time between a client's transmission of an authentication request to a nearby MAP and the receipt of an acceptance confirmation. After a client sends an authentication request, it sets a timer. If it does not receive a confirmation by the time the timer expires, it will re-send the request. The authentication delay is measured starting with the first request. In all experiments, we calculate the *average authentication delay* (AAD), averaged over all mobile clients participating in the experiment. In several cases, we also keep track of the *maximum authentication delay* (MAD), the maximum value among all mobile clients.

In the proposed HAP, after a successful login authentication, the home MAP will send a MAC key it shares with the client to the neighboring MAPs to prepare for a handover in the near future. (This is a one-hop communication, from the home MAP to the neighboring MAPs in one broadcast message.) In Kassab's protocol, after a successful login authentication, the AS sends to every neighbor N of the home MAP a PMK to be shared by N and the client when the client roams and needs to be authenticated by N. (These are multi-hop communications, from the AS to each neighboring MAP.) This pre-distribution of keys/certificates incurs some delay before the next handover. We call this

delay *key pre-distribution delay* (KPDD), which should be minimized to avoid service interruption when clients move from one MAP to another. We compare the proposed HAP with Kassab's protocol in terms of key pre-distribution delay.

3.4.2.2 Simulation Parameters

The common simulation parameters for all experiments are listed in Table 3.4. The transmission range of the wireless routers (MAPs) is 315 m, according to the specifications of wireless routers manufactured by Tropos [103]. The transmission range of mesh clients is 304 m, according to the specifications of wireless adapters manufactured by Cisco [105]. The transmission rate at the physical layer is 2 Mbits/s. Mobility speeds of mobile clients vary from 0 to 30 m/s and the mobility pattern follows the random waypoint model [106]. Each data point in the graphs is the average of 10 runs using different random seeds. Graphs are plotted with a confidence interval of 95%.

We conducted eight sets of experiments:

1. We measured the average authentication latency of the login authentication protocol (LAP) as a function of clients' mobility speed. The 400m x 400m network has one MAP placed in the center of the square. Three scenarios: 20, 40, and 60 clients. In each experiment, all clients have the same mobility speed. The speed is varied from 0m/s to 30m/s.

2. We compared LAP with EAP-TLS and measured both the AAD and MAD. We used the same network as in experiment (a). All clients moved at the same speed

of 20m/s. The number of clients varied from 10 to 60.

3. We measured the AAD of the handover authentication protocol (HAP) as a function of clients' mobility speed. We simulated a network of size 600m x 600m with four MAPs arranged as in Figure 3.4, and three scenarios: 20, 40 and 60 clients in the network, respectively. In each experiment, all clients have the same mobility speed. The speed is varied from 0 m/s to 30 m/s.

4. We measured the AAD and MAD of the HAP as functions of number of clients. We used the same network as in experiment (3). All clients moved at the same speed of 20 m/s. The number of clients varied from 10 to 60.

5. We compared the HAP with EAP-TLS and Kassab's algorithm in terms of the average authentication delay during the handover process. We used the network configuration shown in Figure 3.5. The home MAP H has four neighboring MAPs. The authentication server was located six hops away from each MAP in order to illustrate the high overhead of the multi-hop handover authentication approach used by EAP-TLS. We varied the number of clients from 10 to 60. All clients moved at the same speed of 20m/s.

6. This experiment is the same as experiment (5) above, except that we recorded the maximum authentication delay (MAD) during the handovers.

7. We compare the HAP with Kassab's algorithm in terms of the average key pre-distribution delay (KPDD). The network and simulation parameters are the same

as those in experiment (5) above.

8. This experiment is the same as experiment (7) above, except that we recorded the maximum key pre-distribution delay (KPDD).

The simulation parameters specific to each experiment are summarized in Table 3.5. In all the experiments, the mobile clients were randomly distributed in the networks. To test the scalability of the protocols, we let all clients present in the network send authentication requests to their respective nearby MAPs *simultaneously*.

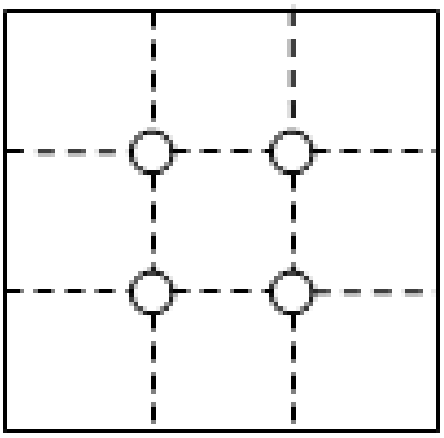


Figure 3.4: Network with four MAPs

3.4.2.3 Result Analysis

The results of the above eight sets of experiments are illustrated by the graphs in Figure 3.6.

1. The graph in Figure 3.6(a) shows the AAD of the LAP as a function of clients'

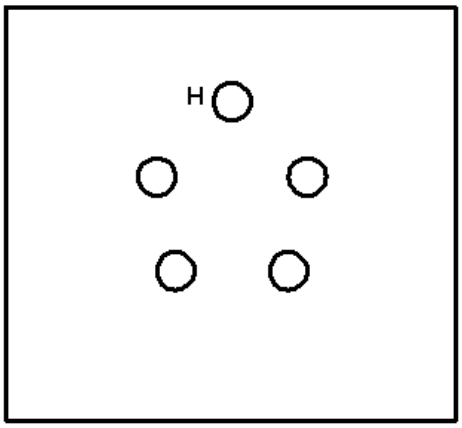
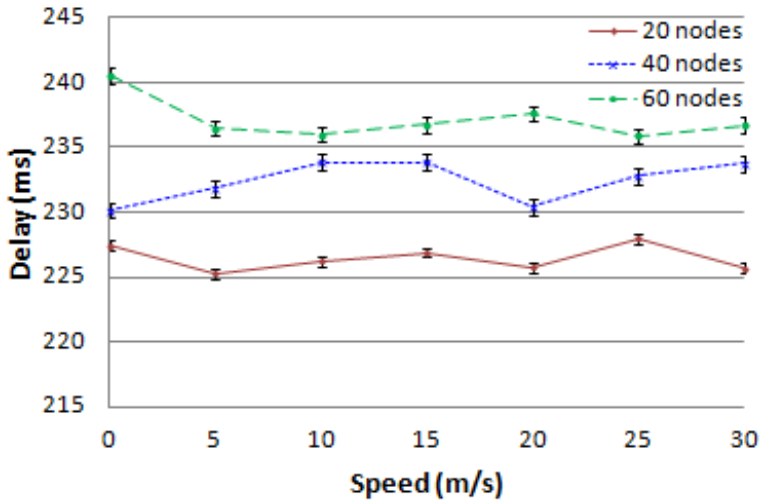


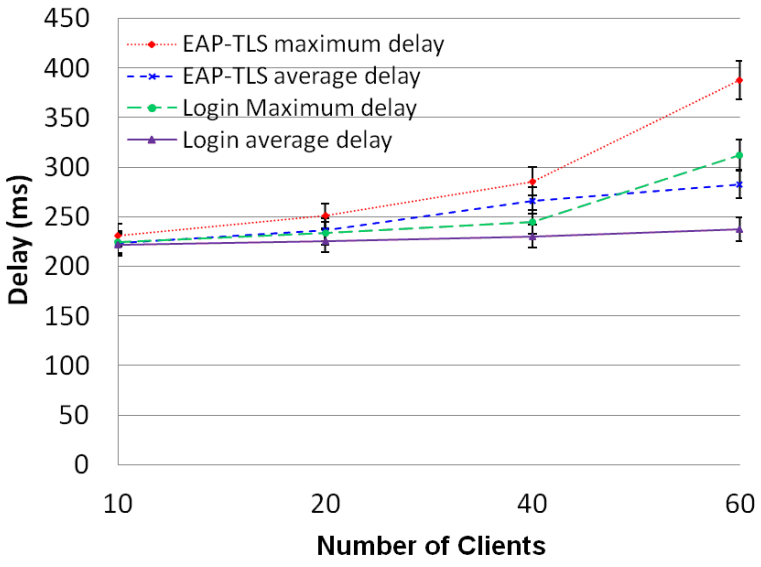
Figure 3.5: Network with five MAPs

mobility speed. There is one MAP placed at the center of the network, serving 10-60 mobile clients. Each client is one hop away from the MAP. We observe that the AAD is not impacted much by the mobility speed, which is a positive attribute of the LAP. On the other hand, as the number of clients increases from 20 to 60, the ADD also increases as expected, by approximately 4% to 6%. More clients imply more authentication requests to be processed by the MAP, and more channel contention around the MAP, resulting in longer delays.

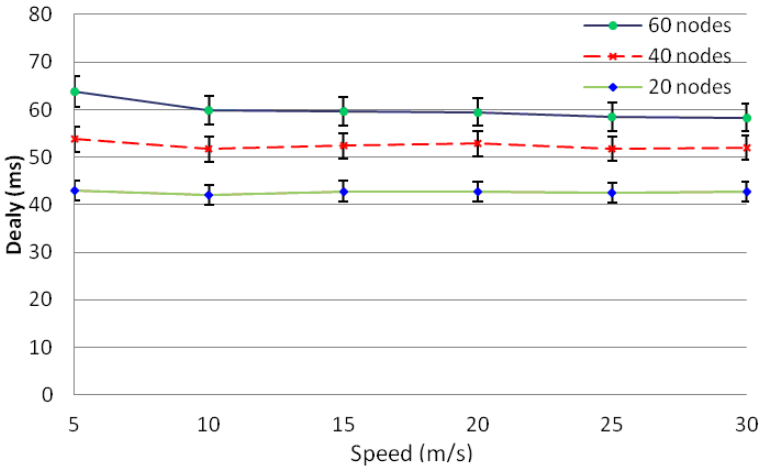
2. Figure 3.6(b) shows the performance of the LAP vs. EAP-TLS under the same network settings as above. When there are only 10 clients in the network, both protocols perform similarly. Given more than 10 clients, the workload and channel contention at the MAP increases. In these cases, the LAP offers lower AAD than EAP-TLS, because the LAP requires less message exchanges than EAP-TLS (6 vs. 9, as shown in the second last row of Table 4.4). In the case of 60 clients, the AAD



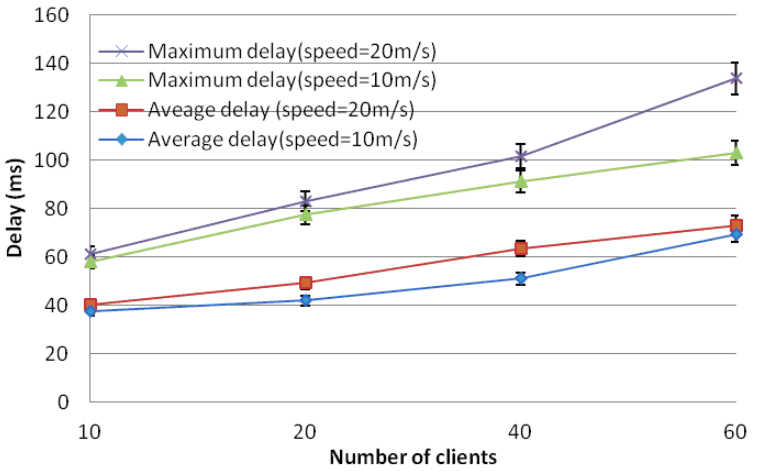
(a) Login protocol (LAP) - Function of mobility speed



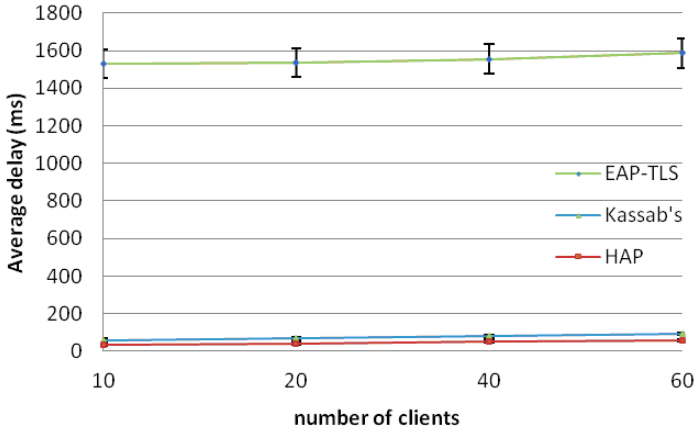
(b) Login protocol (LAP) vs. EAP-TLS - Function of number of clients



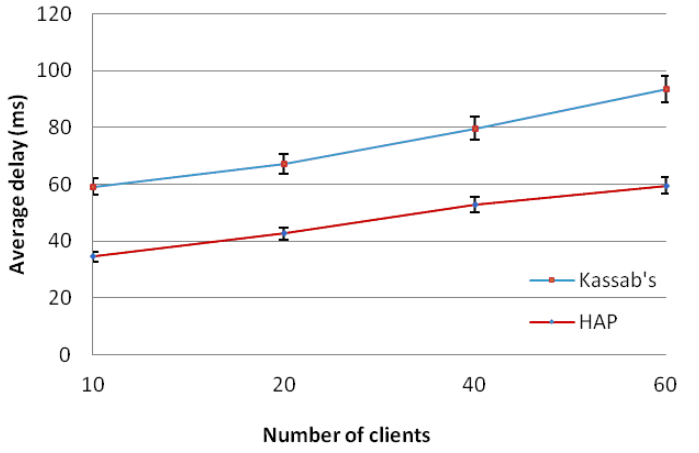
(c) Handover protocol (HAP) - Function of mobility speed



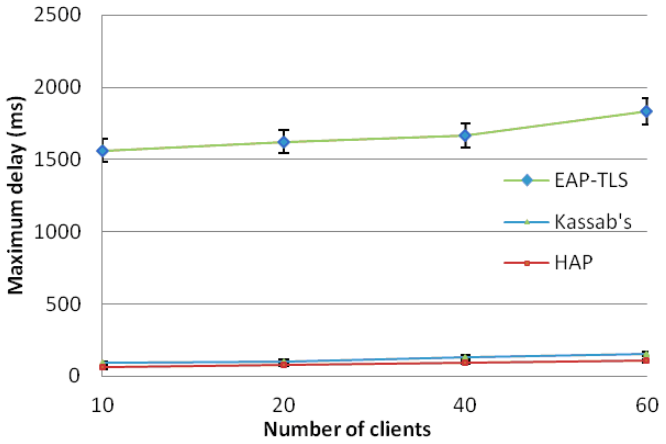
(d) Handover protocol (HAP)- Function of number of clients



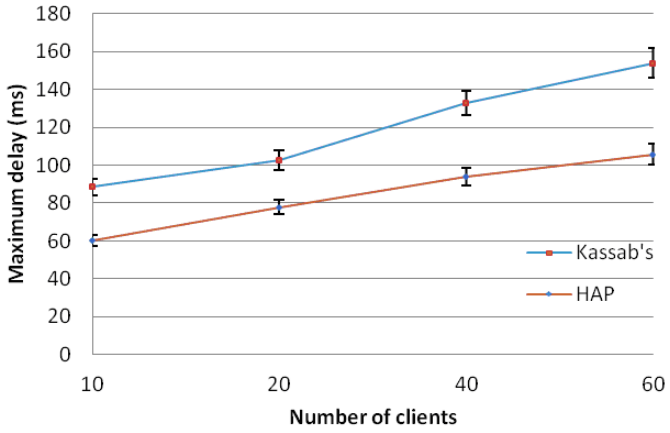
(e) Average authentication delay (AAD) of EAP-TLS, Kassab's protocol and HAP - Function of number of clients



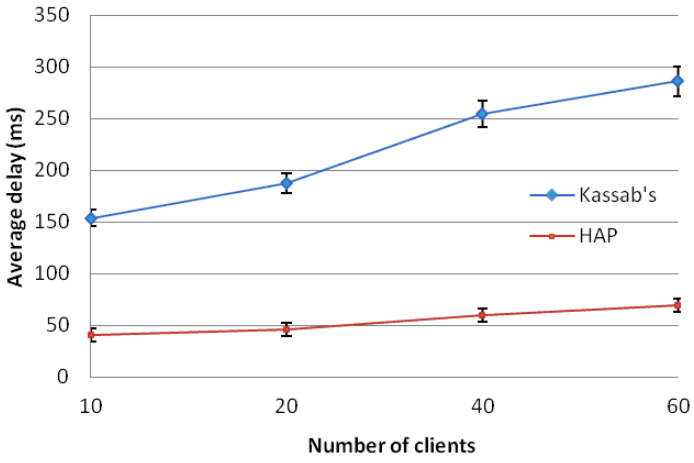
(f) Average authentication delay (AAD) of HAP vs. Kassab's protocol - Function of number of clients (magnification of Fig.(e))



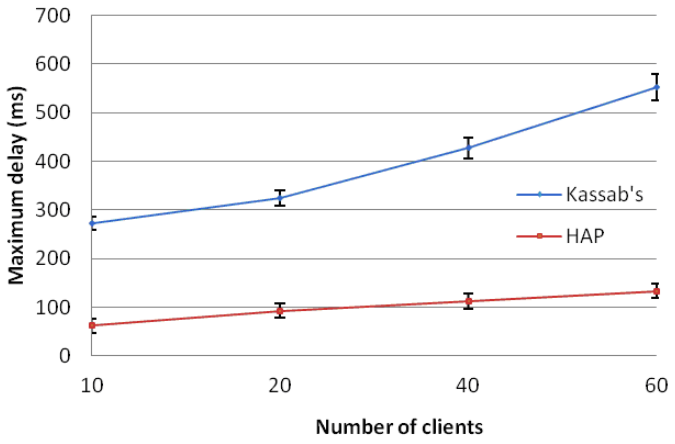
(g) Maximum authentication delay (MAD) of EAP-TLS, Kassab's protocol and HAP - Function of number of clients



(h) Maximum authentication delay (MAD) of HAP vs. Kassab's protocol - Function of number of clients (magnification of Fig.(g))



(i) Average key pre-distribution latency (KPDD) of HAP vs. Kassab's protocol- Function of number of clients



(j) Maximum key pre-distribution latency (KPDD) of HAP vs. Kassab's protocol - Function of number of clients

Figure 3.6: Simulation results

of the LAP is 16% lower than that of EAP-TLS. As the number of nodes increases, the performance gap between the LAP and EAP-TLS enlarges, consistent with the authentication latencies recorded in the last row of Table 3.3 (97.935+6d vs. 97.962+9dh, where $h = 1$). The graph also shows the MAD of both protocols. The MAD of the LAP is about 32% higher than its AAD, which we deem acceptable, and about 20% lower than the MAD of EAP-TLS.

Given 60 mobile clients connecting through the same MAP, the MAD of LAP and EAP-TLS are 321.7 ms and 387.6 ms, respectively, or LAP improves the login authentication delay by 65.9 ms. The amounts of cryptographic computation performed by LAP and EAP-TLS are very similar (97.935 ms vs. 97.062 ms as shown in the last row of Table 3.3). This shows that the gain of LAP over EAP-TLS is mainly due to one-hop communication (between the client and the home MAP in LAP) versus multi-hop communication (between the client and the authentication server (AS) in EAP-TLS), and due to the reduction of the number of messages exchanged from nine (EAP-TLS) to six (LAP).

3. The graph in Figure 3.6(c) shows the AAD of our handover authentication protocol as a function of clients' mobility speed. Four MAPs are uniformly distributed over the network, serving 10-60 mobile nodes. Again, the mobility speed does not have a big impact on the AAD of the HAP, as in the case of the LAP. Also, the more clients send requests, the higher the AAD, as expected. Note very low AADs of the HAP, ranging from 42.96 ms-63.8 ms, compared with the AADs of the one hop

LAP and one hop EAP-TLS which are above 220 ms.

4. The above observations also apply to Figure 3.6(d), which shows the MADs and AADs of our handover authentication protocol as functions of number of clients. In the experiment with 60 nodes moving at a speed of 10 m/s, the MAD of the HAP is 103.2 ms, about 150% of the corresponding AAD, but still very low compared to the authentication delay of EAP-TLS.

5. The graph in Figure 3.6(e) shows the AAD of EAP-TLS, Kassab's protocol and the HAP as functions of the number of clients given the network topology in Figure 3.5. The AS is six hops away from the home MAP. As the number of clients increases from 10 to 60, the AAD of all three schemes increases as expected due to higher loads on the MAPs and more traffic in the network. Both the Kassab's protocol and the HAP outperform EAP-TLS by a large margin in terms of AAD, thanks to one-hop communication (between the client and the foreign MAP) during the handover authentication versus multi-hop communication (between the client and the AS) in EAP-TLS. Moreover, the AAD of the HAP is much lower than that of EAP-TLS due to a reduction in the number of messages exchanged, three vs. nine (see the second last row of Table 3.3).

   We separated the curves of the HAP and Kassab's protocol from Figure 3.6(e) and magnified them in Figure 3.6(f). The new graph shows that our HAP noticeably outperforms Kassab's protocol. For example, when the number of clients is 60, the AADs of HAP and Kassab's scheme are 59.5 ms and 93.3 ms, respectively. HAP

improves the authentication delay by 33.8 ms or by 57% compared to Kassab's scheme, out of which a reduction of 4.3 ms is due to less cryptographic computation. Kassab's algorithm requires three more decryption operations and one more encryption than HAP (see the third last row of Table 3.3). The remaining 29.4 ms (74.26%) authentication delay improvement results from the HAP incurring less message exchanges than Kassab's, three vs. four (see the Appendix A).

6. The above observations and explanations also apply to the graphs in Figure 3.6(g) and Figure 3.6(h), which show the MAD of EAP-TLS, Kassab's protocol, and the HAP as functions of the number of clients. In all cases, the HAP incurs lower MAD than both EAP-TLS and Kassab's protocol.

7. Figure 3.6(i) shows the average key pre-distribution delay of the HAP and Kassab's scheme. As the number of clients increases from 10 to 60, the average KPDD ranges from 273.3 ms to 552.8 ms for Kassab's protocol, and from 61.7 ms to 133.8 ms for the HAP. That is, the average KPDD of HAP is from 55% to 50.3% lower than that of Kassab's scheme. A lower KPDD implies less service interruption, because neighboring MAPs are prepared earlier to connect with a roaming client.

   Given 60 mobile clients trying to join the network via the same MAP, the average KPDDs of HAP and Kassab's scheme are 133.8 ms and 552.8 ms, respectively. The HAP improves the average KPDD by 419 ms compared to Kassab's scheme. The computation cost of the HAP key pre-distribution is n encryptions, where n denotes the number of MAPs adjacent to the home MAP. The computation cost of

Kassab's key pre-distribution is $2n+2$ encryptions and 2 decryptions (see Table 3.3). Given $n = 4$ in this experiment and assuming that the cryptographic operations are performed one after another[2], the computation cost of the HAP is 17 ms less than that of Kassab's. The remaining 200.1 ms (96.17%) out of 419 ms KPDD improvement by the HAP result from the use of intra-network transfer certificates, which eliminate multi-hop communications between the authentication server and the neighboring MAPs, and from the reduction of one additional message exchange ($2n$ messages in the HAP vs. $2n + 1$ messages in Kassab's).

8. The maximum key pre-distribution delays of the HAP and Kassab's scheme are shown in Figure 3.6(j). The above observations and explanations apply to this experiment as well. In short, the HAP offers lower maximum KPDD compared to Kassab's protocol, from 55% to 50.3%. Almost all the gain of the HAP over Kassab's (95%) is the result of the use of intra-network transfer certificates to avoid multi-hop communications between the authentication server and the neighboring MAPs.

Both the performance analysis and simulation results confirm the advantage of the proposed LAP over the EAP-TLS protocol of IEEE 802.11s and the HAP over Kassab's protocol and EAP-TLS. This contributes towards a faster handover process for mobile clients using real-time services in WMNs.

[2]In practice, the neighboring MAPs may perform the cryptographic operations in parallel after receiving the key(s).

3.5 Chapter Summary

The objective of our work is to extend the capabilities of IEEE 802.11s standards to support fast handovers for real-time applications such as VoIP, tele-conferencing, and stock quote distributions. We design a novel trust model and certificate-based authentication protocols to support fast login and handovers in a WMN. A client and a MAP mutually authenticate each other using one-hop communication. The authentication server is not required to participate during the handover authentication process. Fast authentication for roaming from one MAP to another is supported by using intra-network transfer certificates. Our numerical analysis and simulation results confirm that the proposed LAP and HAP outperform the EAP-TLS protocol of IEEE 802.11s and a representative of the certificate-based authentication approach, Kassab's protocol. The proposed protocols are also resilient to various types of attacks.

In this chapter, we studied issues related to *intra-network* handovers. In the next chapter, we present a novel architecture, a trust model, certificates, and a suite of security protocols to support fast inter-network handovers among multiple WMNs.

Chapter 4

Security Protocols for Fast Inter-Network Handovers among Multiple Wireless Mesh Networks

Current research efforts focus mostly on handovers within a single network, and much less attention is devoted to handovers between networks (inter-network handovers). An inter-network handover occurs when a mobile client roams from one network to another, while each network is controlled by a single operator. In this chapter, the terms *different networks, multiple networks, or networks* refer to the networks controlled by different operators. As part of most security policies, a network must authenticate a mobile user roaming from other networks. Similarly, a mobile user has to authenticate the network in order to avoid connecting to an untrustworthy access point. Thus, when roaming from one network to another, a mobile user has to perform a mutual authentication with the

network in order to ensure confidentiality.

Most existing solutions [9, 22, 50, 51, 52, 53] for inter-network handover authentications involve multi-hop communications with a mobile user's home network, and require a pre-established roaming agreement between a home network and a foreign network, which are not acceptable for mobile wireless communications using IEEE 802.11. This may result in long delay, low reliability and thus potential service interruptions. For some applications (e.g. transferring files), this delay is acceptable; however, it is far too long for real-time traffic such as VoIP or video conferencing.

Our work in this chapter contributes towards minimizing authentication latency for mobile clients' handovers between different WMNs Our proposed approach does not require multi-hop wireless communications between a home network and a foreign network. We propose a new network architecture, a trust model and certificates to support inter-network handovers through one-hop communication between a mobile client and a network. Built upon the proposed architecture, trust model and certificates, we propose a suite of key distribution and inter-network authentication protocols for mobile clients' fast inter-network handovers.

In Section 4.1, we describe the proposed scalable inter-network architecture for handovers among WMNs, new certificates for inter-network handovers, and a new trust model that enables one-hop wireless communications between a mobile client and a foreign network. In Section 4.2, we present our new authentication and key distribution protocols for fast inter-network handovers. We present a security analysis of the proposed authenti-

cation and key distribution protocols in Section 4.3. We present performance evaluations of the proposed key distribution and authentication protocols via numerical analysis and simulation results in Section 4.4. We conclude this chapter in Section 4.5.

4.1 The Proposed Architecture, Trust Model, and Certificates for Inter-Network Handovers

Mobile devices such as smart phones and tablets are becoming ubiquitous. In the existing solutions for inter-network handover authentication [9, 22, 50, 51, 52, 53], a foreign network has to communicate with a client's home network via multi-hop wireless communications to authenticate the client. This may result in high handover latency and low reliability, leading to potential service interruptions in WMNs. To support mobile clients' fast handovers among multiple WMNs and to minimize the inter-network handover latency, we propose a scalable architecture, a novel trust model and certificates for inter-network handover authentications. Our proposed architecture, trust model and certificates enable a foreign network to directly authenticate a mobile client through one-hop communication, which significantly reduces the inter-network handover latency.

4.1.1 The Proposed Inter-network Architecture

As discussed in Chapter 3, a WMN consists of mesh clients and mesh points (MPs) (routers) . Mesh clients can be static hosts (e.g., desktops, database servers) or mobile hosts (e.g., smart phones, tablets, laptops, PDAs). The MPs form a wireless mesh back-

bone to provide multi-hop connectivity from one mesh client to another or to the Internet. A small number of mesh points work as gateways, connecting the WMN to the Internet. A subset of mesh points act as mesh access points (MAPs), connecting mesh clients to the WMN. MAPs can be mobile or static. In the thesis we assume MAPs are mostly static.

Based on the existing WMN structure, we propose an architecture consisting of multiple WMNs, to support a mobile client's inter-network handovers, as shown in Figure 4.1. Each WMN is managed by a single operator. A subset of special MAPs in each network are designated as border mesh access points (BMAPs) at the time of network deployment by its operator. Besides performing the same functions as MAPs, BMAPs additionally support mesh clients roaming from one network to another. BMAPs are physically located at the border of their respective networks with much larger storage and faster processors compared to standard MAPs.

In Chapter 3, we discussed the scenario in which a mobile client connects to a MAP and then moves from one MAP to another within a single WMN. In this chapter, we assume scenarios in which a client is currently connected to a border mesh access points (BMAP) and moves from the current network to another network through an adjacent BMAP.

Two BMAPs, $BMAP_x$ and $BMAP_y$, are said to be *adjacent BMAPs* if and only if

– $BMAP_x$ and $BMAP_y$ do not belong to the same WMN, and are managed by different operators.

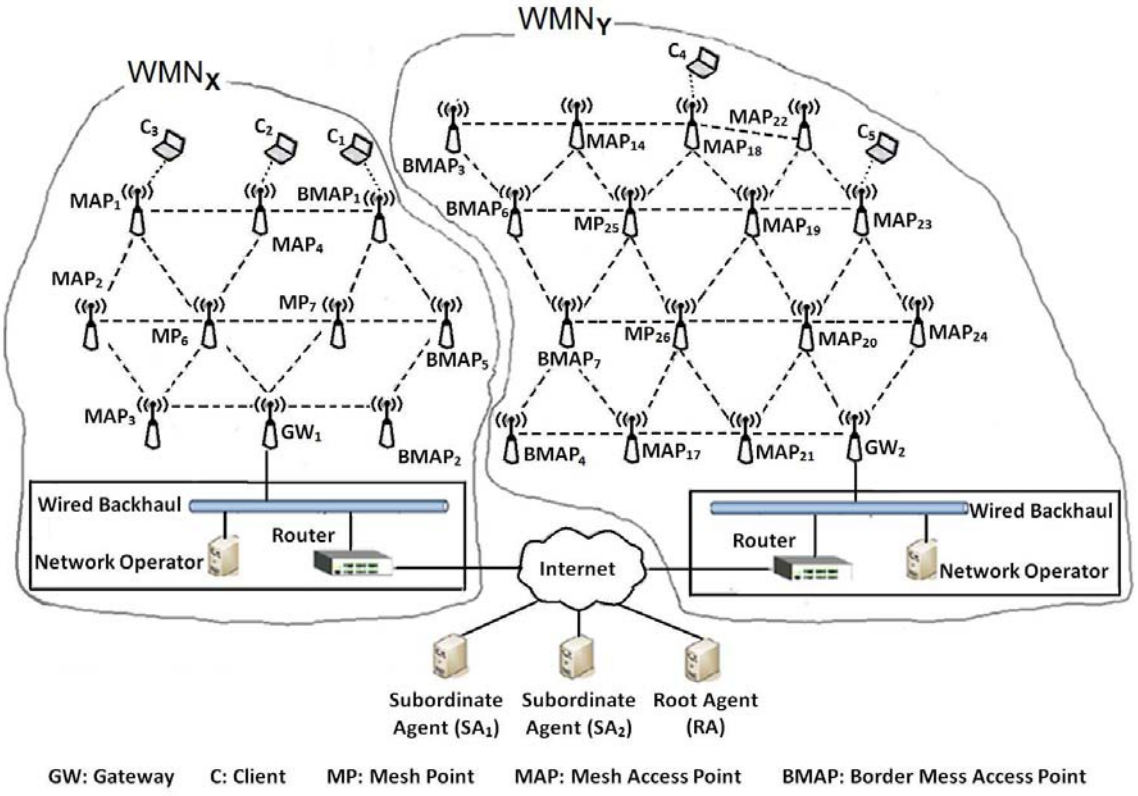


Figure 4.1: Scalable WMN architecture for secure inter-network handovers

– $BMAP_x$ and $BMAP_y$ are within each other's transmission range.

If two BMAPs belong to the same network and are within each other's transmission range, we refer them as *neighboring MAPs* as discussed in Chapter 3. A BMAP is also a MAP within a single WMN, and treated as a regular MAP in the context of intra-network authentication.

Given a mobile client C that is currently connected to network X through $BMAP_X$, $BMAP_X$ is considered to be client C's *local BMAP*, and network X is considered to be client C's *local network*. If $BMAP_X$ and $BMAP_Y$ are within each other's transmission range, and $BMAP_Y$ belongs to network Y, $BMAP_Y$ is considered to be $BMAP_X$'s *adjacent BMAP*. Network Y is considered to be network X's *adjacent network*. Note that network X and network Y are two different WMNs and managed by different operators.

For example, in Figure 4.1, X and Y are two WMNs managed by different operators. Each network designates its BMAPs to support inter-network handover authentications if a mobile client moves from its local BMAP to an adjacent BMAP. $BMAP_1$ and $BMAP_3$ are called adjacent BMAPs as $BAMP_1$ and $BMAP_3$ do not belong to the same network and are within each other's transmission range. So are $BMAP_2$ and $BAMP_4$. A client C_1 currently connects to $BMAP_1$ in network X, as shown in Figure 4.1. $BMAP_1$ is client C's local BMAP and network X is client C's local network. Client C can roam from network X to network Y through $BMAP_3$ as $BMAP_3$ is an adjacent BMAP of $BMAP_1$.

Most existing solutions [9, 22, 50, 51, 52, 53] for inter-network handover authentica-

tions involve multi-hop communications between a mobile client's home network and a foreign network. This may result in long delay, low reliability and thus, potential service interruptions in WMNs. In contrast, our proposed solution requires only one-hop wireless communications between a client and a foreign network for inter-network handover authentications. To support one-hop wireless communications specifically between a client and a foreign network, we design a set of certificates that are used to establish the trust relationship among various entities in WMNs. Certificates are issued and managed by certificate agents who are trusted third parties.

To facilitate fast handovers between networks, we propose a hierarchy of certificate agents that issue certificates to all network entities, including mobile clients, MAPs, and BMAPs. This design allows certificate agents to handle certificate requests from multiple networks in a scalable manner. There are two types of certificate agents: root agent and subordinate agents. The root agent and subordinate agents form a hierarchy of certificate agents to provide scalability for processing a large number of certificate requests.

The hierarchy of certificate agents can be represented by a tree in which the root node represents the root agent, who can directly issue a certificate to a client, a MAP or a BMAP, or delegate the duty to any of its subordinate agents. Each non-root node in the tree represents a subordinate agent. Any subordinate agent can also directly issue a certificate to a client, a MAP, and a BMAP. If a subordinate agent has child subordinate agents, it can further delegate the duty to its child subordinate agents. Figure 4.2 shows an example of a hierarchy of certificate agents, where the root agent RA delegates the duty

to subordinate agents SA_1 and SA_2. The subordinate agent SA_1 can further delegate its duty to its children, subordinate agent SA_3 and SA_4.

As the network grows and the number of certificate requests increases, new subordinate agents can be added to the agent hierarchy as needed. A new subordinate agent can either be added directly under the root agent or under an existing subordinate agent. Thus, the agent hierarchy provides the scalability to meet the demands of a growing number of MAPs and mobile clients in WMNs.

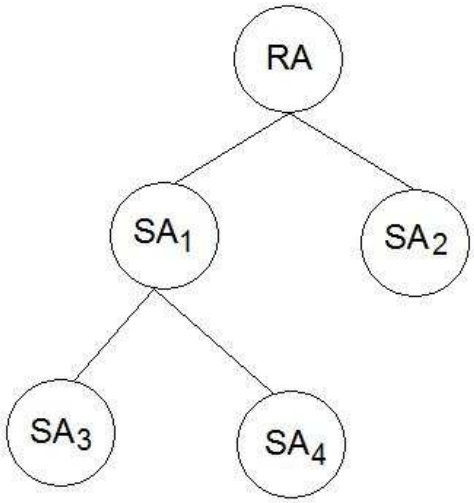


Figure 4.2: An example of an agent hierarchy

4.1.2 The Proposed Inter-network Trust Model

Trust relationships among the entities in the above architecture are the basis for the design of our proposed key distribution and authentication protocols for inter-network

handovers. Unlike the existing home-foreign-domain model discussed in Section 1.1.2, our proposed architecture and trust model provide a mobile client with the freedom to move from one network to another without being bound to its home network. Our proposed architecture and trust model do not require a client's foreign network to contact the client's home network for authentication. Thus, there is no need for several round-trip message exchanges between a mobile client's home network and a foreign network, which may result in long delay, low reliability and thus, potential service interruptions. Since we eliminate multi-hop communications between the home network and foreign network, a new trust model and mechanism are required to replace the home-foreign-domain model for inter-network handover authentications.

Figure 4.3 depicts our proposed inter-network trust model with the trust relationships among network entities. The proposed trust model is built upon the concept of "certificates" and "certificate agents". A certificate is used to establish the trust relationship among entities.

A certificate agent issues a client certificate or a BMAP certificate after verifying the validity of the public key certificate of the client or the BMAP. A public key certificate is issued by a public key certificate authority, such as Verisign or Microsoft. A valid public certificate proves that the certificate holder can be trusted. We assume that each entity in our architecture, e.g., clients, MAPs, BMAPs and certificate agents, holds a valid public key certificate.

A certificate agent is a trusted third party who issues and manages various types of

certificates and can be trusted by various entities in networks. A certificate agent's role can be compared to that of public-key certificate authorities or credit card issuers.

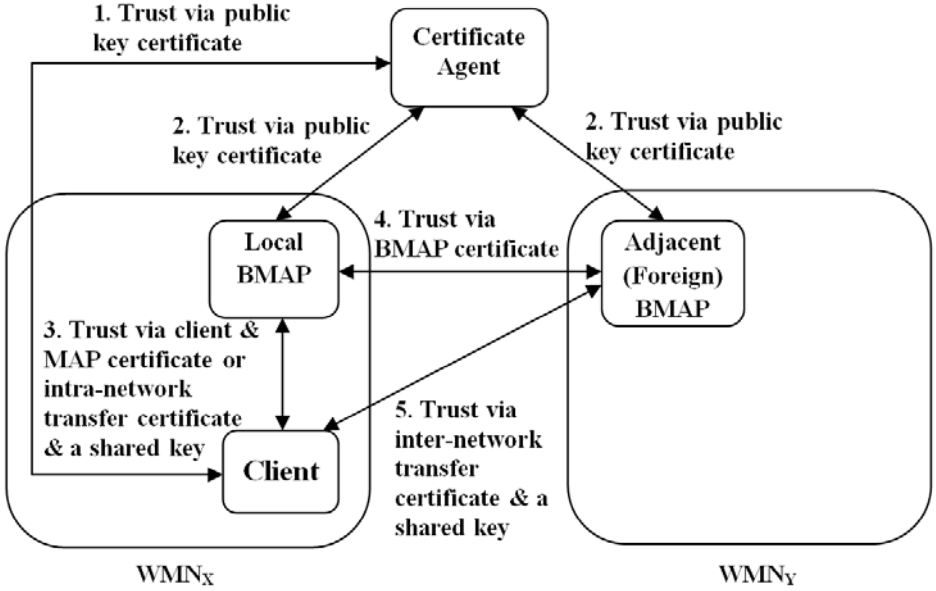


Figure 4.3: Trust model of WMNs

Following are the trust relationships among the network entities as shown in Figure 4.3:

(1) between a certificate agent and a client: By confirming the validity of their respective public key certificates issued by the public key certificate authorities, a client and a certificate agent can trust each other. The mutual trust is established when a client applies for a client certificate from a certificate agent.

(2) between a certificate agent and a BMAP: A BMAP trusts its certificate agent via the certificate agent's public key certificate issued by the public key certificate authority.

The trust between the BMAP and the certificate agent is established when the BMAP applies for a BMAP certificate from the certificate agent.

(3) between a local BMAP and a client: The mutual trust between a client and its local BMAP is established either via their respective client certificate and BMAP certificate, or via an intra-network transfer certificate and a shared key, which will be discussed in Section 4.2.2.4.

(4) between a local BMAP and an adjacent BMAP: Any two adjacent BMAPs trust each other via their respective BMAP certificates. This trust allows a BMAP to accept a client previously authenticated successfully by another BMAP located in a different network.

(5) between a foreign BMAP and a client: A client trusts a foreign BMAP and vice versa via an inter-network transfer certificate and a shared key. After a client and a local BMAP successfully authenticate each other via either the login authentication protocol (LAP) or the handover authentication protocol (HAP) described in Chapter 3, the local BMAP generates an inter-network transfer certificate and a shared key, and sends them to the client. The local BMAP also sends the shared key to the adjacent (foreign) BMAPs. When moving into contact with a foreign BMAP, the client presents the inter-network transfer certificate to the foreign BMAP. The client and the foreign BMAP then use the inter-network transfer certificate and the shared key to authenticate each other (Section 4.2.2.3).

Obtaining a client certificate or a BMAP certificate is done offline before a client joins a network, and not part of inter-network authentication and key distribution process. Thus, the public key operations for obtaining certificates do not affect the efficiency of our inter-network authentication and key distribution protocols presented in Section 4.2.

4.1.3 Certificates for Inter-Network Handovers

We have presented client certificates, MAP certificates and intra-network transfer certificates in Chapter 3. They are required for mutual authentication between a client and a MAP when the client logs in to a network, or roams from one MAP to another MAP within a single WMN. In this chapter, we propose additional certificates to support a client moving from one network to another network.

Certificates are issued and managed by certificate agents who are trusted by all entities of networks to perform such tasks. There can be several certificate agents serving different WMNs. One agent can serve several networks. Certificates are used to establish the trust between a client and a set of BMAPs, and between adjacent BMAPs (see Figure 3.1). The lifetime of a certificate is determined by its issuer's policy.

Three types of certificates are used in our inter-network authentication and key distribution protocols: client certificate, BMAP certificate, and inter-network transfer certificate. They are needed for mutual authentication between a client and a BMAP when the client roams from one BMAP to an adjacent BMAP.

We will use the notations listed in Table 4.1 throughout the chapter to facilitate the

discussion.

Table 4.1: Notations

Notation	Description
C	Client
Γ	Type of certificate
A	Certificate agent
I_x	ID of entity x
λ_C	Inter-network transfer certificate issued to a client
P_x	Public key issued to x
T_x	Certificate issued to x
τ_{exp}	Expiry date and time of a certificate
N_x	A nonce generated by x
Sig_x	Digital signature of entity x
MAC_{alg}	Type of MAC algorithm
$E_{P_x}(m)$	Encryption of message, m using x's public key
$D_{P_x}(m)$	Decryption of message,m using x's public key
$V_k(m)$	Message authentication code (MAC) resulting from the application of a MAC algorithm and a MAC key k on a message m

4.1.3.1 Client Certificates

A client applies for a client certificate from a certificate agent. A client trusts a certificate agent via the agents's public key certificate issued by a central authority. A client certificate is unique to each client and can be used for both intra-network and inter-network handovers.

Following is the structure of a client certificate which has also been described in Chapter 3:

$$\boxed{T_C = \{I_C, I_A, \tau_{exp}, P_C, Sig_A\}}$$

- T_C: client certificate issued by certificate agent A whose ID is I_A.
- I_C: ID of the client that is given this certificate.
- I_A: ID of the certificate agent who issued the certificate T_C.
- τ_{exp}: expiry date and time of certificate T_C. The certificate agent will re-issue a new certificate for the client if the current certificate has expired.
- P_C: public key of client I_C, which is used by a MAP or BMAP to verify the signature signed by the client in the protocol. The certificate agent obtains the public key from the client's public key certificate. We assume that the agent is a trusted party and has access to public key certificates of all clients and MAPs.
- Sig_A: digital signature of certificate agent I_A, which gives a recipient the reason to

believe that the certificate was created by certificate agent I_A, and that it was not altered in anyway.

4.1.3.2 BMAP Certificates

Each BMAP is pre-installed with a BMAP certificate. The network operator of each network obtains a BMAP certificate for each BMAP from a certificate agent. When requesting certificates, the network operator provides the agent with the public key of each BMAP, which will later be embedded in the BMAP's certificate. Since its public key is part of the certificate, each BMAP must be pre-installed with a public key before deployment.

Following is the structure of a BMAP certificate:

$$\boxed{T_{BM} = \{\Gamma, I_{BM}, I_A, \tau_{exp}, P_{BM}, Sig_A\}}$$

- T_{BM}: BMAP certificate issued by a certificate agent A whose ID is I_A.
- Γ: Type of the certificate, indicating "inter-network" or "intra-network".
- I_{BM}: ID of the BMAP that is given this certificate.
- I_A: ID of the agent who issued certificate T_{BM} to BMAP BM.
- τ_{exp}: expiry date and time of certificate T_{BM}. The agent will issue a new certificate for the BMAP once the current certificate expires.

- P_{BM}: BM's public key, which will be used by its neighbors to verify the signature of BM in messages BM sends. The certificate agent obtains the public key of BM from BM's public key certificate.

- Sig_A: digital signature of the certificate agent A, which will be used to identify the sender of the certificate.

4.1.3.3 Inter-network Transfer Certificates

When a client roams from its local BMAP to an adjacent BMAP, the trust relationship between the client and the adjacent BMAP is based on an inter-network transfer certificate. When a client C first logs in to the network through a BMAP, this BMAP becomes C's local BMAP BM_i, which will authenticate the client through the login protocol. If the authentication succeeds, BM_i will issue C an inter-network transfer certificate and become C's local BMAP. When C roams to an adjacent BMAP BM_j, it submits the inter-network transfer certificate to BM_j for authentication. The inter-network transfer certificate proves to the adjacent BMAP that client C had been successfully authenticated by its local BMAP.

The structure of an inter-network transfer certificate λ_C is as follows:

$$\lambda_C = \{\nu, V_{CMK}(\nu)\}, \text{ where}$$
$$\nu = \{I_{cert}, I_{BM_i}, I_C, P_C, I_N, \Gamma, \tau_{exp}, MAC_{alg}\}$$

Message ν stores the information of the client, local BMAP and certificate agent as follows:

- I_{cert}: ID of the inter-network transfer certificate. The combination of I_{cert}, I_{BM_i} and I_C uniquely identifies a transfer certificate n the network.

- I_{BM_i}: ID of the BMAP who issues this inter-network transfer certificate.

- I_C: ID of the client who owns this transfer certificate.

- P_C: public key of the client. The client's home BMAP obtains the client's public key from C's client certificate.

- I_N: ID of the network to which the current BMAP belongs.

- Γ: Type of certificate indicating "inter-network" or "intra-network".

- τ_{exp}: expiry date and time of this certificate.

- MAC_{alg}: message authentication code algorithm. The inclusion of the type of MAC algorithm in transfer certificates is optional. It is not required if the parties agree on an algorithm in advance.

  We now discuss about the value $V_{CMK(\nu)}$ stored in the inter-network transfer certificate and the use of the MAC algorithm. During the authentication between client C and its BMAP BM_i (step (1) in Figure 4.5), they exchange two partial keys N_{C4} and N_{R4} (see Section 4.2.2.1 for details of the authentication procedure). They will both then compute a shared key $CMK = N_{C4}||N_{R4}$, where $||$ denotes a concatenation. BM_i subsequently applies the MAC algorithm and key CMK to message ν to produce a MAC value $V_{CMK}(\nu)$. This MAC value will protect message

ν, and thus the inter-network transfer certificate, against forgery and unauthorized modifications. BM_i combines message ν and $V_{CMK}(\nu)$ to form the inter-network transfer certificate to be sent to C.

When client C moves into contact with an adjacent BMAP BM_j, C submits its inter-network transfer certificate issued by BM_i to BM_j for authentication. BM_i sends the key CMK to the adjacent BMAPs, including BM_j, in advance of the handover via the key distribution to neighbors (KDN) protocol, which will be described in Section 4.2.2.2. BM_j will use key CMK and the MAC algorithm to verify the authenticity and data integrity of the inter-network transfer certificate λ_C submitted by client C in order to authenticate C. BM_j applies the MAC algorithm and key CMK to message ν to produce a MAC value $V'_{CMK}(\nu)$. If $V'_{CMK}(\nu) = V_{CMK}(\nu)$, BM_j can confirm that the inter-network transfer certificate λ_C is valid. It should be noted that each certificate has its own expiration date. The life time of a key CMK is the same as that of the inter-network transfer certificate associated with it. An adjacent BMAP can generate a new inter-network transfer certificate for a mobile client if the mobile client's current inter-network transfer certificate is about to expire.

4.2 The Proposed Security Protocols for Fast Inter-Network Handovers

Based on the above inter-network architecture and trust model, we propose a suite of inter-network security protocols operating in two phases: phase 1 for network initializa-

tion, and phase 2 for inter-network handovers. The adjacent BMAP discovery protocol in phase 1 is to initialize the networks, which prepares the BMAPs for inter-network handover authentication and key distribution. The key distribution and inter-network authentication protocols in phase 2 are to support clients' fast inter-network handovers from one network to another.

Public key operations are computationally intensive. Mobile devices, on the other hand, have limited computing capability and power resources. Therefore, our design of the proposed inter-network key distribution and authentication protocols aims to minimize the number of public key operations performed by mobile devices, thus minimizing their resource consumption. We also minimize the number of messages exchanged between a mobile client and BMAPs, thus minimizing the authentication latency and resource consumption by the mobile device. In addition, we aim to minimize the number of multi-hop wireless communications, thus, minimizing the authentication latency and traffic in the backhaul network. At the same time, we ensure that the protocols are secure and scalable. Note that BMAPs are not computationally constrained and typically have constant power supplies; thus we are not concerned about them regarding public key operations.

4.2.1 Phase 1: Network Initialization

Adjacent BMAPs need to trust each other in order to support mobile clients roaming from one network to another. To establish such trust between adjacent BMAPs, in this

phase, adjacent BMAPs need to successfully authenticate each other.

The network initialization process consists of two stages: BMAP certificate distribution and adjacent BMAP discovery.

- In the BMAP certificate distribution stage, the operator of a mesh network applies for BMAP certificates from a certificate agent, one per BMAP, and distributes them to the BMAPs in the network. The operator is also responsible for requesting and distributing a new BMAP certificate before the current BMAP certificate expires. Since a certificate agent is a trusted authority, a BMAP certificate issued by a certificate agent is the proof of the authentication between the agent and the corresponding BMAP. Thus, each BMAP is pre-installed with its BMAP certificate.

- In the adjacent BMAP discovery stage, a BMAP tries to locate its adjacent BMAPs from the adjacent networks through the adjacent BMAP discovery (ABD) protocol. Two adjacent BMAPs then exchange their respective BMAP certificates, and verify the authenticity of the other party's certificate to authenticate each other.

The ABD protocol identifies the adjacent BMAPs of a given BMAP. The ABD protocol not only allows a BMAP to discover its adjacent BMAPs, but also verifies that the adjacent BMAPs can be trusted. The trust between adjacent BMAPs is established via the ABD protocol.

Figure 4.4 shows the adjacent BMAP discovery protocol, in which BM_i and BM_j are two adjacent BMAPs, and located in two different networks. Following is the order of the messages exchanged in the ABD protocol:

(1) $BM_i \longrightarrow * : Hello, I_{BM_i}, N_i$

(2) $BM_j \longrightarrow BM_i : N_i, N_j, T_{BM_j}, Sig_j(N_i, N_j, T_{BM_j})$

(3) $BM_i \longrightarrow BM_j : N_j, T_{BM_i}, Sig_i(N_j, T_{BM_i})$

Figure 4.4: Adjacent BMAP Discovery (ABD) protocol

(1) BM_i sends hello messages to its neighbors. The message contains BM_i's identity I_{BM_i} and a nonce N_i. A nonce is a number used only once for resisting replay attacks. Details of how nonces are used to resist replay attacks are discussed in Section 4.3.

(2) When a BMAP BM_j receives message (1), BM_j responds with a message which contains the received nonce N_i, its BMAP certificate T_{BM_j} and a new nonce N_j to inform the sender BMAP BM_i of its presence. BM_j also digitally signs the message. When BM_i receives message (2), BM_i first verifies the digital signature of the certificate agent A who issued the BMAP certificate T_{BM_j} using A's public key. BM_i also verifies other information in the BMAP certificate such as the ID of the certificate agent and the expiry date of the certificate. The goal is to prove that the owner of the certificate is a trusted BMAP. If the agent's signature is successfully verified, BM_i extracts the public key of BM_j from T_{BM_j}. BM_i then verifies the digital signature of message (2) using BM_j's public key in order to know if message (2) was indeed sent by BM_j. Thus, once successfully verifying message (2), BM_i authenticates BM_j as its trusted adjacent BMAP, who is the owner of T_{BM_j} and signer of message (2).

(3) If the above verifications are successful, BM_i extracts nonce N_j from message (2)

and generates a message which contains the BMAP certificate T_{BM_i} and the received nonce N_j. BM_i then signs this message using its digital signature and sends it to BM_j. Upon receiving the message, BM_j verifies the digital signature of the certificate agent who issued the BMAP certificate T_{BM_i} using the certificate agent's public key. BM_j then verifies other information recorded in the BMAP certificate T_{BM_i} such as the ID of the certificate agent who issued T_{BM_i} and the expiry date of the certificate. The goal is to prove that BM_j can trust BM_i as an adjacent BMAP for supporting inter-network handovers. If the verification of the certificate agent's signature succeeds, BM_j retrieves the public key of BM_i from T_{BM_i}. To determine if message (3) was indeed sent by BM_i, BM_j uses BM_i's public key to verify the digital signature of message (3). Once BM_j successfully verifies message (3), BM_j consideres BM_i as a trusted neighbor BMAP.

After executing the ABD protocol, a BMAP has discovered its adjacent BMAPs. By verifying BMAP certificates via the ABD protocol, an adjacent BMAP can trust the local BMAP, and vise versa. When a client roams from the local BMAP to an adjacent BMAP, the adjacent BMAP will verify the inter-network transfer certificate issued by the local BMAP to authenticate the client. We discuss the inter-network handover authentication protocol in the next section.

4.2.2 Phase 2: Inter-network Handover

To support fast inter-network handovers for clients moving from one network to another via adjacent BMAPs, we propose a three-stage process: (a) inter-network transfer certificate generation, (b) key distribution, and (c) inter-network handover authentication. In the first stage, inter-network transfer certificate generation, a local BMAP generates and distributes an inter-network transfer certificate to a client after successfully authenticating the client, which is described in detail in Section 4.2.2.1. In the key distribution stage, a BMAP distributes a shared key to its adjacent BMAPs for a client's inter-network handovers, which is described in detail in Section 4.2.2.2. In the inter-network handover authentication stage, a client and its adjacent BMAP execute the proposed inter-network handover authentication protocol (IHAP) to authenticate each other, which is discussed in Section 4.2.2.3.

Figure 4.5 shows a simplified process of an inter-network handover through the above three stages. Suppose a client C connects to network A through a BMAP M_2 and then roams to an adjacent BMAP M_4 of network B. After client C and M_2 successfully authenticate each other via the login authentication protocol through a BMAP (see Section 4.2.2.1), M_2 generates an inter-network transfer certificate λ_C and sends λ_C to C in stage (a). M_2 and C also compute a shared key CMK. M_2 then distributes key CMK to M_4 through the KDN protocol in stage (b) (see Section 4.2.2.2). Client C and M_4 then use the inter-network transfer certificate λ_C and key CMK, and execute the IHAP protocol in stage (c) to mutually authenticate each other (see Section 4.2.2.3).

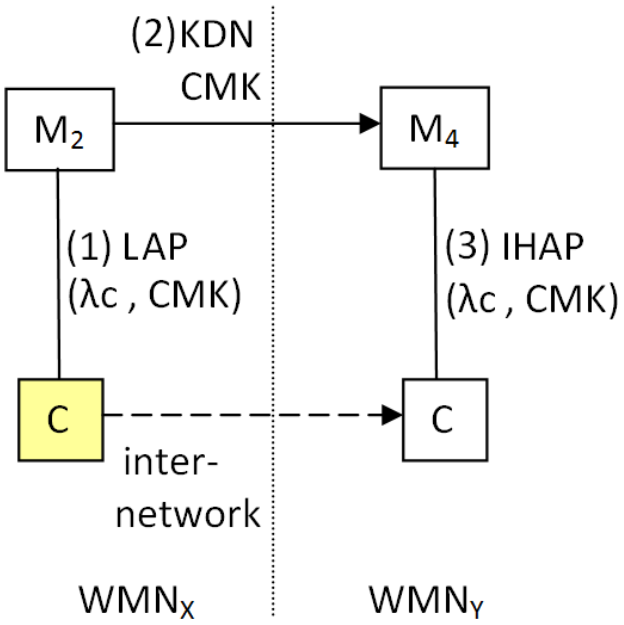


Figure 4.5: Inter-network handover

In the following sections, we describe the three stages of an inter-network handover in detail. A local BMAP first generates an inter-network transfer certificate λ_C for a client. The local BMAP then distributes a shared key CMK to its adjacent BMAPs via the key distribution to neighbors (KDN) protocol. When a client roams to an adjacent BMAP, the client and the adjacent BMAP mutually authenticate each other via the inter-network handover authentication protocol (IHAP) using λ_C and CMK.

4.2.2.1 Stage 1: Inter-network Transfer Certificate Generation

When a mobile client C connects to a network for the first time, client C and its connecting MAP need to successfully authenticate each other through the login authentication

protocol (LAP) as discussed in Section 3.2.1. If the connecting MAP BM_i is also a BMAP, BM_i needs to generate an inter-network transfer certificate for C's subsequent inter-network handovers. We add more information (such as nonces N_{C4} and N_{R4}, and inter-network transfer certificate λ_C as shown in Figure 4.6) to the LAP discussed in Section 3.2.1 in order to generate an inter-network transfer certificate.

The trust between client C and its home BMAP BM_i is established via C's client certificate and BM_i's BMAP certificate. Since an agent is a *trusted* authority, a client (or a BMAP) certificate issued in advance by the agent is the proof of a successful offline authentication between the agent and the corresponding client (or BMAP). After BM_i successfully authenticates C, it creates an inter-network transfer certificate λ_C for C, and subsequently sends a message containing the inter-network transfer certificate to C. C will use λ_C to roam from the current BMAP to a BMAP in another network. The inter-network transfer certificate has to be used in conjunction with a 128-bit MAC key CMK.

Figure 4.6 shows the order of the messages to be exchanged between a client C and a BMAP BM_i in the login authentication protocol.

(1) A client C requests to join a network and associate with a BMAP BM_i. C sends a request message containing its ID number to the BMAP.

(2) A BMAP BM_i replies with a message which contains its BMAP certificate T_{BM_i} to inform C of its presence and ID. Client C verifies the digital signature of the certificate agent A who issued the BMAP certificate T_{BM_i} using A's public key.

(1) $C \longrightarrow BM_i$: I_C

(2) $BM_i \longrightarrow C$: T_{BM_i}

(3) $C \longrightarrow BM_i$: $T_C, E_{P_{BM_i}}(N_{C1}, N_{C2}, N_{C3}, N_{C4})$

(4) $BM_i \longrightarrow C$: $E_{P_C}(N_{R1}, N_{R2}, N_{R3}, N_{R4})$

(5) $C \longrightarrow BM_i$: N_{R2}

(6) $BM_i \longrightarrow C$: $N_{C2}, \Theta_C, \lambda_C$

Figure 4.6: Login authentication protocol through a BMAP

We assume that client C and BMAP BM_i have the public key certificate of the certificate agent. C also verifies other information in the BMAP certificate such as the ID of the certificate agent and the certificate expiry date.

(3) If the above verifications are successful, C extracts the BMAP's public key from the BMAP certificate T_{BM_i} and generates a message which contains C's client certificate T_C and four nonces N_{C1}, N_{C2}, N_{C3} and N_{C4}. C then encrypts the four nonces using the BMAP's public key ($E_{P_{BM_i}}(N_{C1}, N_{C2}, N_{C3}, N_{C4})$). C then sends the encrypted four nonces and its client certificate T_C to the BMAP BM_i. Upon receiving the message, BM_i decrypts the message using its private key to obtain the four nonces, and verifies information recorded in T_C such as the digital signature of the certificate agent who issued the client certificate T_C (using the certificate agent's public key), the ID of the certificate agent who issued T_C and the certificate expiry date.

(4) If the above verifications succeed, BM_i retrieves the client's public key from certifi-

cate T_C, and generates four nonces N_{R1}, N_{R2}, N_{R3} and N_{R4}. BM_i then encrypts the four nonces using the client's public key ($E_{P_C}(N_{R1}, N_{R2}, N_{R3}, N_{R4})$), and sends the encrypted message to client C. C will decrypt the message using its private key to obtain N_{R1}, N_{R2}, N_{R3} and N_{R4}.

Both the client and the BMAP then calculate two 128-bit MAC keys $K_{MAC} = N_{C1}||N_{R1}$ and $CMK = N_{C4}||N_{R4}$, where the operator $||$ denotes a concatenation, and N_{C1}, N_{C4}, N_{R1} and N_{R4} are the nonces generated in steps (3) and (4) above.

BM_i and C also derive a pairwise master key $PMK = N_{C3}||N_{R3}$ to be used after the authentication to compute a shared key called pairwise transient key (PTK) as specified by the IEEE 802.11i security standards. The generation of the PTK is discussed in Section 3.2.3.

(5) Client C sends N_{R2} to the BMAP BM_i. Upon receiving this message, the BMAP BM_i has successfully authenticated the client C, because only C has the knowledge of N_{R2}.

(6) To allow client C to authenticate BM_i, BM_i sends N_{C2} (generated by C in step (3)) to client C. BM_i also creates two transfer certificates Θ_C and λ_C for C's intra-network handovers and inter-network handovers, respectively. BM_i subsequently sends C a message containing nonce N_{C2}, the intra-network transfer certificate Θ_C and the inter-network transfer certificate λ_C. After client C receives N_{C2} correctly, it is considered to have successfully authenticated the BMAP because only BM_i has

the knowledge of N_{C2}. C can use the intra-network transfer certificate Θ_C to roam to the next MAP of the current network. The intra-network transfer certificate Θ_C has to be used in conjunction with the key K_{MAC} generated in step (4). C can also use the inter-network transfer certificate λ_C in conjunction with the key CMK [generated in step (4) above] to roam from its home BMAP BM_i to a foreign BMAP in another network (see Section 4.2.2.3).

If C connects to the network for first time through a MAP, and later roams to the BMAP BM_i, C and BM_i authenticate each other through the handover authentication protocol (HAP) discussed in Section 3.2.2. After a successful authentication through the HAP, BM_i generates and sends an inter-network transfer certificate λ_C to C. BM_i also generates two shared keys, CMK and PMK. BM_i encrypts CMK and PMK using client C's public key. Upon receiving the encrypted message, client C decrypts them using its private key to extract key CMK and PMK. C will use inter-network transfer certificate λ_C and shared key CMK as proof of a successful authentication with BM_i. C and BM_i will use PMK to generate a shared key to be used for secure data exchange after the authentication.

After a client C is authenticated by its home BMAP BM_i and obtains the inter-network transfer certificate and the associated shared key CMK, we proceed to the key distribution stage.

4.2.2.2 Stage 2: Key Distribution

BM_i also needs to distribute the shared key CMK to its adjacent BMAPs, since the adjacent BMAPs need to know the CMK in order to authenticate client C in the future. In the KDN protocol, the local BMAP BM_i broadcasts an encrypted message to its adjacent BMAPs. The message contains the ID of the inter-network transfer certificate, BM_i's ID, client C's ID, a nonce N_i and the shared key CMK. Figure 4.7 shows the steps of the KDN protocol.

$$
\begin{aligned}
BM_i \longrightarrow * : \quad & \{I_{BM_1}, E_{P_1}(\beta)\}, \{I_{BM_2}, E_{P_2}(\beta)\}, \ldots, \{I_{BM_n}, E_{P_n}(\beta)\}, \\
& Sig_i(\{I_{BM_1}, E_{P_1}(\beta)\}, \{I_{BM_2}, E_{P_2}(\beta)\}, \ldots, \{I_{BM_n}, E_{P_n}(\beta)\}) \\
& \text{where } \beta = \{I_{cert}, I_{BM_i}, I_C, N_i, CMK\}
\end{aligned}
$$

Figure 4.7: Key distribution to neighbors (KDN) protocol

Suppose BM_1, BM_2, ..., BM_n are adjacent BMAPs of BM_i which are identified via the adjacent BMAP discovery (ABD) protocol discussed in Section 4.2.1. After client C and BM_i successfully authenticate each other (via the Login authentication protocol through a BMAP discussed in Section 4.2.2.1), BM_i creates a message β, which contains the ID of the inter-network transfer certificate I_{cert}, BM_i's ID I_{BM_i}, client C's ID I_C, a nonce N_i, and key CMK. For each BM_j where $j = 1, 2, \ldots, n$, BM_i encrypts the message β using BM_j's public key P_j. (Since each BMAP's public key is included in the BMAP certificate, BM_i obtains the public keys of its adjacent BMAPs through the ABD protocol discussed in Section 4.2.1.) BM_i then sends the message to BM_j, which

contains BM_j's ID I_{BM_j}, and encrypted message $E_{P_j}(\beta)$.

Since the client may move in any direction, the local BMAP BM_i sends key CMK to all of its adjacent BMAPs in anticipation of client C's mobility. The local BMAP BM_i may combine several such messages $\{I_{BM_j}, E_{P_j}(\beta)\}$, where $j = 1, 2, \ldots, n)$ into one packet, generates the signature of the message $Sig_i(\{I_{BM_1}, E_{P_1}(\beta)\}$, $\{I_{BM_2}, E_{P_2}(\beta)\}, \ldots, \{I_{BM_n}, E_{P_n}(\beta)\})$, and broadcast the packet to all adjacent BMAPs in order to save bandwidth. The digital signature provides the guarantee that the message is from BM_i. Upon receiving the message, the adjacent BMAP BM_j verifies BM_i's signature, decrypts the message using its private key to extract key CMK. Thus, the shared key CMK is distributed to BM_i's adjacent BMAPs before C's inter-network handover. The above public key operations are performed by BMAPs, which are not constrained in terms of computing capability or power supply. To further enhance the reliability for broadcasting the encrypted message in the KDN protocol, the local BM_i can broadcast the key distribution message more than once.

In summary, the key distribution protocols allow BM_i to distribute the shared key CMK to client C and all adjacent BMAPs. When client C roams to an adjacent BMAP, C and the adjacent BMAP will use the shard key CMK and the inter-network transfer certificate λ_C to mutually authenticate each other via an inter-network handover authentication protocol, which will be discussed in the following section.

4.2.2.3 Stage 3: Inter-network Handover Authentication Protocol (IHAP)

After client C roams to an adjacent BMAP BM_j, C and BM_j execute the inter-network handover authentication protocol (IHAP) to mutually authenticate each other. Figure 4.8 shows the order of the messages to be exchanged in the IHAP.

(1) $C \longrightarrow BM_j : \lambda_C, N_C, V_{CMK}(N_C)$

(2) $BM_j \longrightarrow C : \Theta_C, N_B, V_{CMK}(N_C, N_B)$

(3) $C \longrightarrow BM_j : N_B, V_{CMK}(N_B)$

Figure 4.8: Inter-network handover authentication protocol (IHAP)

(1) Client C sends its inter-network transfer certificate λ_C, a nonce N_C and a message authentication code $V_{CMK}(N_C)$ to BM_j. The message authentication code is the result of applying the MAC algorithm and the shared key CMK to nonce N_C (see Section 2.2.4).

When BM_j receives this message, it first verifies the correctness of $V_{CMK}(N_C)$ using the MAC key CMK it received from C's previous BMAP BM_i. If the computed MAC value matches $V_{CMK}(N_C)$, BM_j can confirm that N_C was not altered during transmission. Next, BM_j verifies the validity of the inter-network transfer certificate λ_C. It checks the content of the inter-network transfer certificate, especially the ID of the client's certificate agent and the certificate expiry date. It then applies the MAC algorithm and key CMK received from BM_i to message ν to output a message authentication code $V'_{CMK}(\nu)$. [Recall from Section 4.1.3.3 that an inter-network

transfer certificate consists of two parts: the relevant information stored in a message ν and a message authentication code $V_{CMK}(\nu)$, which is the result of applying a MAC algorithm and a MAC key to message ν.] If $V'_{CMK}(\nu) = V_{CMK}(\nu)$, BM_j can confirm that the inter-network transfer certificate is valid (i.e., C was successfully authenticated by its previous BMAP BM_i).

Note that an attacker may capture the inter-network transfer certificate and attempt to use it, but will not pass the BMAP's authentication, because the attacker cannot produce a valid pair $(N_C, V_{CMK}(N_C))$ without the knowledge of key CMK. The pair $(N_C, V_{CMK}(N_C))$ also enables the protocol to resist denial-of-service attacks (see Section 4.3).

(2) BM_j creates an intra-network transfer certificate Θ_C for C, and subsequently sends to C a message containing a nonce N_B, the intra-network certificate Θ_C, and a message authentication code $V_{CMK}(N_C, N_B)$. After C receives this message, it computes a MAC value $V'_{CMK}(N_C, N_B)$, using nonces N_C and N_B. If $V'_{CMK}(N_C, N_B) = V_{CMK}(N_C, N_B)$, it is considered that C has successfully authenticated BM_j. Nonce N_C serves as a challenge C presents to BM_j. The inclusion of N_C in the MAC computation is the response of BM_j to the challenge. Nonce N_B serves as a challenge BM_j presents to C. C will need to response this challenge to BM_j in the next message upon receiving N_B. If $V'_{CMK}(N_C, N_B) = V_{CMK}(N_C, N_B)$, C can also ensure that nonce N_B has not been altered during the transmission. C will use the intra-network transfer certificate Θ_C to roam from one MAP to another in the network to which

BM_j belongs.

(3) Client C then executes the MAC algorithm using key CMK and nonce N_B as input. The result is a message authentication code $V_{CMK}(N_B)$, which C will send to BM_j along with N_B (the challenge from BM_j). Upon receiving N_B and $V_{CMK}(N_B)$, BM_j repeats the same MAC calculation on N_B. If it obtains the same message authentication code as $V_{CMK}(N_B)$, then this confirms C's identity since C is the only client who has the knowledge of key CMK. Thus, BM_j has successfully authenticated C.

Following are additional implementation issues and discussions.

(a) If the adjacent BMAP BM_j receives the inter-network transfer certificate λ_C from C before the key CMK from BM_i, BM_j will not be able to verify the validity of the transfer certificate because it does not have key CMK in order to apply the MAC algorithm to the certificate. In that case, BM_j sends back an error message to C. C will initiate a login authentication (described in Section 4.2.2.1) instead of the IHAP. However, assuming clients with low to moderate mobility speeds, we expect that this worst-case scenario will not happen often, and the inter-network handover authentication protocol will be used in most cases.

(b) The inter-network handover authentication protocol does not use digital signatures (public key cryptography), but rather a MAC algorithm, to minimize authentication latency during the handover process. (The comparison of computation costs

between MAC operations and digital signatures is discussed in Section 3.2.2 discussion (C)).

(c) Once client C and BM_j successfully authenticate each other, BM_j generates a pairwise master key PMK. BM_j encrypts the PMK using C's public key and sends to C. C uses its private key to decrypt the message and obtains the pairwise master key PMK. The adjacent BMAP and the client will use the pairwise master key PMK to compute a shared key called the pairwise transient key (PTK). The PTK generation procedure follows the four-way handshake protocol defined in IEEE 802.11i [35] (see Section 3.2.3). The PTK will be used to encrypt packets exchanged between two parities for subsequent secure communications.

(d) Since C may move to a next MAP that resides in the same network as BM_j, upon a successful authentication, BM_j also generates a shared MAC key K_{MAC}. BM_j encrypts K_{MAC} with PMK (discussed in (c)) together using C's public key and sends to C. C uses its private key to obtain K_{MAC}. C will use the MAC key K_{MAC} and the intra-network certificate Θ_C for its subsequent roaming from one MAP to another MAP in the network to which BM_j belongs. C and the next MAP then execute the intra-network handover protocol (HAP) discussed in Section 3.2.2.

(e) An inter-network transfer certificate and its associated shared key are recognized only by the issuer and its adjacent BMAPs. The reason is that a BMAP knows only its adjacent BMAPs (neighbors), and not those outside its one-hop communication

range. If C moves into contact with a BMAP for the first time, after successfully authenticating client C, BM_j generates an inter-network transfer certificate λ_C and a shared key CMK for C to roam to one of its adjacent BMAPs. If C visited BM_j before and the inter-network transfer certificate previously issued by BM_j has not expired, C can keep using it. If this previously issued inter-network transfer certificate has expired, BM_j creates a new inter-network transfer certificate for client C.

(f) The life time of a shared key CMK is the same as that of the inter-network transfer certificate associated with it because the key has to be used in conjunction with the certificate.

(g) Additional costs are required in order to support fast inter-network handover authentication, such as offline deployment of certificate agents, generation and maintenace of certificates, and high-performance BMAPs.

4.2.2.4 An Efficient Integrated Authentication System for Intra-network and Inter-network Handovers

In Chapter 3, we propose two authentication protocols for supporting *intra-network* handovers, namely the login authentication protocol (LAP) and the handover authentication protocol (HAP). The LAP is an authentication protocol for supporting clients' initial login to a network. The HAP is an authentication protocol for supporting mobile clients' fast *intra-network* handovers from one MAP to another. The focus of this chapter is on

inter-network handover authentications when a client moves from one network to another via adjacent BMAPs. We now discuss how the proposed intra-network (Chapter 3) and inter-network authentication protocols (Chapter 4) work with each other.

We have presented a simple case of inter-network handover authentication in Section 4.2.2, Figure 4.5. In this subsection, we present an example of our integrated authentication system that involves both intra-network and inter-network handovers illustrated by the flowchart in Figure 4.9. In this example, a mobile client roams from network X to network Y, then to network Z as shown in Figure 4.10.

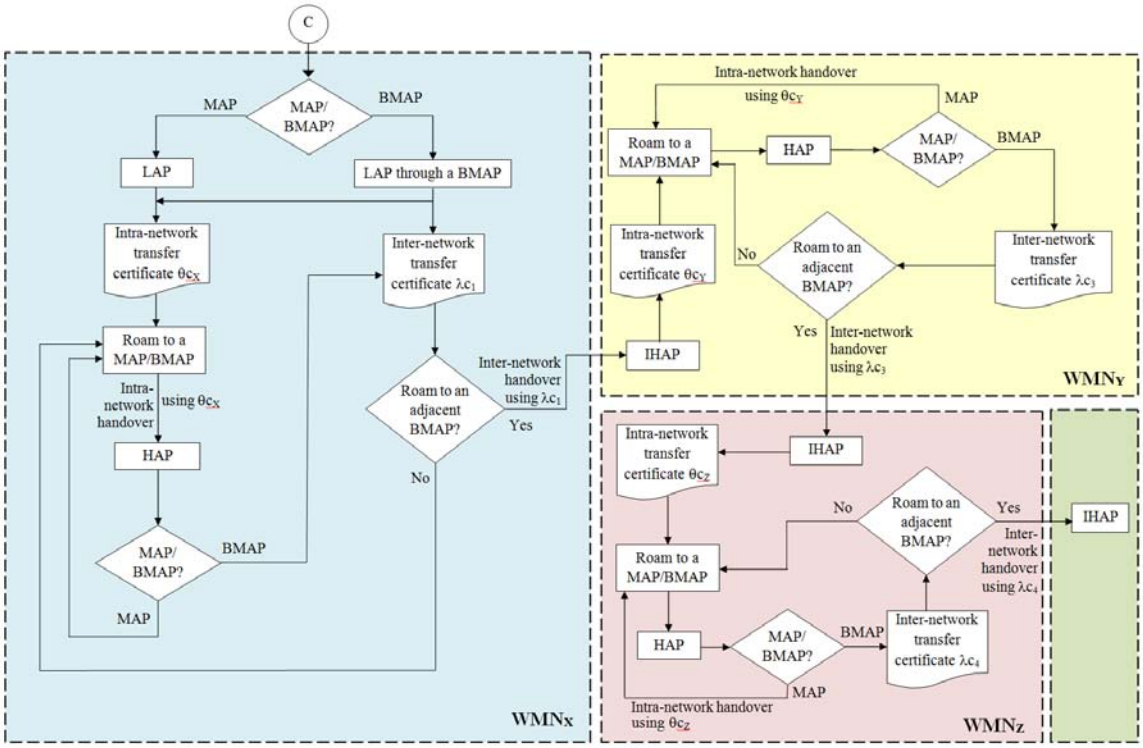
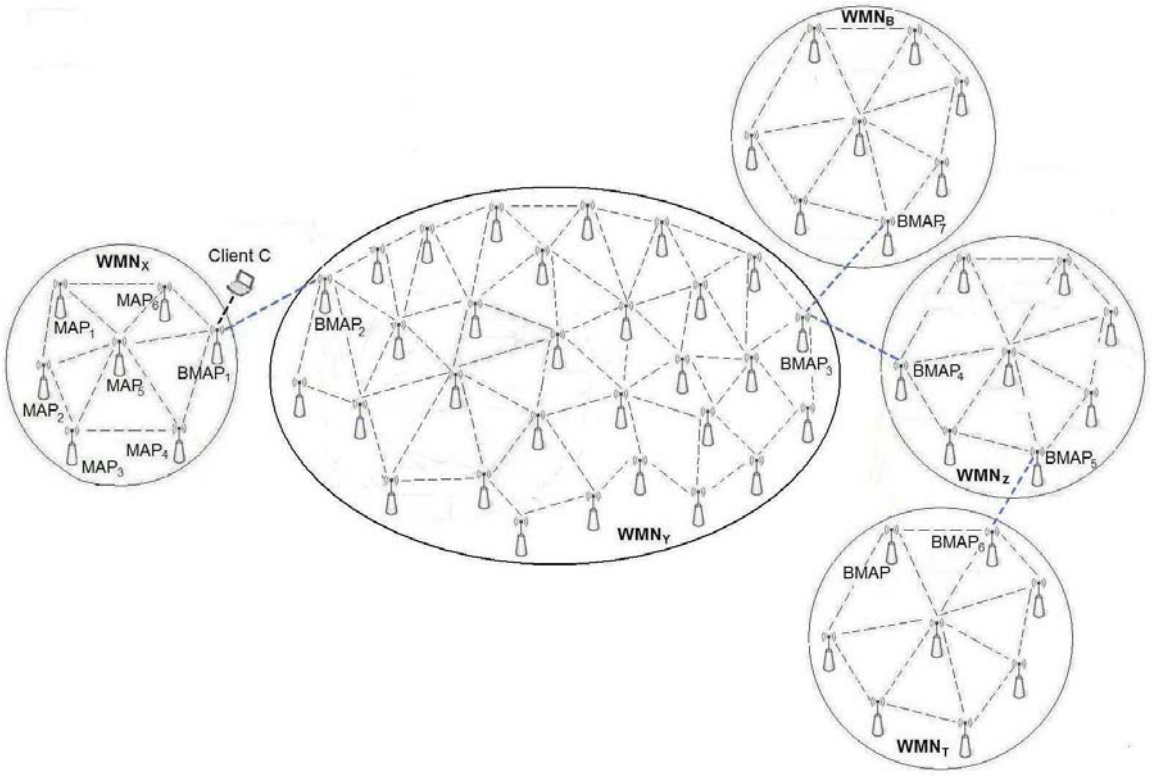


Figure 4.9: An integrated authentication system

When a mobile client C connects to a network X for the first time, client C and its



(Dashed lines represent wireless links between access points)

Figure 4.10: A mobile client roams from network X to network Y, then to network Z

connecting MAP M_i need to successfully authenticate each other through the LAP (see Section 3.2.1). The MAP then generates an intra-network transfer certificate Θ_{C_x} for client C's future intra-network handovers in network X. When moving to the next MAP M_j in network X, C needs to present the intra-network transfer certificate Θ_{C_x} to M_j for intra-network handover authentication through the HAP (see Section 3.2.2). Using the intra-network transfer certificate Θ_{C_x}, C can seamlessly roam from one MAP to another in network X. This process has been discussed thoroughly in Chapter 3.

If C connects to network X for the first time through a BMAP, e.g. $BMAP_1$, C and $BMAP_1$ need to authenticate each other via the login authentication protocol through a BMAP (see Section 4.2.2.1). After successfully authenticating client C, $BMAP_1$ generates an inter-network transfer certificate λ_{C_1} for C's future inter-network handover authentication. Both C and $BMAP_1$ also compute a shared key CMK_1. Since adjacent BMAPs need to know CMK_1 to authenticate C in the future, $BMAP_1$ also distributes the key CMK_1 to its adjacent BMAPs (e.g. $BMAP_2$ as shown in Figure 4.10) through the KDN protocol (see Section 4.2.2.2). This shared key CMK_1 is used in conjunction with client C's inter-network transfer certificate λ_{C_1} in the inter-network handover authentication protocol (IHAP).

C now can roam either inside network X using the intra-network transfer certificate Θ_{C_x} or to an adjacent network Y using the inter-network transfer certificate λ_{C_1}.

Case 1: If C stays inside network X, C needs to present the intra-network transfer certificate Θ_{C_x} to the next MAP in network X, they execute the intra-network

handover authentication protocol (HAP) (Section 3.2.2) to authenticate each other.

Case 2: If C roams to an adjacent BMAP $BMAP_2$ in another network Y, C and $BMAP_2$ need to mutually authenticate each other. C presents its inter-network transfer certificate λ_{C_1} to $BMAP_2$. $BMAP_2$ verifies λ_{C_1} and uses the shared key CMK_1 to authenticate client C through the IHAP (Section 4.2.2.3). $BMAP_2$ then generates an intra-network transfer certificate Θ_{C_Y} for client C's future intra-network handover authentication in network Y.

Case 2a: If C stays inside network Y and moves from $BMAP_2$ to a neighboring MAP M_i, by using the intra-network certificate Θ_{C_Y}, C and the neighboring MAP M_i can perform the intra-network handover authentication protocol (HAP) to mutually authenticate each other. As long as C stay inside network Y, C can roam from one MAP to another by presenting the intra-network certificate Θ_{C_Y}.

Case 2b: C can roam to another network, e.g., network Z through $BMAP_3$ ($BMAP_3$ belongs to network Y). When C moves into contact with $BMAP_3$ in network Y, after successfully authentication through the HAP, $BMAP_3$ generates and sends to C an inter-network transfer certificate λ_{C_3} and a new shared key CMK_3 (see the flowchart in Figure 4.9). $BMAP_3$ also sends the shared key CMK_3 to its adjacent BMAPs, $BMAP_4$ and $BMAP_7$ through the KDN protocol (Section 4.2.2.2). By presenting the inter-network transfer certificate λ_{C_3}, C can roam to either network Z through $BMAP_3$, or to network B

through $BMAP_7$. If C roams to network Z through $BMAP_4$, $BMAP_4$ will use CMK_3 to verify C's inter-network transfer certificate λ_{C_3}. Once C and $BMAP_4$ successfully authenticate each other through the inter-network handover authentication protocol (IHAP) (Section 4.2.2.3), $BMAP_4$ generates an intra-network transfer certificate Θ_{C_Z} for C's future intra-network handover authentication in network Z.

In existing inter-network authentication methods [9, 22, 50, 51, 52, 53], the authentication procedure requires several multi-hop round-trip message exchanges between either a client and an authentication server, or a client's home network and a foreign network, which may result in long delay, low reliability and thus potential service interruptions. In contrast, our integrated authentication system does not require multi-hop communications for authentication. A client and a MAP (or a BMAP) can mutually authenticate each other using either an intra-network (or an inter-network) certificate through one-hop communication, which significantly reduces the handover latency. Thus, this integrated authentication system provides mobile clients seamless roaming from one MAP to another, and from one network to another.

It is worth discussing the issue of how to implement a practical payment scheme for our proposed integrated authentication system. In the home-foreign-domain model, the payment schemes require a foreign network to contact a visitor's home network to negotiate for the services the visitor needs to pay for. Since clients are not bound to a "home" network in our proposed architecture, the existing payment schemes used in the

home-foreign-domain model are not applicable to our proposed integrated authentication system. Instead, real-time payment methods [93, 94] are more suitable to our integrated authentication system as these methods do not require a foreign network to communicate with a user's home network for the payment. In this case, a user's payment account is set up with a broker, a trusted third party. A user can roam to any network and pay for network services if the network agrees to use the broker's payment services. Following is an overview of this payment approach:

- A mobile client pays for network services through a broker who is a trusted third party. A mobile client first needs to buy pre-paid tokens from the broker (or establish credit with the broker).

- All participating networks agree to accept tokens as a payment method. A mobile client can roam from one network to another and pay tokens to a network while connecting to it.

- Each network reports the number of tokens it collected from the mobile client to the broker in order to be reimbursed for the service(s) it provided.

In Sections 4.3 and 4.4, we present a security analysis and performance evaluations of the proposed inter-network handover protocols, respectively.

4.3 Security Analysis

In this section, we identify the security threats relevant to our proposed protocols and discuss their countermeasures. The security threats related to the proposed security protocols include identity privacy attack, impersonation attack, forgery attack, space-time trade-off attack, replay attack, and DoS attack [16].

4.3.1 Overview

The proposed protocols are protected against various security threats thanks to the following security features:

- Digital signatures: to prevent forgery or unauthorized modifications to the message.
- Public key cryptography: to allow a BMAP to encrypt a message and securely distribute the message to the BMAP's neighbors (Section 4.2.2.2).
- Nonces: to combat replay attacks and denial-of-service (DoS) attacks.
- MAC algorithm and MAC keys: to enable a receiver to verify that a message or an information unit (e.g., a nonce) in a message has not been altered in an unauthorized manner. They also provide assurance that a message has been originated by an entity in possession of the MAC key. A counterfeit message will not be paired with the correct MAC value due to the attacker's lack of knowledge of the MAC key.

The following rules apply to our proposed protocols, namely the adjacent BMAP discovery (ABD), the key distribution to neighbors (KDN) protocols, and inter-network

authentication protocol (IHAP):

[R1] A new message with nonces intended for a specific recipient r must use newly generated nonces and not those previously sent to r. If a message with nonces was lost or damaged and the message is retransmitted, the retransmitted message must use newly generated nonces.

[R2] Each message is associated with a timer. If the timer expires before the sender receives a response from the intended recipient of the message, the sender assumes that the message has been lost or damaged.

[R3] If the key distribution or authentication procedure fails after a pre-determined number of tries, the BMAP will give up and send diagnostic information to the network administrator, which will initiate an investigation to determine the cause of the failure.

[R4] For the ABD protocol and the IHAP, if any message is lost, the protocol will restart from step (1).

The IHAP is also required to follow the following rule:

[R5] When a receiver receives a message with a nonce and a corresponding MAC value, it performs the MAC computation. If the resulting MAC value does not match the MAC value in the message, the receiver assumes that this is a message from an attacker.

Note that message losses and retransmissions discussed in this chapter are meant to be associated with the transport layer. (Loss detections and retransmissions may be done at the data link layer [e.g., by the RTS/CTS/DATA/ACK exchange of the IEEE 802.11 medium access control], but are transparent to the authentication protocols and do not follow the above rules.)

4.3.2 Security Analysis of the Proposed Security Protocols

In this subsection, we identify the security threats [16, 92] relevant to the adjacent BMAP discovery (ABD), key distribution to neighbors (KDN) and inter-network handover authentication protocol (IHAP). The threats include identity privacy attack, impersonation attack, forgery attack, time-memory trade-off attack, replay attack, and Denial-of-Service (DoS) attack. We also discuss the countermeasures against these threats.

4.3.2.1 Identity Privacy Attack

Most clients would like to remain anonymous while roaming in different networks for privacy reasons. To protect a client's privacy, the client ID in the inter-network transfer certificate is a number or a string that is not related to the client's real identity, much like a bank account number or a social security number. Only the issuer of the certificate knows the mapping between the client's real identity and the client ID recorded in the certificate.

4.3.2.2 Impersonation Attack

Impersonation is an attack in which an attacker masquerades as a trusted node. IP address spoofing is a form of impersonation attack, in which the IP address of the source of an IP packet is forged in order to conceal the true identity of the sender. We use digital signatures, message authentication codes (MACs), and encryptions to combat impersonation attacks.

In the ABD protocol shown in Figure 4.4, digital signatures are used to prevent impersonation attacks. The attacker may modify message (2), such as adding his ID and public key, or signing the message using his private key. The attacker then sends the modified message (2) to BM_i. After applying BM_j's public key on the modified message, BM_i can detect the bogus message because the message's signature should be signed using BM_j's private key. If the attacker does not know BM_j's private key, the attacker cannot produce a valid signature on behalf of BM_j.

The attacker may modify message (3) and attempt to impersonate BM_i. For example, the attacker may spoof BM_i's IP address, add his ID to the message, and sign the message. The attacker can sign the message with any key other than BM_i's private key if he does not know BM_i's private key. When applying BM_i's public key to the signature in the modified message, BM_j can detect that the message had been tampered with. Thus, if the attacker does not know BM_i's private key, the attacker cannot produce a valid signature on behalf of BM_i.

Similarly, digital signatures are also used in the KDN protocol (shown in Figure 4.7)

to prevent impersonation attacks.

In the IHAP protocol shown in Figure 4.2.2.3, message authentication codes (MACs) are used to prevent impersonation attacks. The attacker may spoof C's IP address, modify message (1) or (3) with the intent of impersonating C. The attacker may also spoof BM_j's IP address or alter message (2) in order to impersonate BM_j. However, any change to these messages would be detected if the attacker does not know the shared key CMK.

4.3.2.3 Forgery Attack

Forgery attack is an attack in which an attacker manipulates the content of a message with the intent that it would be accepted as a valid message. For example, an attacker may try to forge a BMAP certificate or an inter-network transfer certificate in order to make the receiver believe that the forged certificate is a valid one. We prevent this type of attack by using digital signatures and message authentication code (MAC) for data integrity. Unauthorized changes to the content of a message will result in an incorrect signature value if the attacker does not know the author's private key.

The ABD protocol (Figure 4.4) uses digital signatures to combat forgery attacks. For example, an attacker may change the content of message (2), such as nonce N_i, N_j, a BMAP certificate T_{BM_j}. A receiver BM_i can detect the modifications to the message when verifying BM_j's signature in message (2). Similarly, BM_j can verify BM_i's signature in the message (3) to detect any modifications to message (3). Unauthorized

changes to the content of a message will result in an incorrect signature value if the attacker does not know the private key of the sender.

The KDN protocol (Figure 4.7) also uses digital signatures to combat forgery attacks. If an attacker modifies the encrypted message β ($E_{P_n(\beta)}$), a receiver can detect the modifications to the message. The reason is that BM_i's digital signature is used to detect modifications of a message. Unauthorized changes to the content of a message will result in an incorrect signature value if the attacker does not know the private key of the sender.

In the IHAP protocol shown in Figure 4.8, the integrity of an inter-network transfer certificate $\lambda_C = \{\nu, V_{CMK}(\nu)\}$ is protected by the accompanying MAC value $V_{CMK}(\nu)$. Unauthorized changes to the content of the messages will result in an incorrect MAC value if the attacker does not know the key CMK shared between client C and BM_j (rule [R5]).

Similarly, a receiver can detect an altered message when verifying the MAC value of the message. For example, if an attacker changes nonce N_C to N'_C of message (1), BM_j can detect the changes in the message (1) if $V_{CMK}(N'_C) \neq V_{CMK}(N_C)$. Thus, a modified message will result in an incorrect MAC value if the attacker does not know the shared key CMK.

4.3.2.4 Space-time Trade-off Attack

Space-time trade-off is a situation in which memory usage use can be reduced at the cost of slower program execution (and, conversely, the computation time can be reduced at the

cost of increased memory usage). It has been considered as a cost-effective way of solving certain searching problems such as the knapsack and discrete logarithm problem [108].

The space-time trade-off attack is a type of cryptographic attack where an attacker tries to recover a key when the plaintext and the ciphertext are known. This attack can be applied to data encryption algorithms [109] to break the data encryption key. This attack can also be applied to MAC algorithms where the attacker may try to recover the MAC key of a hash-based MAC algorithm.

A space-time trade-off attack has two phases: the pre-computation phase and the online phase. In the pre-computation phase, the attacker executes an exhaustive search of the MAC keys and stores the hashing results. This is done offline and can take a long time. Once this pre-computation is done, the attacker could quickly recover a key using the pre-computed results stored in the memory. The time taken in the online phase is shortened due to the pre-computation results stored in the memory. To combat this type of attack, we use current state-of-the-art MAC algorithms based on SHA-2 in the proposed IHAP (Figure 4.8), and periodically update MAC keys. (Space-time trade-off attacks are not applicable to the ABD and the KDN protocols as there are no MAC algorithms used in those protocols.)

4.3.2.5 Replay Attack

A replay attack occurs when an attacker intercepts a packet and forwards that packet to a service or application as if the attacker were the user who originally sent the packet.

One of the purposes of a replay attack is to gain access to the network as a valid network entity. We prevent this type of attack by using message encryption, nonces and the security rules listed in Section 4.3.1. We discuss how each of the proposed protocols resists replay attacks.

4.3.2.5.1 The adjacent BMAP discovery (ABD) protocol (Figure 4.4)

Two adjacent BMAPs execute the ABD protocol to identify each other as trusted neighbors. We discuss how each message in the ABD protocol resists replay attacks.

(1) $BM_i \longrightarrow * : Hello, I_{BM_i}, N_i$

(2) $BM_j \longrightarrow BM_i : N_i, N_j, T_{BM_j}, Sig_j(N_i, N_j, T_{BM_j})$

(3) $BM_i \longrightarrow BM_j : N_j, T_{BM_i}, Sig_i(N_j, T_{BM_i})$

Replaying message (1) An attacker may overhear and replay message (1) sent earlier by BM_i to impersonate BM_i.

- If BM_j had successfully received the original message (1) from BM_i, it saved the nonce N_i. When BM_j receives the replayed message, it compares the nonce in the message against the saved nonce N_i, and can detect that this is a replayed message because a new message is supposed to contain a new nonce and not a repeated nonce (rules [R1] and [R2]).

- If BM_j did not receive the original message (1) from BM_i, BM_j may accept the replayed message as a valid message (if the timer associated with message (1) has

not expired yet) and reply with message (2). The attacker may create a message (3) to send to BM_j, but cannot produce a valid signature $Sig_i(N_j, T_{BM_i})$ without the private key of BM_i. BM_j will not continue to the next step and will disregard the attacker as a trusted neighbor.

Replaying message (2) An attacker tries to impersonate BM_j and replays message (2) to BM_i.

- If BM_i had successfully received the original message (2) from BM_j, it saved nonce N_i (which was sent by BM_i in message (1)) and N_j. When BM_i receives the replayed message, it compares the nonces in the message against the saved nonce N_i and N_j, and can detect that this is a replayed message because a new message should contains new nonces, and not repeated nonces (rules [R1] and [R2]).

- If BM_i did not receive the original message (2) from BM_j, BM_i may accept the replayed message as a valid message (if the timer associated with message (2) has not expired yet) and reply with message (3). In this case, the attacker actually helps to "retransmit" message (2) that BM_i lost in the first place. BM_i has to verify that message (2) is indeed from BM_j in order to authenticate BM_j as its neighbor. Message (2) contains BM_j's digital signature that is used to prevent unauthorized modifications. By verifying BM_j's BMAP certificate T_{BM_j} and signature $Sig_j(N_i, N_j, T_{BM_j})$ in message (2), BM_i can confirm that message (2) was from BM_j. BM_i then successfully authenticates BM_j as its neighbor. Since mes-

sage (2) is protected by BM_j's digital signature, by simply replaying message (2), the attacker cannot impersonate BM_j to become BM_i's neighbor. (BM_i then sends message (3) to BM_j, so that BM_j can authenticate BM_i.)

Replaying message (3) An attacker replays message (3) sent earlier by BM_i.

- If BM_j had successfully received the original message (3) from BM_j, it saved the nonce N_j. When BM_j receives the replayed message, it compares the nonce in the replayed message against the saved nonce N_j, and can detect that this is a replayed message because a new message should have a new nonce, not a repeated nonce (rules [R1] and [R2]).

- If BM_j did not receive the original message (3) from $BMAP_i$, BM_j may accept the replayed message and consider it as a valid message (assuming that BM_j receives the replayed messages before it times out on the lost message). Since BM_j authenticates BM_i as its neighbor based on the BM_i's BMAP certificate T_{BM_i} and BM_i's signature $Sig_j(N_j, T_{BM_i})$, BM_j can confirm that message (3) was originated from BM_i, not the attacker. Thus, the attacker cannot be authenticated by BM_j to become a neighbor by simply replaying BM_i's message (3).

4.3.2.5.2 The key distribution to neighbors (KDN) protocol (Figure 4.7)

A local BMAP needs to distribute a share key to the adjacent BMAPs in advance of a client's handover via the KDN protocol. The attacker may replay the message of the

KDN protocol. We discuss how the KDN protocol resists replay attacks.

$BM_i \longrightarrow *:$	$\{I_{BM_1}, E_{P_1}(\beta)\}, \{I_{BM_2}, E_{P_2}(\beta)\}, \ldots, \{I_{BM_n}, E_{P_n}(\beta)\},$
	$Sig_i(\{I_{BM_1}, E_{P_1}(\beta)\}, \{I_{BM_2}, E_{P_2}(\beta)\}, \ldots, \{I_{BM_n}, E_{P_n}(\beta)\})$
	where $\beta = \{I_{BM_i}, I_C, N_i, CMK\}$

- If an adjacent BMAP had successfully received the original message from BM_i, the adjacent BMAP will drop the message because a new message should contain a new nonce, not a repeated one (rules [R1] and [R2]).

- If the adjacent BMAP did not receive the original message from BM_i, the adjacent BMAP will consider the replayed message as valid (if the timer associated with the replayed message has not expired). Digital signatures are used in the message to prevent message tampering. By verifying BM_i's digital signature, an adjacent BMAP can confirm that the message was originated from BM_i, not from the attacker. By decrypting the message β, the adjacent BMAP can further confirm that the message was indeed generated by BM_i as BM_i's identity I_{BM_i} is included in the encrypted message β.

4.3.2.5.3 The inter-network handover authentication protocol (IHAP) (Figure 4.8)

When a mobile client C moves into contact with an adjacent BMAP BMj, C and BM_j execute the IHAP to mutually authenticate each other. We discuss how each message in the IHAP resists replay attacks.

(1) $C \longrightarrow BM_j : \lambda_C, N_C, V_{CMK}(N_C)$

(2) $BM_j \longrightarrow C : \Theta_C, N_B, V_{CMK}(N_C, N_B)$

(3) $C \longrightarrow BM_j : N_B, V_{CMK}(N_B)$

Replaying Messages (1) An attacker overhears and replays message (1) sent earlier by client C.

- If BM_j had successfully received the original message (1) from C, it saved the nonce N_C. When BM_j receives the replayed message, it compares the nonce in the message against the saved nonces, and can detect that this is a replayed message because a new message is supposed to contain a newly generated nonce (rules [R1] and [R2]).

- If BM_j did not receive the original message (1) from C, it may accept the replayed message as a valid message and reply with a message (2). However, the attacker will not be able to compute the correct MAC value $V_{CMK}(N_B)$ in message (3) if it does not know the key CMK, and thus fails the authentication in message (3).

Replaying message (2)

- If the client had successfully received the original message (2) from BM_j earlier, it can detect that this is a replayed message as a new message should contain new nonces, not repeated ones.

- If the client did not receive the original message (2), it may accept the replayed

message as a valid message and reply with a message (3) (before it times out on the lost message). After C receives the replayed message (2), it computes a MAC value $V'_{CMK}(N_C, N_B)$, using CMK (a shared key between C and BM_j). If $V'_{CMK}(N_C, N_B) = V_{CMK}(N_C, N_B)$, C confirms that message (2) is originated from BM_j, not from the attacker. The protocol prevents the attacker from impersonating BM_j if the attacker does not know the key CMK shared only between C and BM_j. Thus, client C has successfully authenticated BM_j, who has the knowledge of the shared key CMK, not the attacker.

Replaying message (3) An attacker may replay message (3) sent by client C.

- If BM_j had successfully received the original message (3) from C earlier, it can detect that this is a replayed message because it had seen nonce N_B sent by C before.

- If BM_j did not receive the original message (3) from C, it may accept the replayed message as a valid message (assuming that BM_j receives the replayed message before it times out on the lost message). Since only C and BM_j know the shared key CMK, by computing the MAC value $V'_{CMK}(N_B)$ using CMK and comparing it with the one in the message (3), BM_j can confirm that message (3) is originated from C, not from the attacker. Thus, BM_j has successfully authenticated client C as C is the only client who knows CMK.

4.3.2.6 Denial-of-Service (DoS) Attack

An attacker may send bogus messages or replay past valid messages repeatedly to force a BMAP to spend resources on processing a large amount of DoS attack messages. To combat DoS attacks, our proposed protocols rely on the security features and rules stated in Section 4.3.1.

We use the IHAP as an example to discuss how DoS attacks are combated. An attacker may repeatedly send copies of message (1) to BM_j. BM_j interprets the duplicates of this message as the losses of messages (2) it has sent. BM_j then stops the authentication procedure after a pre-determined number of failed attempts according to rule [R3] to save resources. An attacker may sniff a valid message (3) from a successful authentication and send the message repeatedly to the involved BM_j in order to overwhelm it. When BM_j receives the replayed message, it compares the nonce in the message against the saved nonces, and can detect that this is a replayed message because a new message is supposed to contain new nonce (rules [R1], [R2] and [R3]). If the BMAP receives the same replayed message several times, it can assume that it is under a DoS attack and take appropriate actions to thwart the attack [78, 79, 80, 81, 82, 83, 85, 86, 87]. Note that an attacker may flood a BM_j with bogus copies of message (3) that the attacker creates itself, but those bogus messages can be detected by BM_j because the attacker cannot produce a valid MAC value if the attacker does not have the knowledge of the MAC key shared only by C, BM_i and BM_j. After processing a number of such bogus messages, the BM_j can assume that it is under a DoS attack and can take appropriate actions to prevent the

attack [78, 79, 80, 81, 82, 83, 85, 86, 87].

Similarly, rules [R1], [R2], and [R3] are also applied to the ABD and the KDN protocols to combat the DoS attacks. In addition, digital signatures are used in the ABD and the KDN protocols to protect against forgery and unauthorized modifications. An attacker cannot generate a valid message if the attacker does not know the sender's private key to generate the sender's digital signature.

4.4 Performance Evaluation

We evaluate the performance of the proposed protocols, especially the efficiency of the key distribution and inter-network authentication protocols discussed in Section 4.2.2, using both numerical analysis and simulations. We use QualNet (version 5.2), a commercial software that provides scalable simulations of wireless networks [102], for our experiments.

We compare our inter-network handover authentication protocol (IHAP) with the adaptive protocol for authentication and key agreement (AP-AKA)[50]. The AP-AKA is a representative of existing inter-network handover authentication protocols. The AP-AKA and existing authentication protocols for inter-network handovers require multi-hop wireless communications between a mobile client and a foreign network's authentication server, which may result in long delay and potential service interruptions. Our proposed IHAP does not require multi-hop wireless communications for an inter-network handover authentication between a client and a foreign network. Instead, a BMAP can directly authenticate a mobile client through one-hop communication by verifying the client's valid

inter-network transfer certificate. A detailed description of the AP-AKA is provided in Appendix B.

4.4.1 Numerical Analysis

We evaluate the latencies of the adjacent BMAP discovery (ABD) protocol, the key distribution to neighbor (KDN) protocol, the IHAP and the AP-AKA via numerical analysis.

The performance of the protocols is measured in terms of

- *communication cost*, which indicates the number of messages exchanged between the entities involved in a protocol.
- *computation cost*, which is the latency (in milliseconds) incurred by the following security operations: encryption using a public key (E_{pub}); decryption using a pubic key (D_{pub}); encryption using a shared key K (E_K); decryption using a shared key K (D_K); generation of a digital signature (G_{sig}); verification of a digital signature (V_{sig}); and computation/verification of a message authentication code (MAC).

4.4.1.1 The ABD Protocol

Two adjacent BMAPs execute the adjacent BMAP discovery (ABD) protocol to identify each other as trusted neighbors. Table 4.2 shows the computation and communication costs of the ABD protocol. The eliptic curve digital signature algorithm (ECDSA) [100] is used for generating and verifying digital signatures. The computational costs of one

G_{sig} operation and one V_{sig} operation are 11.6 ms and 17.2 ms [100], respectively. These values were obtained from a personal digital assistant (PDA) platform with an Intel Xscale 400MHz CPU, 64MB synchronous dynamic random access memory (SDRAM) and 32MB flash read-only memory (ROM) using the open-source Crypto++ library [100].

In the message (2) and (3) of the ABD protocol, a sender needs to generate a signature for the message it sends. Thus, there are two G_{sig} operations in the ABD protocol. Message (2) of the ABD protocol contains BM_j's BMAP certificate T_{BM_j}, where $T_{BM_j} = \{\Gamma, I_{BM_j}, I_A, \tau_{exp}, P_{BM_j}, Sig_A\}$. Thus, BM_i needs to verify two signatures, one for the signature of message (2) $Sig_j(N_i, N_j, T_{BM_j})$, and the other for the certificate agent's signature Sig_A. Similarly, message (3) contains two signatures that require verifications. Thus, a total of four V_{sig} operations are required in the ABD protocol. The total computation cost of the ABD protocol is 92 ms, based on the data provided in Table 4.2 and incurred by two G_{sig} operations and four V_{sig} operations.

The numbers of messages exchanged in the ABD protocol is three. The total communication cost is $3d$, where d is the average delay of a one-hop transmission incurred by a message. Thus, the total latency of the ABD protocol is $92 + 3d$ as shown in the last row of Table 4.2.

We run a Qualnet experiment to obtain the average delay d of a one-hop transmission, which is 10.2ms. The average delay d is defined as the time interval between a sender's transmission of a 128-byte message and the receipt of the message by the intended receiver in the Qualnet simulation. We use IEEE 802.11a physical layer protocol for the

simulation. The transmission rate is 54 $Mbits/s$ at the physical layer. The transmission range of each node is 315m. The distance between a sender and a receiver is 150m. A sender sends a 128-byte message to a receiver every five seconds, for a total 100 messages. We measure the latency of each message and average over 100 messages to obtain the average delay d of a one-hop transmission.

Thus, the total latency of the ABD protocol is 122.6 ms ($92+3d$, where $d = 10.2ms$). Since the ABD protocol is executed offline at the network initialization stage, its latency does not affect the online efficiency of a mobile client's inter-network handovers.

Table 4.2: Computation and communication costs of the ABD protocol

Operation	Algorithm	Time (ms)	ABD
G_{sig}	ECDSA [100]	11.6	2
V_{sig}	ECDSA	17.2	4
Total computation cost (ms)			92
Number of messages			3
Latency (ms)			$92 + 3d$

4.4.1.2 Key Distribution to Neighbors (KDN) Protocol

In the key distribution stage, a local BMAP distributes a shared key CMK to its adjacent BMAPs via the KDN protocol (Figure 4.7). The key distribution latency consists of computation and communication costs.

Table 4.3 shows the computation and communication costs of the KDN protocol. In

the second to eighth rows, the first two columns list the types of operations the KDN protocols perform, and the corresponding algorithm implemented for each operation, respectively.

The third column shows the computational latency each algorithm incurs [91]. These latency values were obtained from a PDA platform with an Intel Xscale 400MHz CPU, 64MB SDRAM and 32MB flash ROM using the open-source Crypto++ library [91].

The fourth column list the number of times each operation is performed in the KDN protocol. In the KDN protocol, The number of E_{pub} operations is n, as there are n adjacent BMAPs. We assume that all adjacent BMAPs perform the decryptions and verify the digital signatures in parallel after receiving the message of the KDN protocol. Therefore, the latency of the KDN protocol is equivalent to the sum of n E_{pub} operations, one D_{pub} operation, one G_{sig} operation, and one V_{sig} operation (assuming parallel decryptions and signature verifications).

By multiplying the latency of each operation (the third column) by the number of times it is executed (the fourth column), and summing up the costs of all operations executed, we obtain the total computation cost of the KDN protocols as shown in the third last row.

The second last row of Table 4.3 lists the number of messages exchanged in the KDN protocols. The last row shows the total latency as the sum of computation and communication costs, where d is the average delay of a one-hop communication, and n is the number of adjacent BMAPs. The average delay $d = 10.2ms$ was obtained via

simulations using Qualnet as discussed in Section 4.4.1.1.

Figure 4.11 shows the key distribution latency as a function of the number of adjacent BMAPs. As the local BMAP needs to prepare for the client's inter-network handover authentication, the local BMAP executes the KDN protocol to securely distribute a shared key CMK to its adjacent BMAPs.

Since a client may move in any direction, adjacent BMAPs need to receive the shared key CMK before authenticating the client through the IHAP. Thus, the larger the number of adjacent BMAPs, the longer the latency of the KDN protocol. We observe that the key distribution latency increases linearly when the number of the adjacent BMAPs increases from one to eight. To optimize the performance of inter-network handover authentication, the number of adjacent BMAPs should be carefully considered. Too many adjacent BMAPs may increase the key distribution latency, but too few adjacent BMAPs may limit a client's mobility and cause service disruptions.

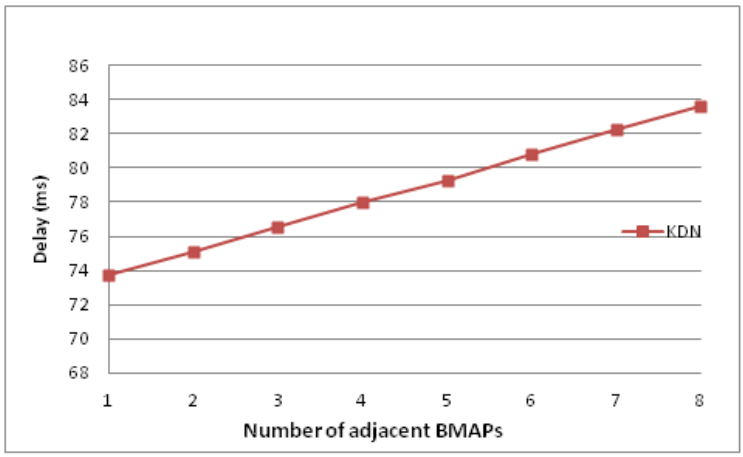


Figure 4.11: Key distribution delay

Table 4.3: Computation and communication costs of key distribution

Operation	Algorithm	Time (ms)	KDN (section 4.2.2.2)
E_{pub}	RSA [99]	1.42	n
D_{pub}	RSA	33.3	1
G_{sig}	ECDSA [100]	11.6	1
V_{sig}	ECDSA	17.2	1
Total computation cost (ms)			$1.42n + 62.1$
Number of messages			1
Latency (ms)			$1.42n + 62.1 + d$

4.4.1.3 IHAP vs. AP-AKA

Table 4.4 provides a comparison of the inter-network handover authentication latency between the IHAP and the AP-AKA. Both the IHAP and the AP-AKA implement MAC operations, which is a HMAC algorithm. The computation time of the HMAC algorithm is 0.015 ms [74, 91], which was obtained on a PDA platform with an Intel Xscale 400MHz CPU, 64MB SDRAM and 32MB flash ROM using the open-source Crypto++ library. The second row lists the number of MAC operations the IHAP and the AP-AKA execute. The total computation costs of the IHAP and the AP-AKA are shown in the third row, obtained by multiplying 0.015 ms by the numbers of MAC operations shown in the second row. The fourth row provides the numbers of messages exchanged in the IHAP and the

AP-AKA. The IHAP requires three messages exchanged between a client and a BMAP. The AP-AKA requires $4+2h$ messages: four messages are exchanged between a client and a BMAP, and $2h$ messages are between the BMAP and the authentication server, where h is the number of hops between the BMAP and the authentication server. The latencies shown in the last row are the summations of the computation costs and communication delays, where $d = 10.2ms$ is the average delay of a one-hop transmission incurred by a message. The average delay d was obtained via simulation using Qualnet as discussed in Section 4.4.1.1.

Table 4.4: Computation and communication costs of authentication

Operation	IHAP (section 4.2.2.3)	AP-AKA [50]
Number of MACs	8	10
Total computation cost (ms)	0.12	0.15
Number of messages	3	$4 + 2h$
Latency (ms)	$0.12 + 3d$	$0.15 + (4 + 2h)d$

Figure 4.12 shows the latencies of the IHAP and the AP-AKA for one client scenario when the number of hops between a BMAP and an authentication server increases from zero to six. In the IHAP, a client and a BMAP directly authenticate each other without the involvement of an authentication server. Thus, the number of hop between the BMAP and the authentication server is zero in the IHAP. If we assume that the BMAP also acts as an authentication server in the AP-AKA (i.e., the number of hops h between a BMAP

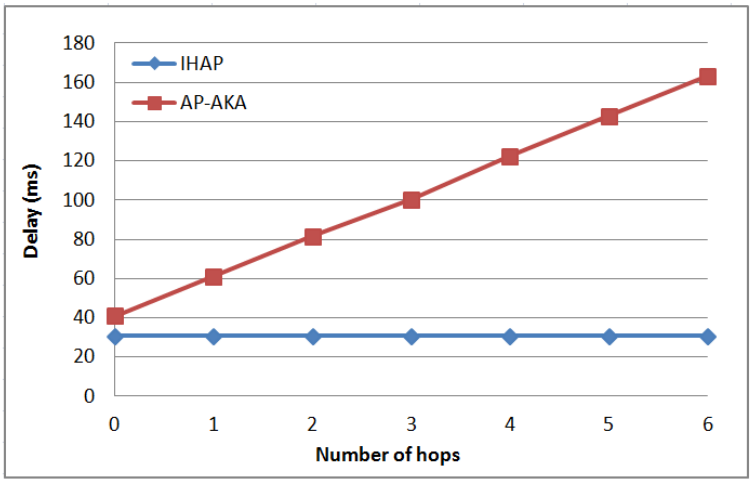


Figure 4.12: IHAP and AP-AKA

and an authentication server is zero), the latency of the IHAP and the AP-AKA is 30.72 ms and 40.95 ms, respectively. The IHAP improves the authentication delay by 10.23 ms or by 33.3% compared to the AP-AKA. However, when the number of hops between the foreign BMAP and the authentication server increases from zero to six, the AP-AKA requires multi-hop wireless communications, which results in much higher authentication latency compared with one-hop wireless communications using the IHAP. For example, when $h = 6$, the delay incurred by the AP-AKA is 163.35 ms or 5.3 times longer than that by the IHAP (30.72 ms).

4.4.2 Simulation Results

We further evaluate and compare the performance of the key distribution, the IHAP and the AP-AKA protocols under realistic network settings using simulations in Qualnet

version 5.2. (We did not simulate the adjacent BMAP discovery (ABD) protocol as it is executed offline at the network initialization stage. The performance of the ABD protocol does not affect the online efficiency of a mobile client's inter-network handovers.)

4.4.2.1 Performance Metric

We use the following performance metrics:

(a) *key distribution delay* (latency), which is defined as the time when a local BMAP's transmission of a MAC key and the receipt of the MAC key by all adjacent BMAPs. The local BMAP broadcasts the MAC key to its adjacent BMAPs through the KDN protocol. We calculate the *average key distribution delay* (AKDD), averaged over all clients and BMAPs participating in the experiment. We also keep track of the *maximum key distribution delay* (MKDD), the maximum value among all clients and BMAPs.

(b) *message delivery ratio of the key distribution protocols*, which is defined as the ratio of successfully received messages over all transmitted key distribution messages. Since the local BMAP broadcasts the shared key CMK to its adjacent BMAPs through the KDN protocol, for each client's shared key CMK, a total of N_{BMAP} messages are delivered to the adjacent BMAPs, where N_{BMAP} is the number of adjacent BMAPs. In this experiment, we recorded the number of messages that were lost during the key distribution stage. We calculate the message delivery ratio of key distribution as follows:

$\psi = \frac{N_{BMAP} \times N_C - N_{lost}}{N_{BMAP} \times N_C}$,

where ψ denotes the message delivery ratio of the key distribution protocols, N_{BMAP}, N_C and N_{lost} denote the number of adjacent BMAPs, the number of mobile clients, the number of lost messages, respectively.

Each experiment was run ten times with different random seeds and we averaged the message delivery ratio over ten runs.

(c) *authentication delay* (latency), which is the time interval between a client's transmission of an authentication request to an adjacent BMAP (message (1) of the IHAP shown in Figure 4.8) and the receipt of an acceptance confirmation (message (3) of the IHAP shown in Figure 4.8). We calculate the *average inter-network authentication delay* (AIAD), averaged over all mobile clients participating in the experiment. We also keep track of the *maximum inter-network authentication delay* (MIAD), the maximum value observed by all mobile clients.

4.4.2.2 Simulation Parameters

In all experiments, one local BMAP is placed in the center of a 600m x 600m area. We consider 3 scenarios: the local BMAP has one, four and eight adjacent BMAPs as shown in Figure 4.13(a), Figure 4.13(b), and Figure 4.13(c), respectively. The transmission range of the wireless routers is 315 m, according to the specifications of wireless routers manufactured by Tropos [103]. The transmission range of mesh clients is 304 m, according to the specifications of wireless adapters manufactured by Cisco [105]. The transmission rate

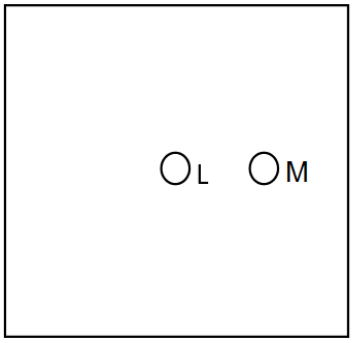
at the physical layer is 54 $Mbits/s$ assuming IEEE 802.11a. The mobility speed of mobile clients is 10 m/s and the mobility pattern follows the random waypoint model [106]. Each data point in the graphs is the average of 10 runs using different random seeds. In each run, the simulated time is 150s. The graphs are plotted with a confidence interval of 95%. The common simulation parameters for all experiments are listed in Table 4.5.

Table 4.5: Common simulation parameters

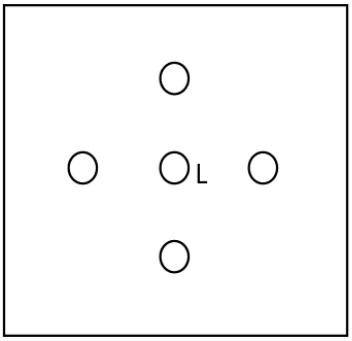
Parameter	Value
Movement model	Random way point
Speed	10 m/s
Propagation fading model	None
Transmission range of MAPs	315 m
Transmission range of mesh clients	304 m
Transmission rate at physical layer	54 $Mbits/s$
Physical layer protocol	PHY802.11a
Number of runs per data point	10
Confidence interval	95%
Simulation time	150s

4.4.2.2.1 Performance of the Key Distribution

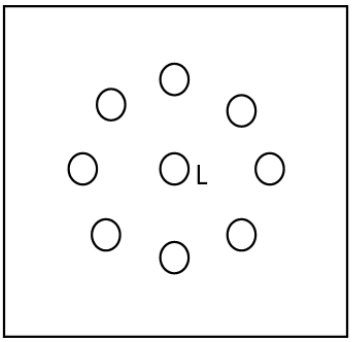
We conducted three sets of experiments for the key distribution protocols by varying:



(a) A local BMAP L with one adjacent BMAP



(b) A local BMAP L with four adjacent BMAPs



(c) A local BMAP L with eight adjacent BMAPs

Figure 4.13: Topologies of BMAPs

(1) *the number of adjacent BMAPs.* We measured the *average key distribution delay* (AKDD), the *maximum key distribution delay* (MKDD), and the message delivery ratio of the key distribution protocols as a function of the number of adjacent BMAPs. The number of adjacent BMAPs varied from one to eight. We conducted experiments for two scenarios: 10 and 50 clients.

(2) *background traffic load.* Since the overall load of a local BMAP directly affects the key distribution latency, we measured the AKDD, the MKDD and the message delivery ratio of the key distribution protocols in the presence of background traffic to the BMAP. In this experiment, an additional MAP is placed in the network as a source to transmit background traffic to the local BMAP at a specific constant bit rate (CBR). We vary the background traffic rate from 0 to 50 Mbits/s. There is no background traffic when the data rate is 0. We simulated a topology with four adjacent BMAPs as shown in Figure 4.13(b), and two scenarios: 10 and 50 clients.

(3) *the number of clients.* We measured the AKDD, the MKDD, and the message delivery ratio of the key distribution protocols as functions of the number of clients. In each experiment, we used three BMAP topologies as shown in Figure 4.13(a), Figure 4.13(b), and Figure 4.13(c). All clients moved at the same speed of 10m/s. The number of clients varied from 10 to 50.

4.4.2.2.2 Performance of the Authentication Protocols

In this section, we compare the inter-network handover authentication protocol (IHAP) with the AP-AKA. Since an authentication protocol is only executed between a client and a BMAP, we used the BMAP configuration shown in Figure 4.13(a), in which the local BMAP L has one adjacent BMAP M. An authentication server is located six hops away from M. The use of the authentication server helps to illustrate the high overhead of the multi-hop inter-network handover authentication approach used by the AP-AKA.

We conducted two sets of experiments for the authentication protocols by varying:

(1) *background traffic load.* We measured the *average inter-network authentication delay* (AIAD) and *maximum inter-network authentication delay* (MIAD) incurred by each protocol as functions of the background traffic. In this experiment, the BMAP that is executing the IHAP or the AP-AKA is also receiving data (background traffic) from another source (not shown in Figure 4.13(a)). The background traffic is transmitted at a specific constant bit rate (CBR). We vary the background traffic rate from 0 to 50 Mbits/s. A data rate of 0 implies no background traffic. We conducted the experiment for two scenarios: 10 and 50 clients.

(2) *the number of clients.* We compared the IHAP with the AP-AKA in terms of the *average inter-network authentication delay* (AIAD) and the *maximum inter-network authentication delay* (MIAD). The number of clients varied from 10 to 50. All clients moved at the same speed of 10m/s.

The simulation parameters specific to each experiment are summarized in Table 4.6. In all experiments, the mobile clients were randomly placed in the networks. To test the scalability of the protocols, we let all clients present in the network send key distribution or authentication requests to their respective nearby BMAPs simultaneously (i.e., the worst case scenario for the BMAPs).

4.4.2.3 Result Analysis

The results for the performance of the key distribution protocols are illustrated in Figures 4.14, 4.15 and 4.16. The results for the performance of the authentication protocols are illustrated in Figures 4.17 and 4.18.

4.4.2.3.1 Performance of the Key Distribution

Experiment (1): Function of the number of adjacent BMAPs

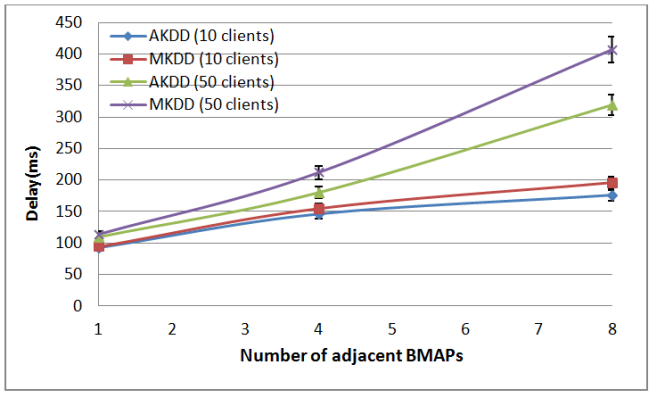
The performance of the key distribution as a function of the number of adjacent BMAPs is given in Figure 4.14.

- *Key distribution delay*

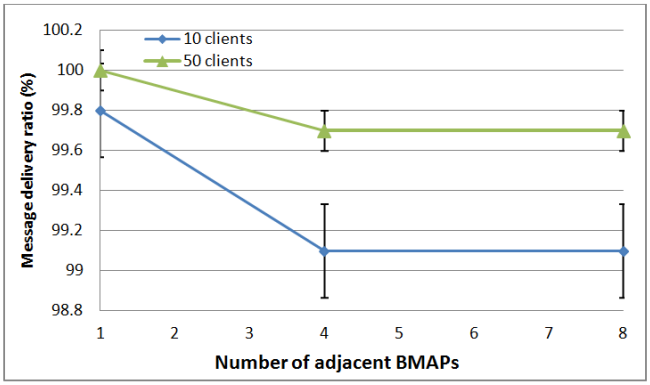
  Figure 4.14(a) shows that the higher number of adjacent BMAPs, the higher the AKDD and the MKDD. For example, the AKDD in the case of eight adjacent BMAPs is 1.9 times (3.6 times) higher than that of one adjacent BMAP for 10 clients (50 clients). A larger number of adjacent BMAPs implies more key distribution messages to be transmitted by the local BMAP and

Table 4.6: Simulation parameters for different experiments

Experiment		Figure	Network	Clients
Performance of KDN	(1) Function of number of adjacent BMAPs	Figure 4.14(a), key distribution delay	A local BMAP has 1, 4, 8 adjacent BMAPs	10 clients, 50 clients
		Figure 4.14(b), message delivery ratio of the key distribution protocols		
	(2) Function of background traffic load	Figure 4.15(a), key distribution delay	A local BMAP has 4 adjacent BMAPs. Background traffic rate: 0-50Mbits/s	10 clients, 50 clients
		Figure 4.15(b), message delivery ratio of the key distribution protocols		
	(3) Function of number of clients	Figure 4.16(a), key distribution delay	A local BMAP has 1 adjacent BMAP	10 to 50 clients
		Figure 4.16(b), message delivery ratio of the key distribution protocols		
Performance of IHAP	(1) Function of background traffic load	Figure 4.17, IHAP vs. AP-AKA	A local BMAP has 1 adjacent BMAP. An authentication server is located six hops away from each BMAP. Background traffic rate: 0-50Mbits/s	10 clients, 50 clients
	(2) Function of number of clients	Figure 4.18, IHAP vs. AP-AKA	A local BMAP has 1 adjacent BMAP. An authentication server is located six hops away from each BMAP.	10 to 50 clients



(a) Key distribution delay



(b) Message delivery ratio of the key distribution protocols

Figure 4.14: Function of number of adjacent BMAPs

adjacent BMAPs, resulting in longer key distribution delay.

The key distribution delay should be minimized so that an adjacent BMAP could receive a shared key before the client associated with the key connects to it. In the worst case scenario, the client and an adjacent BMAP may have to execute the login authentication protocol (LAP) if the adjacent BMAP has not received the shared key in advance.

In a WMN deployed by Cisco, typical MAP-to-MAP distances are 500 feet (152.4m) to 1000 feet (304.8m) [107]. Given a mobile client moves at 60 km/hour (16.7 m/s) (e.g., a car running in the city), it takes 9.1 second to 18.3 second to travel from one MAP to another given a distance of 500 to 1000 feet. If a mobile client moves at 100 km/hour (27.8 m/s) (e.g., a car running on highways), it takes 5.5 seconds to 11 seconds to cross a distance of 500 to 1000 feet between two MAPs.

In Figure 4.14(a), as the number of the adjacent BMAPs increases from one to eight, the maximum latency of our key distributions for 50 clients ranges from 113.8ms to 406.9ms. The longest latency of our key distributions is much less than the shortest time for a car crossing a 500 feet distance at 100 km/hour, 406.9ms vs. 5.5 seconds. Thus, our key distribution protocols allow a BMAP to distribute a shared key to its adjacent BMAPs long before the client associated with the key connects to one of the adjacent BMAPs.

- *Message delivery ratio of the key distribution protocols*

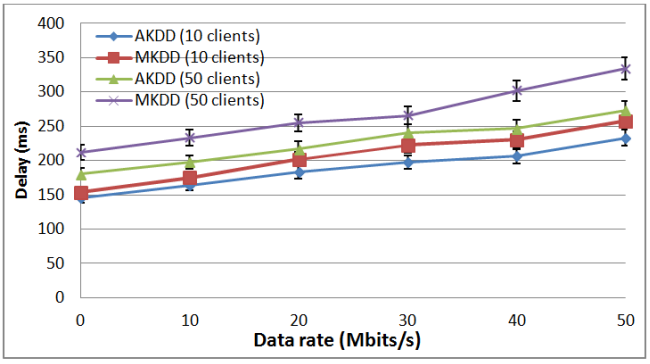
Figure 4.14(b) shows the message delivery ratio of the key distribution protocols as a function of the number of adjacent BMAPs. We observe that the number of adjacent BMAPs does not have a big impact on the message delivery ratio of the key distribution protocols. When the number of adjacent BMAPs increases from one to eight, the message delivery ratio for 10 clients are 100% and 99.1%, respectively, a difference of 0.9%. For the 50 clients, the message delivery ratio changes from 99.8% to 99.7% when the number of adjacent BMAPs increases from one to eight, a difference of 0.1%.

In summary, the more adjacent BMAPs, the higher the key distribution latency. However, too few adjacent BMAPs may limit the client's mobility for inter-network handovers. In practice, the optional number of adjacent BMAPs depends on several factors such as the number of adjacent networks, the size of each network, and the transmission range of an BMAP.

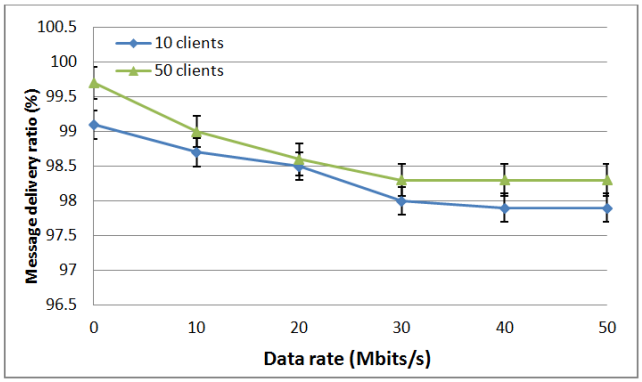
Experiment (2): Function of background traffic load

In this set of experiments, we examine how background traffic load may affect the performance of the key distribution protocols.

An additional MAP was placed in the network as a source to transmit background traffic to the local BMAP at a specific constant bit rate (CBR). We varied the data rate of CBR from 0 to 50 Mbits/s. We simulated a topology with four adjacent BMAPs as shown in Figure 4.13(b).



(a) Key distribution delay



(b) Message delivery ratio of the key distribution protocols

Figure 4.15: Function of background traffic load

- *Key distribution delay*

  The graph in Figure 4.15(a) shows the AKDD and the MKDD as functions of background traffic for both scenarios, 10 and 50 clients. As the background traffic load increases, the AKDD and the MKDD increase as expected. The AKDD with 50Mbits/s background traffic for 10 (50) clients is 41.8% (38.5%), higher than the AKDD with 10 Mbits/s background traffic.

  A larger number of clients also results in longer AKDD's and MKDD's. When there is no background traffic, the AKDD (MKDD) of 50 clients is 24.1% (37.2%) higher than that of 10 clients. More clients imply more requests to be processed by the local BMAP, and more channel contention around the local BMAP, resulting in longer key distribution delay.

  In Figure 4.15(a), given the heaviest background traffic of 50 Mbit/s, the highest latency of key distributions for 50 clients is 333.9ms. The latency is much shorter than 5.5 seconds, the shortest time for a car crossing a 500 feet distance between two BMAPs at 100 km/hour. Thus, even with the heaviest background traffic of 50 Mbit/s, a BMAP is able to send a shared key to its adjacent BMAPs before the client associated with the key moves into contact with one of the adjacent BMAPs.

- *Message delivery ratio*

  Figure 4.15(b) shows the impact of the background traffic load on the message delivery ratio of the key distribution protocols. As the data rate of the

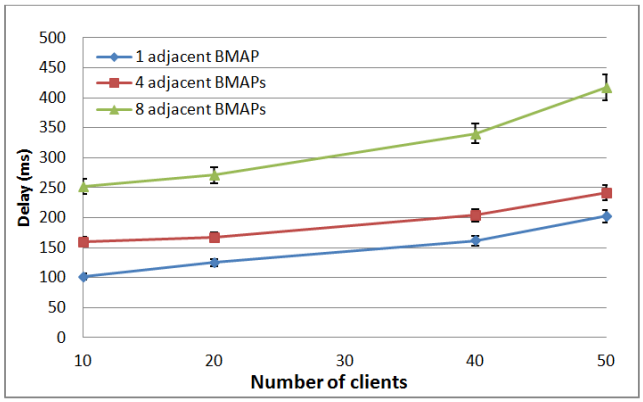
background traffic increases, the message delivery ratio decreases. When the background traffic increases from 0 to 50 Mbits/s, the message delivery ratio for 10 (50) clients decreases from 99.1% (99.7%) to 97.9% (98.3%).

In summary, the latency of key distributions increases as expected when the background traffic load increases. The reason is the additional load imposed on the local BMAP by the background traffic. However, in all simulation scenarios with light to heavy traffic, the latency of key distribution is sufficiently short for shared keys to be received by adjacent BMAPs before clients associated with the keys make contact with the BMAPs.

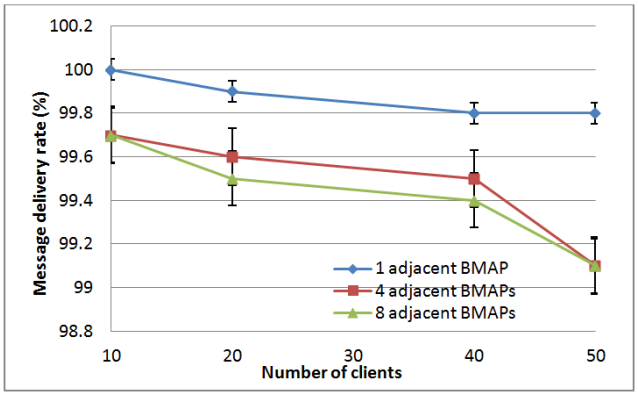
Experiment (3): Function of number of clients

- *Key distribution delay*

  The performance of the key distribution as a function of the number of clients is shown in Figure 4.16(a). In this experiment, we simulated three topologies with one, four and eight adjacent BMAPs as shown in Figure 4.13. As the number of clients increases from 10 to 50, the AKDD ranges from 101.2ms to 202.2ms, from 159.3ms to 241.7ms, and from 251.8ms to 417.2ms for the network of one, four and eight adjacent BMAPs, respectively. All these key distribution latencies are much shorter than 5.5 seconds, which is approximate the time for a car to travel 500 feet from one BMAP to another at 100 km/hour. In all three networks, the AKDDs of 10 clients are much lower than those



(a) Key distribution delay



(b) Message delivery ratio of the key distribution protocols

Figure 4.16: Function of number of clients

of 50 clients, approximately 49.9%, 34%, and 39.8% lower in the network of one, four and eight BMAPs, respectively. More mobile clients imply more key distribution messages to be processed by the local BMAP, and more channel contention around the local BMAP, resulting in longer key distribution delay. We also observe that a larger number of adjacent BMAPs results in longer AKDD, which is consistence with the simulation results of Experiment (1) shown in Figure 4.14(a).

- *Message delivery ratio of the key distribution protocols*

  In this experiment, we examine how the number of clients affects the message delivery ratio of the key distribution protocols. Figure 4.16(b) shows the message delivery ratio of the key distribution protocols as a function of the number of clients. We implemented three networks as above, which have one, four and eight BMAPs, respectively.

  In all three networks, the message delivery ratios are in the range of 99.1% to 100%. As the number of clients increases from 10 to 50 in the networks of one, four and eight adjacent BMAPs, the message delivery ratio decreases 0.2%, 0.6%, and 0.6%, respectively. We also observer that a larger number of adjacent BMAPs leads to a lower message delivery ratio.

In summary, the key distribution latency is affected by all three factors, the number of adjacent BMAPs, background traffic load and the number of clients. An adjacent BMAP should receive a shared key in advance before authenticating a mobile client,

so that the key distribution delay will not increase the inter-network authentication latency. The simulation results show that our proposed key pre-distribution protocols are effective for use in real-world wireless mesh networks.

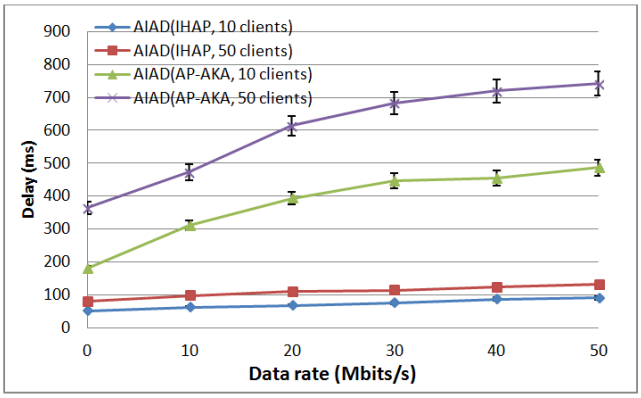
4.4.2.3.2 Performance of the Authentication Protocols: IHAP vs. AP-AKA

Experiment (1): Function of background traffic load

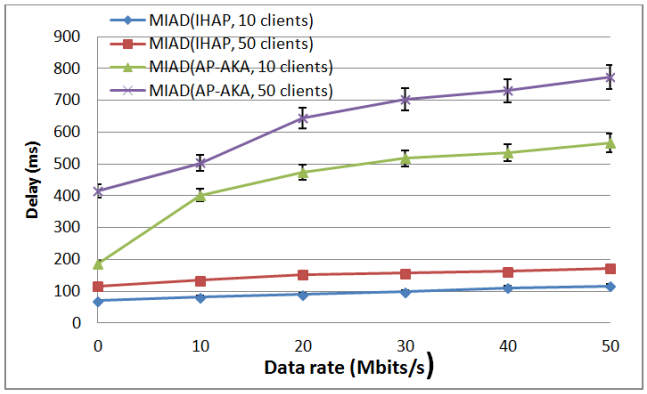
In this set of experiments, we compare the IHAP with the AP-AKA and examine how background traffic may affect the inter-network handover authentication.

In this experiment, an additional MAP was placed in the network as a source to transmit background traffic to the BMAPs at a specific constant bit rate (CBR). We vary the CBR of background traffic from 0 to 50 Mbits/s. An authentication server is located six hops away from a BMAP. The use of the authentication server helps to illustrate the high overhead of the multi-hop wireless inter-network handover authentication approach used by the AP-AKA.

Figure 4.17 shows the *average inter-network authentication delay* (AIAD) and the *maximum inter-network authentication delay* (MIAD) of the IHAP and the AP-AKA as functions of background traffic load. We observe that the background traffic has less impact on the IHAP compared to the AP-AKA. For example, when the number of clients is 10 and the background traffic rate increases from 0 to 50 Mbit/s, the AIAD (MIAD) of the IHAP increases approximately 1.75 (1.66) times while the AIAD (MIAD) of the AP-AKA increases 2.71 (3.04) times.



(a) IHAP vs. AP-AKA - AIAD



(b) IHAP vs. AP-AKA - MIAD

Figure 4.17: Function of background traffic load

The IHAP outperforms the AP-AKA by a large margin in terms of the AIAD and the MIAD. For example, when the background traffic rate is 50Mbits/s, the AIAD of the AP-AKA for 50 clients is approximately six times higher than that of the IHAP. This outperformance is the result of one-hop communications between the client and the adjacent BMAP during an inter-network handover authentication versus multi-hop communication between the client and the authentication server done by the AP-AKA. Moreover, the AIAD and the MIAD of the IHAP is much lower than that of the AP-AKA due to a reduction in the number of messages exchanged, 3 vs. $4 + 2h$ (see the second last row of Table 4.4).

Compared with the AP-AKA, the IHAP offers significantly lower AIAD and MIAD in both 10-client and 50-client scenarios, thanks to one-hop communication between the client and the adjacent BAMP, and a reduction in the number of messages exchanged in the IHAP.

Experiment (2): Function of the number of clients

In this set of experiments, we use the same network settings as above except there is no background traffic transmitted to the adjacent BMAP. We vary the number of clients from 10 to 50. We compare the IHAP with the AP-AKA in terms of the AIAD and the MIAD for inter-network handover authentications.

The graph in Figure 4.18 shows the performance of the IHAP and the AP-AKA. As the number of clients increases from 10 to 50, the AIAD and the MIAD of both schemes increase as expected. The reason is that more clients implies more

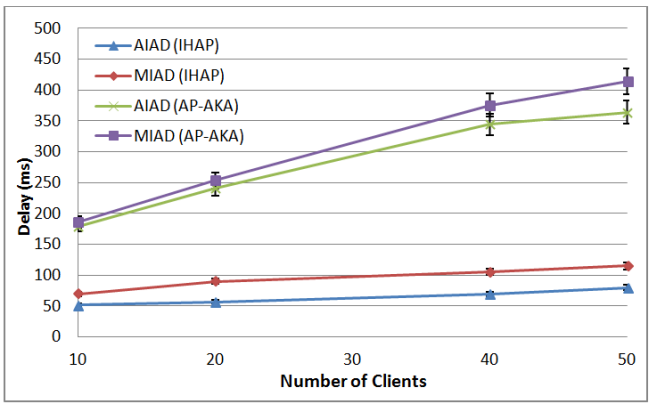


Figure 4.18: Function of number of clients

authentication messages transmitted to the adjacent BMAP, which results in higher workload for and channel contention around the adjacent BMAP.

Given 10 mobile clients connecting to the same BMAP, the AIAD of the IHAP and the AP-AKA are 51.5ms and 179.5ms, respectively. The IHAP improves the inter-network authentication delay by 128ms or 71.3% compared with the AP-AKA. As the number of clients increases, the performance gap between the IHAP and the AP-AKA becomes larger. In the case of 50 clients, the AIAD of the IHAP is 78.03% lower than that of the AP-AKA.

The graph also shows the MIAD of both protocols. The MIAD of the IHAP is about 62.3% lower than the MIAD of the AP-AKA. The amounts of cryptographic computation performed by the IHAP and the AP-AKA are very similar (0.12ms vs. 0.15ms as shown in the third row of Table 4.12). This shows that the gain of the IHAP over the AP-AKA is mainly due to one-hop communication versus multi-hop

communication, and due to the reduction of the number of messages exchanged from $4 + 2h$ (AP-AKA) to 3 (IHAP).

In summary, both the numerical analysis and simulation results confirm that the proposed security protocols outperform existing home-foreign domain solutions in terms of authentication delay. Our proposed authentication framework thus contributes towards the development of a faster inter-network handover process for mobile clients using real-time services in WMNs.

4.5 Chapter Summary

In this chapter, we present a novel inter-network authentication framework to minimize the inter-network authentication delay. The framework includes an inter-network authentication architecture, a trust model, certificates, and key distribution and authentication protocols. A client and a foreign BMAP mutually authenticate each other using the client's inter-network transfer certificate via one-hop communications. The foreign BMAP does not require to communicate with an authentication server through multi-hop wireless communications. This approach allows the authentication delay to be minimized. Security analysis shows that our proposed protocols are resilient to various types of attacks. The numerical analysis and simulation results demonstrate that our security protocols are effective with respect to minimizing authentication delay, and confirm that our proposed solution outperforms the home-foreign domain authentication approach in terms of the delay of authentication among multiple wireless mesh networks.

In the next chapter, we present a new implementation of group key management at the data link layer in order to reduce the key update latency from linear time as currently done in IEEE 802.11 standards to logarithmic time. This contribution is to minimize the latency of the handover process of members in a multicast or broadcast group.

Chapter 5

Efficient Group Key Management for Wireless Mesh Access Points

Real-time applications need very fast rekeying so that changes in group membership are not disruptive to the real-time applications. Group key management (GKM) refers to the actions taken to update and distribute the group key upon a client joining or leaving a multicast group. The scheme defined in the IEEE 802.11s standard for group key management at the data link layer is not scalable, because the rekeying latency grows linearly as a function of the mutlicast/broadcast group size.

To minimize the group key update latency, we propose a new implementation of group key management at the data link layer in mesh access points (MAPs) of a WMN. (We often use the term "data link" instead of "medium access control" in the dissertation because the former is shorter, and the abbreviation "MAC" is used to denote "message authentication code".) Our new GKM implementation is based on the logical key hi-

erarchy (LKH) [13] and one-way function tree (OFT) [14] algorithms. We evaluate the performance of the proposed implementation via numerical analysis and simulations. We present our simulation results obtained from various network conditions under realistic settings. Numerical analysis and simulation results confirm that our proposed implementation of group key management reduces the rekeying latency from linear time (as currently done in IEEE 802.11 standards) to logarithmic time.

In Section 5.1, we present an overview of the LKH and OFT algorithms. In Section 5.2, we describe our new implementation of GKM at the data link layer, which is based on the LKH and OFT algorithms. We analyze the performance of each group key management scheme in Section 5.3. In Section 5.4, we present simulation results and discuss the findings. We summarize the chapter in Section 5.5.

5.1 Overview of LKH and OFT Algorithms

Both the LKH and OFT algorithms use a hierarchical key structure called a *logical key tree* to ensure scalable key updates. In this section, we first describe the structure of a logical key tree, along with definitions and notations to be used in this chapter. We then describe the operations of the LKH and OFT algorithms.

5.1.1 Logical Key Tree

A key tree is a *logical* data structure used in hierarchical GKM schemes. A logical key tree is not to be confused with the physical multicast routing tree of the same group [132],

or the recovery tree used in reliable multicast for retransmissions of lost/damaged data packets [133]. Logical key trees provide scalable computation, maintenance, and updates of multicast group keys.

Logical key trees are generated and maintained by a group key server (GKS). For group key management at the application layer, a GKS is a computing server located in the Internet. A GKS is a trusted entity in charge of generating and maintaining logical key trees for multicast groups in the network it serves. A GKS processes requests from group members, updates group keys, and distributes new keys to members using rekeying messages. The GKS can deliver rekeying messages to members using either unicast or multicast communications.

Given a logical key tree, let K_i denote the content of the key stored in a node i in the key tree. Every leaf node i is associated with a group member C, and contains the member's individual key, which is generated by the GKS and known only to member C and the GKS. We say that "group member C owns node i" or "is associated with node i".

Consider a multicast group having six members C_A, C_B, C_C,C_D,C_E, and C_F whose logical key tree is shown in Figure 5.1. The leaf nodes 6, 7, ..., 11 contain the individual keys $K_6, K_7, \ldots, K_{11}$ of the six group members $C_A, C_B, \ldots, C_F$, respectively. Non-leaf nodes are not associated with any members. The contents stored in non-leaf nodes are keys called *intermediate keys*, which are used to compute the final group key in a scalable manner. In the above example, nodes 2, 3, 4, 5 store intermediate keys.

The content of the root, node 1, is the *group key* denoted by K_1. The group key is used by the source of the group to encrypt data packets before sending them to the group members, and by the group members to decrypt the encrypted packets. The group key K_1 needs to be updated when a member joins (leaves) the group to ensure forward (backward) secrecy. It also needs to be refreshed periodically even when there are no membership changes to prevent an attacker from gathering sufficient time or resources to compute the group key.

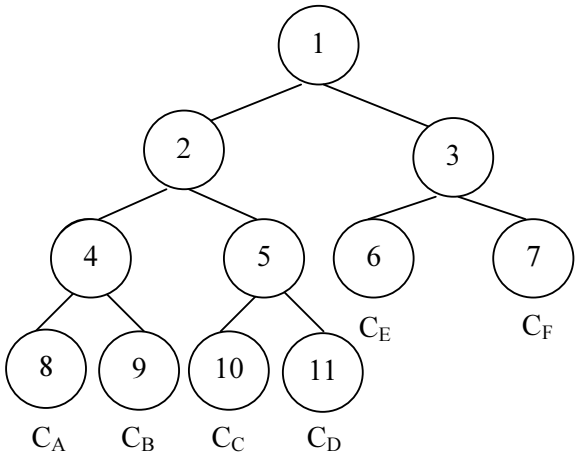


Figure 5.1: A logical key tree

To simplify the discussions in this article, we assume logical key tress of the binary form, although trees of higher degrees can be used with LKH, as discussed in [13]. Furthermore, binary key tress used in OFT must be proper binary trees [14]; that is, every non-leaf node must have exactly two children.

Given a proper binary tree, we use the following notation to identify the tree nodes. The root of the tree has index 1. Given a non-leaf node with index i, the left child of node

i has index $2i$, and right child of node i has index $2i + 1$. To simplify the discussions, we assume that a logical key tree is a complete binary tree: A binary tree T with n levels is complete if all levels except possibly the last. At the last level, all nodes must be as far left as possible [112]. That is, nodes in a complete binary tree are added to the key tree from top to bottom and left to right. Nodes are removed from the key tree from right to left and bottom to top. Given a group with n members and a complete binary tree, the height of the key tree is thus $h = \lceil \log_2 n \rceil$, and the height of the root is $h_{root} = \lceil \log_2 n \rceil - 1$.

Following are the notations used in this chapter:

- K_i is the current content of node i and K_i' is a new key generated/computed to replace K_i when there is a membership change which requires the key tree to be updated.
- The notation $\{X\}_Y$ denotes the encryption of content X using key Y.
- $sib(i)$ denotes the sibling node of node i.
  - If i is an even number, $sib(i) = i + 1$
  - If i is an odd number, $sib(i) = i - 1$

When we say "node i performs an action," we mean that "the group member owning (associated with) node i performs the action."

Following is a high-level description of how the structure of a logical key tree is updated when a member joins or leaves the multicast group.

5.1.1.1 Join

After receiving a join request from a client, the GKS first identifies the *joining point*. If the current key tree is full, the joining point is the leftmost leaf node located at height 0 − the lowest level of the tree. If the key tree is not full, the joining point is the leftmost leaf node located at height 1 of the key tree.

Let i be the index of the joining point. The GKS creates two new leaf nodes and makes node i become the parent of the new nodes. The group member currently owning node i will no longer be associated with node i, but rather with node $2i$. With this updated association, the GKS will assign the previous key K_i as key K_{2i} to the member now owning node $2i$, i.e. though renamed the member will retain its previous key and key contents. The new member joining the group will be associated with node $2i+1$ as a sibling of node $2i$. The GKS will generate a new key K_{2i+1} for the newly joining member. Node i will become a non-leaf node storing an intermediate key. The GKS generates an intermediate key K_i' for node i. All the intermediate keys along the path from the joining point i to the root and the group key at the root will be updated, as will be discussed in Sections 5.1.2.1 and 5.1.3.2.

Consider an example illustrated by Figure 5.2 in which client C_H joins the group by sending a join request to the GKS. Upon receiving the join request, the GKS determines the joining point as node 7, which is currently owned by member C_G. The GKS creates two new leaf nodes of node 7; node 14 and node 15 respectively. At this point, client C_H joins the group and is associated with node 15. *After the join operation, member* C_G

will change its ownership from node 7 to node 14. To prevent the newly joined client C_H from accessing past messages, all keys along the path from the joining point node 7 to the root node 1 need to be changed. As shown in Figure 5.2, the group key server updates the key K_7, K_3 and K_1 to K_7', K_3' and K_1', respectively.

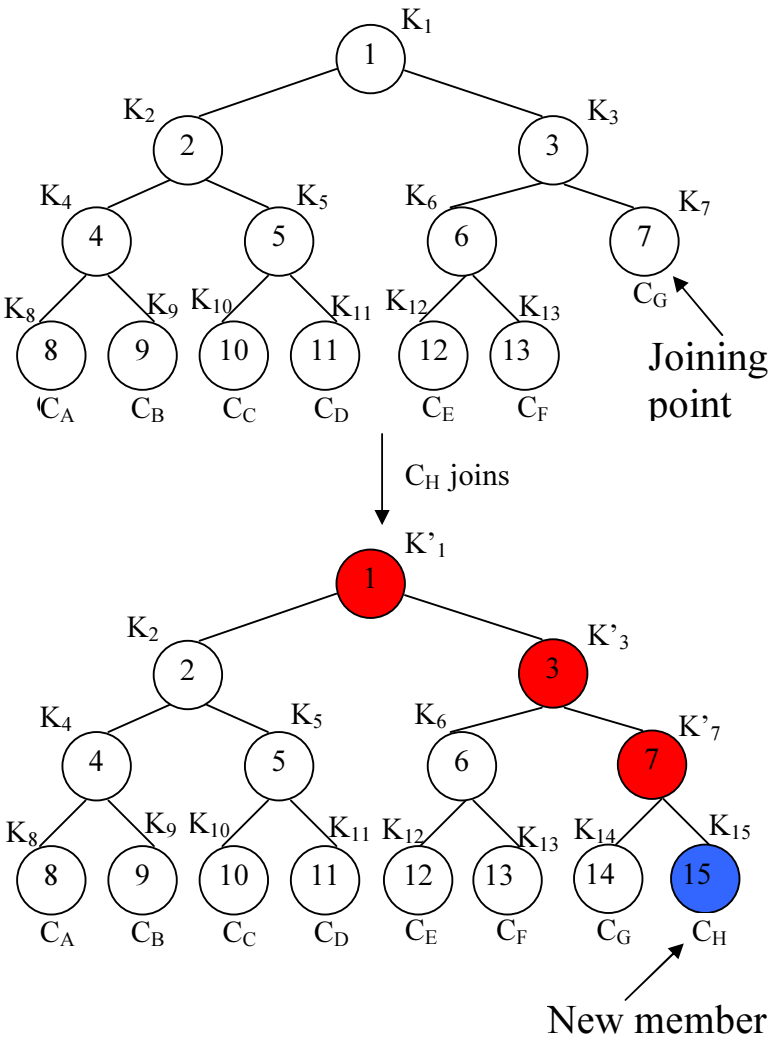


Figure 5.2: Client C_H joins the group

5.1.1.2 Leave

After receiving a leave request from a member, the GKS first identifies the *leaving point*. The leaving point, node $\lceil j/2 \rceil$, is the parent of the leaf node j owned by the member that is leaving. The member currently owning node $sib(j)$ will be moved to be associated with node $\lceil j/2 \rceil$, the parent node. The GKS copies the content of node $sib(j)$ to node $\lceil j/2 \rceil$, and deletes both nodes j and $sib(j)$ from the key tree. All the intermediate keys along the path from the leaving point to the root and the group key at the root will be updated as will be discussed in Sections 5.1.2.2 and 5.1.3.3.

Consider an example illustrated by Figure 5.3 in which member C_H leaves the group. The leaving point is node 7, the parent of node 15 owned by C_H. The GKS deletes the leaf node 15 from the key tree, and moves the content of node 14 to the location of the parent, node 7. To prevent the leaving member C_H from accessing future messages, all keys along the path from the leaving point node 7 to the root node 1 need to be changed. As shown in Figure 5.3, key K_7, K_3 and K_1 are changed to the new key K_7', K_3' and K_1', respectively.

5.1.2 LKH Operations

The LKH algorithm [13] is a tree-based GKM scheme using symmetric-key cryptography. In the LKH algorith, the root node of the key tree stores the group key for data encryption/decryption, which is shared by all members in the group.

Each group member is associated with a leaf node, which contains an individual key of

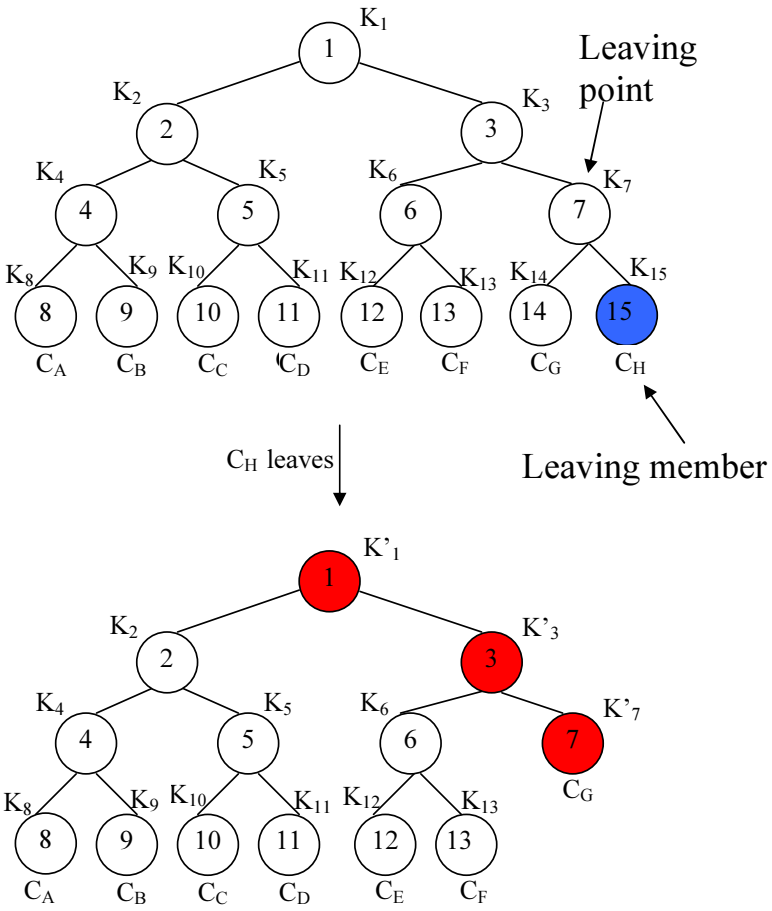


Figure 5.3: Client C_H leaves the group

each member. An individual key is shared only between the member owning the key and the group key server, and is used for pairwise secure communication between the member and the group key server. The non-leaf nodes, except the root node, store intermediate keys that are used to encrypt other intermediate keys or the group key (not data) during a rekeying process. Each member of the group stores the keys along the path from the root to the leaf node assigned to that member (i.e., $\lceil \log_2 n \rceil$ keys, where n is the number of group members). The storage requirement from each member is thus $O(logn)$.

5.1.2.1 LKH Join Operation

After receiving a join request from a new member, the GKS performs the following actions:

- Identifying the joining point i as discussed in Section 5.1.1.
- Creating two new leaf nodes with indices $2i$ and $2i + 1$, which are the children of node i.
- Copying the content of node i to node $2i$: $K_{2i} = K_i$ The member previously owning node i is now associated with node $2i$.
- Generating an intermediate key K_i' for node i.
- Assigning node $2i + 1$ to the new joining member and generating a new key K_{2i+1} for this leaf node (for the new member).

Consider the example illustrated by Figure 5.2 in which client C_H joins the group. After receiving a join request from C_H, the GKS identifies a joining point, node 7, and

creates two new leaf nodes, node 14 and node 15, which are the children of node 7. The GKS then copies the content of node 7 to node 14 as member C_G previously owning node 7 is now associated with node 14. An intermediate key K'_7 is then generated by GKS for node 7. The GKS also assigns node 15 to the new joining member C_H, and generates a new key K_8 for the new member C_H.

The GKS then updates the key tree as follows.

1. The GKS updates all keys on the path from node i to the root: it generates new keys $K'_i, K'_{\lceil i/2 \rceil}, K'_{\lceil i/4 \rceil}, \ldots, K'_1$ for nodes $i, \lceil i/2 \rceil, \lceil i/4 \rceil, \ldots, 1$, respectively. Given the example in Figure 5.2, the GKS generates new keys K'_7, K'_3, and K'_1 for nodes 7, 3 and 1 (the shaded nodes), respectively.

2. The GKS sends the above new keys to the new member in a secure message encrypted with the new member's individual key K_{2i+1}: $\{K'_i, K'_{\lceil i/2 \rceil} K'_{\lceil i/4 \rceil}, \ldots, K'_1\}_{K_{2i+1}}$. After decrypting the message, the new member obtains the intermediate keys associated with the nodes on the path from itself to the root and the new group key K'_1. In the above example, the GKS sends the following message to the new member C_H: $\{K'_1, K'_3, K'_7\}_{K_{15}}$.

3. The GKS encrypts the new keys $K'_i, K'_{\lceil i/2 \rceil}, K'_{\lceil i/4 \rceil}, \ldots, K'_1$ using the current keys $K_i, K_{\lceil i/2 \rceil}, K_{\lceil i/4 \rceil}, \ldots, K_1$, respectively, to obtain $\{K'_i\}_{K_i}$, $\{K'_{\lceil i/2 \rceil}\}_{K_{\lceil i/2 \rceil}}$, $\{K'_{\lceil i/4 \rceil}\}_{K_{\lceil i/4 \rceil}}$ $\ldots$, K'_1. The GKS then sends each encrypted new key $\{K'_j\}_{K_j}$ to the existing members associated with the leaf nodes in the subtree rooted at node j. In the above

example, the GKS sends

- $\{K_7'\}_{K_7}$ to node C_G;
- $\{K_3'\}_{K_3}$ to nodes C_E, C_F and C_G;
- $\{K_1'\}_{K_1}$ to nodes C_A, C_B, ..., C_F and C_G.

Upon receiving the above messages, the members decrypt them using the current keys to obtain the new intermediate keys and the new group key K_1'.

Note that the GKS may combine multiple messages into a physical packet before sending to the members in order to save network bandwidth. Each member will then extract the rekeying message(s) it needs. In Section 5.2.1.2, we will discuss how this is implemented when the LKH algorithm is applied to group key management in WMNs.

5.1.2.2 LKH Leave Operation

After receiving a leave request from a member associated with a leaf node v, the GKS performs the following actions:

- Deleting the leaf node v from the key tree. The parent node $i = \lceil v/2 \rceil$ is the leaving point.
- "Moving" the sibling of node v to the location of the parent node i. That is, $K_i' = K_{sib(v)}$.

- Deleting the sibling of node v from the key tree (whose content had been copied to node i in the previous step).

Consider the example illustrated by Figure 5.3 in which client C_H leaves the group. After receiving a leave request from C_H, the GKS identifies a leaving point, node 7, and deletes the leaf nodes 15 from the key tree. The GKS then copies the content of node 14 to node 7 as member C_G previously owning node 14 is now associated with node 7. Node 14 is then deleted by the GKS from the key tree as its content had been copied to node 7.

The GKS then updates the key tree as follows.

1. The GKS updates all intermediate keys on the path from the leaving point i to the root. That is, the GKS generates new keys $K'_{\lceil i/2 \rceil}, K'_{\lceil i/4 \rceil}, \ldots, K'_1$ for nodes $\lceil i/2 \rceil, \lceil i/4 \rceil, \ldots, 1$, respectively. (The leaving point i is now a leaf node containing the individual key of a group member.) Given the example in Figure 5.3, the GKS generates new keys K'_7, K'_3, and K'_1 for nodes 7, 3 and 1 (the shaded nodes), respectively. Node 7 now contains the individual key of member C_G.

2. For every new key K'_j generated in the above step, the GKS encrypts K'_j using the key of the left child of node j to obtain $\{K'_j\}_{K_{2j}}$. The GKS then sends $\{K'_j\}_{K_{2j}}$ to the leaf nodes in the left subtree rooted at node j (i.e., the subtree rooted at node $2i$). Upon receiving $\{K'_j\}_{K_{2j}}$, these leaf nodes use the key K_{2j}, which they know, to decrypt the message to obtain the new key K'_j. When $j = 1$, that is the new group key K'_1.

In the above example, the GKS sends

- $\{K_3'\}_{K_6}$ to node C_E and C_F;
- $\{K_1'\}_{K_2}$ to node C_A, C_B, C_C and C_D.

Similarly, the GKS encrypts K_j' using the key of the right child of node j to obtain $\{K_j'\}_{K_{2j+1}}$. The GKS then sends $\{K_j'\}_{K_{2j+1}}$ to the leaf nodes in the right subtree rooted at node j. Upon receiving $\{K_j'\}_{K_{2j+1}}$, these leaf nodes use the key K_{2j+1}, which they know, to decrypt the message to obtain the new key K_j'.

In the above example, the GKS sends

- $\{K_3'\}_{K_7'}$ to node C_G;
- $\{K_1'\}_{K_3'}$ to node C_E, C_F and C_G.

Upon receiving the above messages, the members decrypt them using the current keys to obtain the new intermediate keys and the new group key K_1'. In practice, the GKS can combine all the above messages into one single packet and broadcast to all members in the group to save network bandwidth.

5.1.3 OFT Operations

The OFT [14] algorithm is a scalable centralized scheme based on the application of one-way function trees and a bottom-up approach to calculate the group key of a multicast group.

5.1.3.1 OFT Overview

Each node i in an OFT logical tree is associated with three types of keys as follows:

- K_i: node secret
- $f(K_i)$: blinded node key
- $g(K_i)$: node key

If node i is a leaf node, the node secret K_i is known only to the GKS and the group member owning node i. (The member's node secret K_i in OFT plays the same role as a member's individual key used in LKH.) When the group member owning node i joins the group, the GKS generates the node secret K_i and sends it to the new member (securely using a shared key for one-to-one communications between the member and the GKS).

A blinded node key $f(K_i)$ is defined as a one-way function of the node secret K_i. Blinded keys are generated by a pseudo-random function f. (Pseudo-random functions are used to generate random numbers.) It is blinded in the sense that a computationally limited adversary may know $f(K_i)$, yet cannot compute K_i. Group members use blinded keys and node keys to compute the group key K_1.

A node key $g(K_i)$ is used to encrypt other keys such as blinded node keys. A pseudo-random function g is used to compute node key $g(K_i)$ for each node i. Every group member knows the pseudo-random functions f and g.

Following is the sequence of operations performed by a group member associated with a leaf node i in order to compute the group key, K_1. We also provide an example, using

the group in Figure 5.4, which shows how member C_C associated with node 10 computes the group key K_1.

Action	Output	Example
Compute	$f(K_i)$, $g(K_i)$	$f(K_{10})$, $g(K_{10})$
Receive from GKS	$\{f(K_{sib(i)})\}_{g(K_i)}$	$\{f(K_{11})\}_{g(K_{10})}$
Decrypt the above message	$f(K_{sib(i)})$	$f(K_{11})$
Compute parent's node secret	$K_{\lceil i/2 \rceil} = f(K_i) \oplus f(K_{sib(i)})$	$K_5 = f(K_{10}) \oplus f(K_{11})$
Receive from GKS	$\{f(K_{sib(\lceil i/2 \rceil)})\}_{g(K_{\lceil i/2 \rceil})}$	$\{f(K_4)\}_{g(K_5)}$
Decrypt the above message	$f(K_{sib(\lceil i/2 \rceil)})$	$f(K_{4)}$
Compute grand-parent's node secret	$K_{\lceil i/4 \rceil} = f(K_{\lceil i/2 \rceil}) \oplus f(K_{sib(\lceil i/2 \rceil)})$	$K_2 = f(K_4) \oplus f(K_5)$
Receive from GKS	$\{f(K_{sib(\lceil i/4 \rceil)})\}_{g(K_{\lceil i/4 \rceil})}$	$\{f(K_3)\}_{g(K_2)}$
Decrypt the above message	$f(K_{sib(\lceil i/4 \rceil)})$	$f(K_3)$
...	...	
Compute group key	$K_1 = f(K_2) \oplus f(K_3)$	$K_1 = f(K_2) \oplus f(K_3)$

In the above sequence, node i receives from the GKS several encrypted values $f(K_{sib(j)})$ for $j = i, \lceil i/2 \rceil, \lceil i/4 \rceil, \ldots$. In practice, the GKS can combine all these encrypted values into one message as shown below and sends it to node i as follows:

$\{f(K_{sib(i)})\}_{g(K_i)}, \{f(K_{sib(\lceil i/2 \rceil)})\}_{g(K_{\lceil i/2 \rceil})}, \{f(K_{sib(\lceil i/4 \rceil)})\}_{g(K_{\lceil i/4 \rceil})}, \ldots$

In summary, every leaf node i (the member owning node i)

1. needs to store only the blinded node keys of its sibling and the siblings of its ancestors (except the root) in the key tree: $f(K_{sib(i)})$, $f(K_{sib(\lceil i/2 \rceil)}), f(K_{sib(\lceil i/4 \rceil)}), \ldots$.

2. applies the following equation recursively to compute the node secrets of its ancestors, including the group key K_1, from the bottom to the top:

$$K_{\lceil j/2 \rceil} = f(K_j) \oplus f(K_{sib(j)}), j = i, \lceil i/2 \rceil, \lceil i/4 \rceil, \ldots \tag{5.1}$$

3. can compute the ancestors' node keys $g(K_{\lceil i/2 \rceil})$, $g(K_{\lceil i/4 \rceil})$, ... (which are used by the GKS to encrypt the blinded keys of the ancestors' siblings $f(K_{sib(\lceil i/2 \rceil)})$, $f(K_{sib(\lceil i/4 \rceil)})$, ..., respectively, when the GKS updates the key tree due to members joining or leaving.)

In the following sub-sections, we describe in detail how the key tree is updated and a new group key is computed when a member joins or leaves the group.

5.1.3.2 OFT Join Operation

After receiving a join request from a new member, the GKS performs the following actions:

- Identifying the joining point i as discussed in Section 5.1.1.
- Creating two new leaf nodes with indices $2i$ and $2i + 1$, which are the children of node i. Node i now becomes a non-leaf node storing intermediate keys.
- Generating a new individual key (node secret) K_{2i} and computing a blinded node key $f(K_{2i})$ for the member previously owning node i, who is now associated with node $2i$.

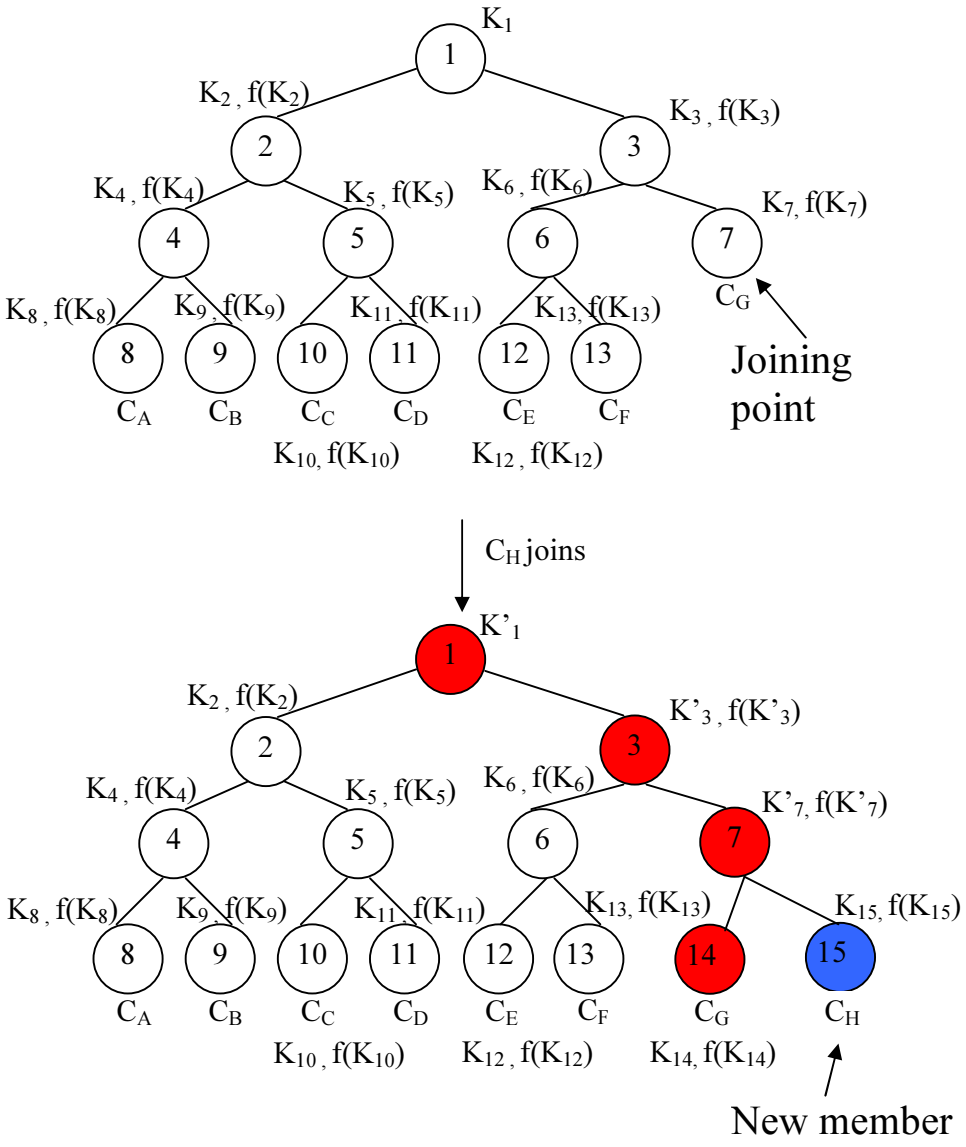


Figure 5.4: Client C_H joins the group - OFT

- Assigning node $2i + 1$ to the new joining member, generating an individual key (node secret) K_{2i+1} and computing a blinded node key $f(K_{2i+1})$ for this leaf node (for the new member).

Consider the example illustrated by Figure 5.4 in which client C_H joins the group. After receiving a join request from C_H, the GKS identifies a joining point, node 7, and creates two new leaf nodes, node 14 and node 15, which are the children of node 7. The GKS then generates a new individual key K_{14} and computes a blinded node key $f(K_{14})$ for member C_G who previously owned node 7 and now is associated with node 14. The GKS also assigns node 15 to the new member C_H, generates an individual key K_{15} and computes a blinded node key $f(K_{15})$ for this new leaf node 15.

The content of the key tree is then updated as follows.

1. The GKS re-computes the node secrets of the nodes on the path from node i to the root, $\lceil i/2 \rceil$, $\lceil i/4 \rceil$, $\ldots, 1$ recursively using Eq. (1), K_{2i}, K_{2i+1} and blinded keys remain unaffected by the membership change. The new node secrets are K'_i, $K'_{\lceil i/2 \rceil}$, $K'_{\lceil i/4 \rceil}$, $\ldots$, K'_1. For example,

   $K'_i = f(K_{2i}) \oplus f(K_{2i+1})$,

   $K'_{\lceil i/2 \rceil} = f(K'_i) \oplus f(K_{sib(i)})$ and so on.

   Given the example in Figure 5.4, the GKS re-computes the node secrets of node 7, 3 and 1 recursively using Eq. (1) as follows:

   - $K'_7 = f(K_{14}) \oplus f(K_{15})$;

- $K'_3 = f(K_6) \oplus f(K'_7)$;
- $K'_1 = f(K_2) \oplus f(K'_3)$.

2. The roup key server GKS computes the new blinded node keys $f(K'_i)$, $f(K'_{\lceil i/2 \rceil})$, $f(K'_{\lceil i/4 \rceil})$, ..., $f(K'_c)$ on the path from node i up to node c where c is a child of the root ($c = 2$ if the joining point i is in the left subtree and $c = 3$ otherwise).

   In the above example, the GKS computes the new blinded node keys $f(K'_7)$ and $f(K'_3)$.

3. For each new blinded key $f(K'_j)$ ($j = i, \lceil i/2 \rceil, \lceil i/4 \rceil, \ldots, c$), the GKS encrypts it using the node key of the sibling of node j: $\{f(K'_j)\}_{g(K_{sib(j)})}$. The GKS then sends $\{f(K'_j)\}_{g(K_{sib(j)})}$ to the leaf nodes in the subtree rooted at node $sib(j)$. Upon receiving $\{f(K'_j)\}_{g(K_{sib(j)})}$, these leaf nodes decrypt the message using $g(K_{sib(j)})$, which they know because node $sib(j)$ is one of their ancestors (see item (3) of the summary at the end of Section 5.1.3.1).

   In the above example, the GKS sends

   - $\{f(K_{14}), f(K_6), f(K_2)\}_{g(K_{15})}$ to node C_H;
   - $\{f(K_{15})\}_{g(K_{14})}$ to node C_G;
   - $\{f(K'_7)\}_{g(K_6)}$ to node C_E and C_F;
   - $\{f(K'_3)\}_{g(K_2)}$ to node C_A, C_B, C_C and C_D.

4. After a leaf node receives all the necessary new blinded keys (sent by the GKS in

the above step), it re-calculates the group key using Eq. (1), the new blinded keys, and applicable keys that are unchanged.

In the example above, member C_G and C_H can compute the group key using Eq. (1) recursively as follows:

- $K'_7 = f(K_{14}) \oplus f(K_{15})$;
- $K'_3 = f(K_6) \oplus f(K'_7)$;
- $K'_1 = f(K_2) \oplus f(K'_3)$.

Client C_E and C_F can re-calculate the new group key K'_1 recursively as follows:

- $K'_3 = f(K_6) \oplus f(K'_7)$;
- $K'_1 = f(K_2) \oplus f(K'_3)$.

Client C_A, C_B, C_C and C_D re-calculate the new group key K'_1 as follows: $K'_1 = f(K_2) \oplus f(K'_3)$.

5.1.3.3 OFT Leave Operation

After receiving a leave request from a member associated with a leaf node v, the GKS performs the following actions:

- Deleting the leaf node v from the key tree. The parent node $i = \lceil v/2 \rceil$ is the leaving point.

- "Moving" the sibling of node v to the location of the parent node i. The member previously owning node $sib(v)$ is now associated with node i. The GKS generates a new individual key K_i' for this member and stores it in node i.

- Deleting the sibling of node v from the key tree.

Consider the example illustrated by Figure 5.5 in which client C_H leaves the group. After receiving a leave request from C_H, the GKS identifies the leaving point, node 7, and deletes the leaf node 15 from the key tree. The GKS then copies the content of node 14 to the location of node 7 because member C_G previously owning node 14 is now associated with node 7. The GKS then generates a new individual key (node secret) K_7' for member C_G and stores it in node 7. Node 14 is then deleted by the GKS from the key tree.

The content of the key tree is then updated as follows.

1. The GKS re-computes the node secrets of the nodes on the path from i's parent to the root $\lceil i/2 \rceil$, $\lceil i/4 \rceil$, $\ldots, 1$ recursively using Eq. (1), K_i' and blinded keys remain unaffected by the membership change. The new node secrets are $K'_{\lceil i/2 \rceil}$, $K'_{\lceil i/4 \rceil}$, $\ldots$, K_1'. For example,

   $K'_{\lceil i/2 \rceil} = f(K_i') \oplus f(K_{sib(i)})$ and so on.

   Given the example in Figure 5.5, the following blinded keys are not affected by the membership change: $f(K_2)$ and $f(K_6)$. The GKS re-computes the node secrets of node 3 and 1 recursively using Eq. (1) as follows:

   - $K_3' = f(K_6) \oplus f(K_7')$;

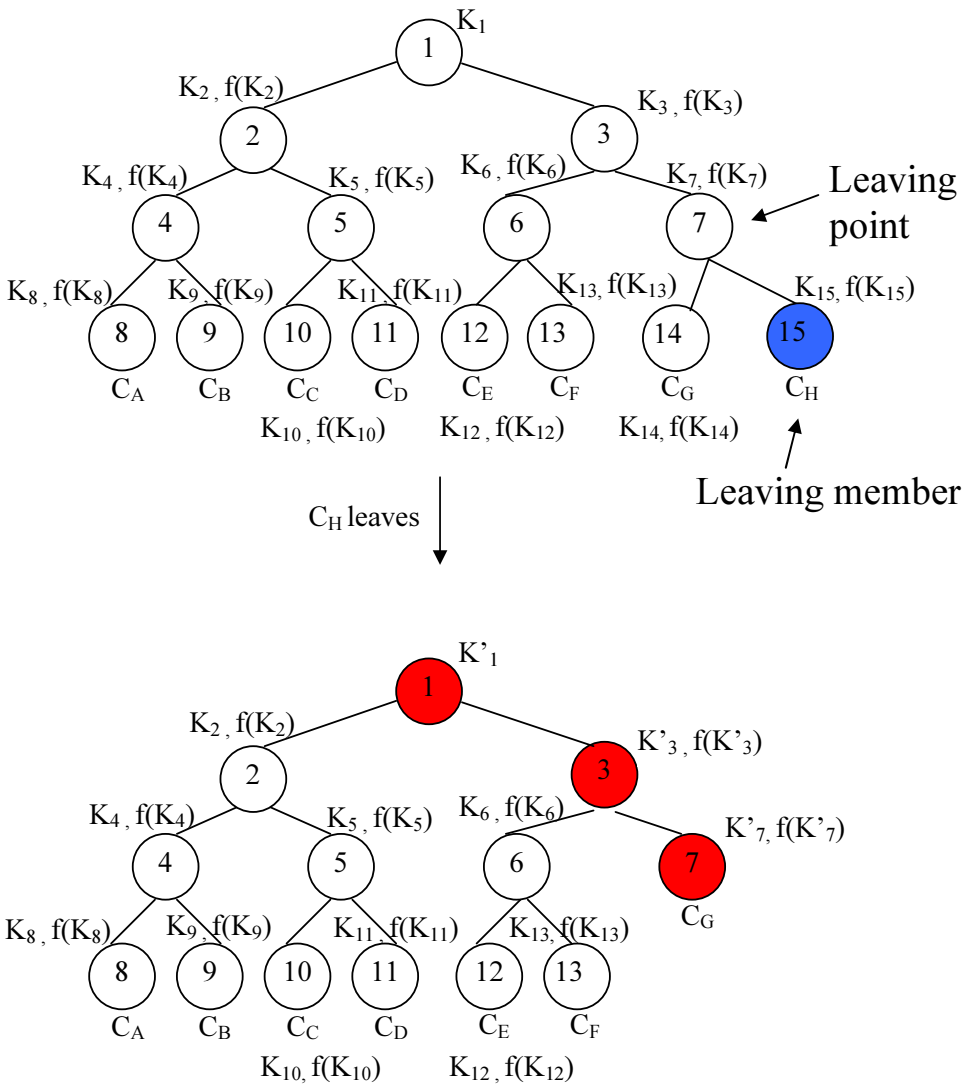


Figure 5.5: Client C_H leaves the group - OFT

- $K'_1 = f(K_2) \oplus f(K'_3)$.

2. The group key server computes new blinded node keys $f(K'_i)$, $f(K'_{\lceil i/2 \rceil})$, $f(K'_{\lceil i/4 \rceil})$, $\ldots$, $f(K'_c)$ on the path from node i up to node c where c is a child of the root ($c = 2$ if the leaving point i is in the left subtree and $c = 3$ otherwise).

   In the above example, the GKS needs to compute the new blinded node keys $f(K'_7)$ and $f(K'_3)$.

3. For each new blinded key $f(K'_j)$ ($j = i, \lceil i/2 \rceil, \lceil i/4 \rceil, \ldots, c$), the GKS encrypts it using the node key of the sibling of node j: $\{f(K'_j)\}_{g(K_{sib(j)})}$. The GKS then sends $\{f(K'_j)\}_{g(K_{sib(j)})}$ to the leaf nodes in the subtree rooted at node $sib(j)$. Upon receiving $\{f(K'_j)\}_{g(K_{sib(j)})}$, these leaf nodes decrypt the message using $g(K_{sib(j)})$, which they know because node $sib(j)$ is one of their ancestors (see item (3) of the summary at the end of Section 5.1.3.1).

   In the above example, the GKS sends

   - $\{f(K'_7)\}_{g(K_6)}$ to node C_E and C_F;
   - $\{f(K'_3)\}_{g(K_2)}$ to node C_A, C_B, C_C and C_D.

4. After a leaf node receives all the necessary new blinded keys (sent by the GKS in the above step), it re-calculates the group key using Eq. (1), the new blinded keys, and blinded keys unaffected by the membership change.

   In the example above, member C_E, C_F and C_G can re-compute the group key using Eq. (1) recursively as follows:

- $K'_3 = f(K_6) \oplus f(K'_7)$;
- $K'_1 = f(K_2) \oplus f(K'_3)$.

Client C_A, C_B, C_C and C_D re-calculate the new group key K'_1 as follows: $K'_1 = f(K_2) \oplus f(K'_3)$.

5.2 Implementing LKH and OFT in Mesh Access Points

We apply the LKH and OFT algorithms described above to group key management at the data link layer in WMNs. The group key server is now the mesh access point (MAP) and the group members are the clients associated with the BSS under the control of the MAP. The MAP maintains and updates the logical key tree. As mentioned earlier, the MAP may combine several rekeying messages for different members into one packet to save network bandwidth. In this section, we describe how the LKH and OFT algorithms are incorporated into group key management at the data link layer in mesh access points of a WMN. The main design objective is to minimize the communication cost between the MAP and the members during the rekeying process. The communication cost is the amount of data sent by the MAP to the group members for the purpose of group key updates.

5.2.1 LKH Implementation

In this section, we implement the LKH algorithm described in Section 5.1.2 for group key management at the data link layer in WMNs. We propose a space-saving message

structure for rekeying messages to minimize the communication cost between the MAP and the members. We then apply this message structure to the group key rekeying process for LKH implementation.

5.2.1.1 LKH Key Storage

Recall that each group member in the LKH algorithm is associated with a leaf node, which contains the individual key of each member. The mesh access point maintains and updates the intermediate keys, which are stored in the non-leaf nodes of the logical key tree. Each intermediate key is used to encrypt other intermediate keys or the group key during the rekeying process. Every member must know all the keys on the path from the leaf node (with which the member is associated) to the root. This means that every member needs to store $O(\log_2 n)$ keys, since every path from leaf to the root is at most $\log_2 n + 1$ long. Consider the example illustrated in Figure 5.2 in which a new member C_H joins the group. Table 5.1 (a) shows all keys that each existing member stores before C_H joins the group. Table 5.1 (b) shows all keys that each existing member and the new member C_H maintain after C_H joins the group. For example, before the new member joins the group, member C_G stores its own individual key K_7 with node ID 7, intermediate key K_3 with node ID 3 and group key K_1 with node ID 1. After C_H joins the group, as node 7 in the key tree is the joining point, two new nodes 14 and 15 are created (see Section 5.1.1.1). Member C_G's individual key, currently denoted as K_7, is copied to and stored at node 14 and renamed to K_{14}. After the GKS generates the new keys K'_7, K'_3,

K_1' for node 7, node 3 and node 1 respectively and distributes them, each member will update its key table accordingly, as shown in Table 5.1(b).

Table 5.1: Each member's key structure for LKH

(a) Before new member C_H joins

C_A		C_B		C_C		C_D		C_E		C_F		C_G	
ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key
8	K_8	9	K_9	10	K_{10}	11	K_{11}	12	K_{12}	13	K_{13}	7	K_7
4	K_4	4	K_4	5	K_5	5	K_5	6	K_6	6	K_6	3	K_3
2	K_2	2	K_2	2	K_2	2	K_2	3	K_3	3	K_3	1	K_1
1	K_1	1	K_1	1	K_1	1	K_1	1	K_1	1	K_1	–	–

(b) After new member C_H joins

C_A		C_B		C_C		C_D		C_E		C_F		C_G		C_H	
ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key
8	K_8	9	K_9	10	K_{10}	11	K_{11}	12	K_{12}	13	K_{13}	14	K_{14}	15	K_{15}
4	K_4	4	K_4	5	K_5	5	K_5	6	K_6	6	K_6	7	K_7'	7	K_7'
2	K_2	2	K_2	2	K_2	2	K_2	3	K_3'	3	K_3'	3	K_3'	3	K_3'
1	K_1'	1	K_1'	1	K_1'	1	K_1'	1	K_1'	1	K_1'	1	K_1'	1	K_1'

5.2.1.2 LKH Message Structure

The GKS (mesh access point) can send out updated keys to each member individually. It can also combine the updated keys into one message and broadcast to all members to save network bandwidth. Because of the broadcast nature of a wireless transmission

in a WMN, one message can reach many nodes within the transmission range of the transmitter. Thus, in our design, the GKS broadcasts rekeying messages to all members to save network bandwidth. A rekeying message contains every updated key and its corresponding node ID. We use the same encryption algorithm as specified in the IEEE 802.11 standard, which is AES [101]. The size of a key is 16 bytes (128 bits). The size of a node ID is 2 bytes (16 bits), which can represent 2^{16} nodes in a key tree. A mesh access point such as Motorola WPA 400 AP could support up to 256 (2^8) active clients [139], which is much less than 2^{16}, the maximum number of nodes represented in the key tree. The message structure is shown in Figure 5.6 and explained as follows:

- The first two bytes indicate the current node ID of the joining (leaving) point, which is also the node ID of a member who currently occupies the joining (leaving) point.
- The next two bytes indicate the new node ID of the node previously occupying the joining (leaving) point.
- The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and a 16-byte encrypted key value.
  - The 2-byte node ID identifies the node storing the key K that is used to encrypt one or more new keys.
  - The 16-byte encrypted key value is obtained from encrypting one or more newly generated keys using key K.

The mesh access point stores and maintains the whole logic key tree. For a complete binary tree, the index of each array element is also the index of each node in the key tree. If we assume complete binary trees, there is no need for storing the new node ID. A node in a complete binary tree is always added to the key tree from top to bottom and left to right. Thus, the new node ID after the node addition is $2i + 1$, where i is the index of the joining point in the array. A node in a complete binary tree is always removed from the key tree from right to left and bottom to top. Thus, the new node ID after the node deletion is $\lceil i/2 \rceil$, where i is the index of last array element before removing a node. However, in practice, the tree may not be a complete binary tree, and thus the mesh access point needs to store both pieces of information (joining point or leaving point, new node ID) explicitly to save array space.

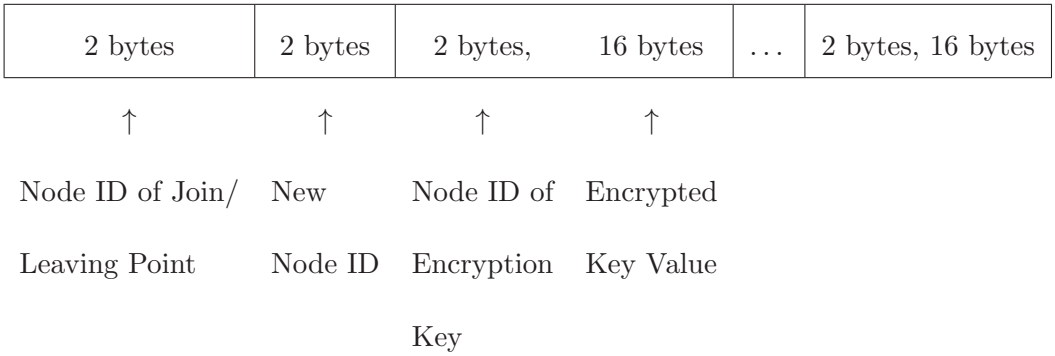


Figure 5.6: Rekeying message format

Given the example illustrated in Figure 5.2, after a new member C_H joins the group, member C_G changes its association from node 7 to node 14. The GKS then broadcasts a LKH rekeying message to all existing group members (see Figure 5.2.1.2), and unicasts another LKH rekeying message to the new member C_H (see Figure 5.2.1.2). Both of these

messages are instances of the implementation of the message structure in Figure 5.6. Figure 5.2.1.2 shows the content of the LKH rekeying message broadcast by the GKS after C_H joins the group. The first two bytes indicate the node ID (7) of the joining point, which is also the node ID of member C_G who currently occupies the joining point. The third and fourth bytes indicate that member C_G's new node ID is being changed to 14. The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and 16-byte encrypted key value. For example, "$1, \{K_1'\}_{K_1}$" denotes that the newly generated key K_1' is encrypted using key K_1. The encryption key K_1 is located at node 1 in the key tree. Similarly, "$14, \{K_7'\}_{K_{14}}$" denotes that the newly generated key K_7' is encrypted using key K_{14}. The encryption key K_{14} is located at node 14 in the key tree. When a member receives the message, it only extracts the updated key it needs. In the above example, member C_E and C_F only need to get the updated keys K_1' and K_3' from the message, while member C_A, C_B, C_C and C_D only need to extract the updated key K_1'. Figure 5.2.1.2 shows the content of the LKH rekeying message the MAP sends to the new member C_H. The first four bytes are the same as those shown in Figure 5.2.1.2. The first two bytes indicate the node ID (7) of the joining point. The third and fourth bytes indicate the node ID (14) of new member's sibling. The next 18 bytes consist of C_H's node ID (15) and the encrypted keys 0.00,0.00,1.00($\{K_1', K_3', K_7'\}_{K_{15}}$) C_H needs to know. When C_H receives this message, it can use its individual key K_{15} to decrypt the message and get the updated keys K_1', K_3', and K_7'.

Given the example shown in Figure 5.3, after a member C_H leaves the group, member C_G changes its association from node 14 to node 7. Figure 5.8 shows the content of a LKH rekeying message sent by the GKS after member C_H leaves the group. The first two bytes indicate the current node ID (14), which is also the node ID of member C_G who currently occupies this node and whose sibling is leaving the group. The third and fourth bytes indicates the node ID of the leaving point, which is also the new node ID (7) of member C_G. The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and 16-byte encrypted key value. For example, "$3, \{K_1'\}_{K_3}$" denotes that the newly generated key K_1' is encrypted with key K_3. The encryption key K_3 is located at node 3 in the key tree. Similarly, "$6, \{K_3'\}_{K_6}$" denotes that the newly generated key K_3' is encrypted with key K_6. The encryption key K_6 is located at node 6 in the key tree. When a member receives the message, it extracts only the updated keys it needs. In the above example, member C_E, C_F, and C_G only need to get the updated keys K_1' and K_3' from the message, while member C_A, C_B, C_C and C_D only need to extract the updated key K_1'.

5.2.2 OFT Implementation

In this section, we apply the OFT algorithm described in Section 5.1.3 to group key management at the data link layer in WMNs. Our objectives are low communication cost and space saving between the MAP and the multicast members. We apply the same

7	14	1, $\{K_1'\}_{K_1}$	3, $\{K_3'\}_{K_3}$	14, $\{K_7'\}_{K_{14}}$

C_G's node

ID is changed	$\longleftarrow$	Updated Keys	$\longrightarrow$

from 7 to 14.

(a) Broadcast message for client C_A to C_G after client C_H joins the group

7	14	15, $\{K_1', K_3', K_7'\}_{K_{15}}$

C_G's node

ID is changed	$\longleftarrow$ Updated Keys $\longrightarrow$

from 7 to 14.

(b) Unicast message for client C_H after client C_H joins the group

Figure 5.7: LKH message example after client C_H joins the group

14	7	2, $\{K_1'\}_{K_2}$	3, $\{K_1'\}_{K_3}$	6, $\{K_3'\}_{K_6}$	7, $\{K_3'\}_{K_7'}$

C_G's node

ID is changed	$\longleftarrow$	Updated Keys	$\longrightarrow$

from 14 to 7.

Broadcast message for client C_A to C_G after client C_H leaves the group.

Figure 5.8: LKH message example after client C_H leaves the group

rekeying message structure proposed in Section 5.2.1.2 (Figure 5.6) to the group key rekeying process for OFT implementation.

5.2.2.1 OFT Key Storage

Using the OFT algorithm, a member C stores the following information:

- its own individual key K_i,
- the blinded node key of its sibling $f(K_{sib(i)})$,
- the blinded node key of the sibling of every ancestor of node i (except the root): $f(K_{sib(\lceil i/2 \rceil)})$, $f(K_{sib(\lceil i/4 \rceil)})$, ...

Each of the above keys is accompanied by the ID of the node where the key is stored.

Consider the example in Figure 5.4 in which a new member C_H joins the group. Table 5.2 shows the keys each member stores before and after C_H joins the group.

For example, before C_H joins the group, member C_G stores its own individual key K_7 (node ID 7), its sibling's blinded node key $f(K_6)$ (node ID 6), and the blinded node key $f(K_2)$ of its parent's sibling (node ID 2). After C_H joins the group, because node 7 in the key tree is the joining point, two new nodes 14 and 15 are created as children nodes of node 7. The MAP (the group key server) creates a new individual key K_{14} for member C_G and stores this new key at node 14. Since the keys along the path from the joining point to the root are updated, the MAP sends the updated blinded node keys $f(K'_7)$ and $f(K'_3)$ (accompanied by the corresponding node IDs) to all members. Each member will

update its key table accordingly after receiving the new keys, as shown in Table 5.2.

Table 5.2: Each member's key structure for OFT

(a) Before new member C_H joins

C_A		C_B		C_C		C_D		C_E		C_F		C_G	
ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key
8	K_8	9	K_9	10	K_{10}	11	K_{11}	12	K_{12}	13	K_{13}	7	K_7
9	$f(K_9)$	8	$f(K_8)$	11	$f(K_{11})$	10	$f(K_{10})$	13	$f(K_{13})$	12	$f(K_{12})$	6	$f(K_6)$
5	$f(K_5)$	5	$f(K_5)$	4	$f(K_4)$	4	$f(K_4)$	7	$f(K_7)$	7	$f(K_7)$	2	$f(K_2)$
3	$f(K_3)$	3	$f(K_3)$	3	$f(K_3)$	3	$f(K_3)$	2	$f(K_2)$	2	$f(K_2)$	–	–

(b) After new member C_H joins

C_A		C_B		C_C		C_D		C_E		C_F		C_G		C_H	
ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key	ID	Key
8	K_8	9	K_9	10	K_{10}	11	K_{11}	12	K_{12}	13	K_{13}	14	K_{14}	15	K_{15}
9	$f(K_9)$	8	$f(K_8)$	11	$f(K_{11})$	10	$f(K_{10})$	13	$f(K_{13})$	12	$f(K_{12})$	15	$f(K_{15})$	14	$f(K_{14})$
5	$f(K_5)$	5	$f(K_5)$	4	$f(K_4)$	4	$f(K_4)$	7	$f(K'_7)$	7	$f(K'_7)$	6	$f(K_6)$	6	$f(K_6)$
3	$f(K'_3)$	3	$f(K'_3)$	3	$f(K'_3)$	3	$f(K'_3)$	2	$f(K_2)$	2	$f(K_2)$	2	$f(K_2)$	2	$f(K_2)$

5.2.2.2 OFT Message Structure

To apply the OFT algorithm to GKM in WMNs, we use the same rekeying message structure proposed for the LKH algorithm in Section 5.2.1.2, as shown in Figure 5.6.

Given the example illustrated in Figure 5.4, after the new member C_H joins the group, member C_G changes its association from node 7 to node 14. Thus, the individual key of member C_G also moves from node 7 to node 14. The GKS (MAP) then broadcasts a OFT rekeying message to all existing group members, and unicasts another OFT rekeying message to the new member C_H as shown in Figure 5.9. The first two bytes records the original node ID "7", and the next two bytes represents the new node ID, "14". In the next 18 bytes, the first two bytes store the node ID of the encryption key followed by the

16-byte encrypted key values.

Figure 5.2.2.2 shows the content of the OFT rekeying message broadcast by the GKS after C_H joins the group. The first two bytes indicate the current node ID of the joining point (node 7 in this case), which is also the node ID of member C_G who currently occupies the joining point. The third and fourth bytes indicate that the member C_G's new node ID changes to 14. The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and 16-byte encrypted key value. The 2-byte node ID identifies the node storing the key K that is used to encrypt one or more new keys. The 16-byte encrypted key value is obtained from encrypting one or more newly generated keys using key K. For example, $[2, \{f(K_3')\}_{g(K_2)}]$ indicates that the newly generated blinded node key $f(K_3')$ is encrypted with node key $g(K_2)$, where $g(K_2)$ is the result of applying function g to the node secret K_2 of node 2. Similarly, $[14, \{K_{14}, f(K_{15})\}_{g(K_7)}]$ indicates that the newly generated node key K_{14} is encrypted with client C_G's previous node key $g(K_7)$, where $g(K_7)$ results from applying function g to client C_G's previous node secret K_7. Each client extracts appropriate information from the messages. For example, members C_A, C_B, C_C and C_D extract the new blinded node key $f(K_3')$, while member C_E and C_F need to get the new blinded node key $f(K_7')$, and member C_G only needs to know the new blinded node key $f(K_{15})$ from the message.

Figure 5.2.2.2 shows the content of the OFT rekeying message the MAP sends to the new member C_H. The first four bytes are the same as those shown in Figure 5.2.2.2. The first two bytes indicate the node ID (7) of the joining point. The third and fourth bytes

indicate the node ID (14) of the new member's sibling. The next 18 bytes consist of C_H's node ID and the encrypted blinded node keys C_H needs to know. When C_H receives this message, it can use its node secret (individual key) K_{15} to decrypt the message and get the blinded node keys $f(K_{14})$, $f(K_6)$, and $f(K_2)$.

Given the example illustrated in Figure 5.5, after a member C_H leaves the group, member C_G changes its association from node 14 to node 7. Figure 5.10 shows the content of a OFT rekeying message broadcast by the GKS after C_H leaves the group. The first two bytes indicate the current node ID (14), which is also the node ID of member C_G who currently occupies this node and whose sibling leaves the group. The third and fourth bytes indicate the node ID of the leaving point, which is also the new node ID (7) of member C_G. The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and 16-byte encrypted key value. The 2-byte node ID identifies the node storing the key K that is used to encrypt one or more new keys. The 16-byte encrypted key value is obtained from encrypting one or more newly generated keys using key K. For example, $6, \{f(K_7')\}_{g(K_6)}$ indicates that the newly generated blinded node key $f(K_7')$ is encrypted with node key $g(K_6)$, where $g(K_6)$ is the result of applying hashing function g to the node secret K_6 of node 6.

7	14	2, $\{f(K_3')\}_{g(K_2)}$	6, $\{f(K_7')\}_{g(K_6)}$	14, $\{K_{14}, f(K_{15})\}_{g(K_7)}$

C_G's node

ID is changed	$\longleftarrow$	Updated keys	$\longrightarrow$

from 7 to 14.

(a) Broadcast message for client C_A to C_G after client C_H joins the group

7	14	15, $\{f(K_{14}), f(K_6), f(K_2)\}_{g(K_{15})}$

C_G's node

ID is changed	$\longleftarrow$ Updated keys $\longrightarrow$

from 7 to 14.

(b) Unicast message for client C_H after client C_H joins the group

Figure 5.9: OFT message example after client C_H joins the group

14	7	2, $\{f(K_3')\}_{g(K_2)}$	6, $\{f(K_7')\}_{g(K_6)}$	7, $\{K_7'\}_{g(K_{14})}$

C_G's node

ID is changed	$\longleftarrow$	Updated keys	$\longrightarrow$

from 14 to 7.

Broadcast message for client C_A to C_G after client C_H leaves the group.

Figure 5.10: OFT message example after client C_H leaves the group

5.3 Performance Evaluation

Real-time applications require very fast rekeying so that changes in group membership are not disruptive. In this section, we analyze and compare the rekeying latencies of the LKH and OFT algorithms with that of the IEEE 802.11 GKM scheme (which will be denoted as 802.11 in the remainder of this section).

We recall that in the IEEE 802.11 GKM scheme, a MAP shares a pairwise key with each mobile device, and additionally a group key with all the members of the multicast group. When a client joins (or leaves) the WMN, the MAP has to update the group key to ensure backward (or forward) secrecy. The MAP generates a new group key, encrypts it using the pairwise key the MAP shares with each trusted member, and sends the new group key to the members one by one. Thus, in this scheme the communication cost of a group key update is $O(n)$, where n is the number of associated members.

5.3.1 Computational Complexity

Let n be the number of members in a multicast group, and assume a proper binary key tree. The computational complexity is measured in terms of the complexity of encryption, decryption and hashing for a rekeying operation.

Table 5.3 summarizes the computational complexity of the 802.11, LKH and OFT algorithms, which shows that 802.11 takes $O(n)$ time while LKH and OFT both take $O(\log n)$ time. Following is a brief explanation of how to derive the computation complexities given in Table 5.3.

Table 5.3: Computational complexity

Operation		802.11	LKH	OFT
Join	Encryption	n	$\log_2 n + 1$	$\log_2 n + 1$
	Decryption	1	$\log_2 n$	$\log_2 n$
	Hashing	–	–	$3 \log_2 n + 1$
	Total	$n \times E + D$	$(\log_2 n + 1) \times E$ $+ \log_2 n \times D$	$(\log_2 n + 1) \times E$ $+ \log_2 n \times D$ $+(3 \log_2 n + 1) \times H$
	Complexity	O(n)	O($\log_2 n$)	O($\log_2 n$)
Leave	Encryption	n	$2 \log_2 n$	$\log_2 n + 1$
	Decryption	1	$\log_2 n$	1
	Hashing	–	–	$2 \log_2 n + 1$
	Total	$n \times E + D$	$2 \log_2 n \times E$ $+ \log_2 n \times D$	$(\log_2 n + 1) \times E$ $+D + (2 \log_2 n + 1) \times H$
	Complexity	O(n)	O($\log_2 n$)	O($\log_2 n$)

In the 802.11 algorithm, every time the MAP updates the group key, it has to encrypt the group key n times using each member's individual key, and then each member performs one decryption after receiving the group key, hence $O(n)$ time for the join operation and also for the leave operation.

In the LKH algorithm,

- when a client joins the group, the MAP updates $\log_2 n$ keys along the path from the joining point to the root. Thus, the MAP performs $\log_2 n$ encryptions for the existing members (step 3 in Section 5.1.2.1). The MAP also performs one encryption using the new member's individual key (step 2 in Section 5.1.2.1). Thus, the MAP performs $\log_2 n+1$ encryptions. When performing the numerical analysis, we assume that all members receive the message and decrypt it at the same time (in parallel). We alleviate this assumption in simulations, when we consider realistic conditions. Since the new member has to know all the keys along the path from itself to the root, the new member needs to perform at most $\log_2 n$ decryptions (step 2 in Section 5.1.2.1). The total computation latency is $(\log_2 n+1)\times E+\log_2 n\times D$ as shown in Table 5.3, where E is the computation time for one encryption operation, and D is the computation time for one decryption operation.

- when a member leaves the group, the MAP updates $\log_2 n$ keys along the path from the leaving point to the root in order to prevent the leaving member from accessing future keys. The MAP encrypts the updated key K_i using the key of the left child of node i to obtain $\{K_i\}_{K_{2i}}$, and sends it to the leaf nodes in the left

subtree rooted at node i. The MAP also encrypts the updated key K_i using the key of the right child of node i to obtain $\{K_i\}_{K_{2i+1}}$, and sends it to the leaf nodes in the right subtree rooted at node i. Thus, the MAP performs $2\log_2 n$ encryptions for the remaining members (step 2 in Section 5.1.2.2). We assume all members perform the decryption operations in parallel. Since $\log_2 n$ keys are updated, each member performs at most $\log_2 n$ decryptions (step 2 in Section 5.1.2.2). Thus, the total computation complexity is $(2\log_2 n) \times E + \log_2 n \times D$ as shown in Table 5.3.

In the OFT algorithm,

- when a client joins the group, the MAP updates $\log_2 n$ keys along the path from the joining point to the root. Thus, the encryption and decryption complexities are same as those in LKH (step 2 and 3 in Section 5.1.2.1). However, OFT employs hash functions f and g for the group key computation as discussed in Section 5.1.3.1. We assume the clients and the MAP perform the hash operations in parallel. The clients and the MAP perform a total of $3\log_2 n + 1$ hashing operations as shown in Table 5.4 when a new member joins the group. The total computation latency is $(\log_2 n + 1) \times E + \log_2 n \times D + (3 \times \log_2 n + 1) \times H$ as shown in Table 5.3.

- when a member leaves the group, the MAP recalculates $\log_2 n + 1$ node keys along the path from the leaving point to the root. Since each member needs to know the updated blinded node keys to recompute the new group key, the MAP preforms $\log_2 n + 1$ encryptions for the updated blinded node keys (step 3 in Section 5.1.3.3). Each member only needs to decrypt once to get its required corresponding blinded

Table 5.4: Number of hash operations in OFT

Description		$f(K)$	$g(K)$	Total
Join	Client	1	1	2
	MAP	$2\log_2 n$	$\log_2 n + 1$	$3\log_2 n + 1$
Leave	Client	1	1	2
	MAP	$\log_2 n$	$\log_2 n + 1$	$2\log_2 n + 1$

node keys (step 3 in Section 5.1.3.3). We assume the clients and the MAP perform hash operations in parallel. The clients and the MAP perform a total of $2\log_2 n+1$ hashing operations as shown in Table 5.4. The total computation latency is $(\log_2 n+1) \times E + 1 \times D + (2\log_2 n + 1) \times H$ as shown in Table 5.3.

5.3.2 Communication Complexity

The communication complexity is measured in terms of the number of keys to be broadcast and unicast during a rekeying operation. More keys imply a longer message and thus longer transmission time.

Table 5.5 summarizes the communication complexity of the join and leave operations incurred by the 802.11, LKH and OFT algorithms. Following is an explanation of the communication complexities given in Table 5.5.

In the 802.11 algorithm, every time the MAP updates the group key, it has to send (unicast) the new key to the members one by one. Hence $O(n)$ time is required for both join and leave operations.

Table 5.5: Communication complexity

Operation/ Algorithm		Unicast (number of keys)	Broadcast (number of keys)	Total	Complexity
Join	802.11	n	–	n	O(n)
	LKH	$\log_2 n$	$\log_2 n$	$2\log_2 n$	O($\log_2 n$)
	OFT	$\log_2 n$	$\log_2 n + 1$	$2\log_2 n + 1$	O($\log_2 n$)
Leave	802.11	n	–	n	O(n)
	LKH	–	$2\log_2 n$	$2\log_2 n$	O($\log_2 n$)
	OFT	–	$\log_2 n + 1$	$\log_2 n + 1$	O($\log_2 n$)

In the LKH algorithm,

- when a client joins the group, the MAP broadcasts a message containing $\log_2 n$ updated keys to the group (step 3 in Section 5.1.2.1). The MAP also unicasts a message containing $\log_2 n$ updated keys to the new member (step 2 in Section 5.1.2.1). The total number of updated keys is $2\log_2 n$.

- when a member leaves the group, the MAP broadcasts a message containing $2\log_2 n$ updated keys to the group (step 2 in Section 5.1.2.2). The total number of updated keys is $2\log_2 n$.

In the OFT algorithm,

- when a client joins the group, the MAP broadcasts a message containing $\log_2 n$ new blinded node keys to the group (step 3 in Section 5.1.3.2). The new key assigned

to the sibling node of the new member is also part of the broadcast message, which therefore contains a total of $\log_2 n+1$ keys. In addition, the MAP unicasts a message containing $\log_2 n$ new blinded node keys to the new member. The total number of updated keys is $2\log_2 n + 1$.

- when a member leaves the group, the MAP broadcasts a message containing $\log_2 n$ new blinded node keys to the group (step 3 in Section 5.1.3.3). The new key assigned to the modified leaf node is also part of the broadcast message, which contains a total of $\log_2 n + 1$ keys. The total number of updated keys is $\log_2 n + 1$.

5.3.3 Numerical Analysis

In this section, we derive the group key update latency functions for the 802.11, LKH and OFT algorithms.

The rekeying latency is defined as the interval starting when the MAP receives a join/leave request from a member and initiates the rekeying process, and ending when all members receive or finish computing the new group key.

5.3.3.1 System Model

Given the message format in Figure 5.6 and the number of keys K carried by the MSDU (media access control service data unit) , the size of an MSDU in bytes is

$$MSDU(K) = 4 + 18K \tag{5.2}$$

The 802.11, 802.11a and 802.11b standards specify a maximum MSDU size of 2304 bytes [113, 114, 115]. One message can store at most 127 updated keys given $MSDU(K) = 4+18K$ for our message structure designed in Section 5.2.1.2, which means it can support up to $2^{127} - 1$ keys in a full binary key tree. Hence, $2^{127} - 1$ as the maximum number of members for a multicast group in a WMN is more than enough.

We use the IEEE 802.11 protocol with distributed coordination function (DCF) for medium access control. Multicast and broadcast transmissions use CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). Unicast transmissions also use RTS/ CTS /DATA /ACK exchanges in addition to CSMA/CA. We consider two spread spectrum technologies: direct sequence spread spectrum (DSSS) and orthogonal frequency division multiplexing (OFDM). DSSS supports data rates of 1 Mbit/s, 2 Mbits/s, 5.5 Mbits/s and 11 Mbits/s while OFDM can support data rate of up to 54 Mbits/s. In our evaluation and experiments, we choose a transmission rate at the physical layer of 2 Mbits/s for DSSS to represent a low speed network and 54 Mbits/s for OFDM to represent a high speed network, respectively.

We also make the following assumptions in the numerical analysis:

- No bit error.
- No losses due to collision.
- No packet loss due to buffer overflow at receiver nodes.
- No fragmentation at the data link layer.

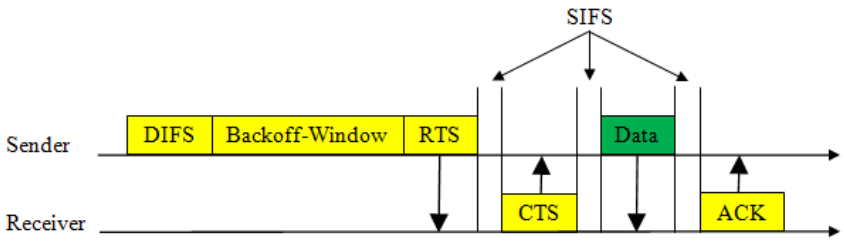
- No management frame considered (e.g. beacon, association frames).

The contention window size (cw) does not increase exponentially since we assume that there are no collisions. Thus, cw is always equal to the minimum contention window size (cw_{min}) whose value depends on the spread spectrum technology. The backoff time is selected randomly following a uniform distribution from (0, cw_{min}) according to IEEE 802.11, resulting in an expected value of $cw_{min}/2$. The minimum size of the contention window, as defined in the IEEE 802.11g standard, is dependant on the requestor's characteristic rate. For DSSS, the minimum size of the contention window cw_{min} is equal to the length of 31 time slots as defined in IEEE 802.11b [116], where each slot length is 20μs. Thus, on an average, the number of back-off slots between successive transmissions is 15.5, which when multiplied by the slot time of 20μs, yields 310μs. For OFDM, the minimum size of the contention window cw_{min} is equal to the length of 15 time slots as defined in IEEE 802.11g [117], where each time slot length is 9 μs. Thus, on average, the number of back-off slots between successive transmissions is 7.5, which yields 67.5 μs.

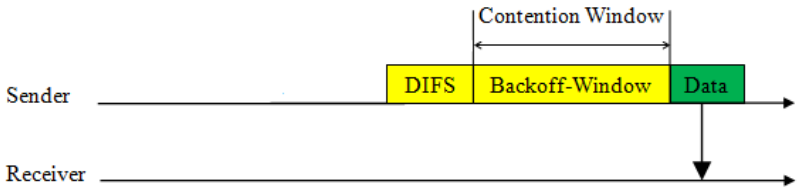
The rekeying latency upon a join or leave operation consists of two costs: communication cost and computation cost. We discuss the calculations of these two costs next.

5.3.3.2 Communication Costs

Figure 5.11(a) shows the sequence of messages exchanged to complete a unicast transmission at the data link layer. According to the diagram, the latency $T_u(K)$ incurred by a successful wnicast transmission is calculated as follows:



(a) Unicast



(b) Broadcast

DIFS: distributed inter-frame space; SIFS: short inter-frame space;

RTS: request to send; CTS: clear to send; ACK: acknowledgment

Figure 5.11: Unicast and broadcast message exchanges at the data link layer

$$T_u(K) = T_{DIFS} + T_{BO} + T_{RTS} + T_{CTS} + 3T_{SIFS} + T_{Data}(K) + T_{ACK}$$

The definition and value of each variable in the above function is listed in Table 5.6. The values are taken from [113]– [115]. As shown in this table, the latencies incurred by the request to send (RTS), clear to send (CTS), distributed inter-frame space (DIFS), short inter-frame space (SIFS), and acknowledgment (ACK) frames, and the average back-off time are constant. $T_{Data}(K)$ is the latency incurred by the rekeying message. This latency depends on the size of the message, and thus on the number of keys K stored in the message. The calculation of $T_{Data}(K)$ will be shown in Section 5.3.3.5.

Broadcast messages in IEEE 802.11 media access control do not use RTS, CTS or ACK [134]. Following is the latency $T_b(K)$ incurred by a successful broadcast transmission, according to vhe diagram illustrating the timeline of a broadcast transmission at the data link layer in Figure 5.11(b).

$$T_b(K) = T_{DIFS} + T_{BO} + T_{Data}(K)$$

5.3.3.3 Computation Costs

Let E, D and H denote the latency of an encryption, decryption and hashing operation, respectively.

Using the encryption, decryption, and hashing costs listed in Table 5.3, we obtain the computation costs of the join and leave operations for the three algorithms, which are listed in Table 5.7.

Table 5.6: Latency Components

Parameter	Description	Value	
Spread spectrum technology		OFDM [113]– [115]	DSSS [113]– [115]
T_{SIFS}	Latency of short interframe space	9 μs [138]	10 μs [138]
T_{DIFS}	Latency of distributed interframe space	34 μs	50 μs
T_{BO}	Latency of backoff	67.5 μs	310 μs
T_{RTS}	Latency of request to send	24 μs	352 μs
T_{CTS}	Latency of clear to send	24 μs	304 μs
T_{ACK}	Latency of acknowledgement	24 μs	304 μs
$T_{Data}(K)$	!Latency for transferring		
K	Number of keys	–	
E	Encryption	$2.1ms$ [101]	
D	Decryption	$2.2ms$ [101]	
H	Hashing	$0.009ms$ [73]	

Table 5.7: Rekeying Latency

Operation/ Algorithm		Communication		Computation		
		Unicast	Broadcast	Encryption	Decryption	Hashing
Join	802.11	$T_u(n)$	–	$n \times E$	D ‘	–
	LKH	$T_u(\log_2 n)$	$T_b(\log_2 n)$	$(\log_2 n + 1) \times E$	$\log_2 n \times D$	–
	OFT	$T_u(\log_2 n)$	$T_b(\log_2 n + 1)$	$(\log_2 n + 1) \times E$	$\log_2 n \times D$	$(3 \log_2 n + 2) \times H$
Leave	802.11	$T_u(n)$	–	$n \times E$	D	–
	LKH	–	$T_b(2 \log_2 n)$	$2log_2 n \times E$	$\log_2 n \times D$	–
	OFT	–	$T_b(\log_2 n + 1)$	$(\log_2 n + 1) \times E$	D	$(2 \log_2 n + 2) \times H$

5.3.3.4 Total Rekeying Latency

Functions T_u and T_b are defined in Section 5.3.3.2 as the latencies incurred by a successful unicast transmission and a successful broadcast transmission, respectively. The arguments of the T_u and T_b functions are the numbers of keys stored in the messages as listed in Table 5.5. Table 5.7 shows the communication costs incurred by the three algorithms (see columns “Unicast” and “Broadcast” in Table 5.5).

The rekeying latency − a combination of computation and communication costs listed in Table 5.7 − incurred by a join operation in each of the three algorithms is as follows:

$$\begin{aligned} J_{802.11} &= T_u(n) + n \times E + 1 \times D \\ J_{LKH} &= T_u(\log_2 n) + T_b(\log_2 n) + (\log_2 n + 1) \times E + \log_2 n \times D \\ J_{OFT} &= T_u(\log_2 n) + T_b(\log_2 n + 1) + (\log_2 n + 1) \times E + \log_2 n \times D \\ &+ (3 \log_2 n + 1) \times H \end{aligned} \tag{5.3}$$

The rekeying latencies incurred by a leave operation are as follows:

$$\begin{aligned} L_{802.11} &= T_u(n) + n \times E + 1 \times D \\ L_{LKH} &= T_b(2\log_2 n) + 2\log_2 n \times E + \log_2 n \times D \\ L_{OFT} &= T_b(\log_2 n + 1) + (\log_2 n + 1) \times E + 1 \times D + (2\log_2 n + 1) \times H \end{aligned} \quad (5.4)$$

To increase the reliability of broadcast messages, we can broadcast a message multiple times. In our implementation of the LKH and OFT algorithms, we borrow the idea from the Internet Group Management Protocol (IGMP) [135, 136], where a message is broadcast three times for enhanced reliability. We denote LKH and OFT with this implementation as LKH-3 and OFT-3, respectively. Following are the rekeying latencies of the LKH-3 and OFT-3 algorithms in the 3-time broadcast implementation. (In the 802.11 algorithm, the MAP unicasts the messages to each group member. Delivery reliability of each unicast message is ensured by the RTS/CTS/DATA/ACK exchange.)

$$\begin{aligned} J_{LKH-3} &= T_u(\log_2 n) + 3 \times T_b(\log_2 n) + (\log_2 n + 1) \times E + \log_2 n \times D \\ L_{LKH-3} &= 3 \times T_b(2\log_2 n) + 2\log_2 n \times E + \log_2 n \times D \\ J_{OFT-3} &= T_u(\log_2 n) + 3 \times T_b(\log_2 n + 1) + (\log_2 n + 1) \times E \\ &+ \log_2 n \times D + (3\log_2 n + 1) \times H \\ L_{OFT-3} &= 3 \times T_b(\log_2 n + 1) + (\log_2 n + 1) \times E + 1 \times D \\ &+ (2\log_2 n + 1) \times H \end{aligned} \quad (5.5)$$

5.3.3.5 Performance Graphs

To visualize the performance comparison of the three algorithms as given by the mathematical equations (3), (4), and (5), we plotted graphs of the above rekeying latency functions $J_{802.11}$, J_{LKH}, J_{LKH-3}, J_{OFT}, and J_{OFT-3}.

The numerical data for plotting the graphs are given in Table 5.6. We use the AES encryption algorithm [137] with 128-bit keys. The wireless transmission delay components such as DIFS, SIFS, RTS, CTS and ACK are from the IEEE 802.11 standards [113]− [115]. The transmission time $T_{Data}(K)$ of an MSDU depends on its size (or the number of keys it stores) and the data rate at the physical layer (which is determined by the spread spectrum technology), and is calculated as follows [114].

For OFDM,

$$\begin{aligned} T_{Data}(K)_{OFDM} &= T_{PREAMBLE} + T_{Signal} + T_{SYM} \\ &\times \lceil \frac{16 + 6 + 8 \times (34 + MSDU(K))}{N_{DBPS}} \rceil \end{aligned} \tag{5.6}$$

where the service field of the physical layer header is 16 bits long; the PSDU tail of the physical layer header is 6 bits long; the maximum media access control header length is 34 bytes, and $MSDU(K)$ is given by Eq. (2). By multiplying $(MSDU(K)+34)$ by eight, we convert the maximum media access control header length and MSDU from bytes to bits. The values of $T_{PREAMBLE}$, T_{Signal}, T_{SYM}, and N_{DBPS} are from the IEEE 802.11 standards and provided in Table 5.8.

After substituting the above values into Eq. (6), we obtain:

Table 5.8: Timing Related Parameters

Parameter	Description	OFDM	DSSS
$T_{PREAMBLE}$	PLCP preamble duration	16 μs	144 μs
T_{SIGNAL}	Duration of the SIGNAL BPSK-OFDM symbol	4 μs –	48 μs –
T_{SYM}	Symbol interval	4 μs	–
N_{DBPS}	Data bits per OFDM symbol	216 μs –	– –
R_{Data}	Data Rate	54 Mbits/s	2 Mbits/s

$$T_{Data}(K)_{OFDM} = 20 + 4 \times \lceil \frac{163 + 72K}{108} \rceil \tag{5.7}$$

For DSSS,

$$\begin{aligned} T_{Data}(K)_{DSSS} &= T_{PREAMBLE} + T_{Signal} \\ &+ \frac{8 \times (34 + MSDU(K))}{R_{Data}} \end{aligned} \tag{5.8}$$

After substituting the appropriate values given in Table 5.8 into Eq. (8), we obtain:

$$T_{Data}(K)_{DSSS} = 496 + 144K \tag{5.9}$$

After substituting $T_{Data}(K)$ (defined in Eq.(7) or Eq.(9)), $T_u(K)$ and $T_b(K)$ (defined in section 5.3.3.2), and the appropriate values into the total rekeying latency functions

Eq.(3)-(5) (defined in Section 5.3.3.4), we plotted the graphs as functions of n, where n is the number of clients in a multicast/broadcast group controlled by a MAP.

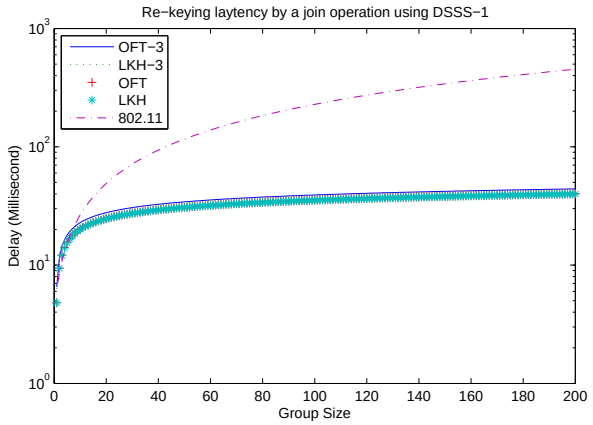
Figures 5.12(a) and (b) illustrate the latency functions of the join operation using DSSS and OFDM, respectively, on a log scale. To demonstrate the linear function of 802.11 and logarithmic behaviors of LKH and OFT functions, we magnify the above graphs for n values from 1 to 50. The magnified graphs are shown in Figures 5.12(c) and (d).

Similarly, the latency functions of the leave operation using DSSS and OFDM are illustrated by the graphs in Figures 5.13(a) & (b), respectively. The corresponding magnified graphs are given for $n = 1$ to 50 in Figures 5.13(c) & (d).

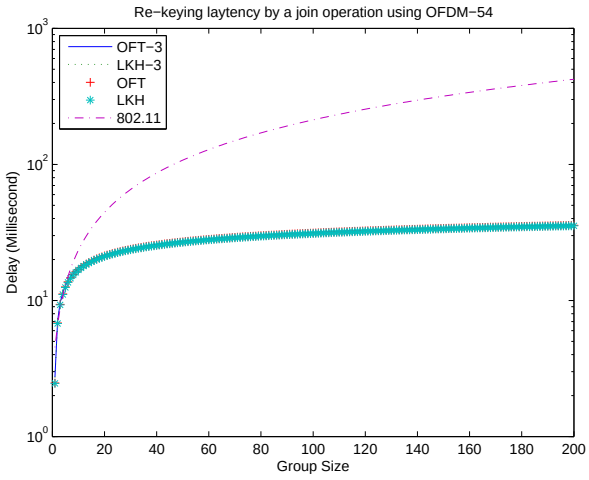
5.3.3.6 Discussion

From the above graphs, we draw the following conclusions.

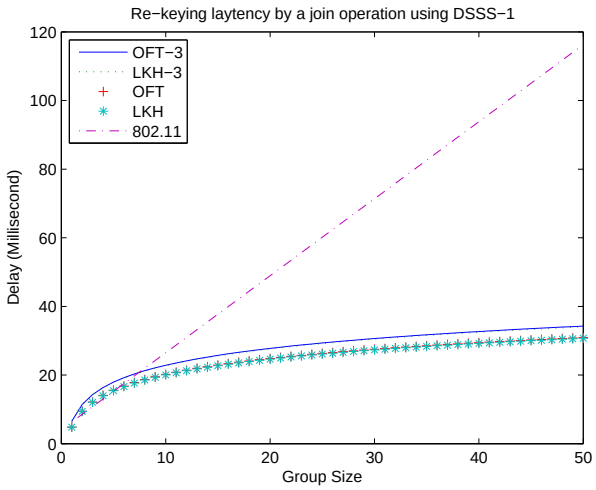
- The join rekeying latencies of LKH and OFT are comparable, and much lower than that of IEEE 802.11 algorithm when $n > 5$.
- The leave rekeying latency of OFT is slightly better than that of LKH. The reason is that the OFT algorithm incurs 50% less bits to be broadcast for the key update required by a leave operation [14]. Both algorithms, given their logarithmic running time, perform better than the linear function of IEEE 802.11 when $n > 7$.
- Although a message is broadcast three times in the J_{LKH-3}, J_{OFT-3}, L_{LKH-3} and



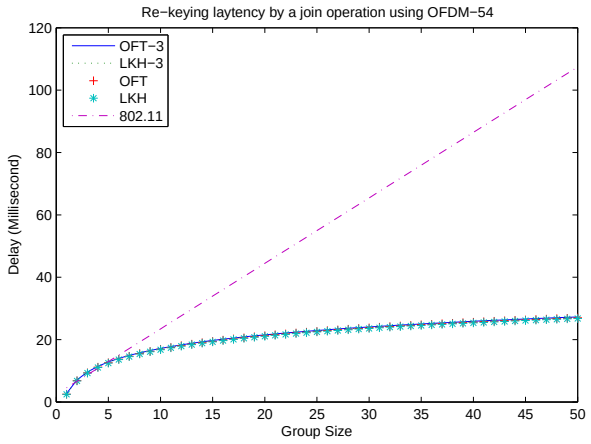
(a) Join - DSSS



(b) Join - OFDM

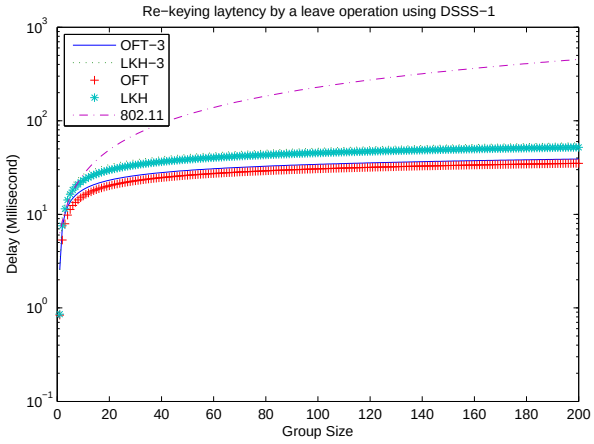


(c) Join - DSSS (magnified)

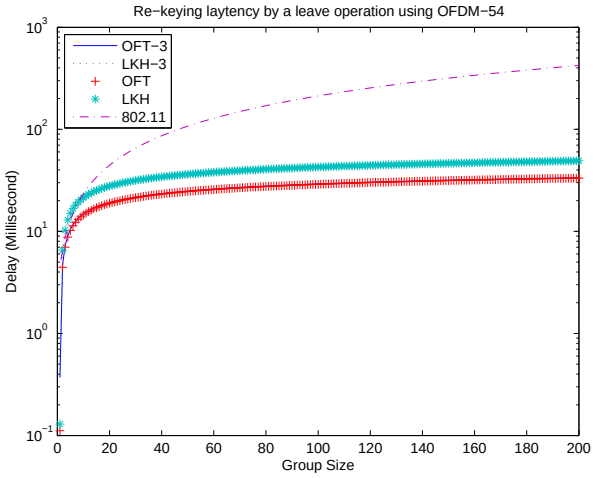


(d) Join - OFDM (magnified)

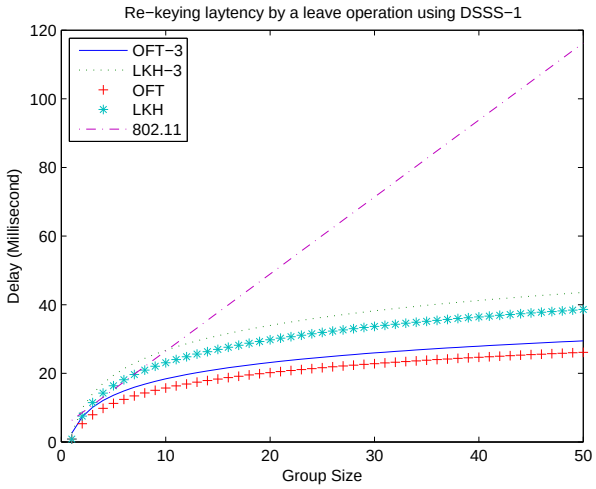
Figure 5.12: Rekeying latency incurred by a JOIN operation



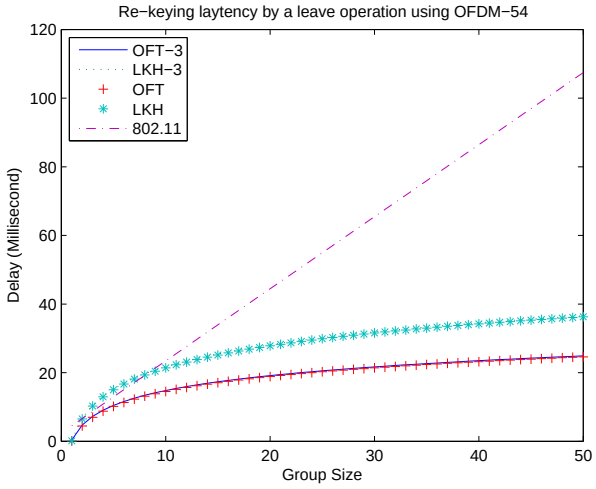
(a) Leave - DSSS



(b) Leave - OFDM



(c) Leave - DSSS (magnified)



(d) Leave - OFDM (magnified)

Figure 5.13: Rekeying latency incurred by a LEAVE operation

L_{OFT-3} functions for enhanced reliability, the increase of the rekeying latency of this implementation is very minor in the theoretical analysis.

5.4 Simulation Results

The above numerical analysis shows that the LKH and OFT algorithms can reduce the rekeying latency of the current scheme used in the IEEE 802.11 GKM from linear time to logarithmic time. We further evaluate and compare the performance of the 802.11, LKH and OFT algorithms under realistic network settings using simulations. We use QualNet version 5.2, a commercial software that provides scalable simulations of wireless networks [102], for our experiments.

5.4.1 Performance Metric

The performance metric is the *rekeying latency* (delay), which is defined as the interval starting when a MAP receives a join/leave request from a member and initiates the rekeying process, and ending when *all* members receive and finish computing the new group key. To enhance the reliability of rekeying messages, the MAP sends each broadcast message three times. We will show how this approach affects the performance of rekeying latency of LKH and OFT in comparison with the one time broadcast scheme. In addition, our experiments demonstrate how background traffic can affect the rekeying latency of the 802.11, LKH and OFT protocols.

5.4.2 Simulation Parameters

A mesh access point such as Motorola WAP 400 AP could support up to 256 active clients [139]. Thus, we conducted our simulation using two group sizes: one has 50 members and the other has 200 members, which are representatives of a moderately busy and highly busy MAP, respectively. The simulation parameters of all experiments are listed in Table 5.9. The simulated time is 150s. The transmission range of the wireless router (MAP) is 315m, according to the specifications of wireless routers manufactured by Tropos [103]. The transmission range of clients is 304m, according to the specifications of wireless adapters manufactured by Cisco [105]. The node mobility is random way point. The mobility speed of each member is 10m/s. Each data point in the graph is the average of 10 runs using 10 random seeds. The graphs are plotted with a confidence interval of 95%. In all the experiments, the mobile clients are randomly placed in a 300m x 300m area while a MAP is placed in the center of the area. Thus, any mobile client can reach the MAP in the 300m x 300m area.

We use the IEEE 802.11b and IEEE 802.11a configurations specified in Qualnet to run simulations for DSSS and OFDM, respectively. DSSS supports data rates of 1 Mbit/s, 2 Mbits/s, 5.5 Mbits/s and 11 Mbits/s while OFDM supports data rate of up to 54 Mbits/s. For comparison purposes, in our experiments we choose a low transmission rate of 2 Mbits/s and a high transmission rate of 54 Mbits/s at the physical layer for DSSS and OFDM respectively, as shown in Table 5.10.

For each type of spread spectrum technology, DSSS and OFDM, we conduct three

Table 5.9: Simulation parameters

Parameter	Value
Transmission range of MAPs	315m
Transmission range of clients	304m
Movement model	Random way point
Client mobility speed	10m/s
Simulation time	150s
Network dimensions	300m x 300m
Number of runs per data point	10
Confidence interval	95%

Table 5.10: IEEE 802.11 standards used in the experiments

Spread spectrum technology	Physical layer Protocol	Transmission rate at physical layer
DSSS	PHY802.11b	2 Mbits/s
OFDM	PHY802.11a	54 Mbits/s

sets of experiments to measure the rekeying latency as a function of

(1) *multicast group size.* We measure the average rekeying latency of the protocols when a new member joins or an existing member leaves a group. We conducted simulations for both join and leave operations with group sizes of 50 and 200 clients. For each group size S, we measure the average latency by: (a) varying S from 1 to 50 for the smaller group, (b) varying S from 1 to 200 for the larger group. We repeat the same experiments for the leave operation. For each data point in a graph, we ran an experiment 10 times using 10 different random seeds and obtained the the average rekeying latency.

(2) *multicast group size with enhanced broadcast reliability.* Unicast uses CSMA/CS and RTS/CTS/ACK to enhance the reliability of a data transmission, while broadcast does not use RTS/CTS/ACK. To enhance the reliability of broadcast, each rekeying message is broadcast three times. We repeat the experiment (1) described above, but broadcast each rekeying message three times.

(3) *background traffic load.* Since the MAP has to update the keys when a member joins/leaves the group, the total workload of the MAP directly impacts the group key rekeying latency. Thus, we measure the average rekeying latency of the protocols in the presence of background traffic send to the MAP, to get a more practical measurement of realistic scenarios. In each experiment, an additional client outside the group is placed in the network as a source to transmit background traffic to the MAP at a

specified constant bit rate (CBR). This client does not count as part of the group size S. We vary the CBR from 0 to 2 Mbits/s for DSSS and 0 to 50 Mbits/s for OFDM in our simulation. The data rate of standard video streams, such as MPEG-2, can be up to 50 Mbits/s [140]. In this experiment, each members joins (or leaves) the group one after another. For example, in the case of the smaller group with 50 members, we measure the rekeying latency as a function of background traffic load when the 50^{th} member joins the group. In the case of the larger group with 200 members, we measure the rekeying latency when the 200^{th} member joins the group.

Table 5.11 summarizes the important parameters and lists the figures containing the graphs of the DSSS and OFDM experiments.

5.4.3 Simulation Results of DSSS

Following is a detailed discussion of the experimental results with DSSS.

5.4.3.1 Function of Multicast Group Size

The rekeying latency of the join operation of the protocols as functions of the group size is given in Figure 5.14(a) for group size S varying from 1 to 50 and in Figure 5.14(b) for group size S varying from 1 to 200. We observe that the join rekeying latencies of LKH and OFT are comparable, and much lower than that of the 802.11 algorithm when the number of members $n > 5$ because the rekeying latency of 802.11 increases linearly, while that of LKH and OFT increases logarithmically. As the number of joining nodes

Table 5.11: Simulation Results

Experiment		DSSS	OFDM	Network
(1) Function of multicast group size	Join	Figures 5.14 (a) Figures 5.14 (c)	Figures 5.17 (a) Figures 5.17 (c)	300m x 300m 50 or 200 nodes Node mobility is 10m/s DSSS transmission rate is 2 Mbits/s OFDM transmission rate is 54 Mbits/s
	Leave	Figures 5.14 (b) Figures 5.14 (d)	Figures 5.17 (b) Figures 5.17 (d)	
(2) Function of multicast group size with enhanced reliability	Join	Figures 5.15 (a) Figures 5.15 (c)	Figures 5.18 (a) Figures 5.18 (c)	
	Leave	Figures 5.15 (b) Figures 5.15(d)	Figures 5.18 (b) Figures 5.18 (d)	
(3) Function of background traffic load	Join	Figures 5.16 (a) Figures 5.16 (c)	Figures 5.19 (a) Figures 5.19 (c)	
	Leave	Figures 5.16 (b) Figures 5.16 (d)	Figures 5.19 (b) Figures 5.19 (d)	

increases, the performance gap between LKH (OFT) and 802.11 enlarges.

For example, in the case of the smaller multicast group of 50 nodes, the maximum latencies of LKH, OFT, and 802.11 are 32.2ms, 32.8ms, and 110.7ms, respectively. In the case of the larger group of 200 nodes, they are 38.1ms, 40.9ms, and 441.2ms, respectively.

The rekeying latency of the leave operation of the protocols as a function of the group size is given in Figure 5.14(c) for group size S varying from 1 to 50 and in Figure 5.14(d) for S varying from 1 to 200. The leave rekeying latency of OFT is approximately 18% to 25% lower than that of LKH. The reason is that the OFT algorithm requires 50% less bits to be broadcast for the key update caused by a leave operation [14]. Again, both OFT and LKH rekeying latency is better than that of the 802.11 algorithm. The rekeying latency of LKH is much lower than that of 802.11 when the number of members $n > 7$. In Figure 5.14(c), LKH is approximately 3.1% to 61.8% lower than IEEE 802.11 when the number of members $n \geq 7$. OFT is approximately 0.9% to 79.8% lower than IEEE 802.11 when $n \geq 1$.

Comparing with the numerical analysis in Section 5.3.3, both LKH and OFT, given their logarithmic running time, perform better than the linear function of 802.11. Thus, these simulation results are consistent with our numerical analysis.

5.4.3.2 Function of Multicast Group Size with 3-time Broadcast

We repeated the above experiment, but each rekeying message was broadcast three times for enhanced reliability.

Figure 5.15 (a) and Figure 5.15 (b) show the join rekeying latencies of 802.11, LKH, and OFT. As 802.11 is unicast in nature, we transmit the 802.11 rekeying message only one time. We transmit the broadcast messages of both LKH and OFT three times each to enhance reliability. We observe that the join rekeying latencies of LKH and OFT are comparable when the broadcast message is sent three times (denoted by LKH-3 and OFT-3 in the graphs). The latency of LKH-3 (OFT-3) is obviously longer than that of its one time broadcast counterpart denoted by LKH (OFT).

In Figure 5.15 (a), when the group size S varies from 1 to 50, the latency of OFT is approximately 29% to 48% lower than that of OFT-3. The latency of LKH is approximately 25% to 47% lower than that of LKH-3. In Figure 5.15 (b), when the group size S varies from 1 to 200, the latency of OFT (LKH) is approximately 25% to 37% (25% to 32%) lower than that of OFT-3 (LKH-3).

We observe that the latencies of LKH-3 and OFT-3 are much lower than that of 802.11 when the number of members $n > 10$. We observe here that when considering smaller group sizes (i.e. $n < 10$), all 3 algorithms perform similarly. 802.11 observably does a bit better due to less overhead. However, for larger group sizes the performance of 802.11 worsens and is not efficient because the communication overhead starts to become greater than the computation overhead. As the number of joining nodes increases, the

performance gap between the LKH-3 (OFT-3) and 802.11 enlarges. In Figure 5.15 (a), the latencies of LKH-3 (OFT-3) is 43.3% (45%) lower than that of 802.11. In Figure 5.15 (b), LKH-3 (OFT-3) is 83.7% (83.1%) lower than that of 802.11.

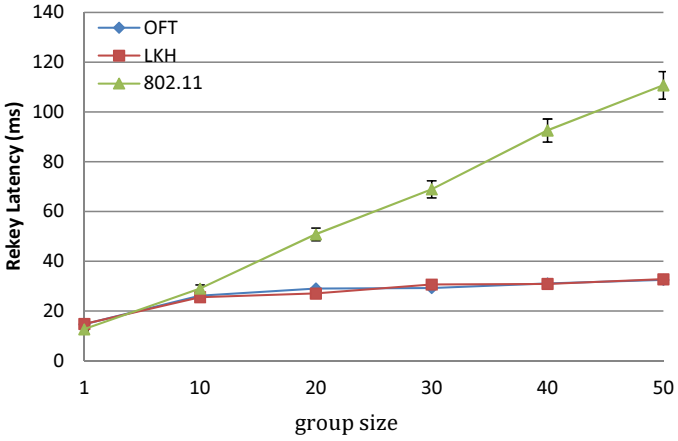
The rekeying latency of the leave operation of the protocols using three-time broadcast is given in Figures 5.15 (c) and (d) for group size varying from 1 to 50 and from 1 to 200 , respectively. Similar to the case of one-time broadcast shown in Figure 5.14 (c) and (d), the rekeying latency of OFT-3 is slightly lower than that of LKH-3. The latency of OFT-3 is approximately 18% to 23% longer than that of OFT. The latency of LKH-3 is approximately 13% to 19% longer than that of LKH.

We observe that the latencies of LKH-3 and OFT-3 are much lower than that of the 802.11 when the number of members $n > 20$. As the number of leaving nodes increases, the performance gap between the LKH-3 (OFT-3) and 802.11 enlarges. In Figure 5.15 (c), LKH-3 (OFT-3) is 53% (62%) lower than 802.11. In Figure 5.15 (c), LKH-3 (OFT-3) is 84% (87%) lower than 802.11.

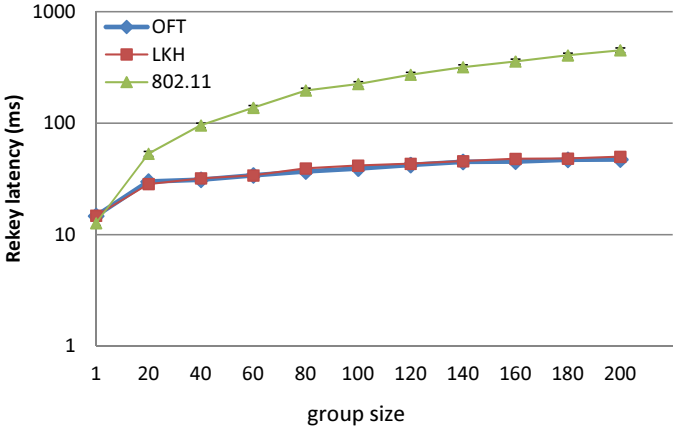
Thus, with 3-time broadcase, the rekeying latency for both join and leave operations is slightly higher than that of one-time broadcast, but is still much better than that of 802.11 and additionally provides reliability in broadcasting messages.

5.4.3.3 Function of Background Traffic Load

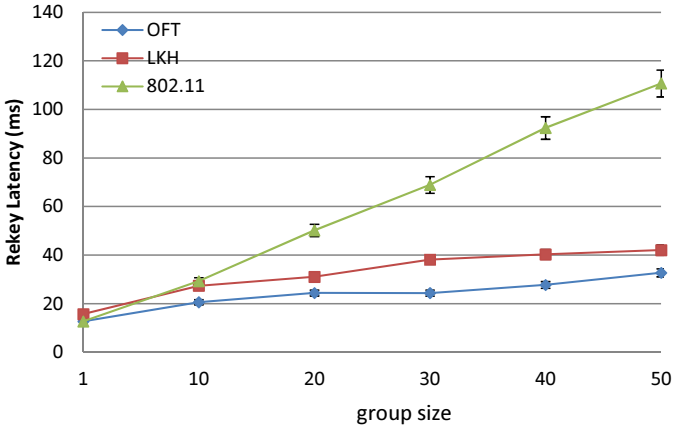
As described in experiment (3) in Section 5.4.2, an additional node is placed in the network as a source to transmit background traffic to the MAP at a specified CBR. The



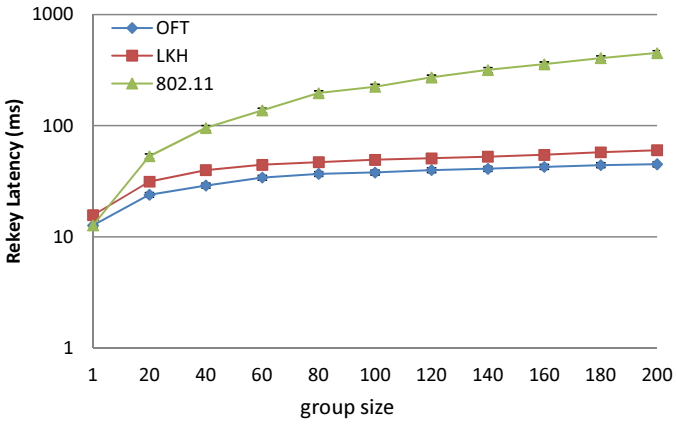
(a) DSSS - Group size from 1-50 - Join



(b) DSSS - Group size from 1-200 - Join

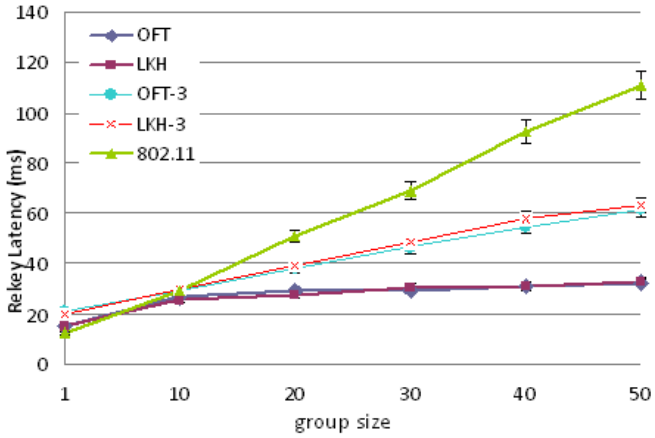


(c) DSSS - Group size from 1-50 - Leave

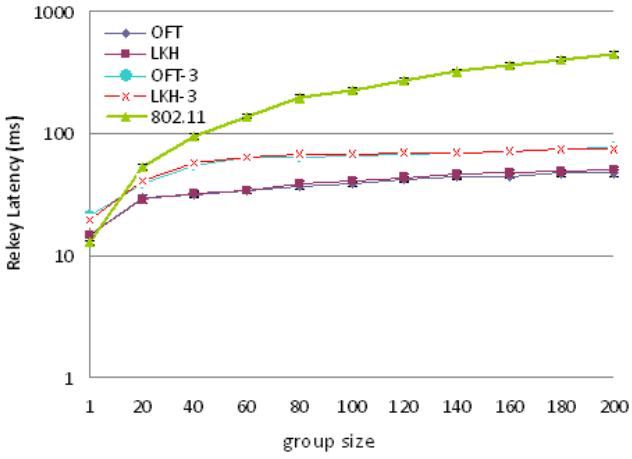


(d) DSSS - Group size from 1-200 - Leave

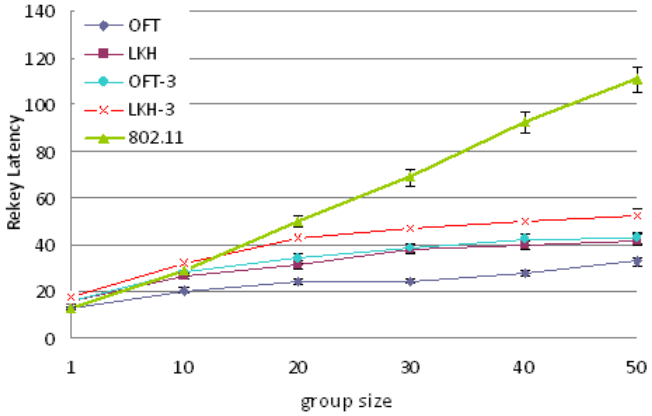
Figure 5.14: DSSS - Rekeying latency of Join/Leave operation



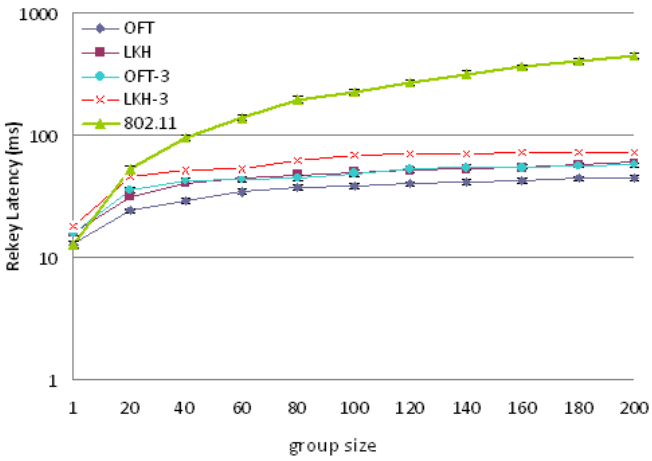
(a) DSSS - Group size from 1-50 - Join



(b) DSSS - Group size from 1-200 - Join



(c) DSSS - Group size from 1-50 - Leave



(d) DSSS - Group size from 1-200 - Leave

Figure 5.15: DSSS - Each broadcast message is sent 3 times

CBR varies from 0 to 2 Mbits/s for DSSS. In this set of experiments, each members joins or leaves the group one after another. For the smaller (larger) group with 50 (200) members, we measure the rekeying latency as a function of background traffic load when the 50^{th} (200^{th}) member joins (leaves) the group.

The graphs in Figure 5.16(a) (Figure 5.16 (b)) shows the average rekeying latency when the 50^{th} (the 200^{th}) member joins the smaller (larger) group with the data rate of type CBR varying from 0 to 2Mbits/s. The latency is calculated as the interval starting when the MAP receives a join request from the 50^{th} (the 200^{th}) member and initiates the rekeying process, and ending when all members receive or finish computing the new group key. As the data rate of the background traffic increases, the average rekeying latency for a join operation in LKH, OFT and 802.11 increases slightly due to the increased background traffic the MAP needs to process. We observe that the join rekeying latencies of LKH and OFT are comparable, but much better than that of 802.11.

The graphs in Figure 5.16(c) (Figure 5.16 (d)) shows the average rekeying latency when the 50th (the 200th) member leaves the smaller (larger) group with the data rate varying from 0 to 2Mbits/s. As the data rate of the background traffic increases, the average rekeying latency for a leave operation in LKH, OFT and 802.11 slightly increases. We observe that the leave rekeying latency of OFT is slightly lower than that of LKH. They both perform better than 802.11 rekeying.

In summary, both LKH and OFT, given their logarithmic running time, perform better than the linear function of 802.11 when the group size increases. Experiments

show that the latency of 3-time broadcast for LKH and OFT is still much better than that of 802.11. Also, with the increase in the background traffic to the GKS (MAP), LKH and OFT perform better than 802.11.

5.4.4 Simulation Results of OFDM

The results are shown in Figure 5.17, Figure 5.18 and Figure 5.19.

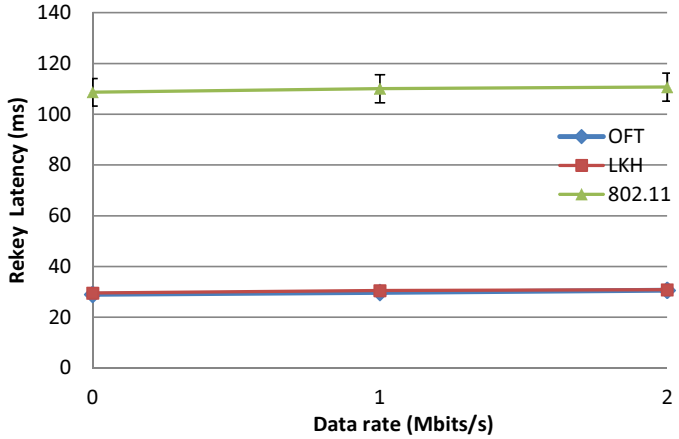
5.4.4.1 Function of Multicast Group Size

Figure 5.17 shows the rekeying latency of the join and leave operations of the protocols as a function of the group size. The group size in this set of experiments is varied from 1 to 50 for the smaller group of 50 nodes, and from 1 to 200 for the larger group of 200 nodes.

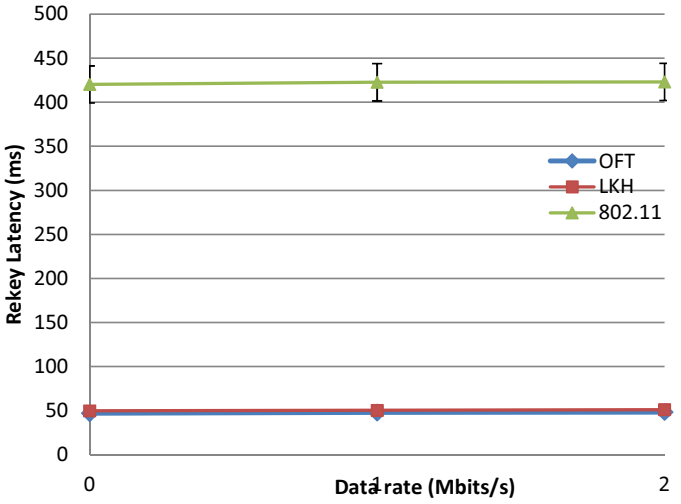
The rekeying latencies of the join operation are given in Figure 5.17(a) and Figure 5.17(b) for the 50-node group and the 200-node group, respectively.

- The rekeying latencies of LKH and OFT are comparable, and much lower than that of the 802.11 algorithm when the number of members $n > 5$. We observe that the average rekeying latency of 802.11 increases linearly, while that of LKH and OFT increases logarithmically.

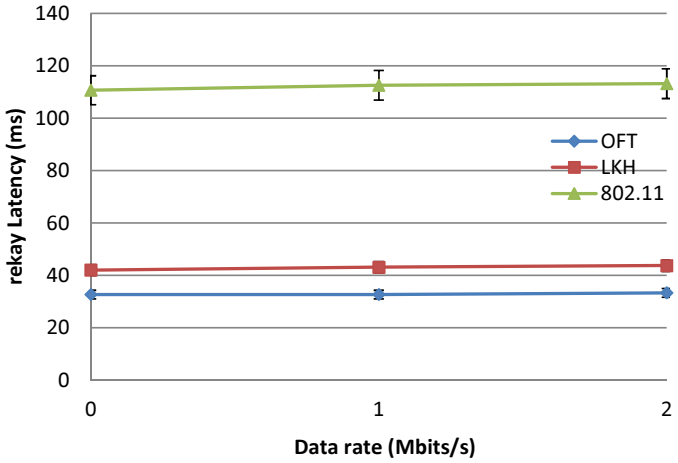
- As the number of joining nodes increases, the performance gap between the LKH (OFT) and 802.11 enlarges. For example, in Figure 5.17(b), the performance gap



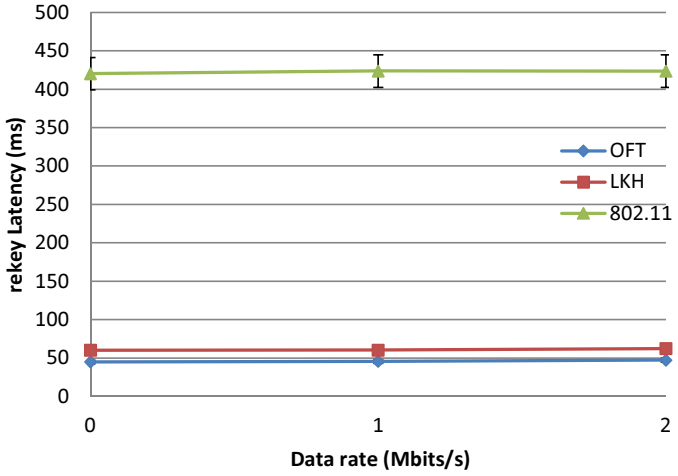
(a) DSSS - The 50^{th} member joins



(b) DSSS - The 200^{th} member joins



(c) DSSS - The 50^{th} member leaves



(d) DSSS - The 200^{th} member leaves

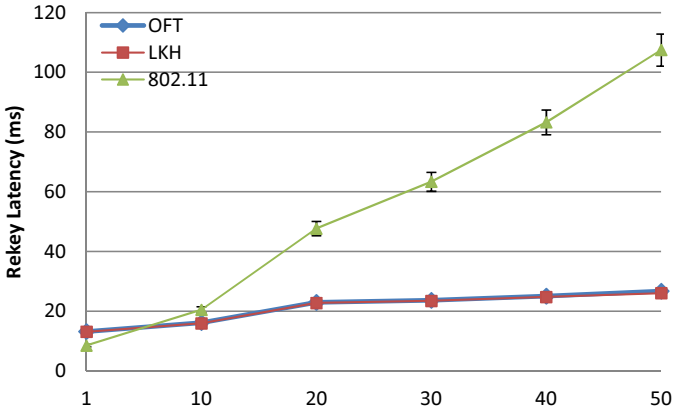
Figure 5.16: DSSS - The impact of background traffic

between LKH (OFT) and 802.11 enlarges from 81.43ms (80.71ms) to 384.27ms (383.98ms) when the group size increases from 50 to 200.

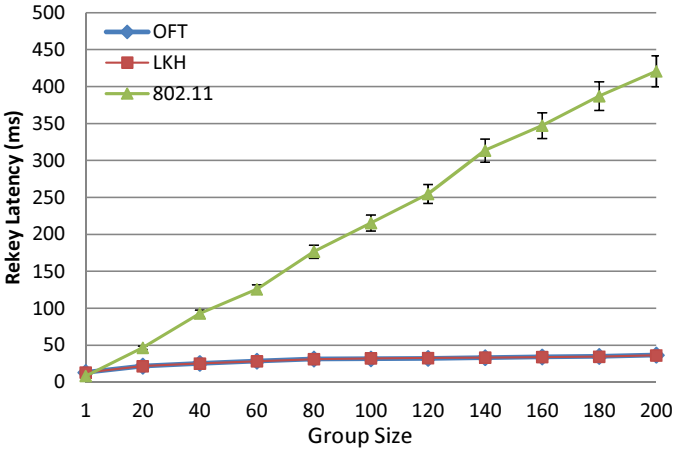
Figure 5.17 (c) and (d) shows the rekeying latency of the leave operation of the protocols, with the group size varying from 1 to 50, and from 1 to 200, respectively.

- In Figure 5.17(c), as the number of members increases from 1 to 50, the latency of OFT is approximately 33% to 82% lower than that of LKH.
- In Figure 5.17(d), as the number of members increases from 1 to 200, the latency of OFT is approximately 32% to 66% lower than LKH.
- Both OFT and LKH perform better than that of the 802.11 algorithm. The rekeying latency of LKH is much shorter than that of 802.11 when the number of members $n > 7$. In Figure 5.17(c), LKH's latency is approximately 3.2% to 64.7% lower than IEEE 802.11. OFT's latency is approximately 2.8% to 76.8% lower than IEEE 802.11.

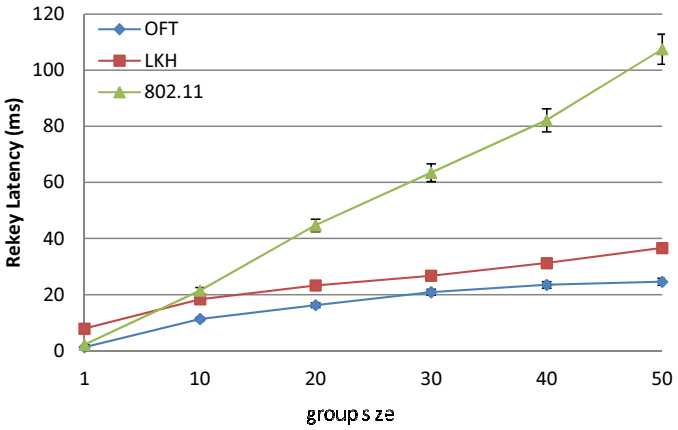
These simulation results are consistent with our numerical analysis. We observe that the rekeying latencies of three algorithms for join and leave operations implemented in OFDM are lower than those implemented in DSSS. For example, when the 50th member joins (leaves) the group, the rekeying latencies of LKH, OFT, and 802.11 implemented in OFDM are much lower than those implemented in DSSS, by 20.43% (13.06%), 17.85% (24.77%), and 2.89% (2.98%), respectively. The main reason is that OFDM supports a much higher data rate than that of DSSS (54 Mbits/s vs. 2 Mbits/s in our experiments).



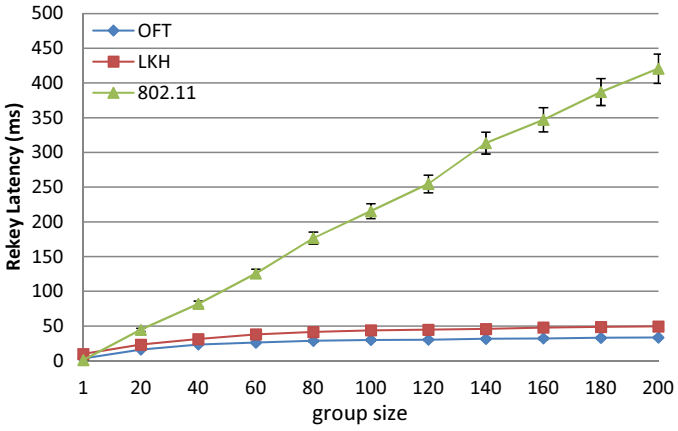
(a) OFDM - Group size from 1-50 - Join



(b) OFDM - Group size from 1-200 - Join



(c) OFDM - Group size from 1-50 - Leave



(d) OFDM - Group size from 1-200 - Leave

Figure 5.17: OFDM - Rekeying latency of Join/Leave operation

5.4.4.2 Function of Multicast Group Size with 3-time Broadcast

Figure 5.18 shows the join and leave rekeying latencies of LKH and OFT when each broadcast message is transmitted one time and three times. These results for OFDM are consistent with those from the DSSS experiments. LKH and OFT perform better than 802.11 in both cases, one-time and three-time broadcast.

- In Figure 5.18(a), given the group size varying from 1 to 50, the latency of OFT is approximately 8.6% to 9.7% lower than that of OFT-3. The latency of LKH is approximately 6.5% to 14.6% lower than LKH-3.

- In Figure 5.18(b), when the group size changes from 1 to 200, the latency of OFT is approximately 2.6% to 10.5% lower than that of OFT-3. The latency of LKH is approximately 2.6% to 6.5% lower than that of LKH-3.

- The latencies of LKH-3 and OFT-3 are much better than that of the 802.11 when the number of members $n > 10$. As the number of joining nodes increases, the performance gap between the LKH-3 (OFT-3) and 802.11 enlarges. For example, in Figure 5.18(b), the latency of LKH-3 is 73.4% (91.1%) lower than that of 802.11 for group size of 50 (200) nodes.

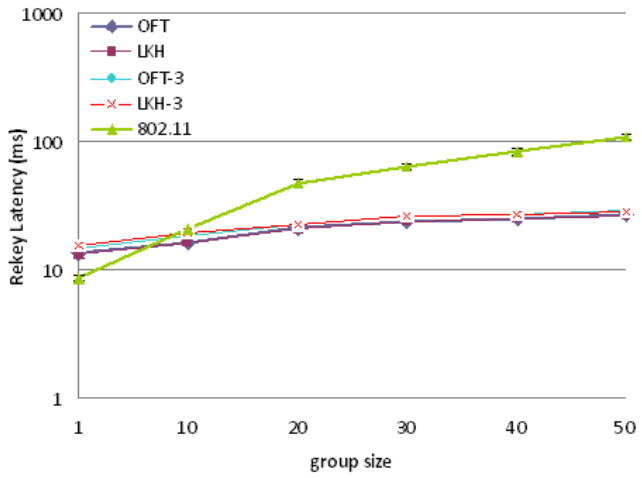
The rekeying latencies of the leave operation of the protocols using three-time broadcast are given in Figure 5.18(c) and Figure 5.18(d). These graphs show similar patterns as those from the DSSS experiments.

- In Figure 5.18(c), the latency of OFT (LKH) is approximately 4.6% to 6.7% (2.1% to 19%) lower than that of OFT-3 (LKH-3).
- In Figure 5.18(d), the latency of OFT (LKH) is approximately 7% to 42% (7% to 22%) lower than that of OFT-3 (LKH-3).
- The latencies of LKH-3 and OFT-3 are much lower than that of 802.11 when the number of members $n > 20$. As the number of leaving nodes increases, the performance gap between the LKH-3 (OFT-3) and 802.11b widens. For example, in Figure 5.18(d), the latency of LKH-3 (OFT-3) is 64.5% (74.7%) lower than that of 802.11 for group size of 50 nodes, and 87.6% (91.4%) lower than that of 802.11 for group size of 200 nodes.

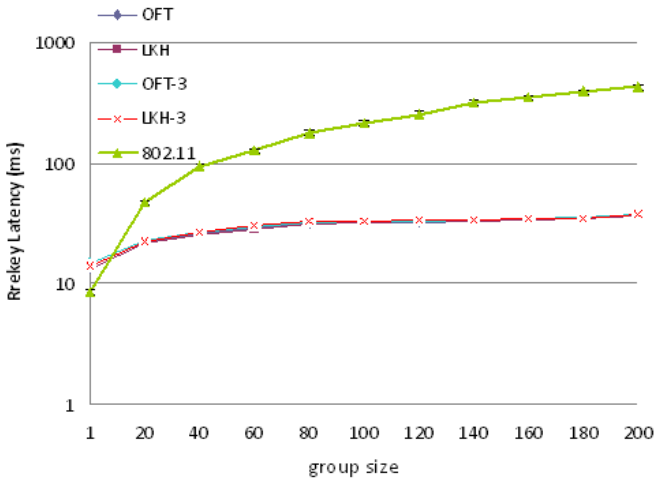
In general, the latency of three-time broadcast for both join and leave operations is slightly higher than that of one-time broadcast, but is still much lower than that of 802.11. We observe that the rekeying latency of three-time broadcast for join or leave operations implemented in OFDM is much lower than that implemented in DSSS because of the much higher data rate used in OFDM (54 Mbit/s vs. 2Mbit/s). For example, when the 200^{th} member joins (leaves) the group, the OFT rekeying latency of three-time broadcast for join (leave) operations for OFDM is 49.93% (35.88%) lower than that for DSSS.

5.4.4.3 Function of Multicast Traffic Load

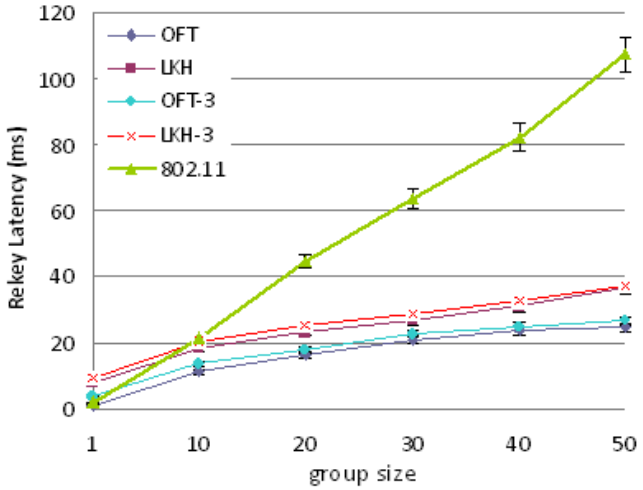
The graph in Figure 5.19(a) (Figure 5.19(b)) shows the average rekeying latency when the 50^{th} (the 200^{th}) member joins the smaller (larger) group with a background traffic



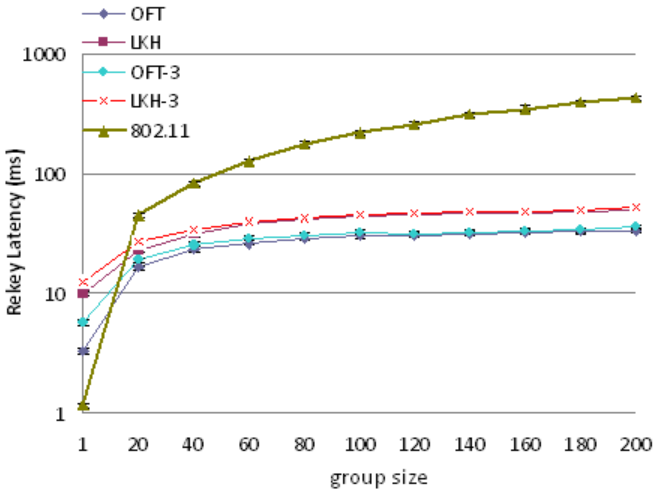
(a) OFDM - Group size from 1-50 - Join



(b) OFDM - Group size from 1-200 - Join



(c) OFDM - Group size from 1-50 - Leave



(d) OFDM - Group size from 1-200 - Leave

Figure 5.18: OFDM - Each broadcast message is sent 3 times

data rate varying from 0 to 50 Mbits/s. As the background traffic data rate increases from 0 to 50 Mbits/s, the average rekeying latency of LKH (OFT) is approximately 26.7% (26.9%) of that of 802.11 for the 50-node group, and 8.5% (8.7%) of that of 802.11 for the 200-node group, respectively.

We observe that the pattern of the graphs for the join operation are similar to those in the DSSS experiments. Due to the higher data rate used in OFDM experiments, the rekeying latencies for all three algorithms are slightly lower than those in DSSS as shown in Figure 5.16(a) and (b) of Section 5.4.3.3). For example, when the 50^{th} (200^{th}) member joins the group with the background traffic data rate varying from 0 to 2 MBits/s, the OFT rekeying latency in OFDM is 15.52% (30.87%) lower than that in DSSS.

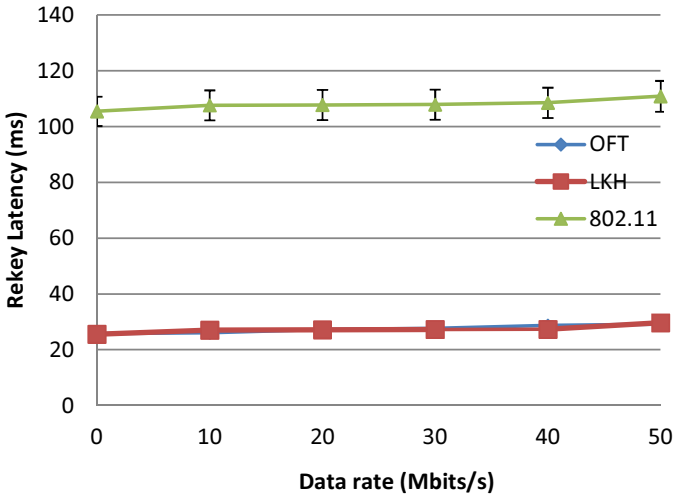
The graph in Figure 5.19(c) (Figure 5.19(d)) shows the average rekeying latency when the 50^{th} (the 200^{th} member) leaves the smaller (larger) group with a background traffic data rate varying from 0 to 50 Mbits/s. When the 50^{th} (200^{th}) member leaves the group, the average rekeying latency of LKH, OFT,and 802.11 increases as expected, by 7.3%, 9.3%, and 4.7%, (6.1%, 8.8%, and 1.6%,) respectively, due to the increase in background traffic load. The graph pattern and rekeying latency increase trend for the leave operation of OFDM are consistent with those for DSSS shown in Figure 5.16(c) and (d) of Section 5.4.3.3. The observed results indicate that the rekeying latency of the leave operation in OFDM is lower than that of DSSS. For example, when the 50^{th} (200^{th}) member leaves the group with the background traffic data rate varying from 0 to 2 Mbit/s, the OFT rekeying latency in OFDM is lower than that in DSSS by 28.98%

(34.88%). The main reason is the higher data rate used in OFDM, 54 Mbits/s vs. 2 Mbits/s in DSSS.

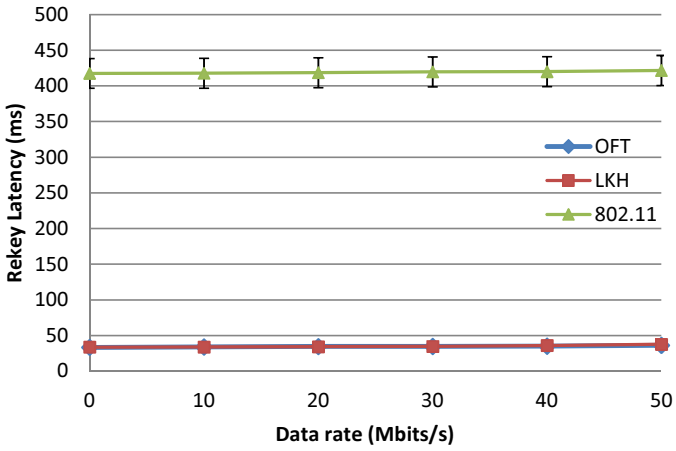
5.5 Chapter Summary

The group key management scheme defined in the IEEE 802.11s standard is neither efficient nor scalable because the rekeying latency grows linearly as a function of number of the clients connected to a mesh access point. The objective of our work is to provide an efficient and scalable rekeying method for group key management at the data link layer in WMNs in order to support real-time applications such as VoIP, tele-conferencing, and online HDTV.

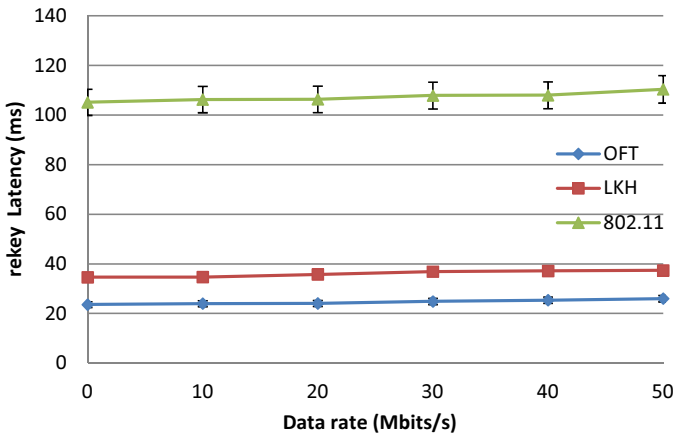
We show how the LKH and OFT algorithms can be applied to group key management (GKM) at the data link layer in WMNs in order to improve its performance. We evaluate and compare the performance of IEEE 802.11 GKM, LKH and OFT algorithms under realistic network settings using simulations. Our numerical analysis and simulation results confirm that the LKH and OFT algorithms reduce the rekeying latency of GKM at the data link layer from linear time to logarithmic time. Based on the numerical analysis and simulation results, we provide insights into the performance of IEEE 802.11 GKM, LKH and OFT algorithms in WMNs.



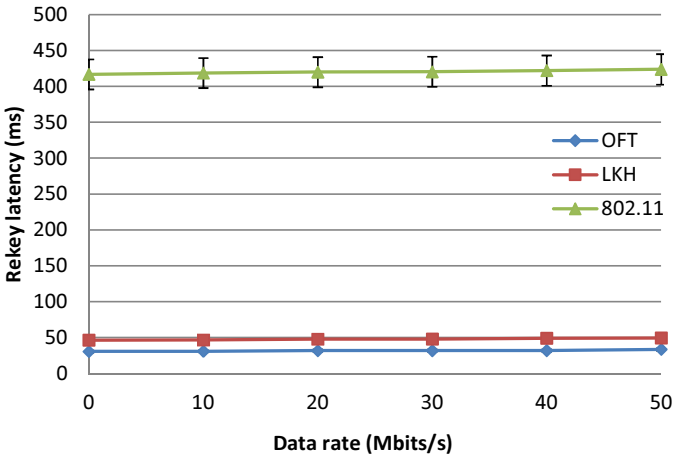
(a) OFDM - The 50^{th} member joins



(b) OFDM - The 200^{th} member joins



(c) OFDM - The 50^{th} member leaves



(d) OFDM - The 200^{th} member leaves

Figure 5.19: OFDM - The impact of background traffic

Chapter 6

Conclusion and Future Research Directions

In this chapter, we summarize the main results of the thesis, identify open issues, and outline research directions for future work.

6.1 Summary

We extend the capabilities of the IEEE 802.11s standard to support fast intra-network handovers for real-time applications such as VoIP, tele-conferencing, and stock quote distributions. We propose a novel security framework consisting of a new trust model, certificates and a suite of security protocols. Our authentication protocols support fast login and intra-network handovers in IEEE 802.11s networks. A client and a MAP mutually authenticate each other using one-hop communications. Fast authentication for roaming from one MAP to another is supported by using intra-network transfer certifi-

cates, which does not require an authentication server's participation during the authentication process. Instead, mobile clients and MAPs mutually authenticate each other directly, avoiding multi-hop wireless communications. Numerical analysis and simulation results show that our login authentication protocol improves the authentication latency of IEEE 802.11s, and our authentication protocol supports fast authentication during the handover process, which is lacking in IEEE 802.11s.

Our second contribution is a scalable security architecture, a trust model and certificates to support mobile clients roaming from one WMN to another. Our proposed architecture and trust model eliminate the involvement of a client's home network during the inter-network handover process. Based on the proposed architecture and trust model, we propose a suite of authentication and key distribution protocols. A mobile client and a foreign access point authenticate each other via one-hop communications using the client's inter-network transfer certificate, minimizing the authentication latency. Our security analysis shows that our protocols are resilient to various types of attacks. Our numerical analysis and simulation results demonstrate that our protocols dramatically reduce the inter-network handover latency in comparison with the home-foreign domain authentication approach.

Our third contribution is a new implementation of group key management at the data link layer in mesh access points of a WMN to minimize the group key update latency. Our proposed GKM implementation takes advantage of the logical key tree structure to reduce the rekeying latency of IEEE 802.11 GKM from linear time to logarithmic

time. The performance of the proposed implementation is evaluated via simulations under realistic network settings. Numerical analysis and simulation results confirm that the new implementation reduces the rekeying latency of GKM in WMNs from linear time to logarithmic time. We also provide insights into the performance of IEEE 802.11 GKM, LKH, and OFT algorithms in WMNs as well as recommendations of suitable GKM approaches to support fast handovers in WMNs.

6.2 Future Research Directions

In our future work on intra-network authentications, to take advantage of WMNs with extended network coverage through multi-hop communications, to reduce data traffic in the core network, and at the same time to satisfy the demand for mutual authentication among neighboring mesh clients, we will identify the criteria to support authentication and key distribution between clients via client certificates. We will further seek efficient solutions based on our proposed trust model to support intra-network handover security issues such as secure billing, secure routing, and access control.

Our designed security solutions for inter-network handovers are based on the assumptions of the same technology used by all WMNs, such as IEEE 802.11s. To provide more flexibility, we will further investigate the inter-network handover process/techniques to support seamless roaming among heterogeneous WMNs. Different technologies can be used to provide wireless connections to mesh clients moving freely in heterogeneous WMNs. A mobile device may have more than one wireless communication interface, thus

allowing the user to benefit from the advantages of different networking technologies. For example, a user with a PDA supporting both general packet radio service (GPRS) and Wi-Fi (IEEE 802.11) service can use a local area network with high data rates, and also benefit from the large coverage area of a GPRS network. However, inter-network roaming among the heterogeneous wireless mesh networks have several security challenges, such as (1) defining trusts between heterogeneous network entities belonging to different network operators; (2) minimizing the handover delay when users roam across multiple domains; (3) enabling a mobile device and a foreign network to authenticate each other, and to establish a shared key without any prior direct trust relationship. We will further explore how to extend our proposed architecture, trust models and certificates for providing security solutions to these issues.

Mobile clients sometimes are located in somewhere the wireless signal cannot reach. In this case, mobile clients may not be able to receive a new group key. Group key updating messages may also get lost. If a client fail to receive a new group key, it will not be able to continue decrypting the new multicast packets. We will further develop a key recovery mechanism for reliable group key management. To achieve better reliability, it is required that each mobile client will receive new group keys, no matter how large the group size is.

Appendix A

The Handover Authentication Protocol by Kassab et al.

Before a client C moves from a serving MAP M_1 to a target MAP M_2, C generates a certificate for M_2 and forwards it to M_1. M_1 will forward the certificate to M_2. Following is the structure of the certificate:

$$E_{IAPPkey}(I_C; E_{PMK}(I_C; K))$$

The certificate contains C's ID and a key K which C will share with M_2 after a successful authentication. Both C's ID and K are encrypted using the pairwise master key PMK shared by C and M_2 and pre-distributed by the authentication server to C and M_2. The encrypted message is then concatenated with C's ID. The content of the certificate is encrypted again with an IAPP (Inter-Access Point Protocol) key [142] shared by M_1 and M_2. After M_2 receives the certificate from M_1, it decrypts the message using

the shared IAPP key and the PMK to obtain key K to prepare for future authentication of client C.

Following is the authentication protocol (presented in the order of the messages exchanged):

(1) $C \longrightarrow M_2$: I_C, I_M

(2) $M_2 \longrightarrow C$: ACK

(3) $C \longrightarrow M_2$: $N_C, I_C, E_{KEK}(K_{Share}), V_{K_{MAC}}(N_C, I_C, E_{KEK}(K_{Share}))$

(4) $M_2 \longrightarrow C$: $I_M, V_{K_{MAC}}(I_M)$

(1) Client C submits its ID and M_1's ID to M_2. When M_2 receives this message, M_2 generates two keys using the shared key K, a key-encryption key KEK and a MAC key K_{MAC}.

(2) M_2 replies with an acknowledgment (ACK). After C receives the acknowledgment, C generates the same two keys KEK and K_{MAC} using the shared key K.

(3) Client C generates a nonce N_C and K_{share}. C encrypts a new key K_{Share} with key KEK. C sends N_C, the encrypted key K_{Share}, C's ID and a message authentication code

$V_{K_{MAC}}(N_C, I_C, E_{KEK}(K_{Share}))$ to M_2. When M_2 receives this message, it computes a MAC value $V'_{K_{MAC}}(N_C, I_C, E_{KEK}(K_{Share}))$.

If $V'_{K_{MAC}}(N_C, I_C, E_{KEK}(K_{Share}))$= $V_{K_{MAC}}(N_C, I_C, E_{KEK}(K_{Share}))$, M_2 has successfully authenticated the client C. M_2 decrypts $E_{KEK}(K_{Share})$ and obtains the key K_{Share}.

(4) M_2 sends its ID I_M along with the MAC value $V_{K_{MAC}}(I_M)$ to C. Upon receiving the message, C repeats the same MAC calculation on I_M. If it obtains the same message authentication code as $V_{K_{MAC}}(I_M)$, then this proves M_2's identity since M_2 is the only party in the network that has the knowledge of key K_{MAC}.

Appendix B

Adaptive Protocol - Authentication and Key Agreement (AP-AKA)

The AP-AKA [50] is a key distribution and authentication protocol for the universal mobile telecommunication system (UMTS). In the following diagram, a mobile user MS roams from its home network HN to a foreign network SN. AP-AKA is a mutual authentication protocol, which allows the foreign network SN to authenticate the mobile user MS. Also, the user MS authenticates its home network HN to make sure that the home network in the authentication process is a trusted one.

In the AP-AKA, each mobile device and its home network share a MAC key K and two cryptographic algorithms, V and G. V is a message authentication algorithm, and G is a key generation function. In practice, the MAC key K is usually generated by the

home network and programmed into a mobile device when a user initially sets up services with its home network.

The AP-AKA is executed as follows (presented in the order of the messages exchanged):

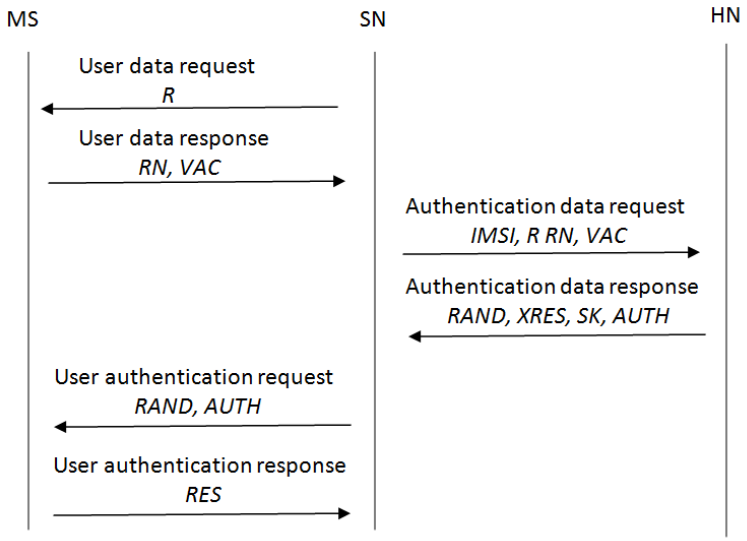


Figure B.1: AP-AKA

(1) A mobile user MS roams to a foreign network SN and requests to be connected to SN. SN sends a user data request message to MS, which includes a random number R.

(2) Mobile user MS generates a random number RN and computes a message authentication code (MAC), denoted by VAC.

$$VAC = V_k(RN||R||IDsn),$$

where K is a key shared by the user MS and its home network HN. ID_{sn} is the ID of SN. MS then sends a user data response back to SN including the random number RN and the MAC value VAC.

(3) Foreign network SN sends an authentication data request to home network HN, including the user's ID denoted by $IMSI$, random number R and RN, and MAC value VAC. Upon receipt of the authentication data request from SN, HN retrieves the user's MAC key K, and verifies the correctness of the received MAC value VAC.

(4) If the above verification is successful, HN generates a batch of authentication vectors. Each authentication vector consists of four components $RAND$, $XRES$, SK, and $AUTH$.

- $RAND$ is a random number generated by HN.
- $XRES = V_K(RAND)$ is an expected response, generated from the random number $RAND$.
- $SK = G_K(RAND)$ is a session key generated from the random number $RAND$.
- $AUTH = idx||RN_{idx}||VAT$ is an authentication token, indexed by an integer idx.
  - idx is an index number to identify the order of an authentication token in the batch.

- $RN_{idx} = V_K(idx||RN)$ is a MAC value generated from index number idx and random number RN.
- $VAT = V_K(RAND||idx||RN_{idx})$ is a MAC value.

HN first allocates an index number idx for the authentication vector, and generates a random number $RAND$. HN then computes an expected response $XRES$, a shared key SK and a authentication token $AUTH$. Finally, HN sends the authentication vectors $RAND, XRES, SK, AUTH$ to SN.

(5) After receiving the authentication vectors from HN, SN takes out one from the batch of the authentication vectors. SN then sends a user authentication request to the user, including $RAND$ and the selected authentication token $AUTH$.

After receiving this message, the user verifies the correctness of VAT in the authentication token $AUTH$. Since idx and RN_{idx} are already in $AUTH$ and $RAND$ is provided in message (5), MS then computes a MAC value $RAND||idx||RN_{idx}$ and compare it with VAT in $AUTH$. If they are equal, the user authenticates the home network HN.

(6) The user then generates $RES = V_K(RAND)$, where K is a key shared by HN and the user.

After receiving this user authentication response from MS, SN compares the RES and $XRES$ it received from HN in message (4). If they are same, SN can successfully authenticate user MS because only HN and MS know the key K.

Bibliography

[1] L. Lazos and M. Krunz, "Selective Jamming/Dropping Insider Attacks in Wireless Mesh Networks," IEEE Network, Vol. 25, No.1, pp. 30-34, 2011.

[2] Y. Zhang, J. Zheng, H. Hu , Security in Wireless Mesh Networks, Auerbach Publications, 2008.

[3] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," Proceedings of ACM Mobicom 2001, pp. 6169, 2001.

[4] H. Velayos and G. Karlsson, "Techniques to Reduce IEEE 802.11b MAC Layer Handover Time," KTH Technical Report, 2003.

[5] H. Dermanilian and F. Saab, "Energy-Efficient Security for Voice over IP," International Journal of Network Security, Vol.17, No.1, PP.11-26, 2014.

[6] Y. Amir and C. Danilov, "Fast Handoff for Seamless Wireless Mesh Networks," ACM MobiSys, 2006.

[7] A. M. Srivatsa and J. Xie, "A Performance Study of Mobile Handoff Delay in IEEE 802.11-Based Wireless Mesh Networks," IEEE International Conference on Communications, 2008.

[8] A. Harbitter and D. Menasce, "The Performance of Public Key Enabled Kerberos Authentication in Mobile Computing Applications," Proceeding of the 8th ACM conference on Computer and Communications Security, pp. 78-85, 2001.

[9] Y. Jiang, C. Lin, X. Shen and M. Shi, "Mutual Authentication and Key Exchange Protocols for Roaming Servbile Networks," IEEE Transaction on Wireless Communications, pp. 2569- 2577, 2006.

[10] B. Bhatt, "Handover Management in GSM Cellular System," International Journal of Engineering Development and Research, Vol.2, No. 2, pp.2096-2099, 2014.

[11] H. Ou, M. Hwang and J. Jan, "A Provable Billing Protocol on the Current UMTS," Wireless Personal Communications, Vol. 55, No. 4, pp.551-566, 2009.

[12] P. Lin and H. Chen, "A Secure Mobile Electronic Payment Architecture Platform for Wireless Mobile Networks," IEEE Transactions on Wireless Communications, Vol. 7, No. 7, pp. 2705-2713, 2008.

[13] Z. Liu, "An Efficient LKH Tree Balancing Algorithm for Group Key Management," ICCECT, pp. 1003-1005, 2012.

[14] J. Liu, "Collusion-Resistant Multicast Key Distribution Based on Homomorphic One-Way Function Trees," IEEE Transactions on Information Forensics and Security, Vol. 6, No. 3, pp. 980 - 991, 2011.

[15] A. J. Menezes, P.C. Oorschot and S. Avanstone, "Handbook of Applied Cryptography," 1996.

[16] S. Goldwasser and M. Bellare, "Lecture Notes on Cryptography," MIT Laboratory of Computer Science, 2001.

[17] S. Vaudenay, "A Classical Introduction to Cryptography: Applications for Communications Security," Springer, 2005.

[18] S. M. Faccin, C. Wijting, J. Kenckt, A. Damle, "Mesh WLAN Networks: Concept and System Design," IEEE Wireless Communication, Vol 13, No. 2, 2006.

[19] P. Yi, Y. Wu, F. Zou, N. Liu, "A Survey on Security in Wireless Mesh Networks," IETE Tech Rev, Vol. 27, pp. 6-14, 2010.

[20] M. Buddhikot, "Design and Implementation of a WLAN/CDMA 2000 Internetworking Architecture," IEEE Commuication Magazine, Vol. 41, No. 11, pp. 90-100, 2003.

[21] M. Shi, "A Certificate ID System for Service Agent Based Authentication in WLAN/Cellular Integrated Networks," IEEE WCNC, pp. 4151 - 4155, 2007.

[22] J. Leu, R. Lai, H. Lin, and W. Shih, "Running Cellular/PWLAN Services: Practical Considerations for Cellular/PWLAN Architecture Supporting Interoperator Roaming," IEEE Communication Magazine, Vol. 44, No. 2, pp. 7384, 2006.

[23] U. Noor, Z. Anwar, W. Aslam, "TrustBook: Web of Trust Based Relationship Establishment in Online Social Networks," FIT 2013, pp. 223 - 228, 2013.

[24] A. Herzberg,"Access control meet public key Infrastructure, Or: Assigning Roles to Strangers," The IEEE Symposium on Security and Privacy, Berkley, California, USA, pp.2-14, May 2000.

[25] Y. Kurniawan, "The design of mini PGP security", ICEEI, pp. 1-4, 2011.

[26] S. Yi and R. Kravets, "Key Management for Heterogeneous Ad hoc Wireless Networks," 10th IEEE International Conference on Network Protocols, ICNP 2002.

[27] D. D. Falconer, F. Adachi and B. Gudmondson, "Time Division Multiple Access Methods for Wireless Personal Communications," IEEE Communications Magazine.

[28] V. Rajaraman, Fundamentals of Computers, 2015.

[29] A. Mondal, Mobile IP, Springer, 2003.

[30] Y. Zou, "Design and Implementation of 802.1x Authentication Module in Security Switch for Industrial Control System Based on FPGA," National Conference on Information Technology and Computer Science (CITCS), 2012.

[31] A. Alabdulatif1 and X. Ma, "Analysing the EAP-TLS Handshake and the 4-Way Handshake of the 802.11i Standard," International Journal for Information Security Research (IJISR), Vol. 3, No. 3, 2013.

[32] U. Meyer and S. Wetzel, "Introducing History-enriched Security Context Transfer to Enhance the Security of Subsequent Handover," IEEE Workshop on Pervasive Computing and Communication Security, 2006.

[33] G. Horn, M. Martin and C. Mitchell, "Authentication Protocols for Mobile Network Environment Value-Added Services," IEEE Transaction on Vehicular Technology, Vol. 51, No. 2, 2002.

[34] M. Kassab, "Securing Fast Handover in WLANs: A Ticket Based Proactive Authentication Scheme," Globecom2007, 2007.

[35] IEEE, "Part11: Wireless Medium Access Control (MAC) and Physical Layer Specifications: Medium Access Control (MAC) Security Enhancement," IEEE Standard 802.11i/D10.0, 2003.

[36] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, "Protocol for Carrying Authentication and Network Access (PANA)," RFC 5191, 2008.

[37] Y. Jiang, C. Lin, X. Shen and M. Shi, "Mutual Authentication and Key Exchange Protocols for Roaming Services in Wireless Mobile Networks," IEEE Transaction on Wireless Communications, 2006.

[38] S. Shen, S. Lin and J. Chiu, "Fast Handover Pre-Authentication Protocol in 3GPP-WLAN Heterogeneous Mobile Networks," International Journal of Communications, Network and System Sciences, Vol.7, No.4, pp. 101-113, 2014.

[39] S. Lin, J. Chiu and G. Lee, "A Fast Iterative Localized Re-authentication Protocol for Heterogeneous Mobile Networks," IEEE Transaction on Consumer Electronic, Vol. 56, No. 4, pp. 2267-2276, 2010.

[40] A. Mishra, M. Shin, N. L. Petroni, T. C. Clancy and W. Arbaugh, "Pro-active Key Distribution Using Neighbour Graphs," IEEE Wireless Communications, vol. 11, no. 1, 2004.

[41] C. Park, J. Hur, C. Kim, Y. Shin, and H. Yoon, "Pre-authentication for Fast Handoff in Wireless Mesh Networks With Mobile APs," WISA'06, Information Security Applications, 2007.

[42] G. Li, "A Ticket-based Authentication Scheme for Fast Handover in Wireless Local Area Networks," Wireless Communications Networking and Mobile Computing (WiCOM) Conference, 2010.

[43] "Draft Amendment: ESS Mesh Networking," IEEE 802.11s IEEE802.11s/D3.0, 2009.

[44] G. R. Hiertz, "IEEE 802.11s: The WLAN Mesh Standard," IEEE Wireless Communications, Vol 17, No. 1, 2010.

[45] P. Huang, "A Fast Handoff Mechanism for IEEE 802.11 and IAPP Networks," IEEE ITC, 2006.

[46] M, Kassab, "Fast Pre-Authentication Based on Proactive Key Distribution for 802.11 Infrastructure Networks," ACM WMuNeP, 2005.

[47] W. Du, J. Deng, Y. Han, P. Varshney, J. Kate and A. khalili, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," ACM Conference on Computer and Communications Security (CCS 03), 2003.

[48] S. Qazi, "Securing Wireless Mesh Networks With Ticket-Based Authentication," Signal Processing and Communication System, 2008.

[49] A. Fu, "A Fast Handover Authentication Mechanism Based on Ticket for 802.16m," IEEE Communication Letter, Vol. 14, No. 12, 2010.

[50] M. Zhang and Y. Fang, "Security Analysis and Enhancements of 3GPP Authentication and Key Agreement Protocol," IEEE Transacions on wireless communications, Vol, 4, No. 2, 2005.

[51] K. Hoeper, "Security Challenges in Seamless Mobility: How to Handover the Keys?" 2008.

[52] J. Lee, and W. Tseng, "A Wide-adapted Bantam Protocol for Roaming Across Wireless Areas," Wireless Network, Vol. 19, No. 4, pp.1423-1440, 2013.

[53] H. K. Aslan, "An Efficient and Secure Handover Protocol for IEEE 802.16m Networks," 2013.

[54] B. Aboba, "Fast Handoff Issues," IEEE-03-155r0-I, IEEE Working Group, 2003.

[55] D. D. Couto, et al., "A High-Throughput Path Metric for Multi-hop Wireless Routing," ACM MobiCom, 2003.

[56] A. Oliva, "An Overview of the IEEE 802.21: the Media-Independent Handover Services," IEEE Wireless Communications, Vol. 15, No. 4, pp. 96-103, 2008.

[57] W. Juang, "Efficient 3GPP Authentication and Key Agreement with Robust User Privacy Protection," WCNC, pp. 2720-2725, 2007

[58] J. Cao, "An Enhanced Authentication Scheme in GLOMONETs," IC-BNMT, pp. 579-585, 2011.

[59] L. Buttyan, C. Gbaguidi, and et al., "Extensions to An Authentication Technique Proposed for the Global Mobility Network," IEEE Transaction on Communications, Vol. 48, No. 3, pp. 373-376, 2000.

[60] K. Hwang and C. Chang, "A Self-encryption Mechanism for Authentication of Roaming and Teleconference Services," IEEE Transaction in Wireless Communications, Vol. 2, No.2, pp. 400-407, 2003.

[61] Y. Tian, "Secret Sharing Scheme with Fairness," TrustCom, pp. 494-500, 2011.

[62] J. Zhang, "Efficient Self-Certified Blind Signature Scheme with Parings," WCSN, pp. 198-203, 2014.

[63] Y.L. Huang, P.H. Lu, J.D. Tygar and A.D. Joseph, "OSNP: Secure Wireless Authentication Protocol Using One-time Key," Computer and Security, 2009.

[64] P. Vijayakumar, "Chinese Remainder Theorem Based Centralised Group Key Management for Secure Multicast Communication", IET Information Security, Vol. 8, No. 3, pp. 179-187, 2014,

[65] J. Park, "A New Centralized Group Key Distribution and Revocation in Sensor Network", International Conference on Computational Intelligence and Security, pp. 721 - 724, 2007.

[66] J. Lin, "Optimizing Centralized Secure Group Communications with Binary Key Tree Recomposition," 18th International Conference on Advanced Information Networking and Applications (AINA), pp. 202-207, 2004.

[67] X. Yanyan, X. Zhengquan, C. Xi, and Y. Zhanwu: "A Scalable De-Centralized Multicast Key Management Scheme," Proceedings of the First International Conference on Innovative Computing, Information and Control, pp. 463-467, 2006.

[68] X. Guo, X. Liu, and Q. Dai: "A Decentralized Key Management Scheme in Overlay Multicast Network," IEEE International Conference on Multimedia and Expo, pp. 1201 - 1204, 2006.

[69] W. Yu, Y. Sun, and K. J. R. Liu, "Optimizing Rekeying Cost for Contributory Group Key Agreement Schemes," IEEE Transactions On Dependable And Secure Computing, Vol. 4, No. 3, pp.228-242, 2007.

[70] CH. V. Raghavendran, G. Naga Satish, P. Suresh Varma, "A Study on Contributory Group Key Agreements for Mobile Ad Hoc Networks," International Journal of Computer Network and Information Security, Vol. 5, No. 4, pp. 48-56, 2013.

[71] R. Dutta and R. Barua, "Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting," IEEE Transactions on Information Theory, Vol. 54, No. 5, pp. 2007-2025, 2008.

[72] X. Wang and A. O. Lim, "IEEE 802.11s Wireless Mesh Networks: Framework and Challenges," Ad Hoc Networks Journal (Elsevier), Vol. 6, No. 6, 2008.

[73] S. Manuel, "Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1," Designs, Codes and Cryptography, Vol. 59, No. 3, 2011.

[74] RFC 2104 - HMAC: Keyed-Hashing for Message Authentication.

[75] A. Biryukov, S. Mukhopadhyay, and P. Sarkar, "Improved Time-Memory Trade-Offs With Multiple Data," 12th Annual Workshop on Selected Areas in Cryptography, Lecture Notes in Computer Science, Springer, 2005.

[76] K. Khoo, "New Time-memory-data Trade-off Attack on the Estream Finalists and Modes of Operation of Block Ciphers," The 7th ACM Symposium on Information, Computer and Communications Security, pp. 20-21, 2012.

[77] G. Dua and N. Gautam, "Replay Attack Prevention in Kerberos Authentication Protocol Using Triple Password," International Journal of Computer Networks and Communications, Vol. 5, No. 2, pp. 59-70, 2013.

[78] A. Mittal, "A Review of DDOS Attack and its Countermeasures in TCP Based Networks," International Journal of Computer Science and Engineering Survey, Vol. 2, No. 4, 2011.

[79] L. Hu and X. Bi, "Research of DDoS Attack Mechanism and Its Defense Frame," Computer Research and Development (ICCRD), 3rd International Conference, pp. 440-442, 2011.

[80] Y. Chen, S. Das, P. Dhar, A. Saddik, and A. Nayak, "Detecting and Preventing IP-spoofed Distributed DoS Attacks," International Journal of Network Security, Vol. 7, No. 1, pp. 70-81, 2008.

[81] Y. Xu and W. Wang, "Detecting and Mitigating DoS Attacks in Wireless Networks Without Affecting the Normal Behaving Nodes," IEEE Military Communication Conference (Milcom'07), 2007.

[82] B. Pooja, "Mitigation of Insider and Outsider DoS Attack Against Signature based Authentication in VANETs," Asia-Pacific Conference on Computer Aided System Engineering (APCASE), pp. 152-157, 2014.

[83] A. Kanthe, "Denial of Service (DoS) Attacks in Green Mobile Ad-hoc Networks," MIPRO, pp. 675-680, 2012.

[84] T. J. McNevin, J. Park, and R. Marchany, "PTCP: A Client Puzzle Protocol for Defending Against Resource Exhaustion Denial of Service Attacks," Technical Report TR-ECE-04-10, 2004.

[85] N. Ruan, "DoS Attack-tolerant TESLA-based Broadcast Authentication Protocol in Internet of Things," iCOST, pp. 60-65, 2012.

[86] S. KhakAbi, "Preventing SYN Flood DoS Attacks: An Improvement to SYN Cookies," IEEE International Conference on Intelligence and Security Informatics, 2009.

[87] Y. Xu and W. Wang, "Detecting and Mitigating DoS Attacks in Wireless Networks Without Affecting the Normal Behaving Nodes," IEEE Military Communication Conference (Milcom07), 2007.

[88] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten. "New Client Puzzle Out-Sourcing Techniques for DoS Resistance," The 11th ACM Conference on Computer and Communications Security (CCS), 2004.

[89] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: Software-based Attestation for Embedded Devices," IEEE Symposium on Security and Privacy, 2004.

[90] A. Seshadri and M. Luk, "Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy System," 20th ACM Symposium on Operating Systems Principles, 2005.

[91] M. Long, "Energy-efficient and Intrusion Resilient Authentication for Ubiquitous Access to Factory Floor Information," IEEE Transaction on Industrial Informatics, 2006.

[92] A. AlSa and C. Meinel,"Secure Neighbor Discovery: Review, Challenges, Perspectives, and Recommendations," IEEE Security and Privacy Magazine, Vol. 10, No. 4, pp. 2634, 2012.

[93] M. Peirce and D. O'Mahony, "Flexible Real-time Payment Methods for Mobile Communications," IEEE Personal Communications, Vol. 6, No. 6, pp. 44-55, 1999.

[94] M. Mahmoud and X. Shen, "A Secure Payment Scheme with Low Communication and Processing Overhead for Multi-hop Wireless Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 2, pp. 209-224, 2013.

[95] International Telecommunication Union, "Seriers G: Transmission System and Media, Digital Systems and Network," ITU-T Recommentation G.114, 2003.

[96] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, Vol. IT-22, No. 6, 1976.

[97] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp130-126, 1978.

[98] E. Knipp, B. Browne, W. Weaver, C. T. Baumrucker, L. Chaffin, J. Caesar, V. Osipov, E. Danielyan, Managing Cisco Network Security, Second Edition, Syngress Publishing, Inc., 2002.

[99] X. Zhou, "Research and Implementation of RSA Algorithm for Encryption and Decryption," 6th IEEE International Forum on Strategic Technology (IFOST), pp. 1118-1121, 2011.

[100] ECDSA, FIPS 186-3, Digital Signature Standard (DSS), 2009.

[101] R. Gupta, "Simulation of AES Encryption and Decryption Algorithm with Parallel Data Execution," International Journal of Electronics Communication and Computer Engineering, Vol. 3, No. 3, 254-258, 2012.

[102] QualNet Simulator, http://www.scalablenetworks.com/.

[103] TROPOS Networks. http://www.tropos.com.

[104] http://www.aircrack-ng.org/

[105] Cisco Aironet 802.11 Wireless Adapter. http://www.cisco.com/.

[106] D. Pong and T. Moors, "The Impact of Random Way point Mobility on Infrastructure Wireless Networks," Proceedings of the 11th International Conference on Parallel and Distributed Systems, pp. 140-144, 2005.

[107] Cisco Wireless Mesh Access Points, Design and Deployment Guide, Release 7.3, 2012.

[108] H. Amirazizi, "Time-Memory-Processor Trade-offs," IEEE Transactions on Information Theory, Vol. 34, No. 3, pp. 505-512, 1988.

[109] J. Katz, Introduction To Modern Cryptography, 2014.

[110] A. Barth, "Robust Defenses for Cross-site Request Forgery," The 15th ACM Conference on Computer and Communications Security, 2008.

[111] S. Lee, "A MPLS-based Micro-mobility Supporting Scheme in Wireless Internet," Lecture Notes in Computer Science, vol. 3314, pp. 160-165, 2005.

[112] Z. Liu, "Embedding Complete Binary Trees Into Parity Cubes," The Journal of Supercomputing, pp. 1-27, 2014.

[113] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," IEEE Standard 802.11, 1999.

[114] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: High-speed Physical Layer Extension in the 2.4 GHz Band," IEEE Standard 802.11, 1999.

[115] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-speed Physical Layer in the 5Gz Band," IEEE Standard 802.11, 1999.

[116] "Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-speed Physical Layer in the 2.4 GHz Band," IEEE Standard 802.11b, 1999.

[117] "Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-speed Physical Layer Extensions in the 2.4 GHz Band," IEEE Standard 802.11g, 2003.

[118] G. Swaminathan, "Key Management for Secure Group communication in Wireless Networks - A survey," International Conference on Contemporary Computing and Informatics (IC3I), pp. 446-450, 2014.

[119] Y, Xu, "A Scalable De-centralized Multicast Key Management Scheme," First International Conference on Innovative Computing, Information and Control (ICICICv), pp. 463-468, 2006.

[120] T. Mapoka, "Efficient Authenticated Multi-service Group Key Management for Secure Wireless Mobile Multicast," Third International Conference on Future Generation Communication Technology (FGCT), pp. 66-71, 2014.

[121] H. Li, "Improvement on WLAN Multicast Key Management Protocol," International Conference on Computational Intelligence and Security, Vol. 2, pp. 419-424, 2008.

[122] A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, 1996.

[123] J. Bian, "A scalable Role-based Group Key Agreement and Role Identification Mechanism," pp. 278-281, 2011.

[124] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture," RFC 2094, 1997.

[125] M. Park, "Enhancement of Cell-based Decentralized Key Management in Vehicular Communication Networks," Wireless VITAE, pp. 1-6, 2011.

[126] H. Um and E. J. Delp, "A Secure Group Key Management Scheme for Wireless Cellular Networks," 2006.

[127] B. Wu, "A Simple Group Key Management Approach for Mobile Ad Hoc Networks," NAS2010, pp. 73-78, 2010.

[128] S. Tu, "The Research of Hierarchical Group Key Management for Ad Hoc Networks," Intelligent Computing and Integrated Systems (ICISS), pp. 121-124, 2010.

[129] H. Lin, "Multi-level and Group-based Key Management for Mobile Ad Hoc Networks," Information Security and Intelligence Control (ISIC), pp. 164-167, 2012.

[130] C. Miao, "A Lightweight Group-Key Management Protocol for Ad Hoc Networks," Genetic and Evolutionary Computing (ICGEC), pp. 288-291, 2012.

[131] Y.Kim, A.Perrig, and G.Tsudik, "Tree-based Group Key Agreement," ACM Transaction Information System Security, No.7, pp. 6096, 2004.

[132] M. Qabajeh, "A Tree-based QoS Multicast Routing Protocol for MANETs," The fourth International Conference on Mechatronics (ICOM), pp. 1-6, 2011.

[133] M. Huang, "Cooperative Loss Recovery for Reliable Multicast in Ad Hoc Networks," International Journal of Communications Network and System Sciences (IJCNS), No. 3, pp. 72-78, 2010.

[134] U. T. Nguyen, "On Multicast Routing in Wireless Mesh Networks," Elsevier Journal of Computer Communications, Special Issue on Resource Management and Routing in Wireless Mesh Networks, Vol. 31, No. 7, pp. 1385-1399, 2008.

[135] RFC3376, tools.ietf.org/html/rfc3376.

[136] P. Marchetta, "Quantifying and Mitigating IGMP Fltering in Topology Discovery," In IEEE GLOBECOM, 2012.

[137] G. Berton, "Efficient Software Implementation of AES on 32-bit Platforms," The fourth International Workshop on Cryptographic Hardware and Embeded System, 2002.

[138] J. Jun, P. Peddabachagari and M. Sichitiu, "Theoretical Maximum Througput of IEEE 802.11 and its Application," The 2nd IEEE International Symposium on Network Computing and Applications, 2003.

[139] M. Mandal, Motorola Solutions WiNG 5.4.1, Access Point System Performance Guide, 2012.

[140] S. Stankovic, Multimedia Signals and Systems, Springer, 2012.

[141] J. Liu and B. Yang, "Collusion-resistant Multicast Key Distribution Based on Homomorphic One-way Function Trees," IEEE Transactions on Information Forensics and Security, Vol. 6, No. 3, pp. 980-991, 2011.

[142] E. Garcia, L. Faixo, R. Vidal, and J. Paradells, "Inter-Access Point Communication for Distributed Resource Management in 802.11 Networks," WMASH, 2006.